Modeling of Test Automation Framework for Femto BSR Modules

Bhargavi D K [1], Dr.Vijaya Prakash A. M [2], Sriharsha B M [3]

[1, 2] Dept. of Digital Electronics & Communication, BIT, Bangalore, India
[3] Technical Specialist, Alcatel Lucent India Ltd, Bangalore, India
[1]dkbhargavi@gmail.com, [2]am_vprakash@yahoo.co.in, [3] sriharsha.bm@alcatel-lucent.com

***Abstract*—This paper describes about a Test Automation Framework based on python for Femto Base Station Router (BSR) modules. The Femtocell is the latest evolution of the mobile network base station. Femtocells are cellular network access points that connect standard mobile devices to a mobile operator's network using residential DSL or cable broadband connections. They are aimed at providing dedicated indoor coverage with enhanced coverage and improved call quality and data rates. This paper presents a reusable and customizable Test Automation Framework that allows unit tests to be executed on Femto BSR modules in a very structured fashion. With this approach, the test fixture for each unit test can be minimized and can speed up the execution of test suites. With the use of this framework, we aim at improving the extensibility and reusability of automated test. The results show that the new software framework improves femtocell software products quality and develop efficiency.**

Keywords—Femtocells, BSR Femto, Automation Framework, Unit Test, Python.

I. INTRODUCTION

Capacity demands of modern mobile telecommunication networks are increasing year by year. People all over the world are using not only more voice call services but also a growing amount of data services with their cell phones. Many of these services, including web surfing, downloading emails, video streaming and video calls require high speed connections and generate large amounts of data traffic to the network. The customer expectations are rising and soon the mobile terminals will have to achieve the same bitrates as the current fixed internet connections [1].

Coverage has always been an important issue in mobile telecom networks. It has traditionally been a problem in rural areas due to the long distance between base stations and in indoor and underground locations due to the wall attenuations. The vendors have to constantly come up with solutions to make the best of the limited radio resources: space and spectrum. Smaller cell sizes such as microcells and nanocells have been used to gain more capacity in urban hotspots like shopping centres and office buildings. Microcells and nanocells as well as Distributed Antenna Systems (DAS) have also been used to improve coverage inside buildings, basements and subway tunnels. These solutions are effective but also expensive [7].

Femtocells offer a different approach to these problems. Femto is a factor denoting one thousandth of nano. Femtocells are very small, low cost base stations and their maximum allowed transmit power level is low. Femtocells are even smaller than nanocells but the biggest difference is not the size of the cell [1]. The devices are integrated to small plastic desktop or wall mount cases and are installed to the customers' premises by the customers themselves. The customers' existing internet connections are used as backhaul connections and the devices are powered from the customers' electricity sockets.

Femtocells are cellular access points that connect to a mobile operator's network using residential DSL or cable broadband connections [2]. They have been developed to work with a range of different cellular standards including CDMA, GSM and UMTS. Femtocell is a recent research area. For the mobile operators, femtocell major features are to improve both coverage and capacity in the in-doors while optimizing the energy consumption and make BS deployment cost effective. Mobile users get better signal quality and longer battery life using Femtocells [8]. Figure 1 shows a home cell that supports 4 users. The other types are enterprise and metro femtocell shown in Figure 2.

The complexity of these Femtocell systems is continuously increasing both in terms of hardware and software [3] [9]. This increased complexity and involvement of multiple streams of engineering (electrical, electronics, software etc) demands Co-design of hardware and software resulting in many interim releases of system. These interim releases of system requires multiple test effort as tests needs to be executed repeatedly as part of regression to identify the defects induced as part of integration and newly added functionalities. This repetition tests results in longer product development life cycle time and increased cost while the ever competitive and demanding market of both suppliers (Original Equipment Manufacturers) and customers (Network Providers) require faster turnaround time and decreased cost for development of products.

Fig.1 Residential Femtocell

Fig.2 (a) Enterprise Femtocell (b) Metro Femtocell

II. FEMTOCELLS ARCHITECTURE

The typical Femtocell architecture is a flat IP architecture shown in Figure 3 [1]. The standard bodies have published formal specifications for femtocells for the most popular technologies, namely WCDMA, CDMA2000, LTE and WiMAX. These all broadly conform to architecture with three major elements:

1. The Femtocell Access Points (FAPs) themselves, which embody greater network functionality than found in Macrocell base stations, such as the radio resource control functions. This allows much greater autonomy within the femtocell, enabling self-configuration and self-optimisation. Femtocells are connected using broadband IP, such as DSL or cable modems, to the network operator's core switching centres.
2. The femtocell gateway, comprising a security gateway that terminates large numbers of encrypted IP data connections from hundreds of thousands of femtocells, and a signalling gateway which aggregates and validates the signalling traffic, authenticates each femtocell and interfaces with the mobile network core switches using standard protocols, such as Iu.
3. The management and operational system which allows software updates and diagnostic checks to be administered. These typically use the same TR-069 management protocol published by the Broadband Forum and also used for administration of residential modems.

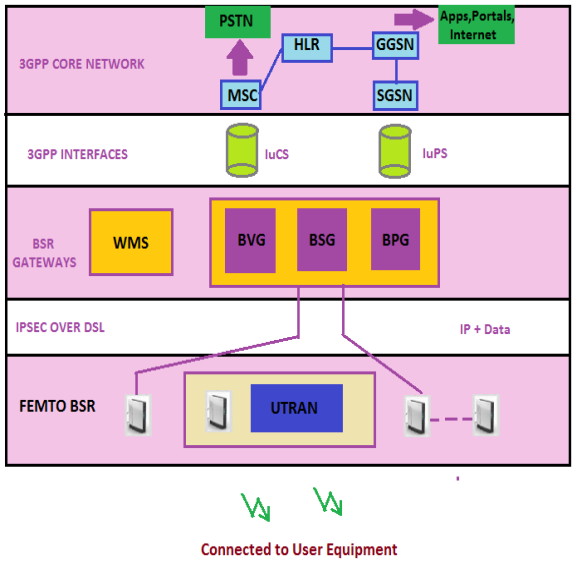

Fig.3. BSR Femto Network Architecture

The Femtocell Network is deployed in clusters of up to 64000 femtocells each. A "cluster" is defined as the set of Small Cells which appears as a single RNS (Radio Network Subsystem) to the UMTS Core Network. To support the cluster there are a number of other network elements which include the cluster gateway elements such as BSR Security Gateway, BSR Voice Gateway, BSR Packet Gateway etc. The Security Gateway, the optional IP Protocol Converter (IPC) (together called the Small Cell Gateway), plus the Operations and Maintenance (OAM) solution made up of Device Manager, Cell, Femto Management System and File Server [1].

In the present UMTS era, the Home Node B (HNB) is usually called as a Femtocell or a Femto Access Point. Typically HNBs operate over the licensed spectrum and connect to the operator's core network by using a residential Internet connection. A residential Internet connection can be based on a DSL, cable broadband connection, and optical fibre or wireless last mile technologies. Like a Wireless Local Area Network (WLAN) Access Point (AP), the HNB is a small device and it is installed by the user.

III. FEMTOCELL DEPLOYMENT AND ITS SOFTWARE DEVELOPMENT ENVIRONMENT

A single Femtocell supports usually at most four to eight simultaneous voice connections (concurrent maximum voice connection support in femtocell is implementation specific i.e. different products support different amount of simultaneous voice connections) in any indoor environment, permitting many authorized users to be able to connect to the femtocell to utilize services other than voice, such as text or real time multimedia streaming etc. the user's subscription model (service and charging) for femtocell services may vary according to user needs and depends upon operators.

There are various factors that affect peak data rates such as the air interface technology used, the user subscription and broadband link capacity. For supporting femtocells operations it requires a network element called Femto Gateway which acts as a Radio Network Controller (RNC) towards the core network [4] [5]. Figure 4 illustrates a basic femtocell deployment model in a real world environment. In order to utilize femtocell services, a user will buy a femtocell and will connect to it through its own fixed broadband access.

Environment refers to the collection of hardware and software tools a system developer uses to build software systems. As technology improves and user expectations grow, an environment's functionality tends to change. By Software Development Environment (SDE) we mean an environment that augments or automates the activities comprising the software development cycle, including programming-in-the-large tasks such as configuration management and programming-in-the-many tasks such as project and team management [6]. We also mean an environment that supports large scale, long-term maintenance of software.

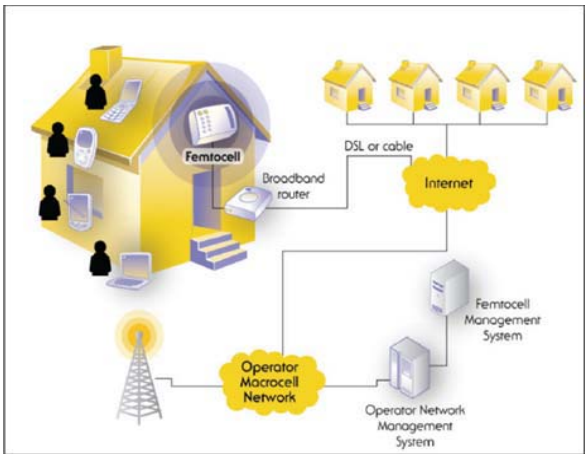

Fig. 4 Typical Femtocell Deployment

The Femto Software Development Environment consists of a number of softwares related to Radio Resource Management, Call Processors, Load Control, Power Control etc. These softwares are implemented on various Operating Systems (OS). Hence, verification and validation of these software components on different OS flavour for multiple features is one of the longest phases of product development cycle, it is that without Automation it would be difficult to achieve the reduction in product cost or improved software quality. Hence all the requirements for test environment were identified and satisfied that supported in achieving the above goals.

The FMS is the central Femto Management System responsible for providing integrated management of the Femto Gateway, the IP Security channel and the Device Manager. It also provides management of the Small Cell in conjunction with the device manager and File Server. The system has been designed to minimize downtime during upgrade, data back-up, data restore or when unexpected outages occur. This offers 3GPP standardized interfaces to interconnect with upstream Operation Support Systems (OSS). In addition, interoperability testing is completed with independent OSS software vendors. As a consequence, this system significantly accelerates time-to-market and considerably reduces the effort of integration with operator's incumbent systems [10] [12].

The file server is a central repository for Small Cell software loads, bulk configuration data, and for uploaded log files and performance files. This is a passive network element. The FMS provides personalized configuration files to file server from where the small cell takes it. Periodically, Small Cells will upload log files and performance data files and they are extracted from file server by FMS. The home device manager is a network element located within the mobile operator’s central office IP network and performs several functions vital to the Small Cell Network. The functions are configuration updates and policy based operations.

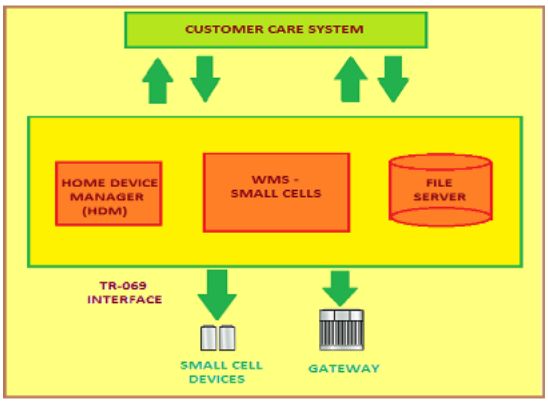

Fig.5. Femto Management System for Femtocells

IV. PROPOSED AUTOMATION FRAMEWORK

The proposed framework model encapsulates all complex components of a Data Driven Test Framework into 5 major modules [11]. These components can be considered for designing and further expanded upon as:

- **Data Repository:** The module containing the test data, for the data driven framework.
- **Test Harness:** This is the heart of any framework architecture; it contains the test driver, suites, and test library. This executes all test cases.
- **Unit Under Test (UUT):** Mostly data driven frameworks are for testing Software Modules or GUI applications, so the UUT driver is a module which actually does all the actions as per the test case on the application, i.e. actual interaction with the application.
- **Validation:** This module is used to design the components for validating whether the configuration done using UUT Driver is correct or not.
- **Log Generation and Reporting:** This module will

be responsible for generation of test case execution logs and the Report Generation for all the test suites executed.

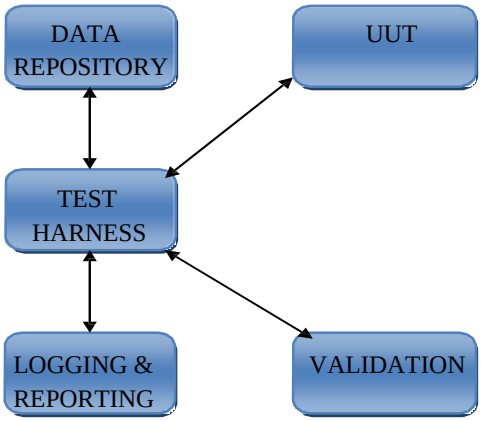

Fig.6 Proposed Automation Framework Architecture

The following requirements for test environment were identified that supported in achieving the above goals.

a) Install the build as selected using User Interface.

b) Run all the applicable test cases in the scheduled suite.

c) Fault/Crash handling mechanism.

d) Give the entire Suite Run Summary.

The Working Principle of Automation Framework is designed in such a way as to provide flexibility to test as many UUTs as possible. The Test Harness is the heart of the Framework. Every UUT will have a corresponding Configuration File which contains all the test case details. The Test Harness will execute the tests as per the Configuration File using the Data Repository for each Unit Under Test and generates the test results. These results are verified by the Validation component which may or may not be the part of Test Harness. Logging and Reporting refers to the Logs generated as part of Test Execution which can be an XML report file or ordinary Summary report file.

V. PROGRAMMING SOFTWARE

The framework is implemented in Python language. Python is a widely used open source, general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. The language provides constructs intended to enable clear programs on both a small and large scale.

Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems. Python programs can be compiled on cross platforms without any modifications except for system dependent features which are minimal.

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object-oriented* languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

Rather than requiring all desired functionality to be built into the language's core, Python was designed to be highly extensible. Python can also be embedded in existing applications that need a programmable interface. This design of a small core language with a large standard library and an easily extensible interpreter is one of the greatest advantages.

VI. IMPLEMENTATION

The proposed framework was implemented in Python 2.7.5 version. Each UUT had a dedicated directory structure which consists of the UUT driver, its corresponding Configuration File, and subdirectories for log files. The central Test Harness would support the execution of any UUT with all or subset of its test cases and debugging of the failed test case. Following steps give a brief description about implementation of the framework:

- First the build installation would be done.
- The Test Harness would parse the Configuration file as per the set instructions and generate an array of test cases to be executed. These would then be executed one by one by passing the corresponding input logs to the UUT.
- The most important aspect of the harness is to handle the test case failures properly. In-case there is a Crash or abnormal failure in a test case, and the failure is not handled properly, it may cause the entire suite to be aborted. This was taken care of by exception handling. In case of crashes, the harness would identify such an issue at the end of test case execution after saving necessary “crash dump” in clean up phase. The harness would immediately reload the host machine so as to regain the original working environment as before.
- After each test case is executed, validations of the output logs generated by the UUT and assert the success or failure of each test case.

- The entire Suite run Summary as the number of cases failed passed and total executed should be calculated at harness level.
- At the end of test suite, the test harness also creates an XML output which is suitable to be processed in Jenkins.

VII. RESULTS AND DISUSSION

The overall outcome of the proposed framework for the BSR Femto Modules is outlined as follows:

a) A single Test Environment/Framework that supports all the phases of testing.

b) Automated Test Framework that supports test development on host PC, test execution on target and test result evaluation and report generation on host PC lucidly.

c) Supports multiple types of embedded development and debugging environments for test execution with reduced effort.

d) Supports data driven test approach for multiple test cases (inputs/outputs set) but common test procedure (test logic) for exercising requirement based testing, robustness test cases and Boundary Value Analysis etc.

e) Configurable test environment which can be used across projects will little hardware and software configuration change or no change at all depending on requirements.

f) Reduced cost and cycle time for developing a test environment which satisfies the wish list.

VIII. CONCLUSION

The above proposed Automation Framework Architecture facilitated us development of the actual framework for the BSR Femto Modules in a properly planned and organized manner. We first developed the design of each block and then the interfaces and interactions between each block of the Framework architecture. The entire implementation helped us the cut down the manual efforts by 1/4th i.e. 25 percent of that without the automation.

REFERENCES

[1] Meghna Sarkar., et. Al., "Next Generation Cellular Network Technology – Femtocells", National Conference on Emerging Technologies in "Broadband for Sustainable Development", May 15 - 17, 2014.

[2] Madhwal E and R Mahapatra, "Performance analysis of femtocell network in macrocell environment", IEEE conference on Electronics, Communication and Instrumentation, Kolkata, pp 1-4, January 2014.

[3] J Weitzen, Li Mingzhe, E Anderland and V Eyuboglu, "Large Scale Deployment of Residential Femtocells", Proc. IEEE, Vol. 101, pp 2367-2380 August 2013.

[4] M. Taranetz, M. Rupp, "Performance of femtocell access point deployments in user hot-spot scenarios", IEEE Conference on Telecommunication Networks and Applications Conference, pp 1-5, November 2012.

[5] Ravinder Kumar, Mr. Karambir Singh, " Enhancing Component Based Testing Using JUnit Tool in Net Beans Environment", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7, July 2012.

[6] Fei Wang., et. Al., "A Test Automation Framework Based on WEB", IEEE/ACIS 11th International Conference on Computer and Information Science, June 2012.

[7] Jeffrey G. Andrews, "Femtocells: Past, Present and Future", IEEE journal on selected areas in communications, Vol. 30, No. 3,pp 275-286, April 2012.

[8] Alan Barbeiri., et Al, "LTE Femto cells: system design and performance analysis", IEEE journal on selected areas of communication, Vol 30, no. 3, April 2012.

[9] J. Liu et al, "Femtocell base station deployment in commercial buildings: a global optimization approach", IEEE lSAC, voL30, no. 3, April 2012.

[10] Mohammad Wahid, Dr Abdullah Almalaise, "JUnit Frameworl: An Interactive Approach for basic Unit Testing Learning in Software Engineering", 3rd International Congress on Engineering Education (ICEED), 2011.

[11] Rohan R. Kachewar, "K model for designing Data Driven Test Automation Frameworks and its Design Architecture "Snow Leopard", International Journal of Computer Applications (0975 – 8887), Volume 31– No.7, October 2011.

[12] Yao Chen; Yuzhang, Wei-Zheng Cai, "The Research and Implementation of Automatic Unit Test Recording Framework", 2nd International Conference on Software Technology and Engineering (ICSTE), 2010.