Framework Development on Automation Testing Using Continuous Integration

Priya B[1], Roopa H[2]

[1]M.Tech. Scholar, [2]Assistant Professor
Bangalore Institute of Technology

Abstract: **In order to improve the efficiency of software development and handling, make sure the quality of application software, with the present mainstream of continuous integration tool Jenkins. Android and IOS operating system of mobile terminal are supported by the framework. And framework can automatically complete the entire process of deployment, testing and log analysis. Not only reduces the testers effort, but also greatly improves the efficiency of development and testing, and establish a solid foundation for program supportable development. In this paper, focus on the implementation of continuous integration with Jenkins tool and how Jenkins can save developers and testers essential work time by automating the entire process.**

Keywords**: Continuous Integration, Jenkins, Automation Framework testing.**

Introduction and Motivation

A big question is why, how and when to do automated test execution. There are various good reasons, during setting up the test environment the time which is spent is gained again during the permanent software programmed into a read-only memory. Every added new test cases are automatically executed and can make stronger the confidence in the code maturity. It can also find immediately regressions.

To do automated test execution another good reason is that all steps compulsory to execute specific tests for setup of the environment at least written down in some scripts (ride). To execute that preparation steps which are mandatory tests successfully the risk is quite high are with the engineer who literally does the tests. Every step's has to be implemented, when automating the execution of tests, otherwise the execution of test cases will fail.

An important advantage which is obtained from test automation is reproducibility. Environment of an automated test allows to rerun tests with exact same conditions many times. To debugging of "random" failures this can be very useful.

The main goal is to increase the efficiency of software development and distribution, also guarantee the application software quality and improve the automated test scripts reusability with help of "Continuous Integration Tool Jenkins".

Entire progress of test cases development, testing, deployment and analysis of logs can automatically complete by this framework. Automation framework using Jenkins is supports for both IOS and android mobile terminal operating systems.

Jenkins is continuous integration tool which helps in complete process of automation, it's reduces the tester and development engineer's works as well by checking the status of each and every step evolution of software's.

Failure of test cases are common because of some environmental issues, but this framework gives complete analysis of logs with complete alarm's, from this can find whether it is because of software issue or environment issue, if it is software issue, based on issue priorities, can provide quality to that particular build.

Advantages of proposed system are as follows:

- It is user-friendly and open source continuous integration tool, installation is easy and additional installations or components are does not require.
- Jenkins is free of cost, according to the continuous integration requirements and continuous delivery Jenkins can be configured.
- Jenkins is platform independent, can support for all platforms and different operating systems, like IOS, android, windows and also linux.
- Modification and extension of test cases by using python scripts is very easy with Jenkins. It generates test log reports by instantly deploys codes.
- Jenkins flexible and allows building, automating across various platforms and deploying by extensive plugins pool, hence it is called "Rich Plugin Ecosystem".
- Most important advantage is any issues can be detected easily and resolved in almost right path, which helps to keep software in a safely state at any time where it can be released.
- Jenkins can trigger jobs at any number of times, it has no limitations for the same. This saves both money and time over the lifespan of a project by automating most of the integration works.

Implementation for Continuous Integration with automation tool Jenkins

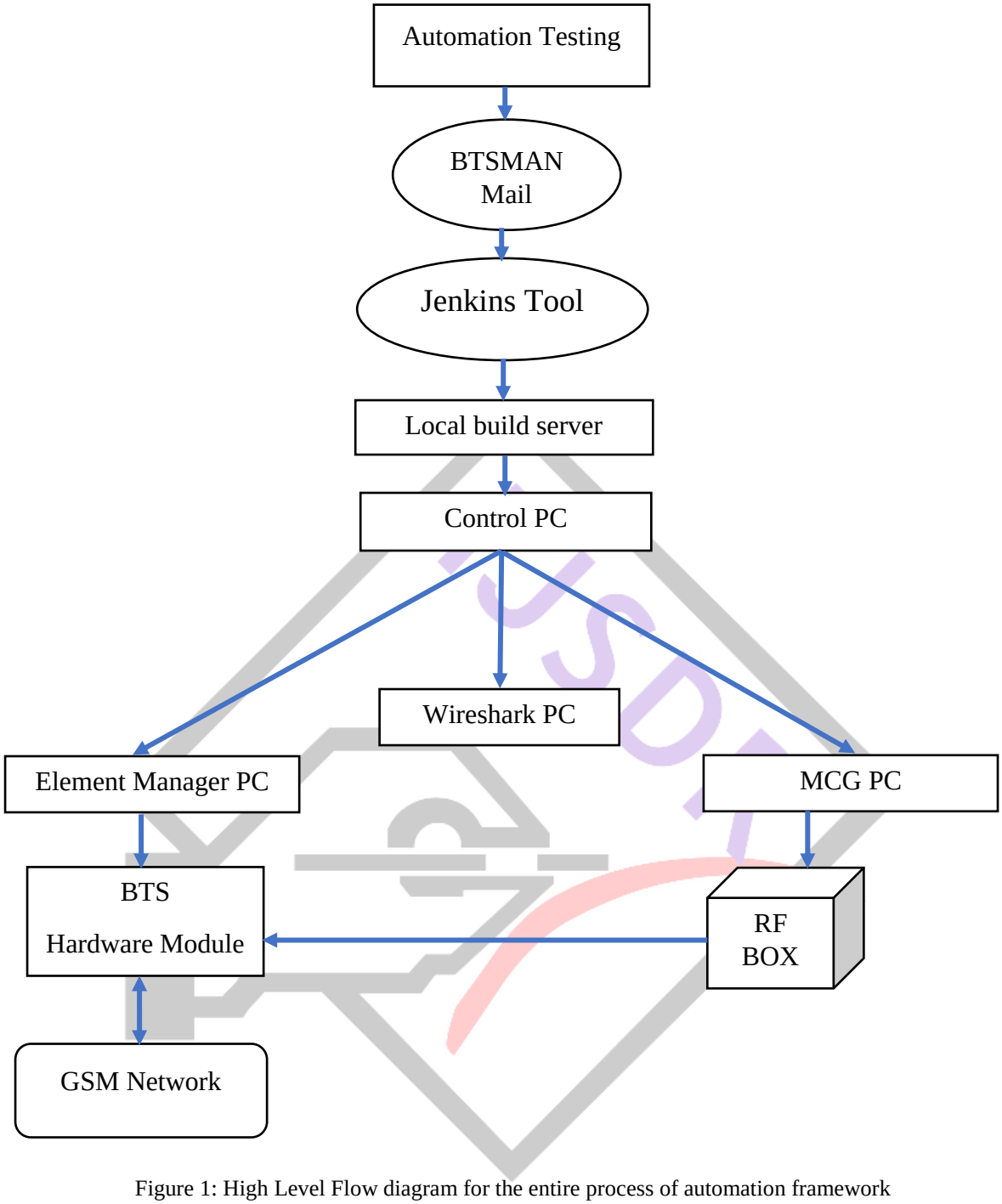

Figure 1: High Level Flow diagram for the entire process of automation framework

This module reads the configurable parameters from "Config_Parameter.txt". Downloads the BTS Build & EM Build, for all setups (EX, GF, FE). Downloads the latest BTS Build & EM Build from Local Build Server to the specified destination directory. Downloads the any specific BTS Build & EM Build as well.

Empties the destination directory (local build folder of EM PC), if any files present. Before downloading the builds, it also deletes the contents of the destination directory, if any earlier copied builds present. After downloading BTS Build (.zip), it un-zips & deletes the actual copied zip file it does same for EM build after copying EM Build (.zip), it unzips and deletes the actual copied zip file. Next step is installing the latest build from local build server to Element manager via control Pc.

Step 1: In Element manager as well as in Control Pc importing all required libraries which supports or in build in python. Import OS (Operating system element (Linux/windows)), re (regular expressions), Shutil (Functionality of Coping, moving file), Zip file (To zip/unzip file), Sys (System module), file comp (File size compare), FTP lib (To do FTP), opt parse (option parse) etc.

Step 2: The functional list is calling to do operations such as to get latest build, specific build, delete all builds, download files, connect ftp, disconnect ftp, unzip file, and also remove Tar zip file. Each and every called functions.

Step 3: Further it checks for specific BTS/EM builds considered as provided by user with source and destination directory, hostname, username, password, reg Ex, label, variant, Latest Build. If BTS/EM build is not available in mentioned path, then it will print "Invalid File Path", build not found.

Step 4: In this step deletes if any folders or tar files or any other text files are present in the Local Build Directory (Destination Directory). If no folders are present, in such case it prints "Empty", if any folders present it prints "There are some folders in the directory", and "Deleting".

Step 5: In this step Connects to Build FTP Server by providing hostname, username, password, source directory, destination directory, when it gets connection to FTP it prints "connected successfully", if not it prints "FTP connection failed" and after required build installation is done it disconnects from FTP with mentioning ftp, destination directory operation will take place.

Step 6: Once the destination directory empties, then it starts downloads builds (.zip) from Local Build Server (Source) to the destination directory (Destination - EM PC). When build found in directory, it prints downloading a file, with FTP connection after download completes, give result with file successfully downloaded. If build not found then its fails, it prints couldn't found latest build in directory entire process should start from beginning.

Step 7: Unzipping the earlier copied BTS.zip and EM.zip file carried out in this module. While calling this unzip file should mention output path of that particular build, after extraction completes it prints "successfully unzipped", if not it prints could not unzip the file and it fails due to size of the file may exceed the actual size.

Step 8: In this module it deletes the specified file such as target and zip file of BTS, EM build and it print deleted files, if no files found in that particular path then it prints file not found or could not delete the files in given path.

Step 9: In the option parse module can mention all mandatory and optional arguments. In python have option parse is more flexible library Arguments. All required arguments for installation.

RESULTS

Main goal of automation testing is to reduce tester effort from doing manual testing. Continuous Integration with tool Jenkins is to automate the entire process of compiling, testing, generating a report with all analysis. Experiments eight scenarios were performed by doing both manual test and automation test. To complete that eight scenarios by traditional manual test it takes 3 days by using 4 persons. With automation testing by continuous integration tool Jenkins takes 11 hours with one machine. The automation testing cycle is not a small process, but also more efficient than manual testing. Hence automated testing framework efficiency is improved by 92%. And also, scenario tested for testing the proposed system is mentioned in detail. Below Table 1 shows the comparison of test results.

Table 1: Results Comparison of Test

Category	Automation testing Framework	Manual testing
Number of test scenarios	Eight (8)	Eight (8)
Test cycle	1 day	3 days
Human and machine consumption	4 persons	1 server
Number of test machines	4 android machines and 4 IOS machines	4 android machines and 4 IOS machines
Effective test time	1*11h	3*9h

CONCLUSION

Integration is essential component of software development for any software industry. Continuous execution of these integration tasks is of at-most important for Automation testing. Jenkins provides a better solution for Continuous Integration i.e, satisfaction. The traditional manual testing methods have many human errors, speed of the operation is also slow and records accuracy is low. Hence manual testing is not flexible enough to deal with complex processes and the script reusability is very poor

Due to success, will now adapt this technique for other projects within next two sprints and will have closer look on different automation test methods to even further improve software quality.

REFERENCES

[1]. Ritu Patider, Anubha Sharma, Rupali Dave. "Survey on Manual and Automation Testing Strategies and Tools for a Software Application", Department of computer science, 2017.

[2]. Fachrul Pralienka Bani Muhamad, Riyanarto Sarno, Adhatus Solichah Ahmadiyah, Siti Rochimah, "Visual GUI Testing in Continuous Integration Environment", IEEE, Department of Informatics Engineering, 2016.

[3]. Muhammad Abid Jamil, Muhammad Arif, Normi Sham Awag Abubakar, Akhlaq Ahmad, "Software Testing Techniques: A Literature Review", Department of computer and information system, 2016.

[4]. Fan Ming, Zude Zhou, Zhengying Li, "The design and implementation of the cross-platform Mobile Automated Testing Framework", Computer Science and Network Technology, 2016.

[5]. Nikita Seth, Rishi Khare, "ACI (Automated Continuous Integration) using Jenkins: Key for Successful Embedded Software Development", IEEE, Computer Science and Engineering Department 2015.

[6]. Eun Ha Kim, Jong Chae Na and Seok Moon Ryoo, "Test Automation Framework for Implementation Continuous Integration", 2009.