UMTS Terminal Testing: A Practical Perspective

Olaf Bergengruen

Optimay GmbH – Agere Systems, Orleansstrasse 4, D-81669 Munich, Germany
Olaf.Bergengruen@Optimay.com

Abstract. This paper presents a framework based on the 3GPP test model for a Virtual Test Environment which is the main tool used by wireless-engineers for development and for testing the complete UMTS terminal software. The main requirements to the system are ease of deployment on all engineering workstations, support the design of test scenarios at a high abstraction level and, when running the test scenarios, the system shall perform much faster than in a real run in order to efficiently exercise and debug the relevant functionalities. Furthermore, the Virtual Test Environment is used to run complete test suites within a couple of hours in order to guarantee the quality of the software when it is delivered to the customer.

1 Introduction

The third generation mobile communication technology is currently being developed world-wide. Components and sub-systems will be produced by different companies (e.g. chipsets, SIM cards, RF units, base stations, IP routers, services) and will be eventually assembled into a running system, the UMTS system (Universal Mobile Communication System). The verification that all these pieces work together, i.e. that equipment or unit X interoperates with equipment or unit Y, is called interoperability testing which is very time consuming and is often the bottleneck during system development. In order to facilitate interoperability testing and to guarantee (to a certain degree of confidence) conformance to the specifications, a set of standard world-wide agreed test suites have been developed.

A key aspect for the success of UMTS will be efficient and thorough testing at all levels of development.

Furthermore, due to the increase in complexity of the next generation wireless standards, it is expected that testing effort will rise comparatively more than the effort required for the proper software and hardware development. For this reason it is important to build advanced testing and debugging tools to aid development and validation. We will see test systems with different requirements, purposes and prices for testing RF-units, for verifying software modules, for regression testing, for Type Approval and for production testing.

In this paper we will concentrate on virtual testing, i.e. pure software testing on top of simulated hardware components running on standard workstations or laptops. The system presented here is the main environment used by wireless-engineers at Optimay - Agere Systems to develop the complete Mobile Station software. The system is

D. Hogrefe and A. Wiles (Eds.): TestCom 2003, LNCS 2644, pp. 10–19, 2003.

also used to verify that all (or almost all) functionality delivered to our customers is running properly. For this purpose we periodically (once a day) run the complete test suites on each customer build. This is called regression testing which is an automatic process requiring one to two hours to complete on a powerful workstation dedicated to this task.

The basic idea of virtual testing is to exercise the 'real' Mobile Station software within a simulated environment consisting of a virtual Test System connected to the Mobile Station via a emulated physical layer. This means that at the lowest level, the hardware drivers (controlling the RF units, SIM card, keypad, serial lines, etc.) do not access the proper hardware but the software emulation which was developed for this purpose.

Of course, most Mobile Station manufacturers use test environments for in-house development. But for a simulation environment to be really useful, it needs to be designed with following basic requirements in mind: The system shall be installed and run on any standard workstation; test scenarios shall be easily adapted or created describing the signalling flow and also the cell configurations (like power levels, Sys Infos and neighbour cells); and finally the test environment needs to be much faster than real time. A wireless-engineer will immediately complain if he or she needs more than, say, one minute to run the test scenario and hit the line of code within the Mobile Station software where the breakpoint has been set.

An additional challenge imposed to a test environment results from the fact that a UMTS terminals comprises also other Radio Access Technologies. The market requires mobile terminals to support GSM/GPRS besides UMTS to connect to the Core Network, and also short range protocols like USB, Bluetooth or IEEE 802.11 for the connection to terminal equipment like laptops. Of course, we need to test each stack independently of the others, but even more important and by far more difficult is to verify the interworking between the different components (e.g. verify that the Mobile Station does not crash during inter-RAT handover from UMTS to GSM/GPRS or when Bluetooth is activated for a data connection). In a later section in this paper we will show how we handle different Radio Access Technologies within the Virtual Test Environment.

We start in the next section discussing the scope and main requirements related to conformance testing for a GSM/GPRS/UMTS terminal. But please note that this is not a theoretical paper presenting best possible testing methodologies but an overview resulting from our practical experiences implementing Optimay's test environment which was very much constrained by a turbulent history.

2 Mobile Station Testing / Logical View

From the Mobile Station conformance testing point of view, we need to test basically three interfaces (see Fig. 1): the Um interface to the GSM/GPRS network (GERAN), the Uu interface (the UMTS air interface) to the UTRAN network and the Cu interface to the USIM/SIM card.

The conformance specification for GSM/GPRS terminals is specified in GSM TS 11.10 (or 3GPP TS 51.010). Although this specification is already around 4000 pages

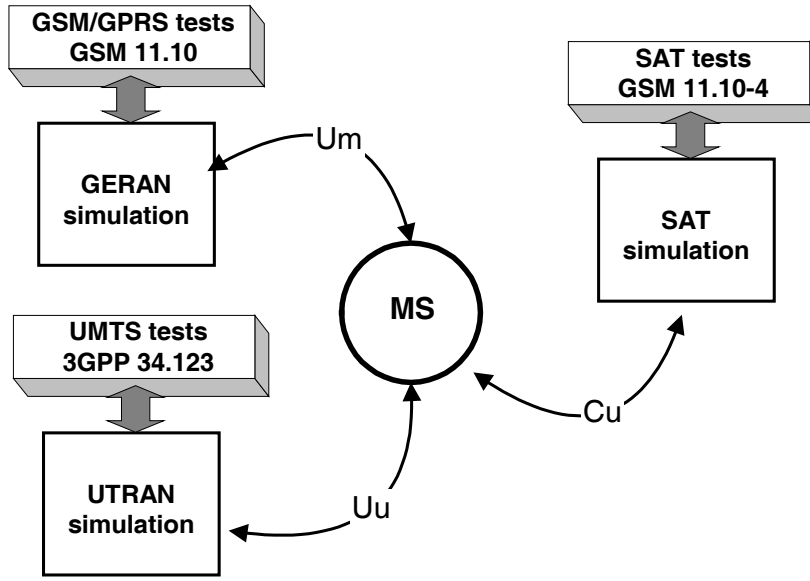

Fig. 1. Mobile Station test environment / logical view

in size, it is in no way sufficient to attain a high degree of confidence in the quality of the Mobile Station software (one of our engineers estimated that GSM 11.10 covers at most 25% of the MS functionality). Thus, we need in addition to GSM TS 11.10 also a substantial number of tests derived from more tricky scenarios found in field testing. And finally, we need performance tests for measuring and optimizing memory and cpu usage. Performance tests are not part of conformance testing nor of Final Type Approval (FTA) procedures but are needed to guarantee and sustain the quality of the overall product.

For the UMTS protocol, conformance tests are specified in 3GPP TS 34.123-1 (the prose specification). Part 3 (3GPP TS 34.123-3) of the specification provides a Test Model and test cases written in TTCN (Tree and Tabular Combined Notation) which are being developed by ETSI Task Force MCC 160. The reason for providing test cases in prose and in TTCN code arises from the difficulties experienced in the past with GSM validation which based on the prose conformance specification. A substantial effort has been invested by test equipment manufacturers, mobile manufacturers and test houses as well in developing proprietary and incompatible test case implementations. Since the prose specification needs a great deal of (human) interpretation and is ambiguous at critical points, inconsistencies and confusion arised resulting for example in a Mobile Station passing a test case from one implementation and failing the same test in an other implementation, and no one being able to tell which is the correct behaviour. For these reasons, there is today consensus in the 3GPP community that an exact test case description in a language like TTCN will simplify conformance testing and save implementation efforts. It is envisaged that eventually the TTCN code will be the mandatory conformance specification and the prose just a scenario description for a better understanding.

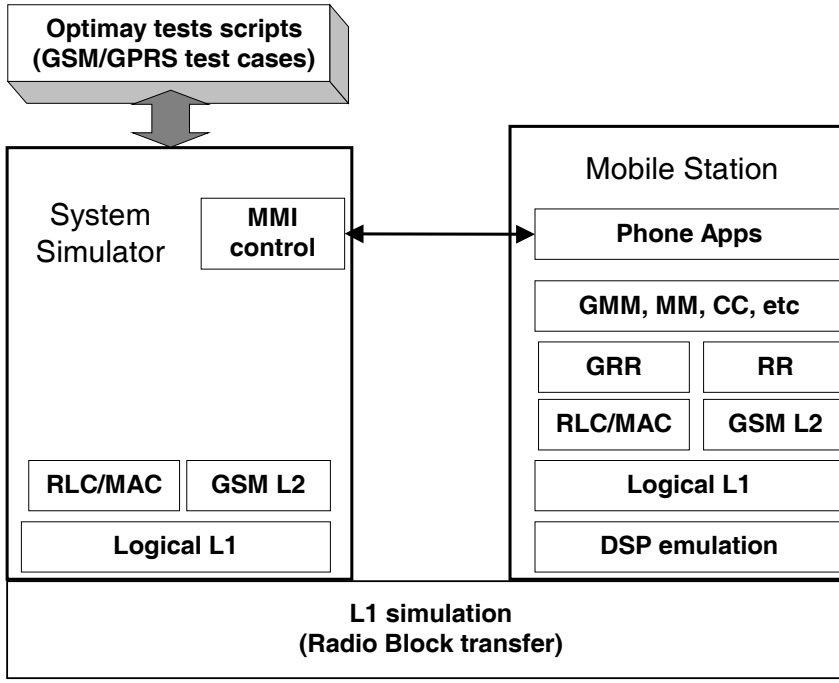

Fig. 2. GSM/GPRS test environment

Finally, the Cu interface enables applications running on the SIM card to interact in a standard way with the Mobile Station and have access to all functionality provided by the Mobile Equipment including call establishment and SMS or mail transfer (access to the ME functionality may happen even without user notice, an ideal situation for developers of 'virus applications' running on SIM cards). To test the SIM Application Toolkit functionality we need a SIM simulation and means to describe the signalling exchanged between ME and SIM in order to implement the conformance tests for SIM Application Toolkit as specified in GSM 11.10-4.

Before we discuss the 3GPP test model, we present in the next section an overview on our GSM/GPRS Test Environment which has been developed over the last 12 years for virtual testing of GSM/GPRS terminals.

3 GSM/GPRS Test Environment

In the good old GSM days, the environment for virtual testing (see Fig. 2) consisted of the System Simulator (SS) connected to the Mobile Station (MS) over a simulated air interface (the lower tester). In order to control the MS and simulate user keystrokes, a simulated MMI interface is connected to the MS (the upper tester) which allows basic user actions like turning on the MS and initiating a call.

The simulated 'air interface' is basically responsible for transferring uplink and downlink radio blocks. Additionally, it handles simulated radio conditions as set by

the test scripts which results in proper signals from the emulated DSP code to the protocol stack like measurement report indications and 'bad block' decoding indications. Finally, the simulated 'air interface' has been extended to enable timer synchronization: timer ticks are transferred from the MS to the SS which need to be acknowledged by the SS before the MS continues with the next radio block. In this way, the timer systems from SS and MS are tightly coupled.

The synchronization between the MS and SS timer systems has two main advantages. Firstly, it facilitates developing and debugging the software. Setting a breakpoint in the MS code results in stopping all MS tasks, including the timer ticks, which in turn results in freezing the System Simulator timers, test scripts and other SS processes. This feature is essential for wireless engineers to understand the system behaviour in complex scenarios by inspecting data structures and states in a running system.

Secondly, we can enormously improve the simulation efficiency: since the MS has full control on the timer ticks, it can issue timer ticks when there is nothing else to do (note that the SS also completed the jobs scheduled for the current frame, since the MS waits for the SS tick acknowledgment). In particular, when the MS is in idle mode (i.e. it is reading only paging blocks corresponding to its paging group and in sleep mode during other frames), then we accelerate timer ticks in order to jump from the frame corresponding to one paging group to the next one. Note that the virtual time flow is not directly related to the workstation time on which the simulation is running.

The results of the mechanism explained above is a substantial reduction of simulation time. This is particularly the case for time consuming tests like those related to MM periodic location area update procedures (GSM TS 11.10 clause 26.7) which require more than 30 minutes 'real' execution time. These tests are run in a couple of seconds on our virtual test environment.

Due to historical reasons, our GSM Test Environment was not carefully designed and it was coded basically using the 'quick and dirty' approach, a methodology which is often used in issues related to testing when marketing people and customers expect results and need within a couple of weeks an environment to demonstrate some functionality. Furthermore, since the test environment was viewed as an internal tool which was not to be sold and produce immediate 'cash flow', there was not much interest at the higher management level to put valuable resources on its development, and so the system was patched over the years to upgrade to the current functionality supported by the Mobile Station (I suppose that this is not an uncommon situation in the industry).

Nevertheless, it turned out that the main ideas mentioned above resulted in an extremely successful Virtual Test Environment. Currently we have around 1300 test cases comprising the GSM 11.10 conformance specification and also proprietary tests which verify functionality not thoroughly tested via GSM 11.10. The execution of these tests in a row is called regression testing which is an essential component of Quality Assurance. A full regression test report is required each time the software is delivered to the customer.

The complete test suites need around one hour execution time on a powerful workstation (actually the test suite is executed many times for different versions of the software build for different customers supporting customer specific functionality).

In addition to running protocol tests, we use the same Virtual Test Environment for developing MMI and SIM application toolkit (SAT) functionality. For SAT, we extended the simulation with an additional serial interface connected to a SIM simulation which in turn is controlled by SAT test scripts basically designed to run the tests specified in GSM 11.10-4.

In the next section we present the 3GPP Test Model and we will discuss the issues involved in designing an overall GSM/GPRS/UMTS virtual test environment.

4 The 3GPP Test Model

Mobile equipment manufacturers, test equipment manufacturers, test houses and network providers are all concerned with conformance testing. Thus, all these parties need a common reference and language for discussing test scenarios, designing test cases and building test equipment. The common reference or Test Model shall clearly describe the complete system involving the Mobile Station (referred as UE, User Equipment, in the 3GPP specifications), the System Simulator and all relevant interfaces. Fig. 3 shows the model adopted by 3GPP for signalling tests (see 3GPP TS 34.123-3). The System Simulator basically implements the PHY, MAC and RLC layers, and provides an API compliant to the 3GPP interface as specified in TS 34.123-3. The behaviour of the RRC layer and NAS (Non-Access Stratum including GMM, MM, CC and SM) is explicitly coded in the TTCN test scripts.

A **test case** running on top of the System Simulator consists basically of following steps:

- Configure each cell within the System Simulator
  - Configure PHY / L1
  - Configure MAC
  - Configure RLC
  - Set power levels for the different cells
- Schedule and send System Information Blocks
- Bring the UE into initial state according to the test case (as an example, the UE is brought into Idle Updated state)
  - Perform Location Update procedure
  - Perform GPRS Attach procedure
- Test case body
  - **Stimulate the UE** (for example the SS sends an IDENTITY REQUEST message to the UE, or the SS changes power levels, or the SS commands the UE via e-MMI to establish a call)
  - **Verify responses** sent by the UE and issue a PASS, FAIL or INCONCLUSIVE verdict
- Test case postamble
  - Complete signalling to bring the UE into a stable state

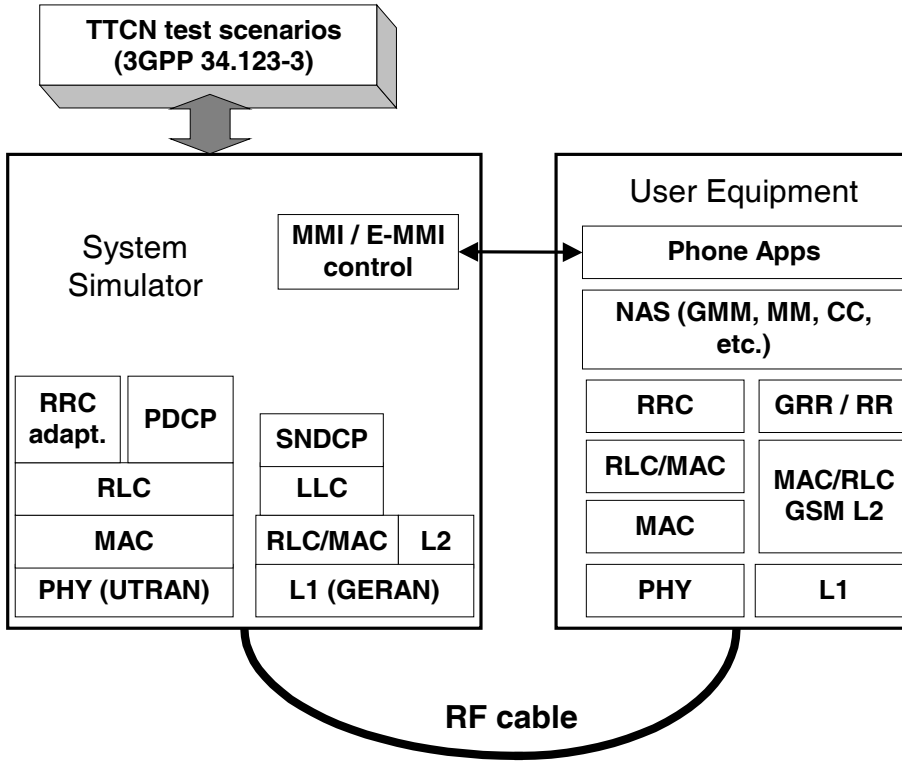

Fig. 3. The 3GPP Test Model (adapted from 3GPP TS 34.123-3)

The Test Model proposed by 3GPP would be the first choice for a model underlying an implementation of a virtual test environment, not only because it is clear structured with well defined interfaces, but also because it facilitates the re-use of software modules and test cases implemented by third parties.

Unfortunately, the 3GPP test model shown above is not sufficient for modeling a complete test environment needed for testing all software components running on a Mobile Station. As already stated, the UMTS stack is only one component in a multimode terminal which also includes a mature GSM/GPRS stack and other functionalities like MMI and STK which need also to be validated. Furthermore, all these components have been developed over the last years using proprietary development tools and running on different environments. For example, we developed a scripting language and wrote about 1300 test cases in that language for the GSM/GPRS protocols. These tests comprise the core components of our regression tests which are run on a daily basis.

Thus, the engineering challenge is now how to build a test environment involving the different protocols, the different testers with their test suites and possibly debugging and tracing tools.

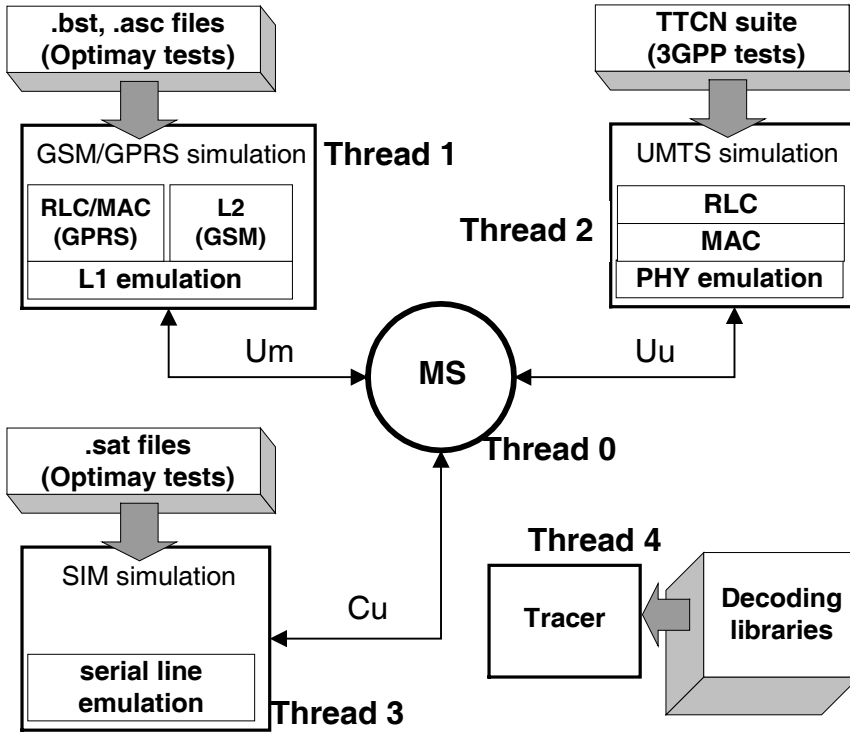

Fig. 4. Architecture of the GSM/GPRS/UMTS Virtual Test Environment

5 UMTS/GSM/GPRS Virtual Test Environment

Our implementation of the Virtual Test Environment (Fig. 4) consists of the proper Mobile Station under test (or User Equipment in 3GPP parlance) and three Testers: the GSM/GPRS System Simulator, the UMTS System Simulator, and the SIM Tester. We decided to apply, as far as possible, the 'golden rule' not to touch running software nor validated test cases unless really necessary. Thus, we encapsulated each Tester into a separated thread implemented as a DLL library. Each Tester thread interfaces with the User Equipment (UE) only via shared buffers protected by semaphores.

The SIM Tester sends and receives SAT command/responses over the simulated serial SIM interface which in turn uses the shared buffers for data transfer. The GERAN and UTRAN Testers use the shared buffers to transfer radio blocks and emulated PHY command/responses.

Additionally, a Tracer thread is responsible for decoding, tracing and 'pretty print' into nice trace logs. The Tracer software is identical to that running on a workstation when testing and tracing the hardware. The Tracer basically fetches trace messages sent by the UE or by the Testers, decodes them using appropriate libraries and out-

puts into a file. The decoding libraries contain functions for decoding GSM/GPRS messages (GSM TS 04.08), for decoding the UMTS ASN.1 messages (3GPP TS 25.331) and also for tracing proprietary messages used for debugging including inter-layer signalling, memory usage on the hardware and DSP states.

There is still one essential component missing in this overview: the synchronization of the threads and, in particular, the coupling of the different timer systems. Each sub-system maintains and manages its own timer system. We need the capability to set breakpoints at any line of code (either at the UE side, at the Testers or even within test scripts) and freeze the whole system when the breakpoint is reached. Also, as explained in a previous section, since the UE is the 'tick master', it can accelerate during a virtual run the rate of its timer ticks sent to the Testers (in particular when the UE is in idle mode only decoding Sys Infos and monitoring neighbour cells). This mechanism greatly improves simulation time. For example, test case 8.3.1.3 Cell update / periodical cell update in CELL_FACH (3GPP TS 34.123-1 version 4.2.0) which at step 13 requires the SS to wait 720 minutes (12 hours!) can be executed in a couple of seconds in our Virtual Test Environment. Of course, this is not a particularly useful test case and hopefully will soon be removed from the test suite.

In order to achieve this tied timer synchronization, and following the ideas developed for GSM/GPRS, we set the UE to issue a timer tick when completing the current frame and then to wait for the tick acknowledgment from each Tester before it goes on to process the next frame. (The code needed for timer synchronization is minimal, but it needs to be thoroughly designed, otherwise it will never work).

6 Future Work

Fortunately for us engineers, there is still plenty of work to be accomplished.

Firstly, we need to add new interfaces to include the W-LAN (IEEE 802.11) and/or Bluetooth stacks in order to run and test a complete TCP/IP application connected to the Mobile Terminal via one of these interfaces. Further, we would like to run 'real' WAP test cases on the Virtual Test Environment which will require a server application running on top of the protocol stacks at the System Simulator side. We still need to investigate the coding effort involved in adapting and connecting TCP/IP and WAP protocols at the System Simulator side for this purpose.

Another issue is related to tools facilitating automatic or semi-automatic design of test cases out of critical scenarios found during field testing. As already mentioned, some software bugs are only found during field testing in tricky situations like Mobility Management or handover procedures at country borders where different providers use different configurations for their PLMNs (Public Land Mobile Networks). It is of high value to exactly reproduce these scenarios within the Virtual Test Environment. For this purpose we need tools supporting the engineers when inspecting the trace files produced during field testing in order to extract base station settings, the System Information broadcasted by the different base stations and the signalling which eventually lead to an erroneous behaviour at the Mobile Station.

Finally, for reducing the simulation time we will distribute the simulation load on different workstations. Several interesting topics on distributed computing need to be investigated like the estimation of memory and cpu potential on idle workstations and the segmentation and distribution of 'simulation patches' over the network.

7 Summary

In this article we presented a practical view of a Virtual Test Environment used in every day work by wireless-engineers developing UMTS Terminal software. Since the environment is software only (i.e. it does not need any UMTS specific hardware) and runs on any standard workstation, engineers can use their preferred developing tools, compilers and debuggers while focusing on the functionality they are currently developing. After selecting test scenarios and setting breakpoints within the Mobile Station software (or also on test scenarios), the system is started and within a couple of seconds it will stop (freezing all timers) at the requested code location for inspecting variables, system states, checking memory usage and so on.

The system is also used for regression testing prior to the delivery of the software to our customers. This is of utmost importance when software modules developed by different groups at different sites and time zones are integrated into a single image which loaded into the Mobile Station hardware will literally execute in your hand.

As one of our engineers remarked 'We will not survive without a proper regression test system verifying at any time all aspects of the phone functionality'. We, as engineers, need to make this point clear also to the company's management, so that the test environment itself is part of the long term company strategy.

References

1. Technical Specification 3GPP 23.010, General UMTS Architecture
2. Technical Specification 3GPP 34.123-1, User Equipment (UE) conformance specification, Part 1: Protocol conformance specification
3. Technical Specification 3GPP 34.123-2, User Equipment (UE) conformance specification, Part 2: Implementation Conformance Statement (ICS) proforma specification
4. Technical Specification 3GPP 34.123-3, User Equipment (UE) conformance specification, Part 3: Abstract Test Suites (ATS)
5. Technical Specification GSM 11.10-4, SIM Application Toolkit conformance specification
6. Technical Specification GSM 11.14, SIM Application Toolkit