Design of CRT setup and Automation of Testcases using Robot framework for Femto access point.

Anusha N , Mtech Student , Bangalore Institute of Technology, INDIA , anu2341994@gmail.com

Dr. R Nagaraja , Professor and PG Co-ordinator, ISE, Bangalore Institue of Technology, INDIA , profrnagaraja@gmail.com

Abstract—Manual testing is a time consuming process to test a Femto Access Point which is used to extend radio coverage. In addition, it is error-prone in few circumstances, so automation is highly desirable. Robot Framework is simple to understand and easily extensible tool which is keyword driven testing approach. Robot Framework is an open source platform which is easy to use tabular syntax that enables creating test cases in uniform way. Ability to create high-level keywords from existing keyword ensures easy reusability and extensibility. Simple library API, for creating test libraries in python, is available, while command line interface and XML based output files are generated that ease integration into existing system. All this features ensure that Robot Framework can be quickly used for test automation of Femtocell Product. Jenkins retrieves and updates the build and run many test cases simultaneously on different femto setups. Therefore, continuous integration can be achieved easily.

Keywords—Backhaul Simulator(BHSIM), Continuous Regression testing(CRT), Evolved Packet Core, Femtocell , Robot Framework , Test Automation

I. INTRODUCTION

Competition in the Information Technology field ends up with higher and challenging so that the organizations and new companies need to make progress towards new thoughts and strategies to endure. Subsequently, the executives and creating groups need to deliver quality items that meet custom needs. Building and keeping up quality programming is a testing undertaking since clients regularly change their prerequisites and tasks get bigger and increasingly muddled. Consequently, software testing is a necessary phase of current activities to guarantee the high quality product. When a femtocell needs to be deployed after all the procedure, Before the deployment of a femtocell, it has to undergo proper testing. All the possible scenarios has to be tested on each and every femtocell in order to give better product performance[1]. In order to execute all possible scenarios manually on all the femtocells is a risky, time consuming and tired job for an individual in order to maintain quality testing[2]. Hence Automation testing comes into picture. Automation testing , which is reliable and gives accurate result needs to be adopted for this method.

Femto cells are completely featured, short variety mobile telephone base stations used to supplement cell telephone offerings from large macrocell towers. these range from very compact residential femtocells to large device used internal industrial places of work or outdoor public spaces. They provide brilliant mobile phone coverage and information speeds at home, inside the workplace and public areas for each voice and data. Femto cells were evolved for 3G and additionally the newer 4G/LTE radio technology.

II. FEMTOCELL IN REAL LIFE NETWORKING

a) Definition of Femto Cell

Femtocells, also called 'home base stations', are small low energy mobile network get right of entry to factors that connect fashionable 3GPP gadgets to a cellular operator's community the usage of residential digital Subscriber Line (DSL), cable broadband connections, optical fibres or wi-fi last-mile technologies. They enable customers to get right of entry to voice and broadband offerings over the already present broadband net connection. As in keeping with UMTS structure, Femtocells are equal to UTRAN (RNC + Node B) blended in a container. A single residential femtocell enables typically at maximum four to 8 simultaneous connections in any indoor surroundings, allowing prison clients to connect to the femtocell to utilize a couple of services inclusive of voice, information or actual time multimedia streaming and many others. The gadgets are incorporated into small plastic computing device or wall mount cases and are established to the customers' premises with the aid of manner of the clients themselves. they may be plug and play devices.

b) Advantages of Femtocell

Femtocells have a number of advantages because of which they are becoming more and more popular in the wireless world. Although there are certain issues to be handled but still the following major advantages can be listed:

DOI : 10.35291/2454-9150.2019.0013

a) Compensates for poor cellular coverage inside the homes or small enterprise

b) Provides plug and play feature thus providing the user with ease of installation and usage.

c) Compact structure, that is the Femtocell is actually the UTRAN in a box and offers all services through existing broadband internet connection

d) Voice and data calls takes place through the secure IP-Sec Tunnel thus offering good quality of encryption and helps in securing the data

e) Helps in offloading the macro traffic and cell phones automatically switch over to the Femtocell coverage when in range

f) Femtocells are compatible with all 3GPP handheld devices and help improve the data speed and coverage in the areas where there are slow speeds and poor coverage

g) Femtocells can not only work efficiently with the existing network infrastructure by extending the cellular macro network but also can serve as primary network if sufficient number of Femtocells are provided in the region thus reducing the number of macrocells reducing equipment cost and complexity.

c) *Deployment of Femtocell*

Femtocells are an opportunity way to deliver the blessings of constant-cellular convergence (FMC) structures. The distinction is that maximum FMC architectures require a brand new (twin-mode) handset which matches with existing unlicensed spectrum domestic/employer wi-fi get entry to factors, whilst a femtocell-based deployment will paintings with present handsets however requires set up of a brand new access factor that uses licensed spectrum. Many operators have launched femtocell carrier such as Vodafone, AT&T, sprint and Verizon.

The standard scenario for deployment of Femtocells in domestic surroundings is proven in the Figure 1. The Femtocell unit generates a private cell cellphone signal within the home and connects this to the operator's network the usage of preferred broadband DSL on Cable provider and commonly supports 2 to five cell phones in a residential placing. this can permit advanced coverage and ability for every user within their domestic. Femtocell is powered by using a Femto-get right of entry to-factor (FAP, additionally called home Node B (HNB) or HeNB in 3GPP literature) and offers operator furnished voice & information services as well as public net get entry to to cellular gadgets in the Femtocell coverage vicinity[9].

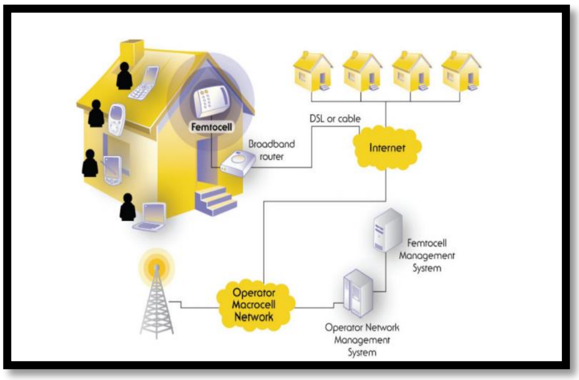

Femtocell
Broadband router
DSL or cable
Internet
Femtocell Management System
Operator Macrocell Network
Operator Network Management System

Figure 1 Typical Femtocell deployment in a residential scenario

III. SCOPE OF WORK

There are various fundamental advantages from test automation like Increases the product quality, reduce manual programming testing tasks and dispense with excess testing endeavors, produce progressively methodical repeatable programming tests, Reduction of repetitive work and create increasingly steady testing outcomes, Greater consistency. Execute significantly more programming tests and accomplish a better testing coverage in an exceptionally constrained timetable. Increments in the efficiency, etc.

Automation has gained popularity throughout the years and will acquire more popularity sooner rather than later; however manual testing is as yet utilized in numerous activities. A few procedures just can't be automated. Therefore, manual testing won't be vanished later rather it will walk one next to the other with Automated testing!

A. *Selection of Open Source Platform*

Robot Framework is chosen as platform for test automation. Robot Framework is flexible to use and can be used to test web applications, desktop application, mobile applications and RESTful and SOAP-based services. Its requires less technical skills than a programming language-based frameworks, and this platform can also be used by the team member with less programming experience[3]. On other hand, Robot Framework includes keyword-based design, those with a high degree of technical skill can write keywords in the preferred language of their choice to test low level functions, and even do unit and integration testing. A two plugins are configured such as Robot Framework Support and Intellibot[4].

B. *Test Automation*

The Robot Framework platform is chosen to automate Femtocell Product. It is open source hence can be chosen to test automation. Further for end-to-end integrity testing Jenkins is used. As new build arrives Jenkins is used to automate software development process, with continuous integration and facilitating continuous delivery

DOI : 10.35291/2454-9150.2019.0013

IV. PROPOSED WORK

An open source platform is chosen for automation of Femtocell. These Femtocell is used for extending radio coverage network in indoor areas. For testing purpose Robot Framework is used for test automation that should also include integration testing. Jenkins is used for continuous integration that provides end-to-end automation testing is continuous regression testing setup as shown in Figure 2. As latest build arrives Jenkins retrieves and update the build and run many test cases simultaneously for different femtocells.

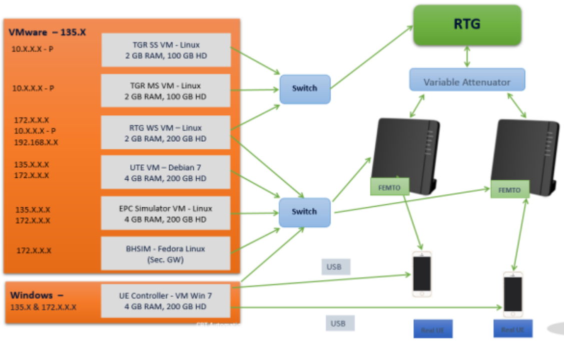


Figure 2. CRT Automation Setup

For the automation of given testcase, plain text format that includes keywords, library functions, variables and low level python scripts are used.In order to execute the automated testcases a complete hardware setup including Windows PC, Mobile devices, Femto access point, USB Cables, RF Cables, Antennas , a sim card, LAN Connection to the Femto and for windows PC is required[7]. To Trigger the testcases the software that must be installed and configured are Mobaxterm, Cygwin installed on UE PC, ADB driver, Wireshark tool, EPCSIM configured with femto, BHSIM Configured, Variable Attenuator is required[8].

A. *Robot Framework*

After careful analysis Robot Framework was found to satisfy all the requirements. Test automation is implemented in python that can be implemented on different major platforms. Therefore, multiplatform requirement is fulfilled.

Figure 3 represents the High level architecture of Robot Framework, this tool is sponsored by Nokia Siemens Networks and released under Apache 2.0 license, which meant to be used for free.

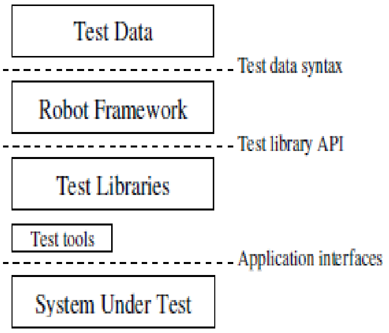


Figure 3. High level architecture of Robot Framework

Robot Framework handles the processing of test data, the controlling of test execution and the reporting of results for testing. The testing and interaction with the target under test is done by test libraries that are plugged into Robot Framework. Test libraries may internally use other test tools, but this is not visible to Robot Framework.

As shown in the figure, Robot Framework's role in a test framework is to take test data, process it into a format that is appropriate for the attached test libraries, invoke the test libraries with the appropriate arguments, receive the data about the test from the test libraries, and then report it[5]. The test libraries are responsible for interacting with the target under test

B. *Test Data*

Test cases are created in test case files. A test case file automatically creates a test suite containing the test cases in that file. A directory containing test case files forms a higher-level test suite. Such a test suite directory has suites created from test case files as its sub test suites. A test suite directory can also contain other test suite directories, and this hierarchical structure can be as deeply nested as needed. Test suite directories can have a special initialization file. In addition to this, there are test libraries containing the lowest-level keywords and resource files with variables and higher-level user keywords. Variable files to provide more flexible ways to create variables than resource files.

C. *TestCase Data Structure Format*

There are three types of Test Data supported file formats, they are HTML format, TSV format(Tab Separated format) and plain text format. Test Data structure contains four divisions

i. *Settings*

Each test case can have their own settings

a. Documentation

The [Documentation] setting allows you to set a free documentation for a test case. That text is shown in the command line output, as well as the resulting test logs and test reports.

b. Force Tags, Default Tags

- Using tags in Robot Framework is a simple, yet powerful mechanism for classifying test cases.

DOI : 10.35291/2454-9150.2019.0013

Tags are free text and they can be used at least for the following purposes:

- Tags are shown in test reports, logs and, of course, in the test data, so they provide metadata to test cases.
- Statistics about test cases (total, passed, failed are automatically collected based on tags).
- With tags, one can include or exclude test cases to be executed.
- With tags, one can specify which test cases are considered critical.

c. Setup, Teardown

A test setup is something that is executed before a test case, and a test teardown is executed after a test case. In Robot Framework setups and teardowns are just normal keywords with possible arguments.Test Teardown is executed also when a test case fails, so it can be used for clean-up activities that must be done regardless of the test case status.

ii. Variables

Variables are an integral feature of Robot Framework, and they can be used in most places in test data. Most commonly, they are used in arguments for keywords in test case tables and keyword tables, but also all settings allow variables in their values. A normal keyword name cannot be specified with a variable, but the BuiltIn keyword Run Keyword can be used to get the same effect.Robot Framework has its own variables that can be used as scalars, lists or dictionaries using syntax ${SCALAR}, @{LIST} and &{DICT}, respectively. In addition to this, environment variables can be used directly with syntax %{ENV_VAR}.Variables can be passed as arguments to any step of test case or keyword

iii. Testcase

Test case should be easy to understand.One test case should be testing one thing.*Things* can be small (part of single feature) or large (end-to-end).One has to select suitable abstraction level , Use abstraction level consistently (single level of abstraction principle).Only include information that is relevant for the test case. Generally has these phases:

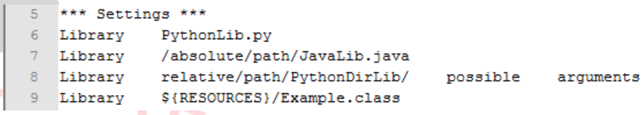
- Preconditions (optional, often in setup)
- Action (do something to the system)
- Verification (must have one!)
- Cleanup (optional, always in teardown to make sure it is executed)

iv. Keywords

Keyword section are used to create new higher-level keywords by combining existing keywords together. These keywords are called user keywords to differentiate them from lowest level library keywords that are implemented in test libraries. The syntax for creating user keywords is very close to the syntax for creating test cases, which makes it easy to learn. Keywords can be defined outside of the test suite and be imported as an external resources. This means you can collect generally useful keywords to one place and share them throughout the project Keyword names has to be unique across the test data file, resources & libraries.

D. Libraries

Test libraries contain those lowest-level keywords, often called library keywords, which actually interact with the system under test. All test cases always use keywords from some library, often through higher-level user keywords. Test libraries are normally imported using the *Library* setting in the Setting section. If the library is a file, the path to it must contain extension. For Python libraries the extension is naturally .py and for Java libraries it can either be .class or .java, but the class file must always be available. If Python library is implemented as a directory, the path to it must have a trailing forward slash (/). Figure 4 demonstrate these different usages.

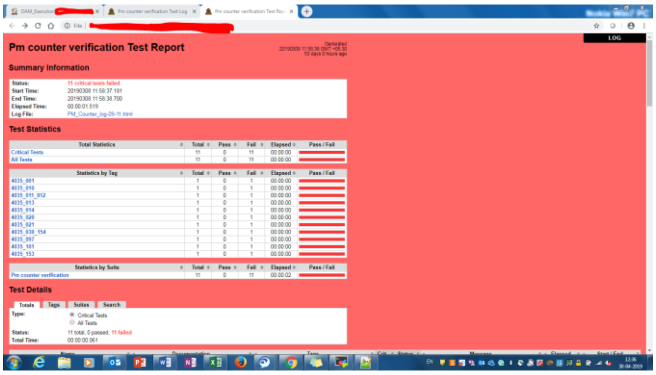


```
*** Settings ***
Library     PythonLib.py
Library     /absolute/path/JavaLib.java
Library     relative/path/PythonDirLib/     possible     arguments
Library     ${RESOURCES}/Example.class
```

Figure 4. Different Library Usages.

V. RESULTS AND DISCUSSION

Since Robot Framework is a command line tool, it is possible to execute test suite or some test cases directly from the command line so that execution is much easier to automate - for example from continuous integration server. The required test suite is run through command line, after executing the test suite HTML report is generated as shown on Figure 6, and background color represents whether the whole test suite is successfully executed or failed. If the critical test cases is passed successfully, the report will be marked as OK.The statistics that will be generated shows the number of test cases failed and specify if any of the test fails



DAM_Execution [redacted]
Pm counter verification Test Log
Pm counter verification Test Rep
File [redacted]
LOG

Pm counter verification Test Report

Generated
20190308 11:58:38 GMT +05:30
53 days 0 hours ago

Summary Information

Status:	11 critical tests failed
Start Time:	20190308 11:58:37.181
End Time:	20190308 11:58:38.700
Elapsed Time:	00:00:01.519
Log File:	PM_Counter_log-09-11.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	11	0	11	00:00:00	🟥
All Tests	11	0	11	00:00:00	🟥

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
4035_001	1	0	1	00:00:00	🟥
4035_010	1	0	1	00:00:00	🟥
4035_011_012	1	0	1	00:00:00	🟥
4035_013	1	0	1	00:00:00	🟥
4035_014	1	0	1	00:00:00	🟥
4035_020	1	0	1	00:00:00	🟥
4035_021	1	0	1	00:00:00	🟥
4035_030_154	1	0	1	00:00:00	🟥
4035_097	1	0	1	00:00:00	🟥
4035_101	1	0	1	00:00:00	🟥
4035_153	1	0	1	00:00:00	🟥

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Pm counter verification	11	0	11	00:00:02	🟥

Test Details

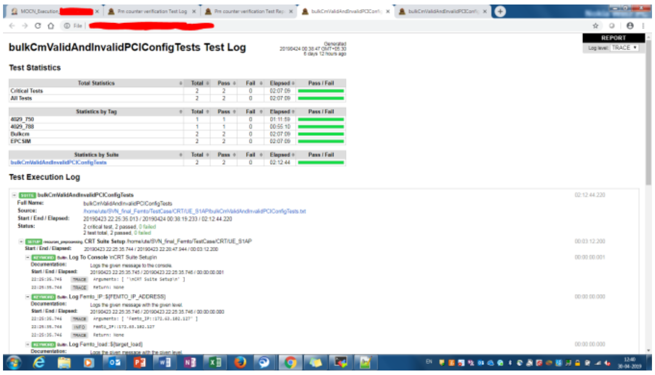
Type:	● Critical Tests ○ All Tests
Status:	11 total, 0 passed, 11 failed
Total Time:	00:00:00.061

Figure 5. Test case report file

For further analysis, there is also detailed log generated as depicted in Figure 6 with all action used in test cases with

DOI : 10.35291/2454-9150.2019.0013

detailed description of input and keyword output parameter along with marked action where that went wrong. Additionally, the keyword “Log” is used to write to log file.As log are generated in HTML format it is convenient for report analysis. Robot Framework also generated XML output file which can be alternatively used.



MOON_Execution [redacted] | Pm counter verification Test Log | Pm counter verification Test Rep | bulkCmValidAndInvalidPCIConfi | bulkCmValidAndInvalidPCIConfi

File | [redacted]

bulkCmValidAndInvalidPCIConfigTests Test Log

Generated
20190424 00:38:47 GMT+05:30
6 days 12 hours ago

REPORT
Log level: TRACE

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	2	2	0	02:07:09	
All Tests	2	2	0	02:07:09	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
4029_750	1	1	0	01:11:59	
4029_788	1	1	0	00:55:10	
Bulkcm	2	2	0	02:07:09	
EPCSIM	2	2	0	02:07:09	

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
bulkCmValidAndInvalidPCIConfigTests	2	2	0	02:12:44	

Test Execution Log

```
- SUITE bulkCmValidAndInvalidPCIConfigTests                                             02:12:44.220
  Full Name:            bulkCmValidAndInvalidPCIConfigTests
  Source:               /home/ute/SVN_final_Femto/TestCase/CRT/UE_S1AP/bulkCmValidAndInvalidPCIConfigTests.txt
  Start / End / Elapsed: 20190423 22:25:35.013 / 20190424 00:38:19.233 / 02:12:44.220
  Status:               2 critical test, 2 passed, 0 failed
                        2 test total, 2 passed, 0 failed
 - SETUP resources_preprocessing. CRT Suite Setup /home/ute/SVN_final_Femto/TestCase/CRT/UE_S1AP   00:03:12.200
   Start / End / Elapsed: 20190423 22:25:35.744 / 20190423 22:28:47.944 / 00:03:12.200
  - KEYWORD BuiltIn. Log To Console \nCRT Suite Setup\n                                  00:00:00.001
    Documentation:      Logs the given message to the console.
    Start / End / Elapsed: 20190423 22:25:35.745 / 20190423 22:25:35.746 / 00:00:00.001
    22:25:35.746  TRACE  Arguments: [ '\nCRT Suite Setup\n' ]
    22:25:35.746  TRACE  Return: None
  - KEYWORD BuiltIn. Log Femto_IP:${FEMTO_IP_ADDRESS}                                    00:00:00.000
    Documentation:      Logs the given message with the given level.
    Start / End / Elapsed: 20190423 22:25:35.746 / 20190423 22:25:35.746 / 00:00:00.000
    22:25:35.746  TRACE  Arguments: [ 'Femto_IP::172.63.182.127' ]
    22:25:35.746  INFO   Femto_IP::172.63.182.127
    22:25:35.746  TRACE  Return: None
  - KEYWORD BuiltIn. Log Femto_load:${target_load}                                       00:00:00.000
    Documentation:      Logs the given message with the given level.
```

12:40
30-04-2019

Figure 6. Test case log file

Results Based on Performance Analysis

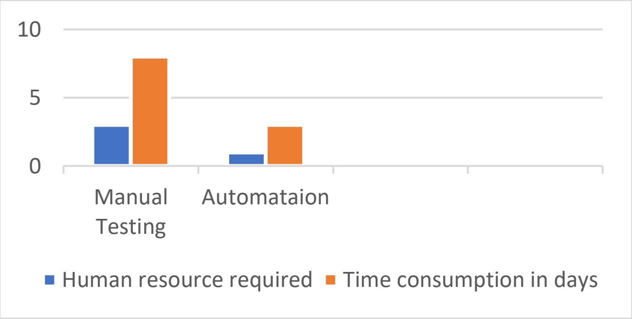


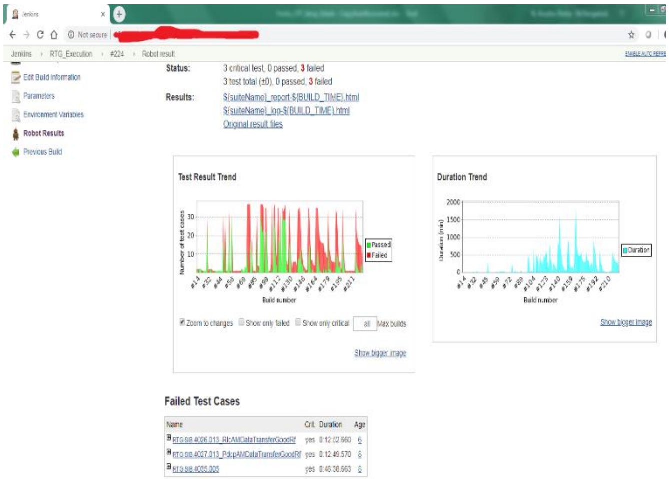
Figure 7. Graph showing Performance Analysis of Manual Testing and Automation Testing

Figure 7 shows the graph which gives the clear picture of Performance Analysis of Manual Testing and Automation Testing. Manual testing requires more man power and also consumes more time to give test results of all the testcases and needs to be executed day and night which is a difficult task for an individual. Whereas in Automation, it takes a single person to write the code and then to run the testcases. Automation is less error prone compared to Manual testing and also consumes less time.

VI. BENEFITS OF TEST AUTOMATION

Manual testing takes longer time to test any of the operation with Femto. So, bugs automation does find will trend sooner after wrong change was made. Debugging is much faster, which also means – cheaper. This is the weightage for automation.

Benefit of using Test automation is it saves time and human resource as well. When there are 1000s of testcases it is a difficult task for an individual to test each and every testcase on a FAP and then to enter whether the test case passed or failed, If failed at which point it failed and what is the reason. To perform this job on every release which is mostly once in a week and sometimes twice a week is very difficult for any individual.



Jenkins

Not secure | [redacted]

Jenkins › RTG_Execution › #224 › Robot result

ENABLE AUTO REFRE

Edit Build Information
Parameters
Environment Variables
Robot Results
Previous Build

Status: 3 critical test, 0 passed, 3 failed
3 test total (±0), 0 passed, 3 failed

Results: ${suiteName}_report-${BUILD_TIME}.html
${suiteName}_log-${BUILD_TIME}.html
Original result files

☑ Zoom to changes □ Show only failed □ Show only critical [all] Max builds

Show bigger image

Show bigger image

Failed Test Cases

Name	Crit.	Duration	Age
RTG.SIB.4026.013_RlcAMDataTransferGoodRf	yes	0:12:52.660	6
RTG.SIB.4027.013_PdcpAMDataTransferGoodRf	yes	0:12:49.570	6
RTG.SIB.4035.005	yes	0:46:38.663	6

Figure 8. Graph generated in Jenkins

With Robot Framework whole process take only few hours, with only one batch command to run the test suite. Jenkin is supported with Robot Framework that execute test suite with latest build that provides continuous integration, Figure 8 shows the resulted graph generated with latest build. Therefore, the person is not busy during test suite execution and can work on other topics. Hence this is why the test automation using Robot Framework is beneficial.

VII. CONCLUSION

Benefits using Robot Framework for automation of Femtocell product is that writing test case follows natural work flow with test case verification, action and finally cleanup. Actual language is used for keyword description, so that it is also easy for non-technical person with simple and easy library that provides great tool for test automation.

By the use of Robot Framework it is very easy to debug the failed testcases as it gives the log , output and report files. These files include the detailed description on how the testcase has been executed, what are the parameters that have been passed to the testcase and their values, which step occurs after which step , at which step the testcase has failed and what may be the reason for the failure. These details help a Tester to debug the failed testcases easily which in turn saves more time.

All process is checked automatically without manual interruption and reports are generated and published on web pages. This saves more time where continuous integration can be introduced. The continuous integration is performed using Jenkins as new build arrives. The cost of automating a test is best measured by the number of manual test cases are prevented from running and cause of bugs which is the biggest strength of the Robot Framework.

REFERENCES

[1] R. Patton, Software Testing, Sams Publishing, 2005

DOI : 10.35291/2454-9150.2019.0013

[2] R.W.Rice, Surviving the top ten challenges of software test automation, In Proceedings of the Software Testing, Analysis & Review Conference (STAR) East 2003. Software Quality Engineering, 2003.

[3] http://code.google.com/p/robotframework/, May 2011

[4] http:// www.opensourcetesting.org/, May 2011

[5] https://github.com/nokia/robotframework-autoitlibrary

[6] http://robotframework.org/robotframework/latest/libraries/Process.html

[7] Roshan Deshmukh, "Global Femtocell Market (Types, Technology, Applications and Geography) - Global Analysis, Trends & Forecast Through 2020", Allied Market Research, IC 14182, April 2014

[8] Jeffrey G. Andrews, et. Al, "Femtocells: Past, Present and Future", IEEE journal on selected areas in communications, Vol. 30, No. 3, April 2012

[9] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2010 to 2015," Whitepaper, Feb. 2011.

[10] D. Kilper, G. Atkinson, S. Korotky, S. Goyal, P. Vetter, D. Suvakovic, and O. Blume, "Power trends in communication networks," IEEE Journal on Selected Topics in Quantum Electronics, Mar.-Apr. 2011, vol. 17, no. 2, pp. 275 –284.

[11] V. Chandrasekhar, J. G. Andrews, and A. Gatherer, "Femtocell networks: a survey," IEEE Communication Magazine, September 2008, vol. 46, no. 9, pp. 59–67.

DOI : 10.35291/2454-9150.2019.0013