Lecture Notes in Computer Science 4745

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany
SDL 2007: Design for Dependable Systems

13th International SDL Forum
Paris, France, September 18-21, 2007
Proceedings
# Table of Contents

## Model Driven Engineering

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Model-Based Standard for SDL</td>
<td>1</td>
</tr>
<tr>
<td>Andreas Prinz, Markus Scheidgen, and Merete S. Tveit</td>
<td></td>
</tr>
<tr>
<td>Model Driven Development and Code Generation: An Automotive Case Study</td>
<td>19</td>
</tr>
<tr>
<td>Michele Banci, Alessandro Fantechi, Stefania Gnesi, and Giovanni Lombardi</td>
<td></td>
</tr>
<tr>
<td>Experiences in Deploying Model-Driven Engineering</td>
<td>35</td>
</tr>
<tr>
<td>Thomas Weigert, Frank Weil, Kevin Marth, Paul Baker, Clive Jervis, Paul Dietz, Yexuan Gui, Aswin van den Berg, Kim Fleer, David Nelson, Michael Wells, and Brian Mastenbrook</td>
<td></td>
</tr>
</tbody>
</table>

## Testing

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTCN-3 Quality Engineering: Using Learning Techniques to Evaluate Metric Sets</td>
<td>54</td>
</tr>
<tr>
<td>Edith Werner, Jens Grabowski, Helmut Neukirchen, Nils Röttger, Stephan Waack, and Benjamin Zeiss</td>
<td></td>
</tr>
<tr>
<td>Using TTCN for Radio Conformance Test Systems</td>
<td>69</td>
</tr>
<tr>
<td>Javier Poncela-González, Juan Gómez-Salvador, Carlos Valero-Roldán, and Unai Fernández-Plazaola</td>
<td></td>
</tr>
<tr>
<td>Testing UML2.0 Models Using TTCN-3 and the UML2.0 Testing Profile</td>
<td>86</td>
</tr>
<tr>
<td>Paul Baker and Clive Jervis</td>
<td></td>
</tr>
</tbody>
</table>

## Language Extensions

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifying Input Port Bounds in SDL</td>
<td>101</td>
</tr>
<tr>
<td>Reinhard Gotzhein, Rüdiger Grammes, and Thomas Kuhn</td>
<td></td>
</tr>
<tr>
<td>Translatable Finite State Time Machine</td>
<td>117</td>
</tr>
<tr>
<td>Krzysztof Sacha</td>
<td></td>
</tr>
<tr>
<td>Enhanced Use Case Map Traversal Semantics</td>
<td>133</td>
</tr>
<tr>
<td>Jason Kealey and Daniel Amyot</td>
<td></td>
</tr>
</tbody>
</table>
Implementation

Automated Generation of Micro Protocol Descriptions from SDL Design Specifications .................................................. 150  
Ingmar Fliege and Reinhard Gotzhein

Synthesizing Components with Sessions from Collaboration-Oriented Service Specifications .................................................. 166  
Frank Alexander Kraemer, Rolv Bræk, and Peter Herrmann

Experiences in Using the SOMT Method to Support the Design and Implementation of a Network Simulator .................. 186  
Manuel Rodríguez and José María Parra

Modeling Experience and Extensions

Consistency of UML/SPT Models ........................................... 203  
Abdelouahed Gherbi and Ferhat Khendek

Formal Verification of Use Case Maps with Real Time Extensions ........ 225  
Jameleddine Hassine, Juergen Rilling, and Rachida Dssouli

Guillaume Châtelet, Benoit Parreaux, and Yves-Marie Quemener

OpenComRTOS: An Ultra-Small Network Centric Embedded RTOS Designed Using Formal Modeling ............................ 258  
Eric Verhulst and Gjalt de Jong

SDL Design and Performance Evaluation of a Mobility Management Technique for 3GPP LTE Systems .......................... 272  
Tae-Hyong Kim, Qi-Ping Yang, Soon-Gi Park, and Yeun-Seung Shin

Author Index ................................................................................. 289
Using TTCN for Radio Conformance Test Systems

Javier Poncela-González, Juan Gómez-Salvador, Carlos Valero-Roldán, and Unai Fernández-Plazaola

ETSI Telecomunicación, Campus de Teatinos, s/n, 29071 Malaga, Spain
{javier,unai}@ic.uma.es

Abstract. While protocol conformance testing methodology is a well formalized field, radio testing methodology still relies on natural language specifications. This paper proposes an improvement on the quality of radio test specifications via the use of formal notation TTCN. This approach, and the fact that protocol and radio conformance testing share most of the underlying concepts, enables the use of a generic architecture for implementations of both types of testers, resulting in a reduction of the development efforts. This architecture has been validated with the implementation of radio test cases for the UMTS technology.

1 Introduction

Since its beginning, the effort in the field of conformance testing has been mainly centered in the area of protocol testing, where the testing methodology has attained a high level of formalization [1]. However, radio conformance testing has not evolved as much further. Radio test specifications are still provided only in natural language making them prone to ambiguous interpretations.

Traditionally, the fields of protocol and radio conformance testing have been considered as distant worlds, as the engineering groups related to each of them usually have non-overlapping backgrounds. While the protocol tests are focused on checking that sequence of messages exchanged between peer entities is performed in the correct order and the proper syntax, the aim of the radio tests is to certify the compliance of an implementation in aspects such as transmission and reception compatibility with other equipment. Nevertheless, most of the concepts are shared in both areas. For example, the set of documents internationally standardized for testing is nearly the same, as well as the documentation handled and provided by test laboratories.

In this paper we propose that TTCN is used to model radio test cases which would enhance the radio test specifications. Nowadays, these specifications are provided in natural language. Its quality could be improved by formalizing the test procedure with the use of TTCN, thus avoiding possible ambiguities and increasing its consistency [2]. With this approach it is possible to use the same architecture for radio and protocol test systems. One example of this architecture for protocol test systems, implemented using ITU description languages (SDL,
TTCN, ASN.1), can be found in [3]. A common architecture for both types of conformance tests can be obtained, which results in a reduction of the time and cost of the development, as some of the internal modules can be reutilized as well as the operator interface.

This paper is structured as follows. Section 2 gives an overview of the existing methodologies for protocol and radio conformance testing. Section 3 briefly comments the radio conformance test procedures. A proposal for a common test system architecture for protocol and radio test systems is proposed in Sect. 4. A detailed description of the main characteristics of this architecture for radio test systems is provided in Sect. 5. The architecture has been applied to the conformance testing of UMTS systems. A brief description of its radio test specification together with an example of application are presented in Sect. 6.

2 Conformance Testing Methodology

Nowadays, the conformance testing methodology is a well understood field. Firstly compiled in the late eighties by ITU [4], this methodology has been reviewed in order to achieve a higher formalization and, at the same time, to tackle distributed protocols testing. ETSI has been deeply involved in the advances of conformance testing, being one of the drivers of the TTCN language [5]. A tutorial, published by ETSI, on the standardized techniques for conformance testing can be found in [6].

Four types of tests can be applied to an implementation:

a) Basic interconnection: to check that main features are implemented and whether interconnection is possible.

b) Capability: to check observable external static capabilities.

c) Behaviour: check the dynamic conformance of the implementation.

d) Conformance resolution: in-depth checking of conformance.

Different configurations, shown in Fig. 1, are considered for the single layer testing process. The test designer will choose the most adequate configuration depending on the level and type of coordination between the UT and LT blocks and the accessibility of the upper IUT boundary. In ratio tests, the name Equipment Under Test (EUT) is used instead of Implementation Under Test (IUT); the term System Under Test (SUT) may be used in any context and encompasses the former two.

Test cases, each with a possible outcome of PASS, FAIL or INCONC, are grouped in Abstract Test Suites (ATSs). Though there could exist one Abstract Test Suite for each Abstract Test Method (ATM), an underlying principle is that only one ATS will be standardized for a given protocol layer.

Compiling a conformance testing standard is a laborious task, which, at the end, must provide one (or several) documents: a) The Test Suite Structure and Test Purposes (TSS&TP); b) One or more Abstract Test Suites (ATSs); c) The Test Management Protocol (TMP) if required. The TSS&TP is provided in natural language, while the ATSs are written in the TTCN formal notation.
When a manufacturer asks a test laboratory for a conformance certificate, several other documents are needed: a) Protocol Implementation Conformance Statement (PICS); b) Protocol Implementation eXtra Information for Testing (PIXIT); c) System Conformance Test Report (SCTR); d) Protocol Conformance Test Report (PCTR). The first two are needed in the test preparation stage; the last two are outputs produced by the test laboratory after the conformance assessment process has been performed.

3 Radio Conformance Testing

The development of conformance testing standards for the radio access of communication systems follows a path quite similar to the one employed for protocols. However, this process lacks the level of formalization achieved with protocols. Specifically, no Abstract Test Suite in formal language is produced, thus stopping the standardization process at the TSS&TP document.
A key difference between radio and protocol conformance tests is that the former require specific equipment capable of carrying out the needed measures in the air interface. Examples of such instrumentation are wave generators, signal modulators, oscilloscopes, spectrum analyzers, etc. Instruments include interfaces to remotely control them, such as RS232, VXI or GPIB. The first one is an all purpose interface, while the latter has been specifically designed for instrumentation control, having a higher flexibility and performance. GPIB [7] is characterized by a parallel bus (8 bits) with a bitrate of 8 MB/s, together with a basic set of high level functions for the equipment management. GPIB can simultaneously control up to 15 devices at high data rates.

At present, ratio test systems have an architecture as shown in Fig. 2 [8]. The Test Case Library block can be seen as scripts that carry out the required actions in order to remotely control the measurement instrumentation and report the outcome of test campaigns. Virtual instrumentation tools, such as LabWindows/CVI or LabView, are widely used to develop such test systems, as they offer an easy and efficient way to code these scripts. Nevertheless, the lack of formality in the test specification may result in different realizations of the same test by two or more test system manufacturers. Note that the communication between the Test System and the EUT can be accomplished as either a conducted or a radiated link. The former is usually implemented using a switching unit.

4 Test System Architecture

Paper [3] describes a methodology that, taking into account the protocol conformance testing methodology, can be used for the implementation of protocol test systems using ITU description languages. The architecture proposed in this paper is shown in Fig. 3-a. Its main components are:

a) Graphical User Interface: controls the test campaign execution and reports and logs its outcome. It’s written in Java to allow for portability to different operating systems.

b) Abstract Test Suite: divided in lower tester and upper tester parts, contains the compiled Test Cases.

c) Lower Subsystem: allows communication between the test system and the IUT. It comprises those layers above the physical layer that the test system must implement as indicated by the corresponding Abstract Test Method.

Looking at the architecture in more detail we can see that some of its parts can be either reused or automatically generated. The GUI can be designed so that it can serve in any protocol test system. Using TTCN C-code generators the Executable Test Suite (ETS) can be automatically generated from the standard TTCN modules. Supporting modules for the Executable Test Suite such as Test

---

1 Originally 1 MB/s, increased to 8 MB/s in 2003.
2 In ratio tests, the name Equipment Under Test (EUT) is used instead of Implementation Under Test (IUT).
Fig. 2. Typical radio test system architecture

Fig. 3. (a) Protocol test system architecture (b) Radio test system architecture
Control, Platform Adapter and I/O can be reused in different test systems; this also happens with the Platform Adapter and I/O modules used by the Lower Subsystem. The Codec modules can be partially reused; the Codec used to communicate the ETS with the Lower Subsystem can be the same in all test systems. Only the specific codec used for peer messages must be generated for each technology under test.

This architecture could also be used in conformance radio testers (see Fig. 3-b) if such test cases were provided in the formal TTCN notation. In this way, code reusing and automatic generation of several blocks would also be achieved in radio test systems. Such an extension would require some changes in several of the blocks of the architecture:

a) Graphical User Interface (GUI): Radio Test Cases usually define a mask or a threshold that the IUT must comply with. Graphically showing the outcome is a useful tool for radio engineers, something not needed in protocol testing.

b) Abstract Test Suite: Its characteristics will be described in detail in Sect. 5.1.

c) Platform Adapter, Test Control, Logging and I/O: No changes would be needed.

d) Codec: As explained above, only the codec used by messages of the technology under test (e.g. UMTS, Bluetooth, WiMax) would be needed.

e) Lower Subsystem: Instead of implementing the behaviour of the lower layers of the protocol stack, it handles the communication with the instruments, hiding specific characteristics implemented by manufacturers. Due to this, it’s more appropriate implementing it in C than in SDL.

f) Equipment: Performs actual measurements as requested by test cases.

An advantage of this architecture is that the components of the test system can be distributed across different platforms without having to modify their implementation. For example, we could think of implementing the lower subsystem in a dedicated card more closely integrated with the equipment, while the ETS and the GUI could run on a specific processor.

The designer could be tempted to merge the Lower Subsystem functionality (see Sect. 5.2 for a full description) into the Executable Test Suite. This would remove the overhead introduced by the Codec modules. However, we must consider that the radio tests are “slow” tests, at least, at the test procedure level. If we accept the overhead by introducing these codecs, it allows us to use the same architecture as for protocol test systems, modularize the design and (as indicated above) distribute the components.

5 Radio Test System Description

This section describes in detail those modules specific to radio test systems. Both the Abstract Test Suite and the Lower Subsystem own particular characteristics that distinguish them from those included in protocol test systems.
5.1 Abstract Test Suite

The main feature of the proposed architecture for radio test systems is the use of TTCN. This language would allow the standardization bodies to provide radio test suites in formal notation, thus leaving out any ambiguity that could arise in the natural language test specifications. At the same time, the validation effort would be reduced. This section describes the decisions taken and the main characteristics of this modelling.

In Sect. 2 the four ATMs defined by the conformance testing methodology have been presented. The main difference among these configurations is how the EUT is controlled during the test execution. There are two options to carry out this control: either automatically configuring the EUT for a test case, or manually. The first option completely automates the test procedure, but radio test specifications do not include a Test Controlling Protocol. Because of this, the second option has been chosen. Thus, the remote ATM (see Fig. 1d) seems to be the most adequate configuration.

As in protocol tests, radio tests cases can be divided in three different sections that deal with the following duties:

a) Preamble: Sets the instrumentation and puts the EUT into the initial conditions demanded by the test case so that the test purpose can be verified.

b) Test body: Carries out the required actions on the EUT and the instrumentation in order to check the test purpose.

c) Postamble: Although this stage does not explicitly appear in radio test specifications, it is required so that both EUT and instrumentation are put back into idle state.

The idle state must be defined so that every preamble starts under the same conditions as there is no fixed order for executing test cases. GPIB instrumentation allows two different states: local and remote. The local state, which does not allow the remote controlling, has been considered as idle.

The communication between the Abstract Test Suite and the instrumentation is made via messages. Each message will request one or several actions, depending on the specific instrumentation used, to be carried out by the Lower Subsystem. The communication must be designed taking into account GPIB bus behaviour and features. We require a confirmation for all messages sent to the instrumentation; it would be possible to only check the correct communication with the instruments at previously defined checkpoints, but we must consider the possibility of an error in the GPIB bus. As the instrumentation is provided by external devices, the probability of errors increases. The logical flow of the test case using the proposed set of messages is depicted in Fig. 4.

For every send event, a timeout is raised if its confirmation message, which can carry a positive or negative result, is not received in due time. If the confirmation is not received, it will mean that a communication error has occurred. The test execution can be stopped just after an error occurs, avoiding unnecessary waits for an INCONC verdict of the test case. This behaviour can be implemented in a parameterizable step in TTCN.
Although the type and number of needed messages may depend on the implementation, we have defined a set of messages that covers all required functionality. In the preamble stage, messages \texttt{INIT\_CONFIG} and \texttt{INIT\_BUS} have been defined. The first one tells the Lower Subsystem to read in the equipment configuration; afterwards, the instrumentation is initialized with the second message, setting them into GPIB remote state. At the same time, message \texttt{CLOSE\_BUS} (used in postambles) will put the instrumentation into idle state.

Three messages have been used for configuring and retrieving data from the instrumentation.

- Message \texttt{SET\_PARAMETER} configures and activates measurements.
- Message \texttt{GET\_PARAMETER} obtains the instantaneous value of the active measurement. The request messages should include an instrument identifier and a command, which indicates the action to be performed by the instrumentation. The response counterparts should include an error code and, for the \texttt{GET\_PARAMETER} message, the measured value.

![Diagram](image-url)
The third message is related to one of the functionalities typically needed in radio tests, the capture of a whole trace from an instrument. The points of the trace can be obtained either in one shot or by issuing a repetitive series of requests. This behaviour should be hidden from the ATS. Consequently, message GET_TRACE has been defined. The Lower Subsystem, which knows the interface with the instruments, will be responsible for implementing the appropriate actions to read in an entire trace.

Finally, one more message has been used for requesting actions from the operator. This message has been called ACTION_REQ and it can carry a text message. It is handled by the Test Control module.

Radio tests usually require an additional processing after having captured a trace by the instrumentation. This processing may require complex mathematical operations. Given that most of these (processing, for example, maximum/minimum search, filtering, bandwidth calculation, demodulation, bit synchronization, power integration) are common to radio test suites for different communication systems, we can think of them as a generic signal processing library that can be reused in different test systems, such as GSM, Bluetooth or UMTS. The functions in this library can be called from the TTCN code and linked at compilation time.

5.2 Lower Subsystem

The Lower Subsystem is the module responsible for the communication between the Abstract Test Suite and the Equipment Under Test. On one hand, it hides the physical communication characteristics; on the other hand, it takes into account possible interface differences as instrumentation can be built by different manufacturers. The implemented Lower Subsystem offers a generic API that can be used by any radio test case that is built using GPIB instrumentation.

One first issue is the type of interface offered by the instrumentation for its remote control. GPIB [7] has been chosen because of its flexibility and performance. The second issue to tackle is the set of remote commands that can be used. In Sect. 5.1 we have shown the messages used in the TTCN test modelling. The Lower Subsystem must map these messages into commands suitable for the specific instrumentation included in the test system. Several aspects must be considered:

a) The test system must be able to integrate instrumentation from different manufacturers.

b) Manufacturers usually only implement a subset of the GPIB standard.

c) Proprietary commands are often included to control capabilities that are specific for the instrument.

d) The same command may have different meanings for each device.

Configuration files can be used to solve these problems. Two types of configuration files have been used. The main configuration file (see Fig. 5) contains, for each instrument, an identifier, specific GPIB information (GPIB card address, GPIB address of the instrument and GPIB signalling mode) and a reference to an equipment configuration file.
# Configuration file for Test System

<table>
<thead>
<tr>
<th>id_equipo</th>
<th>GPIB_board</th>
<th>dir_GPIB</th>
<th>EOT_mode</th>
<th>file</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spec_An</td>
<td>0</td>
<td>20</td>
<td>2</td>
<td>specan.egp</td>
</tr>
<tr>
<td>Wave_Gen</td>
<td>0</td>
<td>29</td>
<td>2</td>
<td>wavgen.egp</td>
</tr>
</tbody>
</table>

**Fig. 5.** Main configuration file

```plaintext
# DEVICE: Spectrum Analyzer (specan.egp)
# Generic command Specific device command
# Non-query commands
Reset *RST
PeakPower CALC1:MARK1:MAX
Span SENS1:FREQ:SPAN
Center SENS1:FREQ:CENT

Trigger TRIG1:SEQ:SOUR
_FreeRun IMM
_Line LINE
_RFPower RFP

# Query commands
PeakPower? CALC1:MARK1:Y?
RefLevel? DISP:WIND1:TRAC1:Y:SCAL:RLEV?
SweepTime? SENS1:SWE:TIME?

Detector? SENS1:DET1:FUNC?
_Average AVER
_Sample SAMP
_Rms RMS

ErrMsg? SYST:ERR?
```

**Fig. 6.** Configuration file for spectrum analyzer FSIQ26 (Rohde&Schwartz)

The equipment configuration files (see Fig.6) will provide the mapping between the command parameter of the primitives and the particular GPIB command for that instrument. This file contains, in one column, the command parameter of the primitives and, in a second column, its translation to the instrument’s GPIB command. When the command requires a response from the instrument then character ‘?’ is added at the end. Such commands are used to retrieve data from the instruments. The commands may carry options that must be also mapped. To distinguish them from the commands, they start with character ‘_’ and appear right after the command itself.

With this mechanism new instruments can replace the existing ones by just defining their corresponding configuration files. Using these new files instead of the old ones will create a new test system with the same functionality as before, but with instrumentation from different manufacturers.
The Lower Subsystem has been written in C. Each message used in the Abstract Test Suite has, in the Lower Subsystem, a routine that implements its expected behaviour. These routines map the commands, hiding the actual implementation of the GPIB commands and can also perform low level error control. Table 1 lists these routines, their parameters and a brief description for each of them. The Instrument data type (see Figure 7) stores the configuration data for each instrumentation device. In case of error, all these routines return an error code and a description in parameter error. The Commands field of the structure holds the list of available commands for this instrument and the parameters that each command can carry.

When a message is received, the Lower Subsystem checks its type and then codifies the corresponding equipment commands, with the appropriate parameters as indicated in the message, looking at the translation table generated from the configuration files. The commands are executed sequentially and each result read from the bus and stored. When all the commands associated with the message have been executed, a response is sent to the Executable Test Suite.

6 UMTS Radio Test System

UMTS is one of the mobile communication systems that are part of the 3G family. This technology provides multiple, simultaneous and flexible connections with bitrates from 64 kbit/s up to 2 Mbit/s, worldwide roaming, security and negotiated QoS according to the user needs. It is expected that, with such speeds, services with high bandwidth demand such as multimedia services can be provided in a mobile environment.

The radio access technology is Direct Sequence CMDA (DS-CDMA), commonly referred to as WCDMA because bands are 5 MHz wide. The main characteristics of this technology are its robustness against interferences, its spectral efficiency, frequency reuse and flexible data rates. There are two working modes in WCDMA: Frequency Division Duplex (FDD), where one carrier is used for uplink and another carrier (separated 5 MHz) is used for downlink; and Time Division Duplex (TDD), where some slots in the carrier are used for uplink while other slots are used for downlink.

```c
typedef struct
{
    char id_instr[L_ID_INSTR];
    int gpib_board;
    short dir_gpib;
    int eot_mode;
    char arch_instr[L_ARCH];
    Commands *coms;
    void *sig;
} Instrument;
```

*Fig. 7* Instrument data type

---

3 Execution of commands is aborted if any of them is not successful. In this case, the error is read and reported back.
Table 1. Routines implemented in the Lower Subsystem

<table>
<thead>
<tr>
<th>Function declaration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int InitConfig (Instrument *instr, char *err)</td>
<td>Reads the configuration files. Must be called at the beginning of the test case.</td>
</tr>
<tr>
<td>int InitBus (Instrument *instr, char *err);</td>
<td>Sets the equipment into remote mode.</td>
</tr>
<tr>
<td>int SetParameter (Instrument *instr, char *id, char *com_gen, char *par_gen, char *err);</td>
<td>Executes the GPIB commands that do not require a response by the equipment.</td>
</tr>
<tr>
<td>int GetParameter (Instrument *instr, char *id, char *com_gen, char *par_gen, char *valor_dev_str, int l_valor_dev_str, char *err);</td>
<td>Executes the GPIB commands that require a response by the equipment.</td>
</tr>
<tr>
<td>int GetTrace (Instrument *instr, char *id, char *tx, char *ty, char *err)</td>
<td>Acquires the measurement trace.</td>
</tr>
<tr>
<td>int CloseBus (Instrument *instr, char *err);</td>
<td>Sets back the equipment into local mode; the front panel controls can be used again.</td>
</tr>
</tbody>
</table>

The radio test cases [10] are classified in four groups: transmitter characteristics (17), receiver characteristics (7), performance requirements (14) and requirements for support of Radio Resource Management (38). The list of transmitter test cases is shown in Table 2 which indicates the type of measurement that must be performed for each test case.

6.1 Example of Test Case Implementation

All test cases in the transmitter group have been implemented. As an example, we will describe the implementation of test case 5.9 (see Table 2), called Spectrum Emission Mask for the FDD variant. The purpose of this test case is to verify that the power of the User Equipment (UE) emission does not exceed the prescribed limits specified in the standard. The UE output power is measured at different offsets and compared with a reference emission mask.

The initial test conditions require that the UE is entered into loopback mode after a call has been setup (as described in [11], [12]). The test procedure must perform the following steps:

a) Set and send continuously Up power control commands to the UE until the UE output power is at maximum level.

b) Measure the power of the transmitted signal with a measurement filter as described in the standard. The centre frequency of the filter is stepped in contiguous steps and the measured power recorded for each step. The bandwidth of the filter is 30 kHz or 50 MHz, depending on the offset from the carrier centre frequency.

c) Calculate the ratio of the measured power compared to the reference power mask.
### Table 2. List of transmitter test cases for UMTS

<table>
<thead>
<tr>
<th>Test</th>
<th>Measurement</th>
<th>Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS 34.121</td>
<td>Test Name</td>
<td>TS 34.121</td>
</tr>
<tr>
<td>5.2</td>
<td>Maximum Output Power</td>
<td>Power level</td>
</tr>
<tr>
<td>5.3</td>
<td>Frequency Error</td>
<td>Frequency</td>
</tr>
<tr>
<td>5.4</td>
<td>Output Power Dynamics in the UL</td>
<td>&lt;group&gt;</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Open Loop Power Control in the UL</td>
<td>Power level</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Inner Loop Power Control in the UL</td>
<td>Power level</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Minimum Output Power</td>
<td>Power level</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Out-of-Sync Handling of Output Power</td>
<td>Power level</td>
</tr>
<tr>
<td>5.5</td>
<td>Transmit ON/OFF Power</td>
<td>&lt;group&gt;</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Transmit OFF Power</td>
<td>Power level</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Transmit ON/OFF Time Mask</td>
<td>Power level</td>
</tr>
</tbody>
</table>

### Test Case Name
TC_TRM08_SpecEmissMask

### Group
TRM/TC_TRM08/

### Purpose
Verificar que la mascara espectral de emision se cumple para distintas varaciones de la frecuencia portadora, tanto para altas como bajas frecuencias

### Configuration
Default: Check_T_global_trm08

### Comments
Selection Ref: TCS_TRM08

### Description
Verificación de la mascara de emision espectral para bajas y altas frecuencias.

<table>
<thead>
<tr>
<th>Nr</th>
<th>Label</th>
<th>Behaviour Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>START</td>
<td>T_global_trm08</td>
</tr>
<tr>
<td>2</td>
<td>+Inicializar sistema</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>+Inic_an_esp_trm08_ftx_low</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>+Calc_SpecEmissMask_low</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>+EUT_ftx_high</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>+Inic_an_esp_trm08_ftx_high</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>+Calc_SpecEmissMask_high</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>+Check_res_trm08</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 8.** Code for test case Spectral Emission Mask
To carry out this test, we only need a spectrum analyzer; in particular we have used the FSIQ26 spectrum analyzer. The test cases were implemented in TTCN-2 because the common blocks (see Fig. 3) with protocol test systems were already built for this language version. However, the same architecture can be used with TTCN-3. In this case the common blocks around the Executable Test Suite would have to be adapted to the particular characteristics (structure, interfaces, data types, ...) of the C code generated by the TTCN compiler.

The test case has been implemented as follows (see Fig. 8 and Fig. 9):

1. The test system is initialized: the equipment configuration files are read (INIT_CONFIG) and the equipment is set into remote mode (INIT_BUS).
2. The spectrum analyzer is configured with the appropriate measurement filter parameters (bandwidth 30 kHz) via SET_PARAMETER messages.
3. The peak power is measured. The value is received in a GET_PARAMETER message.
Using TTCN for Radio Conformance Test Systems

Fig. 11. Sequence of messages for test case Spectral Emission Mask until the first trace is obtained
4. Measurements for frequencies with offset up to +2.5 MHz from the carrier frequency are taken (steps of 5 kHz) using the GET_TRACEx message.
5. Measurements for frequencies with offset down to -2.5 MHz from the carrier frequency are taken (steps of 5 kHz) as before.
6. The spectrum analyzer is configured for a bandwidth resolution of 50 kHz.
7. Measurements for frequencies with offset from -12.5 MHz up to +12.5 MHz from the carrier frequency are taken (steps of 25 kHz).
8. Measurements are compared with the reference emission mask and a verdict is generated.

When the test finishes a graphical report of the measured results is provided on the operator screen such as shown in Fig. 10. The dotted line represents the reference power levels. Figure 11 shows the sequence of messages exchanged between the test case and the Lower Subsystem from the beginning of the execution until the fourth step, where the first trace is read.

7 Conclusion

A methodology for radio conformance testing has been presented, which increases the quality of the radio test specifications as these are currently written in natural language. Using TTCN represents a step forward in the formalization of these test specifications. The validation process for test systems is simplified as some of their parts would have already been agreed by the qualification bodies.

The proposed architecture is derived as an extension from one that has provided good results in protocol conformance test systems. Several modules can be shared between both types of test systems, thus reducing the effort and cost required for the development.

This architecture enables the integration of instrumentation from different manufacturers as well as its straightforward substitution by other instrumentation with equivalent capabilities. This is achieved by the use of GPIB bus for the communication with the instrumentation and the definition of configuration files that particularize the interface implemented by each instrument.

As an example of use, the implementation of one transmitter test case for the UMTS system has been shown. Additionally, this architecture has also been used to implement the set of radio test cases for Bluetooth system.

Acknowledgments

This work has been partially funded by AT4 wireless and Spanish government.

References

10. 3GPP TS 34.121: 3rd Generation Partnership Project; Technical Specification Group Terminals; Terminal Conformance Specification: Radio transmission and reception (FDD)
11. 3GPP TS 34.108: 3rd Generation Partnership Project; Technical Specification Group Terminals; Common Test Environments for User Equipment (UE) Conformance Testing
12. 3GPP TS 34.108: Universal Mobile Telecommunications System (UMTS): Terminal logical test interface; Special conformance testing functions

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATS</td>
<td>Abstract Test Suite</td>
</tr>
<tr>
<td>ATM</td>
<td>Abstract Test Method</td>
</tr>
<tr>
<td>ETS</td>
<td>Executable Test Suite</td>
</tr>
<tr>
<td>EUT</td>
<td>Equipment Under Test</td>
</tr>
<tr>
<td>FDD</td>
<td>Frequency Division Duplex</td>
</tr>
<tr>
<td>GPIB</td>
<td>General Purpose Interface Bus</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>IUT</td>
<td>Implementation Under Test</td>
</tr>
<tr>
<td>SUT</td>
<td>System Under Test</td>
</tr>
<tr>
<td>TDD</td>
<td>Time Division Duplex</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunications System</td>
</tr>
</tbody>
</table>