3GPP-specific commands and must be adapted to the specific mobile phone being tested by means of proprietary commands. This requires a knowledge of Visual C++. It should also be mentioned here that the Automation DLL project basically contains two modules: the comport.c module, which contains all functions that involve the COM port, and the MS_Auto_RS232.cpp module, which contains all mobile-phone-specific commands. For details, refer to the R&S®CRTU-G software manual.

Adapting this DLL to the DUT takes relatively little time if you consider that this up-front work reduces test time and that no personnel is needed during testing. This means that you can use the protocol tester virtually around the clock.

You can monitor the commands exchanged between the R&S®CRTU-G and the mobile phone by using the AutoDLL Traffic Viewer. This tool is also helpful when developing a Customer Automation DLL.

Summary

By providing test-case automation, Rohde & Schwarz has increased the value of the R&S®CRTU-G even further. A universal mechanism that requires only a little up-front effort saves substantial time later on. For details, refer to the R&S®CRTU-G documentation.

Gerhard Götz

Protocol Tester R&S®CRTU-W

Efficient programming interface for UMTS protocol development

The R&S®CRTU-W (FIG 1) has long been known for its convenient and powerful TTCN interface, which was designed and optimized for conformance testing. Research and development, however, require a flexible yet easy to operate programming interface. The new MLAPI software option (R&S®CRTU-WT02) now provides you with the right tool for R&D applications.

Intelligent compromise

The use of the R&S®CRTU-W as a test tool in UMTS protocol development places very high demands on the programming interface. The vast number of parameters specified by the 3GPP standard as well as the variety of applications call for an intelligent compromise between flexibility and convenience. The R&S®CRTU-W now comes with a new software option – MLAPI (Medium Level C++ Programming Interface) – that provides the right answer.

Convenience and flexibility

The test cases specified by 3GPP for conformance testing of UMTS mobile phones [1, 2] are already implemented in the R&S®CRTU-W. Research and development, however, require a programming interface that offers greater flexibility and convenience. It must be able to handle the constraints inherent in the TTCN programming language as well as in the architecture of the 3GPP test suites and the extra tools required.

To meet this need, Rohde & Schwarz has developed the MLAPI programming interface. It enables you to create protocol layer 3 user-defined scenarios in the C++ programming language. The meth-
ods of handling protocol messages as well as the modelling of dynamic operation cover a wide range of applications, from simple to very powerful. For example, you might want to set a mobile phone to a defined signalling state by using a simple sequence of messages and then implement a desired application in this state. Or, you might employ complex alternatives or deliberately cause a faulty response in order to test the signalling function.

Concept of automatic configuration

Due to the quality of service (QoS) architecture of the UMTS standard, higher protocol layers request the transmission quality for a specific application in abstract form, whereas the actual parameters for channel configuration are known only in the lower layers (referred to as access stratum, AS). These parameters are very numerous, however, and can be combined in a variety of ways. UMTS differs from GSM in that the top AS layer, i.e. the RRC [3], is responsible for managing these parameters and their combinations. The strategy for assigning these parameters is implemented by the network operator. The parameter information is coded as RRC protocol data (PDUs) using complex elements described in the ASN.1 language and transmitted to the mobile phone. The base station and the mobile phone must configure their lower protocol layers accordingly before data can be transmitted at the required QoS level (FIG 2). Similar processes take place during handover and other main procedures.

In contrast to other layer 3 messages, the RRC PDUs therefore contain information not only for the peer station but also for local configuration. The configuration interfaces of the lower layers are usually proprietary, whereas the RRC PDU is defined by the 3GPP standard [3].

Since users of the C++ programming interface are usually interested in the detailed definition and full flexibility of the RRC PDU covering all parameters rather than in the complex configuration of the lower layers, offering the actual RRC PDU as a “single-source” interface is a good approach.

As a result, the MLAPI protocol stack provided by Rohde & Schwarz contains not only the lower protocol layers but also the part of the RRC layer that is responsible for the configuration of the lower layers, hence the designation “Medium Level”. Thus, when you send an RRC PDU from your scenario, the required information is automatically extracted and passed on to the lower layers. This considerably reduces error rate and program size without any loss of flexibility.

New dimensions in protocol message handling

The basic software for the R&S®CRTC-W provides a library (MODDB) for handling protocol messages. The library is automatically generated from the message specifications and includes all methods of handling such messages. For example, the Message Analyzer uses the MODDB to completely decode recorded protocol messages and interpret subelement values in text form.

All MODDB methods are available for the C++ programming interface, including read and write access to any desired subelements, cloning of substructures, reading and storage of message instances — in hexadecimal or XML format — and many more. Simple macros are available for transmitting and receiving any layer 3 PDU.

Being able to define message instances is one of the programming interface’s main features. Therefore, a graphical tool which is also based on the MODDB has been implemented for this purpose — the Message Composer (FIG 4). This tool enables you to conveniently perform any modifications to the scenario behaviour that are coded in the protocol messages without having to recompile the scenario.

To allow received messages to be checked against predefined receive constraints, they are not stored in purely hexadecimal format. Rather, they must support wildcards that, for example, allow a defined list of values or any value (ANY) to be assumed by a specific subelement. These values can also be graphically defined with the Message Composer and are highlighted in colour (FIG 4). The C++ programming interface also offers a simple macro that performs the comparison automatically. If there is no match, the discrepancy will be indicated in the subelement tree.

Intelligent organization of dynamic behaviour

The MLAPI offers simple macros for defining state machines to enable efficient programming even for complex, branched operations. The reception of a message — or more generally speaking an event — is represented by a transition within a state. If and to what extent different states or state machines are actually used is up to you.

Further powerful functions are available for optimizing scenario modularity. State machines can be run in parallel or be derived from one another, making it easy for you to define a common means of handling exceptions for a specific number of procedures, for example. Moreover, state machines can mutually call each other, allowing you to design a procedure-oriented scenario. The programming interface further provides convenient means of defining timers, delays,
FIG 2
DoS and UMTS protocol structure.

FIG 3
Modularization of the dynamic behaviour of a scenario.

FIG 4
Message Composer with receive constraints check.

Important abbreviations

3GPP 3rd Generation Partnership Project
AMR Adaptive Multirate
AS Access Stratum
ASN.1 Abstract Syntax Notation One
CS Circuit Switched
EMMI Electrical Man-Machine Interface
HSDPA High Speed Downlink Packet Access
IP Internet Protocol
MAC Medium Access Control
MDDB Message Description Data Base
MLAPI Medium Level C++ Programming Interface
MMS Multimedia Message Service
PDCP Packet Data Convergence Protocol
PDU Protocol Data Unit
PS Packet Switched
QoS Quality of Service
RAT Radio Access Technology
RLC Radio Link Control
RRC Radio Resource Control
TTCN Tree and Tabular Combined Notation
UMTS Universal Mobile Telecommunications System
WAP Wireless Application Protocol
triggers, as well as the parameterization for passing messages or other information to additional state machines.

The MLAPI software option already contains a number of example scenarios in the form of compilable source code. These scenarios show how to use all essential interface elements. They also implement standard applications such as CS registration and PS attach, AMR speech call, PS data call, CS/PS multicall, soft and hard handover. Being of modular design (FIG 3), the scenarios can easily be expanded. To make this possible, the state model described in the 3GPP test specification [4] has been expanded to create a modular system.

All actions carried out by macros of the programming interface are written to a log file that can be decoded by the Message Analyzer. This keeps you constantly informed of what is happening – in the state involved and the scenario activity – in the lower protocol layers. You can also send information of your own to the log file while your scenario is running.

**Tools and peripherals**

The generated programs run under Windows® 2000; the programming interface is integrated into Microsoft® VisualStudio .NET© IDE. User-created scenarios can be organized by means of the TestSuite Explorer and Project Explorer tools included in the R&S®CRTU-W basic software. An example project is also supplied.

The MLAPI supports linking to the EMMI, enabling automatic control of the mobile phone by means of AT commands [5]. It also allows preliminary and final verdicts to be output like in conformance testing. The above functions employ the same modules as those used in the TTCN test cases.

**Summary and future developments**

The MLAPI is a tool for UMTS protocol development that optimally combines flexibility with convenience. Due to the unique interface design, test scenarios can be created with considerably less effort. The concept of automatic configuration of the lower protocol layers further contributes to efficiently generating and maintaining test scenarios. Plus, you can easily integrate your own protocol behaviour into the C++ program source code that is supplied. This enables you to simulate incorrect protocol behaviour, for example.

The versatile capabilities of the base unit are also reflected by the programming interface. The convenient use of IP-based data services (http, WAP, audio/video streaming, MMSS) and video conferences will soon be possible, as well as the scaling of several protocol testers for multicell and handover scenarios. Moreover, Rohde & Schwarz plans to expand the programming interface to enable you to control a Protocol Tester R&S®CRTU-G using the same philosophy as for the R&S®CRTU-W. This will allow you to define 2G3 inter-RAT handover scenarios [6].

Rohde & Schwarz increasingly offers packages of ready-to-run scenarios that simulate typical applications in real networks, i.e. applications in the field of interoperability testing.

Future expansions include the HSDPA standard, interfaces for stimulating a specific behaviour in layer 2, and a convenient graphical user interface for interactively creating scenarios from existing procedures.

Stephan Sandhäger