Verification for software communication architecture compliance of software defined radios

[1]Divyashree K B, [2]Jayanthi .J, [3]Rizwana Kowsar M S, [4]Bhagyashree R

[1,3,4] Dept. of CSE, A.P.S College of Engineering
[2]CSIR-National Aerospace laboratories

Abstract : Software Communication Architecture is a standard which defines the software infrastructure for the management control and configuration of the software defined radio. Main purpose of the SCA specification is to define the operating system environment software. This is mainly used for increasing the flexibility, interoperability and reduces the support costs, reduces the system acquisition cost. Software-defined radio(SDR) is a radio communication system where components that have been typically implemented in hardware(e.g. mixers, filters, amplifiers, modulators/demodulators, detectors ,etc.)are instead implemented by means of software on a personal computer or embedded system. Spectra CX is a tool to Simplify, Accelerate and Validate SCA-based Development. Software Communications Architecture Tool Spectra CX is a model-driven development tool that simplifies, accelerates, and validates a significant proportion of the Software Communications Architecture (SCA)development process. Spectra CX validates SCA compliance at the architectural and unit test level, and generates correct-by-construction SCA compliant artifacts, such as: XML descriptor files, compliance test reports, and validation documentation. Spectra CX enables SCA and non-SCA software aspects to be developed together, integrated early, and thoroughly tested. By using this tool operating system of the software defined radio is tested for SCA compliance.

I. INTRODUCTION

"Software defined radio is a radio that is substantially defined in software and whose physical layer behavior can be significantly altered through changes to its software" A Software Defined Radio (SDR) is one way to realize a radio communication system. In this approach, many of the components that have typically been implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors. etc.) are instead implemented using software. Since the radio communication system is now implemented primarily as software, it can be run on a personal computer (PC) or other embedded computing device with sufficient input, output, and computational capabilities. The Software Communications Architecture (SCA) is a non-proprietary, open architecture framework that tells designers how elements of hardware and software have to operate in harmony within a software defined radio. The SCA is not a system specification, but was intended to be implementation independent. Thus rather than being a specification, it is a set of design constraints. If a developer designs a system according to these design rules, the system will be portable to other SCA implementations regardless of what operating system or hardware that implementation utilizes.

The aim of SCA is to define an Operating Environment (OE), often referred to as the Core Framework (CF). This CF implements the management, deployment, configuration, and control of the radio system and the applications that run on top. JTRS was established to pursue the development of future communication systems, capturing the benefits of the technology advances of recent years, which are expected to greatly enhance interoperability of communication systems and reduce development and deployment costs. The goals set for the JTRS program are:

- greatly increased operational flexibility and interoperability of globally deployed systems;
- reduced supportability costs;
- Upgradeability in terms of easy technology insertion and capability upgrades.
- Reduced system acquisition and operation cost.

In order to achieve these goals, the SCA has been structured to

- provide for portability of applications software between different SCA implementations
- Leverage commercial standards to reduce development cost.
- reduce development time of new waveforms through the ability to reuse design modules
- Build on evolving commercial frameworks and architectures.

II. PROBLEM STATEMENT

The objective of this paper is to verify the Software Communication Architecture compliance of operating system of software defined radios. In this internship program I am concentrating only on the OS part of the

core framework, which is the operating environment for the Software Defined Radios and verifying the SCA compliance of OS of software defined radio by providing the test case scenarios.

The scope of this internship is as follows:

- Reviewing the methodology for verifying the OS whether it is SCA compliant or not.
- By taking the open source check for the SCA compliance.

III. EXISTING SYSTEM

The software defined radios are mainly used in military fields for communication purpose because of its great feature which keeps the information confidential. Every communication devices will be having transmitter and the receiver for transmitting and receiving the signal. These signals will be transmitted at particular frequency .this may cause lot of Power consumption, Prone noise, or it May suppress the defined frequency

To overcome these disadvantages sender should transmit the signals at different frequencies. Changing the frequencies each time and transmitting the information is more expensive. Suppose an opponent hacks the information at a particular frequency the sender has to transmit the information at different frequency.

Problem with the normal existing communication devices is if the transmitting frequency changes then the entire hardware architecture of the transmitter and receiver has to be changed each and every time. Changing the frequency and sending the information is not a big task but changing the entire architecture of the transmitter and receiver is more difficult and expensive.

For example suppose a transmitter is fixed in satellite and the receiver is in earth and opponent hacks the information at a particular frequency then sender has to change the frequency immediately and transmit the information. If sender changes the frequency then the problem occurs. The entire hardware architecture of the transmitter and the receiver has to be changed again. This is very difficult task and more expensive because transmitter and the receiver both are placed at different places. One is in earth and another is fixed in satellite. This is the major problem with the existing communication devices.

Disadvantages of Existing System

The disadvantages of existing systems are as follows

- Entire hardware architecture of transmitter and receiver has to be changed frequently.
- Changing the entire architecture of transmitter and the receiver each and every time is the difficult task.
- More expensive.
- Lot of time consumption.
- Opponent can easily hack the complete information, if the further part of the information transmits at the same frequency without changing the hardware architecture.
- Easy for the opponents to hack the information once he gets the transmitting frequency.

IV. PROPOSED SYSTEM

To overcome the disadvantages of existing system the software defined radios are used. These radios can operate at different frequencies without changing the hardware architecture of transmitter and receiver. Instead of that only the software layer of the radio is changed according to the frequency. Software Defined Radio is the reconfigurable and reprogrammable radios that can show different functionality with the same hardware. This software defined radio uses a standard architecture called software communication architecture. Software Communication Architecture is the standard which defines a software infrastructure for the management, control and configuration of the software defined radios.

The Main purpose of the Software Communication Architecture (SCA) specification is to define the operating environment software. SCA basically describes the software components with in a software defined radio and in particular it defines the interfaces. SCA is an open architecture framework that tells designers how elements of hardware and software are to operate in harmony within a software defined radio. SCA used for defining the software structure and interfaces for the software defined radios and their designs. Software Communication Architecture standard is the set of requirements that has to be compliant with operating system, Middleware, file system, devices, application waveform.

In this internship program I am concentrating on the OS part of the core framework, which is the operating environment for the Software Defined Radios. In this internship I am verifying the SCA compliance of OS of software defined radio by providing the test case scenarios. The scope of this internship is as follows.

- Reviewing the methodology for verifying the OS whether it is SCA compliant or not.
- By taking the open source check for the SCA compliance.

Advantages of SCA Standard

The advantages of software communication architecture standards are as follows

- SCA provides common infrastructure for distributed application deployment.
- SCA allows quicker integration of external applications.

- The SCA allows easier insertion of new technology.
- The SCA provides an infrastructure for extensibility and integration.
- The SCA reduce the development of non recurring engineering for system development.
- To increase the flexibility.
- To increase the interoperability.
- Reduce support costs.
- Provide upgradeability through technology insertion.
- And reduce the system acquisition costs.

V. DESIGN AND ARCHITECTURE

Common Open Architecture: The use of an open, standardized architecture has the advantages of promoting competition, interoperability, technology insertion, quick upgrades, software reuse, and scalability. Multiple Domains: The JTRS family of radios must be able to support operations in a wide variety of domains, including airborne, fixed, maritime, vehicular, dismounted and handheld. Multiple Bands: A JTRS radio can replace a number of radios that use a wide range of frequencies, and it can interoperate with them.

- Compatibility: JTRS radios must be able to communicate with legacy systems to minimize the impact of platform integration.
- Upgrades: The JTRS architecture must enable technology insertion, so that new technologies can be incorporated to improve performance, and to build future-proof radios.
- Security: Security is a very important aspect of military radios. The architecture should provide the foundation to solve issues like programmable cryptographic capability, certificate management, user identification and authentication, key management, and multiple independent levels of classification.
- Networking: The JTRS radios should support legacy network protocols, for the purpose of seamless integration. The architecture should also support wideband networking capabilities for voice, data and video.
- Software Reusability: As with any other software architecture, the JTRS architecture should allow for the maximum possible reuse of software components. The components should support plug-n-play behavior with waveforms being portable from one implementation to the next.
- Conceptually, a SCA radio has three segments: i) The Waveform Deployment; ii) the Core Framework; and iii) the Domain Profile. These three segments are each divided into physical and logical views. In the Waveform Deployment segment, the radio hardware is the physical view of the radio system. However, the waveforms are realized through software that is loaded on the physical radio elements. There are two layers in the logical view of the radio system. The first consists of the set of components that form a waveform application or other service on the system. The second is the application that provides the top-level interface and control for the set of components

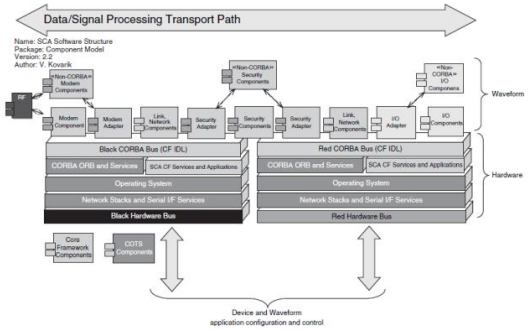

Figure 2 The SCA waveform component organization

The Core Framework segment includes all software required to manage the radio system and deploy applications. It has a physical view and a logical view as well. The physical view of the Core Framework provides high-level management of the physical devices in the radio system. The logical view provides the same for the waveform applications and other services. The Domain Profile segment consists of the set of XML files that describe the hardware resources within the radio system, the waveform application structure, and dependencies between waveform components, connections between components, and dependencies on hardware resources. This is illustrated in Figure 3.2

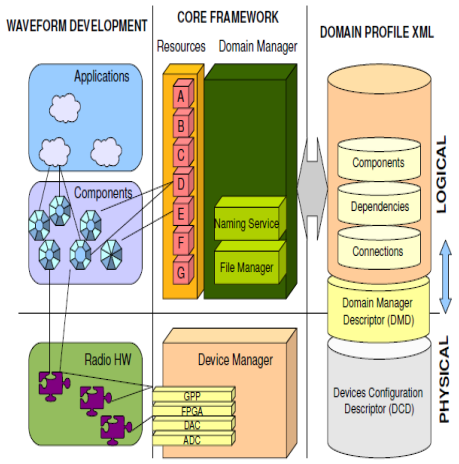

Figure 1 Abstraction layers in an SCA System

SCA Core Framework

The figure below shows a very simplified version of the main layers of a JTRS radio set. With it, you can start to get a feel of where all the components come from. Starting from the bottom and working up, the three

major components are:

- The actual radio hardware that provides all the analog and digital interfaces to the outside world.
- The operating system and middleware.
- The SCA core framework which is the software that interfaces to the hardware.
- The waveforms that are actually what gives the radio its specific characteristics needed to satisfy the application requirement.

The SCA core framework connects the SCA-compliant waveforms shown at the top with the SCA-compliant hardware at the bottom within the JTRS radio set. This allows the platform developers who produced an SCA compliant radio set and the waveform developers, who independently developed their SCA-compliant waveforms, to expect that the waveforms will function properly with the radio hardware.

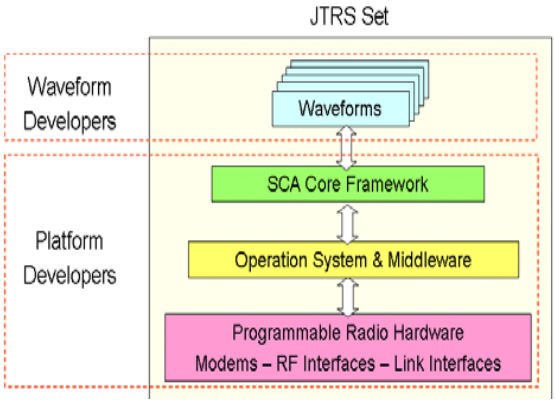

Figure 3 core frame work of SCA

SCA Operating Environment

Shown in Figure 3.5 below, is the SCA Operating Environment, sometimes called the OE. It basically depicts all the layers of the software and how they interact. Figure 3.6 depicts an expansion of the OE components. Starting with the components below the application line, we will work our way down so we can build a foundation for the applications to run. These components consist of the Core Framework, the Operating System, the Board Support Package, CORBA. (Common Object Request Broker Architecture) ORB (Object Request Broker), and the Network Service.

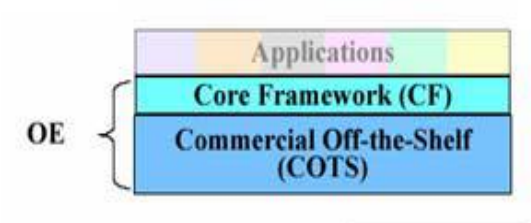

Figure 4: SCA operating environment

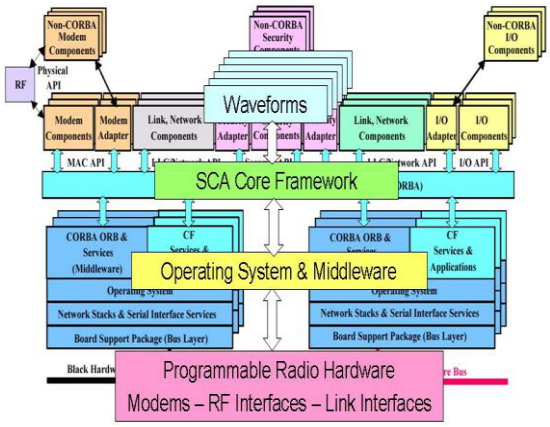

Figure 5: Expansion of the OE components

The SCA recognizes the need for separating the black, or secure components of an application from the red, or non-secure components. So what we have here are basically two parallel and complete OE sets that exist in the same application to support both black (secure) and red (non-secure) components of the system.

The SCA specification defines the Core Framework (CF) as the essential core set of open application layer interfaces and services. The framework provides an abstraction of the underlying software and hardware layers for software application designers. The core framework includes the base application interfaces which can be used by all the software applications and the framework control interfaces that provide system control.

8 Implementation

This section tells about the implementation part which describes the open sources and testing tool used for SCA compliance testing and the testing benefits.

Testing

Software testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application or product: Meets the business and technical requirements that guided it's design and development

Verification

Verification is done at the starting of the development process. It includes reviews and meetings, walkthroughs, inspection, etc. to evaluate documents, plans, code, requirements and specifications. Suppose you are building a table. Here the verification is about checking all the parts of the table, whether all the four legs are of correct size or not. If one leg of table is not of the right size it will imbalance the end product. Similar behavior is also noticed in case of the software product or application. If any feature of software product or application is not up to the mark or if any defect is found then it will result into the failure of the end product.

Hence, verification is very important. It takes place at the starting of the development process.

SCA compliance and testing

One fundamental principle of the SCA is that it maximizes the independence of software from hardware by requiring application and device portability, encouraging code reuse and allowing for rapid technology insertion over time. By this definition, any hardware platform that is provided without any software support technically is SCA-compliant. Since the SCA is only software architecture, providing no software to run on a board has the lowest risk of being in contravention to the SCA.

In fact, there is mention of hardware architecture in the SCA. The hardware must be able to be described in object-oriented terms that are consistent with the Software Architecture description, rein-forcing the concept that application functions can be implemented in either hard-ware or software. Hardware is intentionally not specified to any degree in order to allow the constraints of a particular application to drive hardware choices.

It is important to recognize that the SCA is not implementation architecture but, in fact, constitutes an implementation-independent framework for the development of JTRS software radios. As such, it is comprised of inter-face and behavioral specifications, general rules, waveform Application Program Interfaces (APIs) and security requirements to meet the JTRS program criteria. Implementation of many parts of the SCA must take into account the vagaries of differing hardware and underlying software, as long as the SCA requirements are met. Due to this, different hardware vendors can and will supply different levels of SCA software support for their platforms, all while being "SCA-compliant"

Testing For SCA Compliance

In order that any software can be declared as SCA software it needs to be tested for SCA compliance. In this way the API (Applications Programming Interface) can be determined as compliant and it will operate with other SCA complaint software. Also its performance is tested for correct operation.

SCA Software Communications Architecture is an ideal standard to use for large Software Defined radio SDR projects where different software elements may be brought in from different software houses. It provides a robust interface between the different software modules that allows components to communicate together reliably in a known standard format. However the use of SCA Software Communications Architecture does place an overhead on the complexity of the system. This may mean that SCA may not be the right choice for many smaller projects. Whether to use SCA or not is a design choice that needs to be made at the beginning of the project.

Available Open Source Operating environments

- OSSIE: open source SCA implementation embedded. It is an open source OE for SDR.
- REDHAWK: is SDR framework designed to support the development, deployment and management of real time os.
- Hack RF: is the open source for SDR platforms.
- Transmits or receives any radio signals from 30MHZ to 6000MHZ on.
- Designed to enable test and development of the modern and next generation radio technologies.

Testing Tool Used to check SCA compliance

Spectra CX™ is a component-based development (CBD) tool essential for the design of re-usable software components. Spectra CX (SCX) provides the design, development and maintenance support for system structure, and inter-component communication and control, allowing developers to minimize software complexity and rapidly adapt software to new platforms. SCX's powerful visual system representation uses a UML 2.0 interface to give cross-functional and geographically-dispersed team members critical insight into system structure.

SCX is built on top of the IBM Rational Software Architect (RSA) and utilizes and extends many of its capabilities and functionality. SCX enables developers to model, configure, integrate, and validate applications through a UML 2.0 interface. Using SCX you can verify your model and generate the complete set of artefacts for your domain, reducing development time.

Spectra CX is a tool to Simplify, Accelerate and Validate SCA-based Development. Software Communications Architecture Tool Spectra CX is a model-driven development tool that simplifies, accelerates, and validates a significant proportion of the Software Communications Architecture (SCA) development process. Spectra CX validates SCA compliance at the architectural and unit test level, and generates correct-by-construction SCA compliant artifacts, such as: XML descriptor files, compliance test reports, and validation documentation. Spectra CX enables SCA and non-SCA software aspects to be developed together, integrated early, and thoroughly tested. Spectra CX also reduce development risk due to its consistent model-based approach. Together the above benefits result in faster time-to-market, lower costs, better software quality, and superior compliance for all SCA waveform and platform code developed with Spectra CX.

Validate With Spectra CX

Spectra CX allows developers to produce SCA compliant software from day one. Validation is built right into Spectra CX providing automatic identification of errors in SCA-compliant radio platforms and

waveform applications. In addition to checking the syntax of the Document Type Definition (DTD), Spectra CX validates the semantic correctness of the model. Errors are presented along with a hyperlink to the model construct that violates the SCA standard and a reference to the relevant section of the SCA standard. Spectra CX also provides suggestions for correcting the violation.

Many SCA projects involve the integration of legacy and 3rd party waveforms. Spectra CX facilitates integration by allowing developers to import complete or partial sets of SCA descriptor files (XML), validate them against the SCA standard for correctness, and assemble them into a complete and valid application or platform. Spectra CX provides a complete development environment for SCA compliant software, validating at every stage, up to and including deployment.

- Automatically validate for SCA compliance - applications (waveforms), platforms, and deployments
- Validate model for syntactical and semantic correctness
- Import and validate 3rd party and legacy components, applications, devices, nodes
- Complete validation of SCA deployments
- Guidance on resolving SCA violations with links to the cause and suggested modifications
- Enhanced validation with cross-reference to the standards

Figure 6 : Validation using Spectra CX

Generate with Spectra CX

Accelerate and de-risk SCA development by generating correct-by-construction SCA artifacts Spectra CX provides push-button generation of correct-by-construction descriptor files and documentation. By automatically generating the complete set of SCA compliant descriptor files, i.e. the entire SCA Domain Profile, development time can be reduced from months to days. Generating documentation improves communication between team members. Documentation generation through Spectra CX is completely customizable. Developers can produce documentation relating to a specific view, for example, a list of all components in the waveform or a description of an intended deployment.

Automated generation of code implementing SCA component structure is provided through Spectra's Code Generators. They automate the production of both SCA application code and SCA device code. Device code abstracts the physical hardware in accordance with the SCA specification. Automating the generation of this code benefits both developers of SCA-compliant radio platforms and developers needing to modify an SCA-compliant radio platform, allowing them to make changes to their hardware while ensuring continuous adherence to the SCA specification.

- Push-button generation of the complete set of SCA XML descriptors
- Fully customizable documentation generation
- Extensible code generators
- Generates SCA structural code in C++ or C
- Build environment generation

Code Generators

Spectra CX Code Generators are designed to provide Software Defined Radio (SDR) developers with highly productive and error-free Software Communications Architecture (SCA) compliant source code for SDR components.

Generation of source code from Spectra CX de-risks development since the code precisely reflects the SCA-validated component, preserving the architectural intent of the component so ensuring the consistent and correct implementation of SCA interfaces. Code generation also ensures consistency of coding across the project, resulting in high quality, easy to maintain code.

PrismTech currently offers two Code Generator options:

- C++ Code Generator for general purpose processors (GPP)
- C Code Generator for GPP and digital signal processors (DSP)

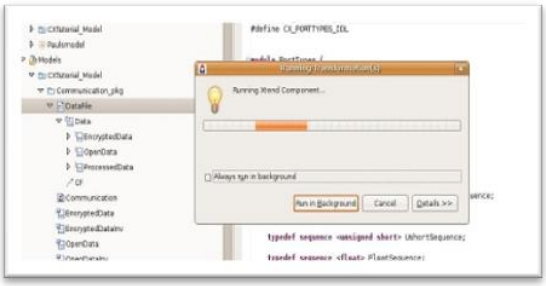

Figure 7: Generate code using Spectra CX

Develop With SpectraCX

Design and develop SCA component behavioral code using Model-Driven Development (MDD).Spectra CX provides developers with a complete model-based

development environment that will significantly reduce the time to develop and maintain their components. Seamless integration with the Eclipse IDE allows developers to use their preferred tools for developing and managing source code that is linked to the model of the waveform. Spectra CX supports the integration of behavioral models created by 3rd party UML, Block Diagram, and State Chart design tools.

- Model-driven development can be used throughout the waveform realization process
- Component developers can work from a fully described application architecture
- Complex components can be designed and implemented using Spectra CX
- Integrated with Eclipse Team system and 3rd party Configuration Management solutions
- Automatic linkage of the model and source code eliminates the need for manual synchronization
- CDT integration allows developers to model and code using Spectra CX
- Supports 3rd party design tools

Execute With Spectra CX

Monitor the component system on its actual target, with the platform running multiple applications for complete testing. In an SCA radio, the actual deployment of software components to hardware devices is done at system initialization time. For this reason, developers require features that enable them to connect to the SCA Core Framework (CF) to thoroughly test their application running live. In addition to features for SCA development, the Spectra CX environment contains comprehensive features allowing developers to quickly and easily test and debug their components and applications.

Spectra CX's runtime monitor allows users to start the SCA CF, load an application to a platform and inspect it in real-time. With runtime monitoring, developers can see if the deployment they expected to have is actually the one dynamically created by the CF. Users can also take their deployment design models and explicitly enforce them during runtime. This feature ensures that all specific test cases are executed thoroughly.

Spectra CX's runtime monitoring feature can be connected to any SCA compliant operating environment (OE). For example, it will run on a PC development host equipped with a PC compatible middleware suite. This is especially beneficial because it allows developers to test their deployments very early on in the development cycle. The run-time monitor will communicate with an embedded target's OE, for testing on the actual target.

The runtime monitor allows multiple applications to be started and stopped with the click of a button. Users can load their SCA platform with many applications, better representing what is likely to happen in the field. Previously set values, such as component properties, can be adjusted on-the-fly so that users can change the behavior of the waveform as it is running, and continue observing it.

Spectra CX captures logs generated by the CF (if available) and presents them to the user within Spectra CX's UI.

Also available from PrismTech is an SDR Development Suite comprising Spectra CX plus a complete SCA operating environment (Spectra CF and CDB) designed to run on a PC host. Users can observe their applications deployed on a commercial-grade SCA CF in record time.

- Start and stop SCA Core Framework on host within Spectra CX environment
- Load and unload multiple applications
- Support for development hosts and embedded targets
- Use design model to force particular deployment
- Adjust component properties while application is running

Test With SpectraCX

Test early and often to minimize development risk. Automated testing of components and subsystems of an application (waveform) is provided with Spectra CX, through the Spectra CX SCA Test framework. Generating specific code for testing the developed components for SCA compliance is critical to ensuring delivered components meet the runtime characteristics demanded by the standard. Spectra CX SCA Test allows users to generate, compile and execute test code, and view test results directly from the toolset. All tests can be executed on host or target systems.

- SCA Component Test: test the component for compliance with the SCA standard
- SCA Functional Test: test the component for compliance with functional requirements
- SCA Scenario Test: Test the interaction of components against specified scenarios

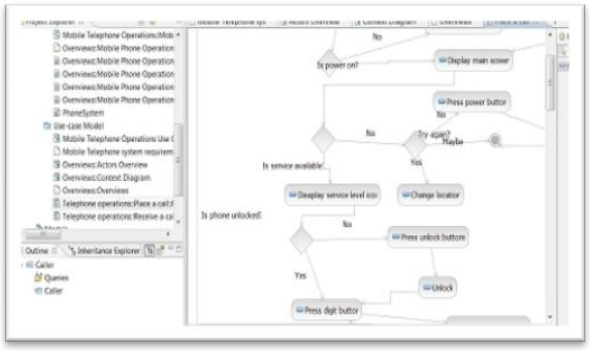

Figure 8: Testing using Spectra CX

VI. CONCLUSION

Hence by using the software defined radios in military or in any field we can keep the information confidential.

- Opponent can not hack the information.
- Mainly we can send the information by changing the frequencies and without changing the hardware architecture of the SDR.
- Hence SCA compliance SDRs are very useful in communication purpose in military fields.

REFERENCES

[1]. "Strategies and insights into SCA-compliant waveform application development By Yun Zhang, Scott Dyer, Nick Bulat The MITRE Corporation Bedford, MA".

[2]. "Open-Source SCA Implementation-Embedded and Software Communication Architecture OSSIE and SCA Waveform Development Edoardo Paone 15 February 2010".

[3]. "Achieving SCA Compliance for COTS Software Defined Radio Second Edition By Robert Sgandurra Product Manager, Pentek, Inc".

[4]. "Software Defined Radio, SDR, Tutorial - tutorial and information about the basics of the software defined radio, SDR, and links of software defined radios to JTRS, and general SDR receiver technology".