# Deployment of Calabash Automation Framework to Analyze the Performance of an Android Application

**Madhuri Kishan Kulkarni**
*Department of Computer Science & Engineering*
*RVCE Bangalore*

**Prof Soumya A**
*Department of Computer Science & Engineering*
*RVCE Bangalore*

## Abstract

The use of mobile phones has grown rapidly and has become a vital part in the present era. The conventional software testing practices are not really feasible for mobile applications. Test Automation does not cover all the necessary features, which makes manual testing also very crucial. The challenge is even greater. The problem is even greater in projects where agile methodologies are deployed. In such projects Test Automation plays a very significant role and is very important during the development cycle. The Automation Framework should be deployable on multiple heterogeneous platforms. The actions that are frequently used in the mobile devices are extracted. They are then mapped into the corresponding functions of Calabash testing framework.The objective and intention of this paper is to bring out significant merits and demerits of different Automation Frameworks and then use the Calabash Automation Framework to develop the Performance Analysis module which can effectively determine the launch time of the mobile application.

**Keywords: Mobile Application, Mobile Test Automation, Calabash Automation Framework, Performance Analysis**

## I. INTRODUCTION

There is a significant leap in mobile platform technologies and mobile applications. The mobile applications are becoming complex. The greatest hurdle is not just to build the applications but to guarantee their effectiveness and stability. Testers have two options doing it manually or with the help of Automation using test scripts. Another option would be to do it on real mobile devices or with the help of emulators. The simulators have restricted processing and storage capacities, the testing team should also test the applications on real mobile devices [1]. There are several test automation frameworks and tools that are to deploy the automated test scripts on mobile devices. The Framework that is deployed for any mobile application should ensure that:

- It is interoperable on both iOS and Android
- It should support both device and simulator
- Parallel test execution on multiple devices
- Flexibility to generate the report in different formats

All the Test Automation Frameworks should be studied extensively before deploying it for a particular mobile application.

Another main component is the Performance of the mobile application is imperative when it is released into the app store. The performance of the application is determined by the launch time. The launch time can be found by manually keeping tracking of time for each launch of the application. But this can be error prone and tedious. In order to overcome this loophole the launch time can be found with the help of the automated test script. The existing automation methods are as classified: Recording/Playback Method, Scripting Method and Screen Capture Method [1].

Table – 1
Test Automation Techniques

| *TYPE* | *CHARACTERISTIC* |
|---|---|
| ***Recording/Playback Method*** | ***In this method it is possible to record events that the user incurs. Test cases can be played back.*** |
| ***Scripting Method*** | ***The test cases are automated with the help of programming language.*** |
| ***Screen Capture Methodology*** | ***The images are captured from the screen and then they are processed to run the automated tests. [2][3]*** |

## II. LITERATURE SURVEY

As proved in some studies [3], [4], mobile applications are not error free and novel software engineering approaches are necessary to test those applications [5].This section gives an overview of the test automation tools and a survey of the most suitable tools that can be deployed for automation on various mobile devices.

### *A. Automation Test Tools:*

Automation Test tool automates all the routine steps that are involved in a test. Each application has a different tool that needs to be used. Test Automation Tools simplify the testing process thus helping to make it easy and effective. Some popular test automating tools that are used for the comparative study are as listed:

*1) Calabash:*

- Calabash supports different mobile platforms that are Android and iOS.
- Supports for native as well as hybrid mobile applications.
- It also helps the non-computer science testers to write the test scripts. The execution of automated acceptance tests is very simple on both Android and IOS mobile devices because of Calabash are easy to understand syntax.
- Calabash's tests are written with the help of Cucumber and are converted during run time to Robotium or Frank.
- Calabash can only support Ruby

Table – 2
Analysis leading to conclusion

| ***Features*** | ***Calabash*** | ***Espresso*** | ***Robotium*** |
|---|---|---|---|
| ***Android*** | ***Yes*** | ***Yes*** | ***Yes*** |
| ***iOS*** | ***Yes*** | ***No*** | ***No*** |
| ***Mobile web*** | ***Yes (Android)*** | ***No*** | ***Yes*** |
| ***Scripting Language*** | ***Ruby*** | ***Java*** | ***Java*** |
| ***Supported API levels*** | ***All*** | ***8, 10, 15-19*** | ***8 and above*** |
| ***Community Support*** | ***Limited*** | ***Google*** | ***Limited*** |
| ***Parallel Execution*** | ***No*** | ***Yes*** | ***No*** |
| ***License*** | ***Open Source*** | ***Open Source*** | ***Open Source*** |

*2) Espresso:*

- Espresso is implemented on Android Instrumentation Framework. It is small API that is supported on API level such as 8 (Froyo), 10 (Gingerbread), and 15 (Ice Cream Sandwich) . It is fairly a simple Automation tool.
- It synchronizes with UI thread and hence this makes it more reliable. It does not require sleeps. When the app becomes idle the tests are executed on the same millisecond.
- Webviews are not supported.
- It is the most recent Framework that was launched by Google and it's open-sourced. This makes it available easily for the developers and testers

*3) Robotium:*

- There are plenty easy to use methods that extends the Junit that can be used for Android testing.
- The black-box test cases that are executed are effective and robust.
- Native and Hybrid mobile web testing is supported.
- The community has good support and there are intermediate releases for this Automaton tool.
- The functions, system and acceptance and test scenarios can be written with the availability of the source code.
- Android Robotium 7 is a open source that allows the black-box testing of acceptance and functional test cases. There is a strong resemblance to Selenium but this tool is for Android application.[6]

Test Complete [7] provides a solution for different Mobile platforms. The See Test plug-in for Test Complete provides a complete test Automation solution that has data driven options and also OCR. The Automation tool is chargeable and is not open sourced [8].

RealMobile™ [9], is a mobile Test Automation Framework. This tool makes use of PC along with tools like QuickTestPro. The tests can be run on the device, But a PC is required to run the test scripts. The mobile device needs to be plugged to the PC with the help of Wi-Fi, cable or with the help of the cellular network.

With the above detailed analysis done for each Automation Frameworks/Tools, it is quite evident that Calabash is the best match. The tool that is selected needs to in accordance with the requirements of the project. One tool may not suffice for every other mobile application and hence careful study and analysis of the various tools is a must

## III. DESIGN OF CALABASH AUTOMATION FRAMEWORK

### *A. Anatomy of the Calabash Automation Framework:*

Calabash helps in providing a bridge that enables the Cucumber tests to execute and validate the test cases for Android and IOS mobile applications as depicted in Figure 1.

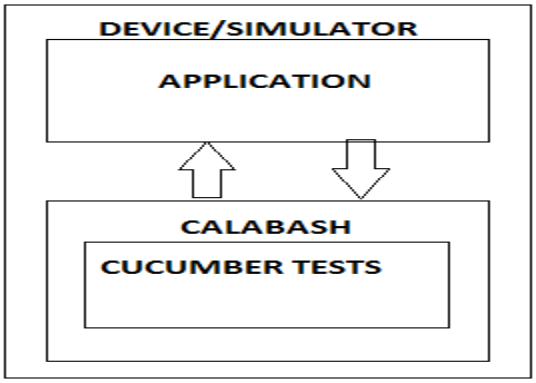


Fig. 1: Anatomy of the Calabash Automation Framework

The specifications are written using plain English language and the grammar rules that are written is called Gherkin, and the glue that helps the Gherkin language tests run is the code behind it called Step Definitions that allows the specification to be tested.

### *B. System Architecture of Calabash Automation Framework:*

The various components that are included in the Calabash Automation Framework is depicted in Figure 2.

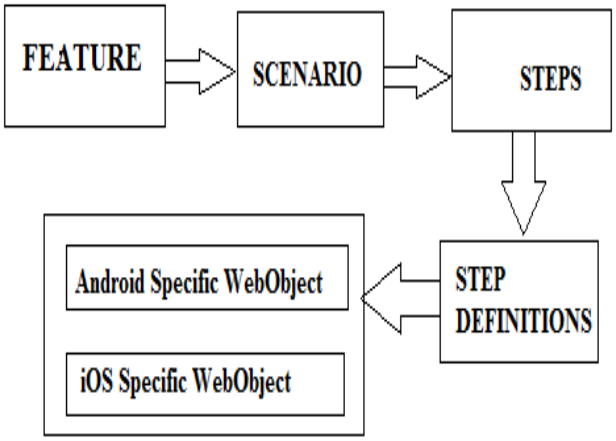


Fig. 2: System Architecture of Calabash Automation Framework

1) Feature: It is a collection of the test cases that are necessary to execute a Sanity/Regression Test Suite.
2) Scenario: These are the different test cases that constitute the Feature file. It can have several test cases and each test case is called the Scenario.
3) Steps: Each scenario consists of several Gherkin language steps. (Plain English)
4) Step Definitions: The backend or the code supporting the Feature file is written in the Ruby file. The Gherkin language steps should have a corresponding Ruby language code/snippet that defines the functionality.

### *C. Test Case Sample:*

Feature: To test the login functionality
Scenario 1: Login Functionality with correct credentials

- Given my app is launched and it's on the login page
- When I type invalid credentials
- And I press the Log-in button
- Then Error message has to appear stating "Invalid username or password"

Here Given symbolizes the state of the system that the application should be present, When describes the actions performed by the user and Then describes the outcomes observed. So in the above test case sample, the series of actions include verifying the user being on the Login page, Entering text, tapping the button, validating the message. The common actions that are supported and their respective class used in the Android application are as shown in Table 3 [10].

Table – 3
The generally used Events and their Classes

| ***Event type*** | ***Corresponding Class Type*** | |
|---|---|---|
| ***Tap*** | ***UIButton*** | ***UIInputText*** |
| ***Text Input*** | ***UICheckBox*** | ***UITextField*** |

| *Scroll/Checkbox* | *UIRadioButton* | *UIScroll* |
|---|---|---|
| *Validate* | *UITableView* | |

Step Definitions:
Given(/^my app is launched and it's on the login page $/)do
wait_for(180) {element_exists("button id:")}
##This row will wait for 180 seconds before loading the login page after installing the app on your mobile device or emulator.
When (/^I type invalid credentials$/) do
steps % {
Then I enter "" into input field number 1
## This row will enter text in the username field (If it is the first input or text field on the page.)}

The above code snippet provides the Ruby code for the Gherkin language steps that are written in the feature file.

## IV. PERFORMANCE ANALYSIS IMPLEMENTATION

The performance launch time of an application depends on various factors mainly the size of the app, the various components that are present in the app, and the language that is used. The performance launch time is very crucial in determining if the application can be launched in the app market. In order to make the app launch ready the performance is a very keen aspect and this has to be taken care of very carefully. The performance metric is very important when the app is tested redundantly several times and if the same consistency is observed. If the app has the same launch time it can be concluded that it is stable and can be released to the market.

### *A. ADB Commands:*

Android helps in providing a mechanism to collect and view the system debug output. The logs are collected in a circular fashion serially from various mobile applications and different portions of the systems as well. The logcat command can be used to view and filter the logs that are collected from the various applications. In order to see the log messages logcat from the ADB shell can be used.

adb logcat –c is used to flush the logs and exits after that.

### *B. UNIX Commands:*

grep can be used to log files and search for strings that have a similarity to a particular pattern list and once it finds a match for the given pattern in the log file it displays to the output log . Any number of output can be requested with the help of various different options. grep matches the input with the help of text and it forces no limit on the input string as long as there is enough memory available. grep also provides a newline, if the binal byte of some input file does not have a newline. Since newline is also a separator for the list of patterns, there is no way to match newline characters in a text. Here, metric that is needed is the am_activity_launch_time.

cmd1 = "grep am_activity_launch_time Example.txt | grep com.cl.di.example | cut -d ',' -f4-4 > Example.csv"

Once the grep command is executed, it will extract the values from the text file and store them in a CSV file. The am_launch_activity_time basically figures out the time that is needed for the application to lunch when it is pushed to the background n number of times. MSYS has a collection of GNU utilities. The utilities that are includes are bash, make, gawk and grep that help in building applications that depend on Unix tools It is proposed to supplement MinGW and the deficiencies of the cmd shell. The example that would be appropriate would be to build a library that makes use of autotools build system.

### *C. Calabash Automation:*

Since Calabash Automation Framework is deployed the launch time of the application is found out by launching the application n number of times. To launch the application several times, one way is to manually do it and the next possible way is to deploy some Automation Framework. This will reduce the errors that can creep up during manual execution. Calabash Automation feature file will have the Gherkin language code.

Below is the feature file:
@Example
Feature: Execute 20 times
Scenario: Execute the execution the app
Given I see the text "Example App is Launched"
When I wait for 1 seconds

The feature file has to be supported by a corresponding Step Definition.

In order to execute the block for n number of times certain changes have to be done in the hooks file of the Calabash. Cucumber provides a number of hooks that allows the tester to execute blocks in the test cycle of Cucumber at different points of time. They are placed in `support/env.rb` file. They can be placed in a file called `support/hooks.rb`. The hooks cab=n be defined anywhere and where each hook is implemented. They can be run for any scenario/step, but the hooks are tagged for

more fine tuning and control from the tester's perspective. When the associated event occurs the defined hooks are deployed at that time.
Example:

Before do
Execute a random number of steps before each scenario
End.
This will run before each step of the scenario.

After do
Execute a random number of steps before each scenario
End.
The after scenario is implemented after the last step. The after scenario is implemented eve if the scenario fails, undefined, pending or skipped.
They will be executed in the reverse order in which they are written. The following code snippet will execute the called Feature file as mentioned.

```
Around('@Example) do |scenario,block|
20.times {block.call}
end
```

***D. Matplotlib:***
matpltlib provides a python 2D plotting library that gives extremely high quality of figures in any hardcopy format that is desired by the user . They can be deployed in various environments and can be deployed across various platforms. matplotlib can be deployed in python scripts or in ipython shell

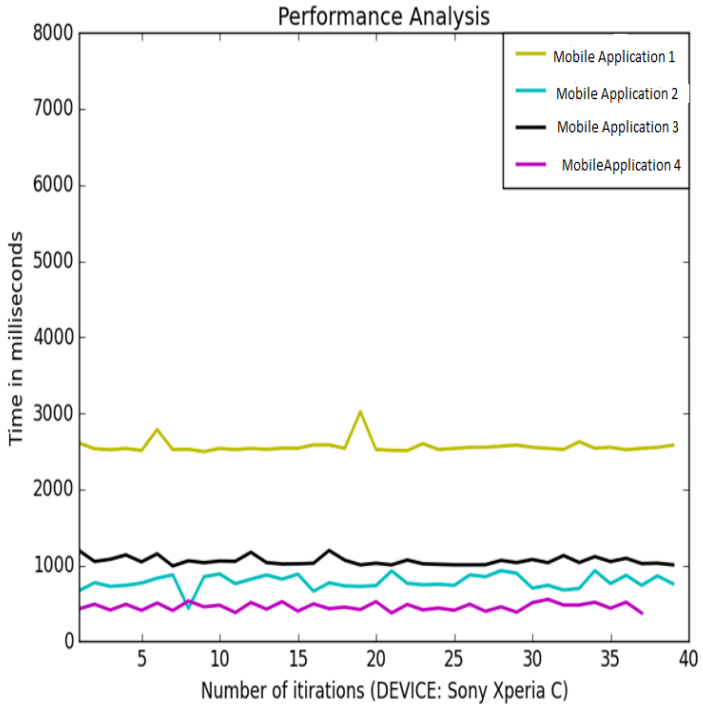

Fig. 3: The Graph that is obtained from the CSV values using matplotlib

Web application servers. matplotlib simplifies things. It can generate plots, histograms, power spectra, bar charts, error charts, and scatterplots, with just a few lines of code. The pyplot interface provides MATLAB similar interface along with IPython. The control to line styles, font properties, and axes properties is with the help of object oriented interface or with the help of functions that are like MATLAB. The system specification used in Figure 3 is as Quad-core 1.2 GHz Cortex-A7.

***E. Enthought Canopy:***
Enthought Canopy gives a complete analysis features. It provides installation that is simple. The installation includes scientific Python and analytic packages, thereby producing a strong platform that can be used to build and envisage on. This can be used to plot the graph from the CSV file.

Figure 3 shows the graph of the launch time of different applications that were taken up for the study. The launch time has to well within 5000 milli-seconds on order to get accepted by the app store.

### *F. NumPy:*

NumPy is very crucial package for computations that are scientific. It is combined along with Python. The features that are present include:

- Robust N-dimensional object array.
- Functions that help in broadcasting
- C/C++ and FORTRAN code can be integrated with the built in tools.
- It also has linear algebra, Fourier transform, and random number capabilities that are very useful.

It has many uses scientifically. NumPy is also as a container for multi-dimensional data and it is very efficient. NumPy allows the seamless integration with a huge variety of databases and various data-types can be defined.

## V. CONCLUSION

Mobile app testing is very necessary in agile environments but at the same time it has a lot of tradeoffs as well. The mobile testing can be done with the help of Test Automation Frameworks or manually. The Automation Frameworks should be chosen carefully before they are deployed for any mobile application. Calabash Framework provides an effective tool that can support both Android and iOS applications. Since this is of utmost importance it is necessary to analyze and study all the tools that are available in the market.

The performance analysis is very crucial for mobile applications before they are released to the app store. The launch time is found with the help of ADB commands that are present only for Android applications and similar commands or component has to be implemented for iOS applications as well. Since the iOS applications do not have the ADB component it is necessary to examine the log in the XCODE and then do an analysis on the same and find the performance metric.

## REFERENCES

[1] Keynote, White Paper on "Testing Strategies and Tactics for mobile Applications"
[2] Tom Yeh, Tsung-Hsiang Chang, Robert C. Miller, "Sikuli: Using GUI Screenshots for Search and Automation", unpublished
[3] Tsung-Hsiang Chang, Tom Yeh, Robert C. Miller, "GUI Testing Using Computer Vision", unpublished
[4] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research," Hanover, NH, USA, Tech. Rep., 2000.
[5] Android Platform", Institute of Mobile and Distributed Systems, University of Applied Sciences Northwestern, Switzerland: pp.1-2.[online]. Available: http://www.fhnw.ch/technik/imvs/publikationen/vortraege/androidtesting
[6] "Software testing of mobile applications: Challenges and future research directions," http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6228987
[7] Automated QA Corporation, "Getting Started with TestComplete 8.0".
[8] Confiair Inc, "RealMobileTM - Mobile Test Automation", Available at: http://www.confiair.com
[9] Nagowah, Leckraj, and Gayeree Sowamber. "A novel approach of automation testing on mobile devices", 2012 International Conference on Computer & Information Science (ICCIS), 2012.
[10] Hyungkeun Song, Seokmoon Ryoo, Jin Hyung Kim. " An Integrated Test Automation Framework for Testing on Heterogeneous Mobile Platforms", 2011 First ACIS International Symposium on Software and Network Engineering, 2011.