Automated Testing Tool for Linux based Embedded Software

Deepika.H
Dept. Of Software Engineering
Bharathidhasan Institute of Technology
Trichy, Tamil Nadu, India
deepika.hari03@gmail.com

Padma.M
Dept. Of Information Technology
Bharathidhasan Institute of Technology
Trichy, Tamil Nadu, India
padmamayan@gmail.com

Manikandan.S
Senior Engineer
Zilogic Systems
Chennai, Tamil Nadu, India
maniksankar86@gmail.com

***Abstract*— Linux is an open source environment and most of the enterprises pick out Linux for cost efficiency and easy to get the source code. Linux based software are suitable for multiprocessor environment. But for testing and debugging in Linux environment is the big concern. Significance of automation in testing a Linux based in vehicle infotainment is to validate the features of the software in efficient manner. Python is an effective interpreted, object-oriented, high-level programming language with dynamic semantics which makes rapid application development. Python scripts are the modest automation testing tool to the software. It is easy to create python scripts because it is easy to learn syntax underscores readability and therefore reduces the cost of program maintenance. Adding the test script using python required its coding knowledge. Minimizing the amount of time for adding the test scripts is difficult. Adding the test cases in the excel sheet is easy and increase the probability of capturing testing cases based on requirement. The major objective of the project is to design and develop an automation tool which gives the test report of the software by creating test scripts using excel sheet. GUI based tool need to be developed for easy of automation and user friendly in implementing test cases. This work distillates on submitting an algorithm for automation testing system. (*Abstract*)**

Keywords—Automation testing;In-Vehicle Infotainment; Robot Framework; Python scripting; test cases

I. INTRODUCTION

In this rapidly developing software environment, it is essential to test software. Testing accomplishes quality control and maintenance. Every organization has their own testing team for testing the software before it will be released to customer. Testing the software can be achieved in various level or various stages of the software development life cycle. Usually the following stages of testing can be performed while the software development process.1) Unit testing – Code level 2) Integration testing – Design level 3) Feature testing – Feature requirement level 4) Verification testing – Software Release level 5) Validation testing – Customer Release level 6) Field testing – End customer release level.

In the current scenario there is need for accelerated software development. Automated testing is the process through which detailed and rapid testing can be done. This quality of automated testing has made it an essential part of software development. Automated testing is also preferred from software engineering point of view because a well-organized approach to automate functional software testing is needed to achieve significant cycle-time and quality improvements. Integrating automated testing into the software development program gains three benefits: 1) Reduced cycle time by decreasing product and integration test time. 2) Improved quality 3) Standardized testing and reproducible results. Software testing is an integral part of software development process. It analyzes a system or a component by providing defined inputs and comparing them with the desired outputs to check the discrepancies between the desired and actual outputs and correct them. Basically it can be divided into two categories. 1) Manual software testing 2) Automated software testing.

A. *Manual Software Testing*

Manual software testing is as the name suggests done manually that is it requires human input, analysis and evaluation.

B. *Automated Software Testing*

Automated software testing is the automated version of manual software testing. Using automated software testing helps in avoiding the human made error due to tiredness of doing the process repeatedly. The automated test program will keep the results of the tests accurately. The results can be automatically fed into a database and can be used to provide useful statistics of the software development process.

C. *Some features of automation testing*

Automation of test cases has some features. 1) It can be run all day and night in unattended mode 2) System continues running even if a test case fails 3) Keep the automated system up and running at all costs 4) Single point maintenance 5) It is easy to update and reuse the modules 6) Easy to understand the scripts 7) Scripts and modules to the system for new features can be added rapidly 8) Tests coverage provide by automated test suites can be tracked 9) Same architecture can be used for Web or GUI based application testing.

II. AUTOMATION TESTING PROCESS

Sequence of actions need to be done for automation of test cases. A) Requirements for Automated Software Testing

B) Process of Automated Software Testing C) Selection of Testing Tool D) Development and Testing of Scripts.

A. Requirements for Automated Software Testing

There are certain requirements for a testing to be automated. They may be financial limitations, limited man power, etc. There a few basic cases to understand the requirements for test cases to be automated. 1) By checking whether the test sequence of actions can be defined 2) Repetition of sequence is needed 3) Possibility of test cases can be automated.

B. Process of Automated Software Testing

Automation of software testing is similar to a software development process .It goes through the same life cycle as in the development of software product .The important think that has to be taken care of conflict between writing the scripts by developer or testing team member. It is advisable that the effort should be a collaborated effort between tester and the developer. Many of organizations followed the same. The automation process goes through a lot of effort taking collaborated work because a lot of emphasis is given for the time and financial constraint. In Fig (1) the lifecycle of the process of automation software testing has been explained in detail.

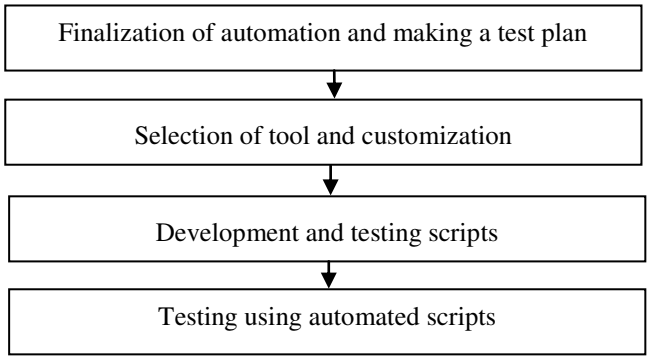

Fig. 1. Test Automation Project Lifecycle. *(1)*

1) Finalization of automation and making a test plan

During this phase the test cases can be automated are listed. The test plan has been created. It takes effort similar as development to automate a test case. And hence the test effort has been split up to testing team.

2) Selection of Testing Tool

Test tool selection is a very important part of test automation which requires the study of the scope of testing and the test plan. It is also needed to know whether the test tool meets the test suite requirements for the particular product and version. The important factors that also come into picture are reusability, reliability and cost. This can be done to see if they can get the maximum benefit out the product being made, bought or customized 1) The test tool should support 2) Scripting interface 3) Facility to give valid and invalid input 4) Result comparison 5) To give the verdict

3) Development and Testing of Scripts

Development process includes both development and testing scripts. This mainly depend on two factors 1) The skill of the person who is writing the scripts 2)On the flexibility of the test tool for developing for all valid and invalid scenarios.

It is desirable to develop scripts in a modular way .This approach towards writing scripts helps in reusability of the module in the different scripts. This approach will use less time and will be enable the use of the scripts across different releases. The scripts should always be tested before being put to use .They can be tested on an already released version of the software. This is done to avoid problems which may arise during the testing process.

4) Testing Using Automated Testing Scripts

This is the face where the actual testing is done. It solely depends on the test plan when a test is automated which one are automated and in which part of the development cycle these tests are done. The scripts that have been made are developed in such manner that if there is another version release of the software product then these scripts could be used for testing purpose for that version too. These thinks have to be kept in mind by the test manager when he plane for test automation.

C. Tools for Automated Software Testing

For many test managers, the decision of which testing tools to use can cause confusion. The first decision to make is which category of tool to use—one that tests specific units of code before the application is fully combined, one that tests how well the code is working as envisioned, or one that tests how well the application performs under stress. And once that decision is made, the team must wade through a variety of choices in each category to determine which tool best meets its needs. Evaluating our needs will narrow down our short list very rapidly. There are many options in market to choose. Depending on what sort of test it will be used for the choice may become easy. The following general categories categorize the tests and tools which makes it easy to choose. 1) Developer oriented tools 2) Functional testing tools 3) Load testing tools 4) Performance monitoring and maintenance tools

1) Developer Oriented Tools

Tools in this category, also called component testing tools or unit testing tools, test individual software components or groups of related components, helping isolate and rectify problems as early in the development process as possible. Areas of testing include memory analysis, function analysis. The important tools in this area are

Examples: DevPartner Studio from Compuware Corp., PurifyPlus from IBM Rational, Optimizeit Enterprise Suite from Borland Software Corp.

2) Functional Testing Tools

Tools in this class help verify that applications will work as expected. It allows developers to record an existing application and modify scripts to meet changes in an upcoming release. They also provide for regression testing on

the new release using the test scripts developers have captured up until that point. The important tools in this area are

Examples: WinRunner from Mercury interactive Corp., Astra QuickTest Mercury interactive Corp., SilkTest from Segue Software Inc., RationalSuite TestStudio from IBM Rational.

3) Load Testing Tools

This category, also called performance or stress testing tools, tests what happens to the code as the application scales with multiple users in a simulated environment. The tools test, among other things, whether performance degrades as the load is increased. Once bottlenecks are found, these tools can determine the source of the problem and begin fixing it. The important tools in this area are

Examples: LoadRunner from MercuryInteractive. SilkPerformer from Segue Software Inc., Testperspective and Load test editor from Keynote Systems Inc., QACentre from Compuware.

4) Performance Monitoring and Maintanence Tools

All the tools in this category are used after the application is already in production by examining how the application is working in a real-world environment. In essence, these tools monitor the production environment to ensure that all requirements and defined thresholds are continually being met. The important tools in this area are 1) Vantage from Compuware. 2) VTuneAnalyzer from Intel Corp. 3) Optimizeit Enterprise Suite from Borland Software Corp. 4) OneSight from Empirix

III. EXISTING SYSTEM

In existing system, there is a methodology has been followed for automation testing tool for testing in vehicle Infotainment. In Luxury vehicles, the IVI is the most important system which interacts with all other modules and updates the driver in user friendly manner. The basic features of the IVI are Tuner AM/FM, USB connectivity audio playback, IPOD connectivity, Bluetooth connectivity, navigation, Car info, etc.

The features may be added in future. Due to faster development of technology the features of the IVI may be increased. And for the low end vehicles cost effective system also can be developed with small amount of features. The hardware and software specification may be varied due to various factors such as quality, cost, improvement, etc. Here in this environment testing the functionality and features of the IVI is most important to avoid customer and field issues. Hence the basic features of the system and the features which are impossible to test in manual are automated. The test cases will be written as scripts. Whenever features are added, the scripts for automation are included for automation testing. The tool allows testing the IVI automatically and generates the report. To avoid man work hours for scripting the test cases will be tabulated and the results will be updated in the excel sheet, the tool takes the control of reading excel sheet and writing excel sheet also it covers which binary need to be tested. This proposal mainly concentrates on automation testing environment for a Linux based embedded software. Robot framework is the vital tool for automation testing of Linux based system. This tool interfaces the binary, runs the test cases automatically and generates the results. But adding the test cases in the python library needed python knowledge and time taking activity of adding the test cases. In a crisp the scope has been listed below: 1) Automation of test cases has been done using robot framework 2) Test scripts are written in python 3) Variables are interfaced in Robot framework.

Test scripts and robot framework are executed by providing commands. The existing automation system contains the python library, robot framework. Robot framework contains auto scheduling of calling python library. Once the process has been started, the log, output and result of the test cases will be logged into the separate files. The test cases are added in the python library which interfaces the actual test binary of the embedded system.

A. Development of Test Scripts

Test cases have been developed in python library interface in which the python script interfaces the actual software. And the scheduling and logging of the test cases have been included in Robot framework.

B. Running the test cases

Running the test cases has been scheduled automatically by robot framework. Once the automation process has been triggered the robot framework calls the python functions with variables.

C. Selecting the failed test cases

After completion of the automation process the failed test cases has been selected for re-running the test cases. It will be selected based on the preferences provided.

D. Disadvantages of Existing System

The main disadvantage of existing system is time taking activity of adding the test cases in which python knowledge is required for adding the test cases and if the features are increased, python library and interfaces will also increase.

IV. PROPOSED SYSTEM

Proposed system is mainly concentrates on evading the main disadvantage of the actual existing system. Adding the test cases in the python library requires python knowledge. It is a time taking process to guide a test engineer to add the test cases in python readable format. And the job for the test does not know python scripts instead the test engineer should add the test case in easier manner. The proposed system which has a GUI which interfaces python library and excel sheet read write module as well. Test engineer can add the test cases in the excel sheet. The robot framework will trigger the execution of excel sheet reading. Once the excel sheet is decoded, the testing progress will be logged into a file. In manual test mode, user can add the excel sheet and test binary using GUI. Once the test cases and actual binary has been loaded in the module, by pressing the test button, the progress can be started. The test case log, result and output will be stored in separate XML files.

While running the test cases the progress can be seen in the GUI itself. Once the test progress is completed, the test results will be shown in the popup.

A. *Development of Test Scripts*

The test scripts have been listed in excel sheet based database. The cases are listed based on the requirements provided.

B. *Running the test cases*

The GUI based tool / Robot framework triggers the process of running the test case.

C. *Selecting the failed test cases*

Selection process has been defined based on failure and index oriented.

D. *Advantages of Proposed System*

The main advantage is adding the test cases are easy because capturing the requirement and converting it into excel sheet is tranquil process. It can be operated in manual mode and automated mode. There is no need of training the python library to the test engineer and readability of test cases which covers unit, integration and feature testing as well.

V. ALGORITHM

Automation of test cases in the proposed algorithm is database oriented listing. The unique python library has been developed for reading and writing the excel sheet, interfacing with GUI and robot framework. And it also interfaces the actual binary to be tested.

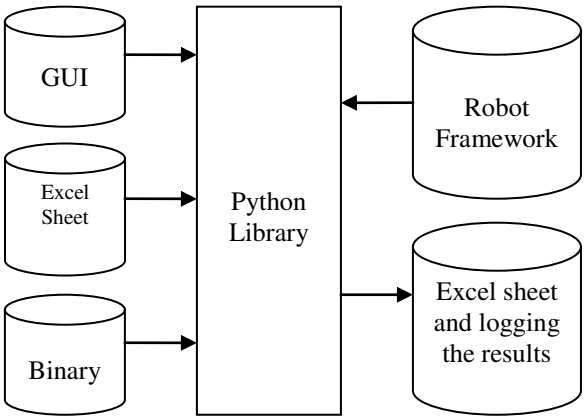

Fig. 2. Block diagram of proposed algorithm

- Initialize main variables and call all other initialization functions.
- Open the main window and wait for the user input
- Open dialog for browse if browse button is pressed
- Select the path and send to python module if the path and file verification gives success
- Enable the load, run, stop, pause buttons if validation of excel sheet and binary are success
- If load button is pressed, call excel library to load the excel contents into a list and send it to python library
- If run button is pressed, call python library to execute the test case and store the results in the same excel sheet.
- If Stop button is pressed, call python library to stop executing the test cases and reinitialize the main window with the results
- If pause button is pressed, call python library to stop the execution and wait for user input

VI. PERFORMANCE MEASURES

The performance comparison between the existing and proposed can be measured by the effort spent. The amount of work, man count and working hours are same.

$$\frac{M_1 H_1 D_1}{W_1} = \frac{M_2 H_2 D_2}{W_2} \quad (1)$$

Where M_1, M_2 are the man count, H_1, H_2 are the Working Hours per day and D_1, D_2 are the days for the work W_1, W_2 respectively.

$$Efficiency = \frac{Standard\ Work\ Hours}{Amount\ of\ time\ worked} X\ 100 \quad (2)$$

From the above equations, the performance of the proposed software has been measured by means of efficiency.

A. *Figures and Tables*

1) Performance measures: Table measures are taken in the real time scenario. (i.e) work based data : taken a single man for assigning the work and the performance has been measured. Work: Adding the test cases in the script is the work scenario and work hours is considered as 8hrs/day

TABLE I. PERFORMANCE MEASURES – MANUAL SCRIPTING

Work Id	Days quoted	Manual Efforts of Python scripting			
		No of test cases	*Scripting(Each test cases) in Hours*	*Executing (Each test cases) in Hours*	*Total In Hours*
1	40	160	0.4	0.2	96
2	160	480	0.3	0.2	240
3	64	192	0.3	0.1	76.8
4	80	240	0.3	0.2	120
5	24	96	0.2	0.2	38.4

TABLE II. PERFORMANCE MEASURES – AUTOMATED SCRIPTING (PROPOSED ALGORITHM)

Work Id	Hours Quoted	Automation Efforts of Python scripting			
		No of test cases	*Scripting(Each test cases) in Hours*	*Executing (Each test cases) in Hours*	*Total In Hours*
1	40	160	0.2	0.1	48
2	160	480	0.2	0.1	144
3	64	192	0.2	0.1	57.6
4	80	240	0.2	0.1	72
5	24	96	0.2	0.1	28.8

Fig. 3. Performance interms of Efficiency

From the above chart it is clear that with the automation test tool the efficiency of the work got increased. In the manual scripting (Highlighted Blue) the efficiency got low when the work load got increased whereas in the highlighted red (Automated scripting) has the merely constant efficiency with higher rate i.e. the efficiency always maintained near to 100% hence the work will be completed within the quoted hours.

CONCLUSION

Thus the proposed automation testing tool minimizes the effort of adding the test scripts and it saves the time of learning the python scripts. As the major advantage of the software, it is very easy to add test cases which can be matched with the actual software requirements.

REFERENCES

[1] https://github.com/robotframework/robotframework
[2] https://wiki.python.org/moin/PyQt
[3] https://www.python.org/about/
[4] http://www.origsoft.com/Issues3-buscase-4save.htm
[5] http://www.sqa-test.com/ATS_Tips.html
[6] http://www.sqa-test.com/articles.html
[7] http://www.origsoft.com/Issues2-comptest-1cov.htm
[8] http://www.cssttechnologies.com/Test%20Requirements%20for%20automated%20Testing.htm
[9] http://www.io.com/~wazmo/succpap.htm
[10] http://www.hssworld.com/whitepapers/whitepaper_pdf/test_automation.pdf
[11] http://www.stsc.hill.af.mil/crosstalk/2002/05/rice.pdf
[12] http://www.testing.com/writings/automate.pdf
[13] http://www.automated-testing.com/PATfinal.htm
[14] http://www.stickyminds.com/sitewide.asp?ObjectId=3583&Function=DETAILBROWSE&ObjectTpe=ART\
[15] http://www.mactech.com/articles/mactech/Vol.13/13.10/SoftwareTestAutomation
[16] http://www.jasmin-infotech.com/domains/testing/test-automation/
[17] http://www.zilogic.com/services.html
[18] Patrice Godefroid, Peli de Halleux, Aditya V. Nori, Sriram K. Rajamani,Wolfram Schulte, and Nikolai Tillmann, *Microsoft Research* Michael Y. Levin, *Microsoft Center for Software Excellence*, **"Automating software testing using program analysis",** IEEE Software, Vol. 25, No. 5, pp 30–37, September/October 2008.
[19] Reinhard knoll, rory yorke, **"Automated testing of simulation and embedded software"**, Copyright (C) 2015 by Denel Dynamics. Published and used by INCOSE SA with permission
[20] Lipka Bose, Sanjeev Thakur, **"GRAFT: Generic & Reusable Automation Framework for Agile Testing"**, Confluence the Next Generation Information Technology Summit (Confluence), 2014 5th International Conference