Int. j. inf. tecnol. (March 2017) 9(1):41–48
DOI 10.1007/s41870-017-0007-x

ORIGINAL RESEARCH

General framework for testing using formal languages

S. Lobato[1] · **J. Poncela**[1] · **M. Aamir**[2]

Published online: 22 February 2017

Abstract Currently, when a technology is defined (Bluetooth, GSM, UMTS or LTE), it is mandatory to think about an infrastructure with an abstract test suite, codecs, timeframes, etc. that will support the design of a new testing architecture. If, in addition, the testing methodology used is for different purposes, such as protocol conformance, radio conformance, radio performance or network conformance, the result is a redundant effort in the design and development of different test systems, increasing costs, development time and reducing reusability. In this paper it is shown how a framework for formal languages such as TTCN can be realized in a common adapter layer capable of sharing a model for any technology and testing methodology, reducing costs, improving time and avoiding redundant designing efforts.

Keywords Testing methodology · Radio conformance testing · Protocol conformance testing · Radio performance testing · Network conformance testing · Test architecture · TTCN-3

1 Introduction

Telecommunication technologies, and especially wireless technologies, have evolved very quickly along the last two decades, from DECT to LTE, going through GSM, Bluetooth, Wi-Fi, WiMAX, UMTS or CDMA2000. All of these technologies require a complex process of testing, such that final user can be guaranteed of the correctness and quality of their systems.

As new technologies rise up, they require a set of testing methodologies to cover conformance and performance testing of equipment. These include mobile devices, base stations or security nodes (for example, implementing AAA on Radius and Diameter). We face a world of complexity that evolves faster and faster in the wireless market, with requirements of more functionality, multimedia access and where security and privacy is a "must". So, Protocol Conformance Testers, Radio Conformance Testers, Radio Performance Testers and Network Performance Testers become a need. Testing may also target distributed systems or applications, expected to run on open non-standardized environments, such as those described in [1, 2].

The development and integration of any kind of technology and any test methodology represents a huge effort for testing vendors and measurement equipment manufacturers. Test systems are required different for test purposes, such as measurement of instruments, automation control of samples, testing of user equipment, etc. All these factors mean a new product life cycle, design, building, integration and maintenance for each technology and test need.

Test systems are different to testbeds. While a test system emulates the behaviour of part of the network and/or other nodes against which a device or part of a device is verified, a testbed, such as those used for ad-hoc networks [3] or replication strategies [4], is an environment in which to deploy an implementation and analyse its behaviour.

Although different technologies and different testing methodologies are on the table, all of them can be based on a common architectural framework, which abstracts any common infrastructure and eases the adaptation to the specific features of any case.

✉ J. Poncela
jponcela@uma.es

[1] ETSI Telecomunicación, Universidad de Málaga, Málaga, Spain

[2] Sir Syed University of Engineering and Technology, Karachi, Pakistan

In the same way as quality is required for the system, the test system itself must be validated with regards to its compliance with the standards defined for each technology and, furthermore, for the test procedure and the test methodology that the test system is defined for. In this sense, the use of testing formal languages, such as TTCN, is a good concept from the point of view of computer semantic standardization. Protocol testing is the area where a higher level of formalization has been achieved [5]. Radio tests can also be written in formal TTCN language, thus removing likely ambiguities [6].

In this paper, we describe a common framework for a general purpose test system, and how we can generalize the definition of particular technologies and testing methods. Once the common framework has been defined and validated, any technology and testing methodology may add their own particularity into the test system (codecs, measurement equipment, ports, etc.) for a specific test purpose.

This paper is organized as follows. First, the standardized conformance testing methodology is briefly described. Section 3 presents a general architecture for test systems. Section 4 describes how the architecture can be tailored for different types of test targets. Section 5 comments on implementation issues. Finally, some conclusions are presented.

2 Conformance testing methodology

The conformance methodology for testing was defined by ITU in 1998 [7]. It has undergone several updates so that it could benefit of a larger formalization, and has incorporated the testing of distributed protocols. The European Standardization Institute (ETSI) has been a central part of this evolution, and has pushed forward the use of the TTCN-3 language [8]. A detailed guide of conformance testing has been published in [9].

TTCN-3 is in fact a set of standards: Core Language, Tabular Presentation Format, Graphical Presentation Format, Operational Semantics, TTCN-3 Runtime Interface (TRI) y TTCN-3 Control Interface (TCI). It constitutes a test language that allows for the specification, execution and automation of test cases for distributed systems.

The testing methodology defines four types of tests, which are: Basic interconnection tests, which check that the main functionalities are implemented and, at the same time, verify that connectivity is possible; Capability tests, that are used to verify external static capabilities that can be observed; Behaviour tests, that verify the dynamic conformance of equipment to the implementation; and Conformance resolution tests, which carry out deeper verifications. Test cases have three valid outcomes: PASS, FAIL or INCONC. Test cases are grouped in Abstract Test Suites (ATSs).

The number of test cases may grow very fast if too many situations are considered. To avoid excessively long test runs at the certification laboratory, a selection must be carried out. Techniques such as those described in [10] may be of help in this task.

Each testing process may employ one or more different test configurations. The designer will choose the most adequate configuration depending on the level and type of coordination between the Upper Tester (UT) and the Lower Tester (LT) blocks, together with the level of accessibility of the upper Implementation under Test (IUT) boundary. In radio tests, the name Equipment under Test (EUT) is used instead of (IUT). The Point of Control and Observation (PCO) is an abstract interface where the IUT can be stimulated and responses can be inspected. Figure 1 shows the test configuration for the remote testing method.

3 Test system architecture

A generalized architecture suitable for any test method is based on, at least, four aspects. First, a Signalling plane, to emulate the node where the Equipment under Test will try to connect. Second, the automation control of the EUT, since it doesn't need of a manual operator (i.e. AT and MMI commands for 3GPP samples). Third, the measurement instrumentation, such as spectrum analyzers, signal generators, etc., and the necessity of communication with them to obtain information or to configure any measurement. Finally, we must consider the communication with an external server, like AAA nodes are used to facilitate security in the test measurement.

Each of these components is a subsystem external to the Test System (see Fig. 2). The relationship among them will depend on the presence or not of such components. Excepting the EUT and the Signalling Unit, which are considered as mandatory, the other components, External Servers and Measurement Instruments, are optional, and their presence is based on the test method. In any case, the Test System must be prepared for them to be connected. In

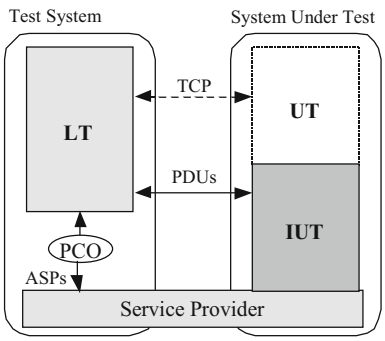


Fig. 1 Remote test method

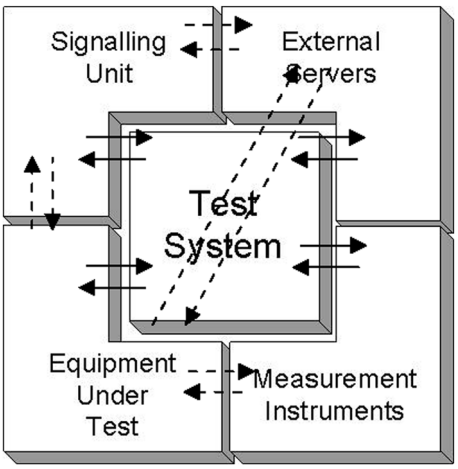


Fig. 2 Test architecture schema

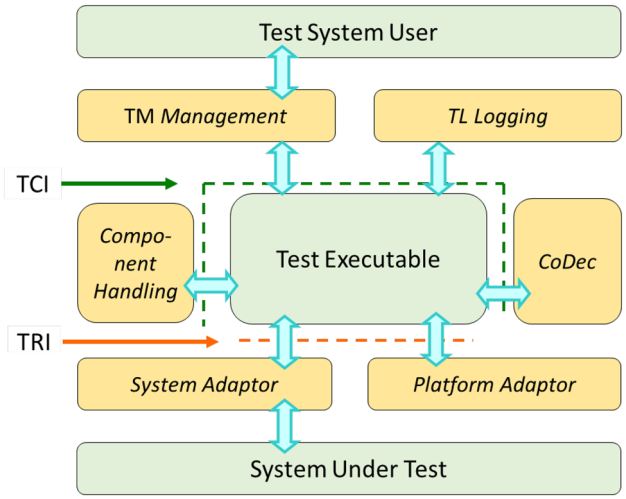


Fig. 3 Architecture of a TTCN-3 test system

addition, each subsystem has to be able to interoperate with any other, in special with the EUT. So, the Signalling Unit will emulate the network or other equipment in order to interoperate with the Equipment Under Test; external servers will be used by the EUT or the Signalling Unit, i.e. to check certificates or for security and encryption.

The implementation of a Test System requires the construction of several modules, which adapt the abstract test cases to a specific software and hardware platform. Figure 3 highlights, in yellow, those modules that must be provided by the manufacturer. Globally, these components are called the adapter layer. The SUT Adapter (SA) manages all communication aspects with the system being tested; the Platform Adapter (PA) has special interest from the point of view of timing; the Codec (CD) module provides encoding/decoding functionality for messages. Other components are the Test Manager (TM), the Test Logging (TL) and the Component Handling (CH), but these are not in the scope of this paper.

The Test Executable (TE) block is the heart of the system, and it is in charge of the normal flow of the test case. This block is provided in the standards and includes the test cases in formal notation TTCN. Interfaces with standardized functions are defined around this module. The TTCN-3 Control Interface (TCI) [11] is used to select one or more test cases and control the test execution. The TTCN-3 Runtime Interface (TRI) [12] provides support functionalities for time management and message communication.

Due to testing purposes, and in order to work in concurrent environment, where any component or subsystem is an independent element, the policy adopted in the TE is using a Parallel Test Component (PTC) for each subsystem. It means that each parallel component is a logical entity, an abstract model of the real implementation of the instrumentation and the subsystem involved in the test.

This requires of a model for each instrumentation, and consequently the SUT Adapter must implement any communication driver supported by the different subsystems (LAN TCP, RS-232, USB, GPIB, etc.). To achieve this goal, it will need to implement the entire physical drivers commonly used by all the architecture components. For each physical port the SUT Adapter will work independently from the point of view of connectivity. This implies that each of these ports will work in a different thread. Table 1 shows the common interfaces that are supported by measurement instrumentation.

Actually, most of the SUT Adapters base their timers on the CPU clock, checking its value to discount from the current time. One of the main problem is that TTCN is not a real-time system, so if brief timers, smaller that milliseconds, need to be considered, the general framework must include some extended functionality to handle these events, and process such timers off-line.

For such purpose, we introduce a new concept, the "virtual timers". A timer is considered as virtual when it is not based in the same clock than the system. An example of this concept is the case of measuring the number of packets received within a small interval of time. It may happen that the interval where hundreds of receptions must be done is smaller than the minimum latency supported by TTCN. In this case, the Signalling unit must add in the interface with TTCN a header with the instant when each packet was received, so, the virtual timer is based on the header of the packets rather than the CPU clock.

The CoDec is usually the most complex and difficult module to develop; recursion and formal specification of the message syntax together with automatic codec generation tools helps to ease the development effort.

Table 1 Common interfaces in measurement instrumentation

Instrument	LAN client	RS-232	GPIB	USB
Signalling unit	X	X	X	X
EUT	X	X		X
Spectrum analyzer	X	X	X	X
Signal generator	X	X	X	X
Power meter	X	X	X	X
AAA server	X	X		

4 Tester architecture configurations

Once we have the basis of the architecture of the test platform, we will apply this to the different testing targets, such as Protocol Conformance Tester, Network Conformance Tester, Radio Conformance Tester and finally Radio Performance Tester. Figure 4 shows partial architectures for these test methodologies.

The architecture for the Protocol Conformance Tester is the simplest of all, as it is based just on the Test System, the Signalling Unit,[1] and the Equipment Under Test (see Fig. 4a). The goal of the protocol conformance tests is checking all the signalling procedures for a specific test purpose, verifying the EUT compliance with the standards.

The Signalling Unit will be able to emulate the signalling node and will emulate a valid or an invalid behavior depending on the test purpose. It is the responsibility of the Test System to configure such device providing the functionality and configuration needed for each test purpose.

Once the Signalling Unit is configured, the EUT can be accessed, and then the Test System will configure the device by means of specific commands or, if an automated mechanism is not available, manually by means of the testing operator. The Test System will send and receive information related to the test purpose, will check timers to study the latency of the procedures and finally will provide a verdict.

The next level of complexity corresponds to the Network Conformance Tester (NCT). The EUT is tested from the point of view of the Network. In this test method, it will be necessary to support not only the signalling standardization but also a set of features and skills that allow the mobile devices to interoperate with the network, i.e. DHCP, ICMP, security and cryptographic capabilities to interchange certificates and keys, etc. Sometimes commercial testers need external servers to emulate security nodes (AAA) based on standardized protocols, such as RADIUS. Components involved in the architecture are shown in the Fig. 4b.

[1] Depending on the test method, the Signalling Unit cam be considered as additional measurement equipment.

Radio Conformance Testers must control and communicate with not only Signalling Units (one or several of them, i.e. to emulate several interferers or for RRM testing), but also control measurement instrumentation (see Fig. 4c) such as vector spectrum analyzers, signal generators, IQ modulators, power meters, switching units, etc.

As Network Conformance Testers extend the architecture of Protocol Conformance Testers, Radio Performance Testers extend the architecture of Radio Conformance Testers. However, in this case the extension consists on the management of the pointing of the antennas and their polarization in an anechoic chamber from the Test System, following the same architecture shown in Fig. 4c.

5 Implementation

Whenever a testing manufacturer begins the design of a test system for a new technology and for any test method, it is necessary to define some particular aspects in the adapter layer. The implementation of each part of the TTCN-3 Adapter components can be designed in such a way that they can be reused in any test method and technology.

5.1 Virtual timers

From the point of view of TRI interface, we will first study the design and implementation of timers. Nowadays, timers might be considered the easiest point in an adapter, and they are supposed to be reused at any tester, but the new concept of "virtual timers" introduces more complexity in order to achieve a general framework.

Virtual timers are defined as timers which are not based on the CPU clock of the computer where the test is running, but they are defined in different timestamps from other systems (Signaling Units, Servers, etc.). These timestamps are wrapped within a PDU or ASP, so when the TTCN Runtime Interface (TRI) receives any information from any port, when present timestamps are extracted and provided to the Platform Adaptor.

The implementation proposed requires of a configuration file where the test designer includes which of the timers declared in TTCN-3 are defined as virtual timer. This configuration file has also information related to the source of the timestamp; so, if a timestamp comes from a Signaling Unit, only virtual timers configured for such instrument tick will consider this timestamp to update their time values. The rest of virtual timers wouldn't be updated until a timestamp from the configured source arrives. Figure 5 shows a visual example of how the Platform Adaptor handles all TTCN-3 timers.

Any timer not defined as virtual, updates its time value using the main time reference. This main reference is,

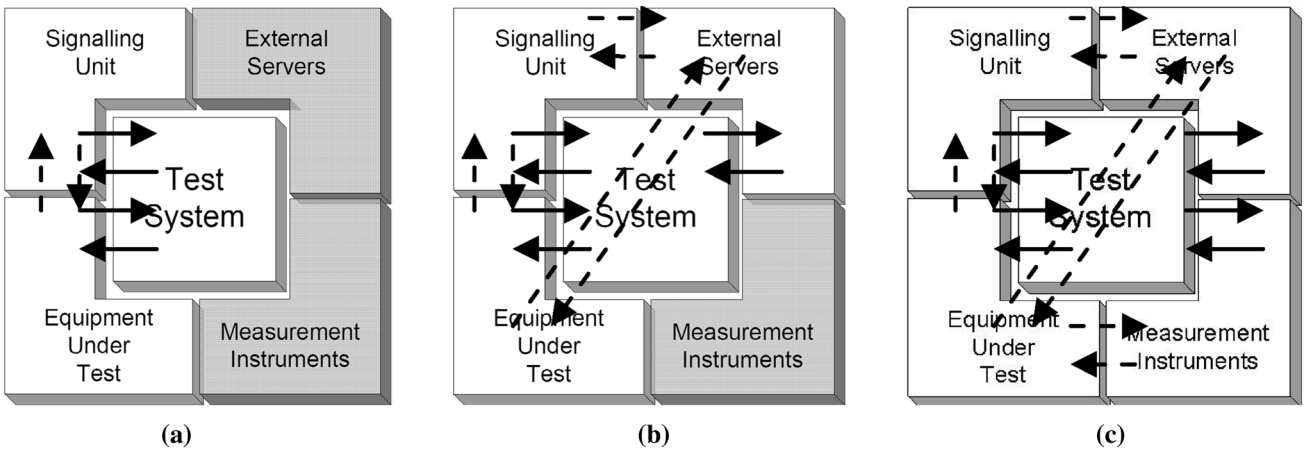


Fig. 4 Partial architecture schemas: **a** Protocol Conformance Tester; **b** Network Conformance Tester; **c** Radio Conformance and Performance Tester

usually, the CPU clock, but if more accuracy is needed, an external GPS reference signal might be used.

5.2 Instrumentation

TTCN-3 is commonly used for protocol testing; this implies that usually the System Adaptor implements LAN connections (TCP or UDP). This paper tries to provide the next stage of this language, as it is using TTCN-3 for any types of tests, not only for protocol conformance testing, but also network conformance, and overall, Radio, conformance and performance. We, thus, try to open this standard to be used for the control and management of measurement equipment.

If a tester wants to use TTCN-3 to control measurement instrumentation, it needs to include in its System Adaptor a set of drivers for different buses and interfaces, as indicated in Table 1.

In the first step of the initialization, TSI ports are loaded, then, for this general architecture, it is needed to define what kind of bus is used by each port. The proposed implementation requires a configuration file where at least are defined LAN TCP, LAN UDP, for sockets, USB for RAW or specific controllers, serial bus RS-232, where parameters such as COM, bauds, parity bits, stop bits, etc. can be configured, and, finally, GPIB, where several GPIB cards are accepted and in the same card several GPIB instruments can be controlled. In addition, the system provides the option of "No Connection" for those test cases that, although TTCN-3 is mapping the port, do not need to create any physical connection, i.e. manual instead of automatic configuration of EUT. It only requires creating a pop-up or any screen message to report the user to configure the EUT. The system is open to adding new drivers as need arise.

Figure 6 shows a us case of a Radio Conformance Tester configuration, where the Main Test Controller (MTC) controls the EUT and a PTC is defined for each equipment. Although the TSI is prepared to connect equipment, such as a signal generator, in the example shown it is not necessary, so there's no mapping for such port.

Once the System Adaptor is initialized and the TSI port information is loaded from the TRI interface and ready for connection using the driver indicated in the configuration file, the System Adaptor waits for the *triExecuteTestCase* function. When called, all the component ports are created in a mapping structure. When a TTCN-3 map sentence is executed in the Test Executable, the System Adaptor links the port of the component to the TSI port, and the real connection from the TSI port to the external component (EUT, Signaling Unit, External Server or Measurement Instrument) is realized.

The communication with the EUT is based on the protocol implemented by the EUT manufacturer, i.e. AT and MMI commands in the case of 3G and LTE mobiles. With external servers, it consists in a protocol used for security, i.e. RADIUS, and for measurement equipment, it is based on SCPI commands and the generic protocol defined in [13]. Figure 7 shows the configuration of the System Adaptor for a Radio Conformance Tester, as well as all the physical connections to external components.

Instrumentation can be modeled using the concept of Virtual Instrumentation (VI). A Virtual Instrumentation is a component which can carry out one or more measurements, where each may be configured with zero or more parameters and has one or more I/O connectors. Commands used with VIs can be standardized, for example, following the set of actions defined by the SCPI standard [14]. Abstracting the instrumentation would increase the

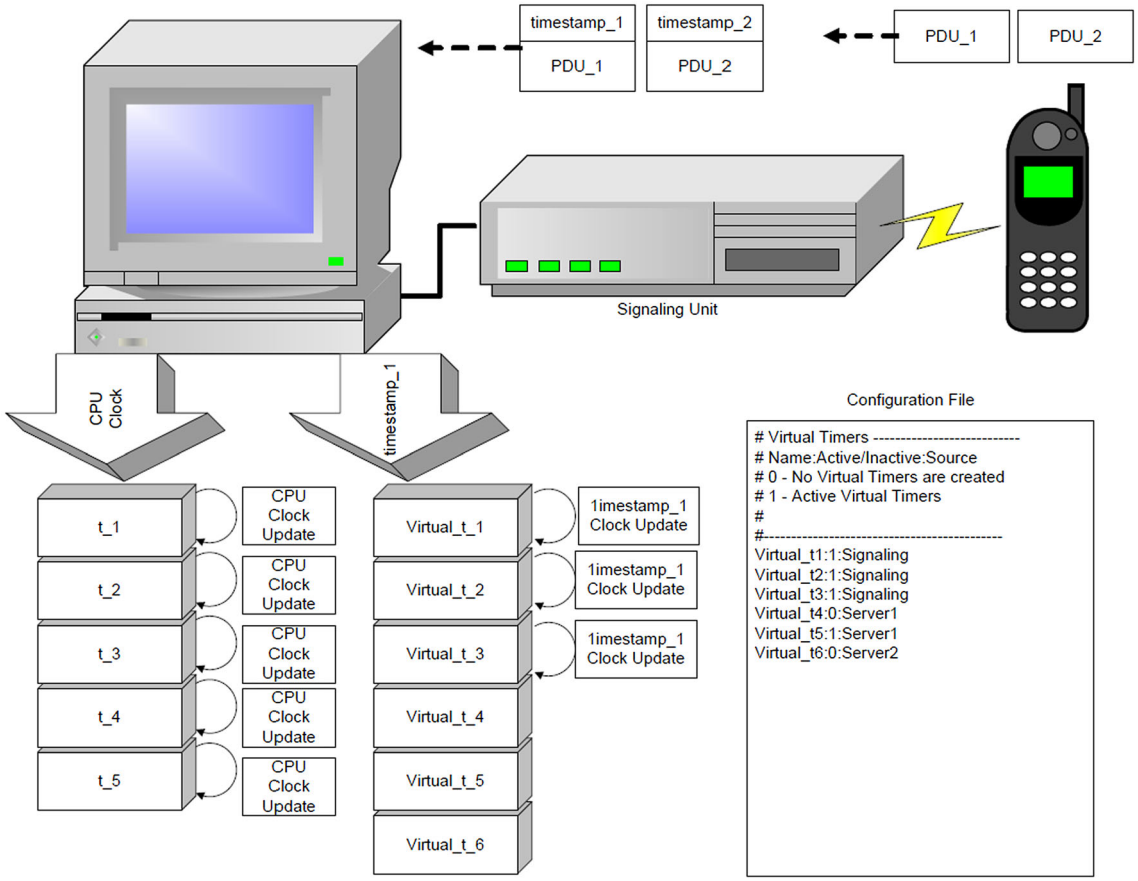


Fig. 5 Timing management by the platform adaptor

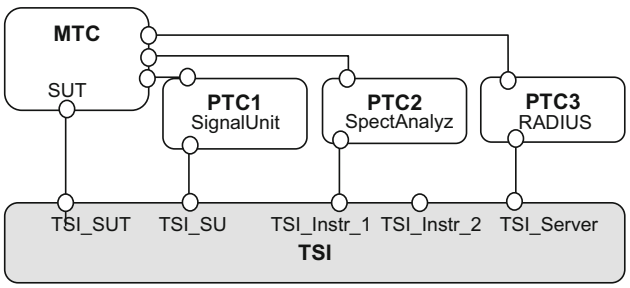


Fig. 6 TTCN-3 test configuration

formalization of radio test cases, and simplify the implementation of test systems.

5.3 Encoding messages

At this point of the implementation, the architecture is capable of handling real and virtual timers and connections can be opened for each part of the system using any type of driver. The next step preparing the codecs.

The TTCN Control Interface for the CoDec module has to be designed so it is capable of working with any technology and any test method. To achieve this target, the implementation uses a recursive algorithm to encode and decode messages.

From the TTCN-3 code, where ports are defined as *in* or *out* ports, we can extract all the relationships among the various message types, from the port definition up to the basic types. This information is represented into a tree of types, and stored as a table where it is indicated the name of the type, the type (predefined, user defined or basic types), the number of elements in the case of structured types and the subtypes which contain such "father" type.

Using this definition, to decode a data stream the system only needs to begin from the hypothesis of the initial type of the port and navigate recursively in the table. Initially, this recursive algorithm has some limitation with some such as set or set of, especially when these contain optional subtypes, but this limitation is easy to work out if upon reception the data stream is pre-studied to check which of the subtypes are present and their ordering.

Figure 8 shows a simple example of a record type (*gmmStatus*) and how the types are saved in a table used by means of recursive algorithms for decoding the stream.

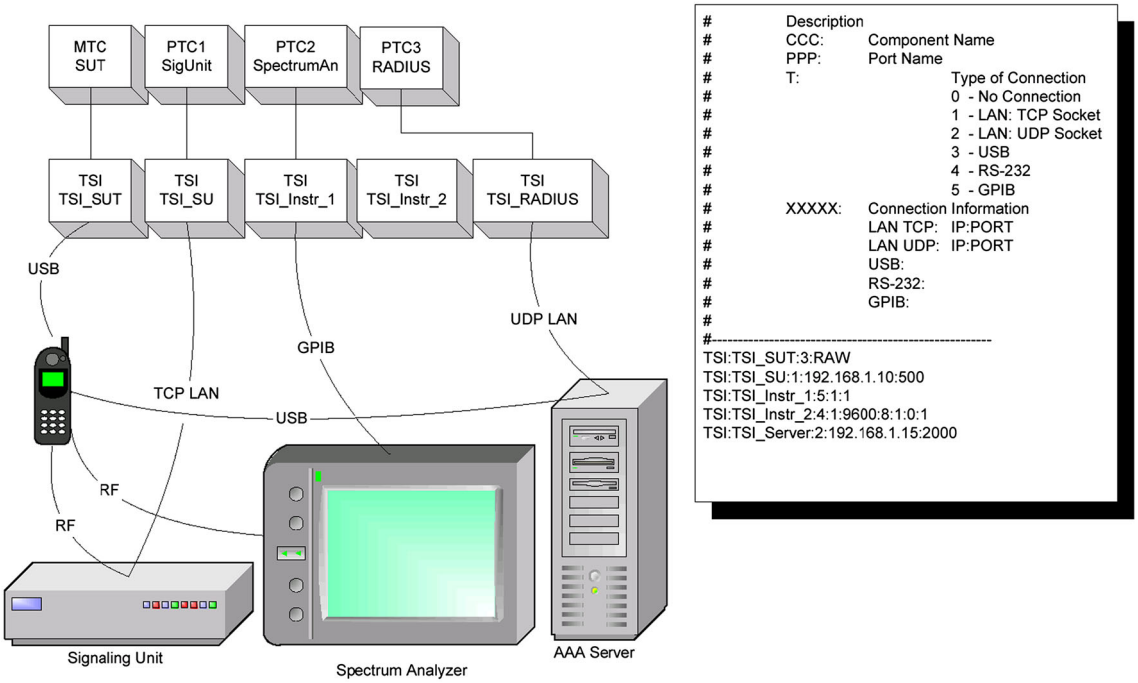
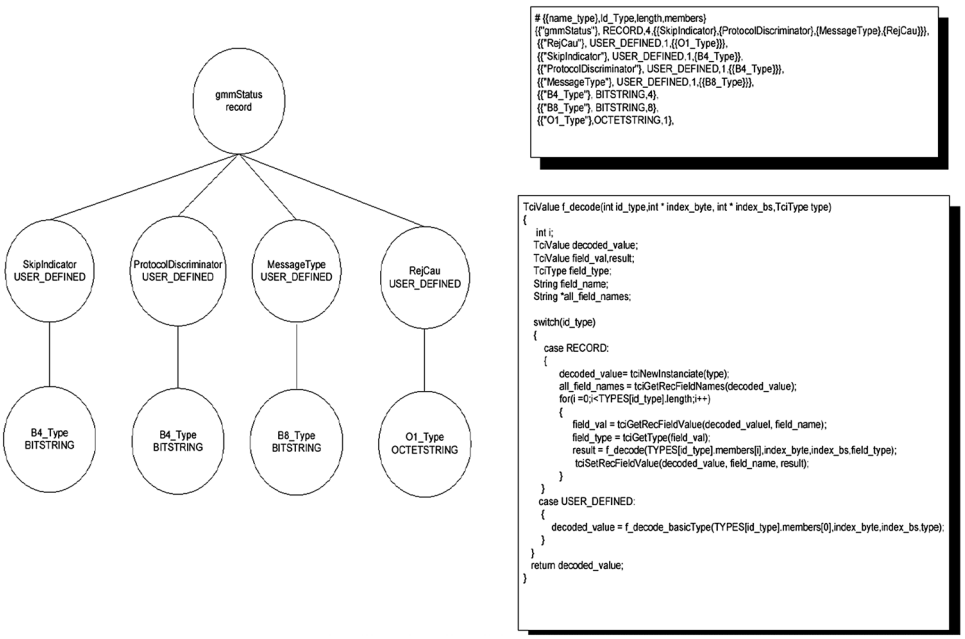


Fig. 7 System adaptor configuration

Fig. 8 Implementation of the CoDec module



6 Conclusions

In this paper it has been shown how to create a general architecture using formal languages for different test methods, and, specially, for different technologies. The first result is the use of a common language for the test definition, which initially was used only for protocol testing. Secondly, we have defined a communication mechanism, as the basis for any testing procedure, capable not only of sending and receiving protocol messages but also of communicating with different measurement instrumentation, authentication servers and of configuring the EUT. The proposed architecture also manages virtual timers, and uses global codecs suitable for any technology.

The result of this architecture is a general framework of formal languages, which saves time and efforts for the hard

process of designing and integrating a new technology or a new test method in a test system. Using formal languages for radio test specifications would increase their quality and provide the means for using a combined test platform for protocol and radio tests, as described in this paper.

References

1. Patel RB, Anu (2009) A mobile transaction system for open networks. BVICAM's Int J Inf Technol 1(1):67–78
2. Krishna Mohan K, Srividya A, Verma AK, Kumar Gedela R (2009) Process centric development to improve quality of service (QoS) in building distributed applications. BVICAM's Int J Inf Technol 1(1):43–54
3. Hasti A (2012) Study of impact of mobile ad-hoc networking and its future applications. BVICAM's Int J Inf Technol 4:439–444
4. Abdul Moiz R, Rajamani L (2010) Replication strategies in mobile environments. BVICAM's Int J Inf Technol 2:170–173
5. Tretmans J (2001) An overview of OSI conformance testing, formal methods and tools group. University of Twente (Translated and adapted from: Tretmans J, van de Lagemaat J, Conformance Testen. In: Handboek Telematica, vol II, pp 4400 1–19. Samson)
6. Kegley K, Stavridou V (1999) The role of formal methods in software standards. In: Fourth IEEE international symposium and forum on software engineering standards
7. ITU-T X.290-X.296 (1998) OSI conformance testing methodology and framework for protocol recommendations for ITU-T Applications
8. ETSI ES 201 873 (2007) Methods for testing and specification (MTS). The Testing and Test Control Notation version 3, v 3.2.1
9. ETR 021 (1991) Advanced testing methods (ATM). Tutorial on protocol conformance testing
10. Jain R, Agarwal N (2013) Additional fault detection test case prioritization. BVICAM's Int J Inf Technol 5(1):623–629
11. ES 201 873-6 ETSI TTCN-3 (2011) TTCN-3 Control interface
12. ES 201 873-5 ETSI, TTCN-3 (2011) TTCN-3 runtime interface
13. IEEE Std. 488.1 (2003) IEEE standard for higher performance protocol for the standard digital interface for programmable instrumentation
14. Poncela J, Entrambasaguas JT, Aguayo MC (2010) Advancing radio testing methodology via formal notations. Wirel Pers Commun 53:409–429