FUNCTIONAL TESTING AND TESTING UNDER LOAD CONDITIONS OF REAL-TIME SOFTWARE IN SPC SWITCHING SYSTEMS VIA A UNIVERSAL ENVIRONMENT SIMULATOR

W. Daniel[1)], H. Friedel[1)], R. Lehnert[1)], W. Lemppenau[2)], P. Tran-Gia[2)]

[1)]Philips Kommunikations Industrie AG, Nuremberg, F.R.G. [2)]University of Stuttgart, F.R.G.

1 INTRODUCTION

The software development for private communication systems is often characterized by the use of an enhanced version of the CCITT's SDL method (1). The software specification at Philips Kommunikations Industrie is done in a coarse SDL, followed by stepwise refinement and further transformations. By this method, the control layer and the data manipulation layer are strictly separated. The code of the control layer is generated automatically from the refined formal specification using a set of tools; the data manipulation layer is modularized using the principle of information hiding and is written in a high level language.

Errors in the control layer are mainly due to special cases not considered in the specification or in the refinement process. Furthermore, the debugging and the evolutionary development of the software may cause new errors.

To validate the control layer, it is necessary to make exhaustive tests. These tests have to be done automatically as far as possible and must be reproducible. The test sequences have to cover concurrent activities of several and different ports.

In the literature a number of environment simulators have been presented, most of them are designed for specific systems to be tested (3-5). Thus, they are system-dependent and can therefore only be applied to dedicated systems. Other approaches deal with more system-independent concepts; they are designed for the use in simulations of a relatively small number of subscribers. Most of the known environment simulators do not take into account the dependency of the subscribers' behaviour on system reactions (feedback effects, e.g. repeated attempts, subscriber impatience, etc.) as well as subscriber - system interactions. They handle call-individual electrical test for a small number of subscribers.

2 Objectives

The aim is to have an universal test machine for the black-box testing of switching systems for voice, data and integrated services. The machine shall consist of a part common to all applications and a dedicated part serving the standardized interfaces of the target system.
Accommodation to the target system has to be done by installing the appropriate test software and adding the specific hardware interfaces.

Users of the test machine will be the development, the manufacturing, and the maintenance departments.
All of them have similar overall goals:

- o automatic test,
- o test repeatability and regression test,
- o extensive test coverage,
- o fast execution of complex test programs,
- o generation of test documents.

They concentrate on different aspects. The development departments want to test primarily conflicting and boundary situations, while the manufacturing departments are more interested in testing all extensions and the features these are authorized to use.
Including the maintenance staff in the specification of the test software and in the system tests with the environment simulator enables them to influence the diagnostic and maintenance concept of the system under development.

For the test under load conditions the user of an environment simulator should be able to test the the process interference in the software of a SPC switching system.
It should be able to generate realistic system loads, which model stationary as well as nonstationary overload conditions. In order to to characterize and to simulate overload traffic streams, e.g. for the investigation of a specific overload control mechanism, we should be able to realize the offered traffic by means of short-term nonstationary load patterns, whereby realistic effects like repeated attempt phenomena can be taken into account.

3 SYSTEM DESCRIPTION

The environment simulator is embedded in our set of test tools, see fig. 1. It simulates real environments and is optimized for the SDL method mentioned above.
It communicates with the target system via an interface named ADAPTER, see fig. 2.

3.1 UNES

The functional/hardware structure of the universal environment simulator UNES is shown in fig. 3. The simulator consists of five functional modules

- system control module	SCM
- subscriber behaviour module	SBM
- random number generator	RNG
- timer handler	THD
- target system interface	TSI

The simulator modules are implemented by means of M68000 microprocessor based control units backed by the VME-bus.

The SCM supervises the whole activities of the simulator in terms of hardware control lines and messages via the system bus.
The SCM is based on a readily available CPU board equipped with 256 kByte dual-ported RAM a terminal and a floppy disc interface.
At present two 5.25" floppy disc drives with 720 kByte capacity each and a control VDU are connected to the SCM.
The hardware of SCM is expanded by two 8" floppy disc drives with 1.2 MByte capacity each, a hard disc with 24 Mbyte capacity an additional terminal port for the interconnection with a host computer and a parallel printer port. There is also a 1 MByte RAM board connected to the VME-bus.
The operating system of SCM is CP/M 68K™, which enables the user to access a variety of standard software, e.g. a screen-oriented editor, compilers for C and Pascal, an assembler and additional support software.
The interconnection of UNES and a host computer is controlled by a support software in which UNES emulates a DEC VT100 (VT220) terminal. The data transfer runs error-protected under the control of UNES without any additional interactions between the user and the host computer.

The SBM is based on an available CPU board with 128 kByte dual-ported RAM. When performing a functional test the real-time operating system MTOS™ and a layered software is used. In the case of testing under load conditions, the main function of the software used in SBM is to model the designed number of subscribers and trunks connected to the target system and to supervise the messaging between SBM, TSI and SCM.

The THD and the TSI are based on CPU boards with 16 kByte dual-ported RAM and 192 kByte local memory for code and data. The programme code for these modules can be stored residently or it can be loaded during a configuration phase from mass storage.
In the case of testing under load conditions, a software timer, controlled by the THD, is allocated to each simulated subscriber process. The message passing between THD and SBM then is done via two message buffers allocated in the dual-ported RAM area of THD. The special functions of THD for controlling the software timers, are supported by five local 16 bit counters in hardware. The simulation clock is provided by THD and can be accessed simultaneously by the SBM without any interference to the local of the THD's control unit.
The TSI is able to support two 16 bit parallel data links for connecting the simulator to the target system via ADAPTER. Basically it manages the flow control between SBM and ADAPTER, using an intermediate message buffer.
In order to enable more accurate measurements of message delay and system reaction times, TSI is able to report the time instants when transmitting and receiving a message to SCM and SBM. The message passing between TSI and SBM is done via two message buffers located in the dual-ported RAM area of TSI. Based on the implemented uniform message format between SBM and ADAPTER, TSI transports messages transparently in both modes.

The RNG provides random numbers based on arbitrary distribution functions to the SBM. It is implemented by means of a multiplicative congruential random number generator producing uniformly distributed random variates in the interval (0,1] in conjunction with a table-driven distribution function decoder DFD.

3.2 ADAPTER

ADAPTER is the link between the universal part of the testing system UNES and the target system, see fig. 3.

To ease the use of UNES with different target systems, ADAPTER has to interface to standardized external ports of the target system.
Usually, slightly modified peripheral boards of the target system can be used.
The structure of ADAPTER depends on the used peripheral boards:

- A small number of VME-bus compatible peripherals with sufficient on-board processing power may be connected to the system bus of UNES.
  We consider this alternative for the adaptation to a small text and data switching system.

- Other boards have to be connected to UNES by means of the above mentioned parallel interface.
  This architecture was chosen for the adaptation to a digital PABX. Our further explanations are based on this realisation.

UNES and ADAPTER have to perform functional testing as well as load testing. Therefore the structure of ADAPTER has been optimized with respect to small delay times.

ADAPTER is composed of two main parts:

- a central part,
- a peripheral part.

The central part is a multi microprocessor system based on the iAPX 86 microprocessor family. The processors communicate via the standardized Multibus™.

The UNES-ADAPTER interface (UAI) is the counterpart of and communicates with the TSI in UNES.

The central unit (CU) performs the translation of the message codes used by UNES to the message codes used by the peripheral part and vice versa. It also transforms the logical process numbers used in UNES into board addresses.

The bus interface (BI) connects the central part of ADAPTER to the peripheral part. It polls the peripheral boards and supplies them with messages received from the CU.

The communication memory (CM) is merely a buffer for the information exchange between the processors of the central part.

All peripheral boards are equipped with one-chip microcontrollers (i8741A). Their task is to debounce the boards' detectors and to generate messages to the central part of ADAPTER.
In the reverse direction, they transform commands into actions on the physical level, e.g. switching relays, tone and ring circuits. At present, the peripheral part contains boards of the following types:

- Analogue subscriber.

The hardware is the original analogue trunk line board of a PABX.

- o Analogue trunk line, subscriber signalling.
  This is an original analogue subscriber line board.

- o Analogue trunk line, direct dialling in (DDI).
  This is a newly developed board, based on the DDI trunk line board of a PABX.

- o A signal generator board supplies the above mentioned boards with clocks and tones.

We intend to add boards with digital interfaces relating to CCITT's S_0 interface as soon as these interfaces are requested by our target system.
The digital interface boards will interface directly to the Multibus of the central part of ADAPTER.

4 SOFTWARE STRUCTURE

4.1 Test Setup Procedure

The man-machine interface offers the functions of an intelligent PC with softkey and menue technique.

UNES has four operational modes:
- installation and configuration,
- test administration,
- test,
- maintenance mode.

Normally the complete test evaluation is done on a host computer after the test run. This evaluation is supported by a tool. The data base for the evaluation is the UNES log-file and, if available, the trace file of the target system.

During the installation and configuration phase, UNES can be accessed and programmed by the user at the man-machine interface. The operation control units of SBM, THD and TSI are deactivated, thus enabling the SCM's operating system to load the dual-ported RAM areas of SBM, THD and TSI. On the other hand, test and simulation results, which are stored on mass memory, can be accessed and printed. Based on the system support, the user is able to produce new software for SBM or other modules. The initalization data required for the different modules for several test and simulation runs is stored on mass memory. The input of this data by the user is assisted by a user-oriented support software, which allows the comfortable generation or modification of files.

4.2 Layered Software For Functional Testing

The software for functional testing consists of a basic part and the test software.
The basic part includes a standard multitasking real-time operating and the test manager.
The test software has a layered design oriented at the OSI layer structure but also fitted to the software structure of a voice-PABX, see fig. 5.
The terminal and the mass storage handler is running on a separate processor-board and under CP/M 68K.

4.2.1 Elementary Operations. The lowest layer, named elementary operations, receives and generates the signals of the used signalling systems. It provides the next higher layer with the logical equivalents of the signals.

Therefore, this layer has to be port oriented. Different kinds of peripheral interfaces are supported by dedicated process types. Every port corresponds to an incarnation of the appropriate process type.

As the elementary operations layer communicates with the next upper layer by means of elements of the used signalling system, with our architecture it is not intended to test the correct behaviour of the signalling link. This layer executes all time-critical tasks originating from the used signalling system. For that reason, the upper layers only have to consider the delay times of the application software of the target system.

The elementary operations layer is implemented in ADAPTER. Its main functions are performed by the peripheral boards.

4.2.1.1 Software in the peripheral part. The analogue subscriber has to perform all possible operations of a simple telephone set, such as
- o closing and opening the loop
- o dialling
- o operating a relay to simulate the earth key.

Furthermore, the peripheral boards have to
- o distinguish different tone and ring sequences,
- o test the speech channel by transmitting a test tone.

In conjunction with a board bearing multiple DTMF transmitters, we may as well do sole DTMF signalling as well as mixed rotary dialling and DTMF dialling.

The analogue trunk line board with subscriber signalling behaves similar to an analogue subscriber line board.
That is
- o switching ringing signals and dial tones,
- o observing the loop state to detect on-hook, off-hook and outgoing digits.

DTMF dialling on trunk lines is possible with additional DTMF registers as used in PABXs, but not yet supported.

The analogue trunk line board with direct dialling in requires the most complex software of the peripheral boards. The software has to deal with several voltage level detectors and reversly to switch different line terminations.

4.2.1.2 Software in the central part. The physical interface between UNES and ADAPTER is a high speed, full duplex, 16 bit parallel interface. For the sake of speed, the software is written in assembly language.
The UAI writes data received from UNES to a queue in CM, from were it also reads the data to be transmitted to UNES. The link between TSI of UNES and UAI of ADAPTER is transparent.

The BI interconnects the central and the peripheral part. It runs a handshake protocol to the peripheral boards' microcontrollers. It communicates with the CU by means of

queues, located in CM.

The CU translates the message codes used by UNES to message codes usable by the peripheral part and vice versa.
These conversions hide the specific hardware structure of ADAPTER from the upper layers of the software. Further features of the CU are

- o trace facility to record the message conversion
- o error logging
- o interface to an auxiliary terminal for test purposes, providing
  - a concurrent monitor to substitute an emulator in some cases
  - selective enable/disable of the trace facility
  - procedures to test and configure the peripheral hardware of ADAPTER as well as to check and change the translation tables.

Especially the trace and the error log are very useful during development and test of ADAPTER.

4.2.2 Primitives. The next layer, the so-called primitives, is also port-oriented.
To provide a first evaluation of the target system's reactions the incoming signals are analyzed with respect to the context of the signalling procedure.
The implementation is done using SDL for the control layer and C for the data manipulation layer.

4.2.2.1 Voice PABX. Here signalling procedures are split into distinct signals, which are sent to the elementary operation layer. In the reverse direction one or more signals are assembled to signalling procedures.

4.2.2.2 Text Switch. Here the standardized services of level 2 and level 3 of the specific interfaces are offered (e.g. HDLC and connection supervision in case of Teletex).

4.2.3 Complex Operations. This layer is link oriented and controls the
complete lifetime of a connection. The physical and logical states of the connections and the related events are controlled with respect to the actually activated features. Moreover this requires the sampling and administration of configuration and connection data.
The implementation is done in the C language.

4.2.3.1 Voice PABX. Standard switching procedures like off-hook including waiting for dialling, dialling, internal call, etc. are the basic building blocks (software procedures using all the services provided by the lower levels) with which the test manager can run complex switching procedures.

4.2.3.2 Text Switch. The basic building block of the complex switching procedures are the standardized services of levels 4, 5, 6 of the specific interfaces, e.g. transport, session and document layer.
In addition to these interface-relevant procedures there exists a set of feature-relevant procedures simulating the features of the target system, e.g. retrieval or operation and maintenance functions.

4.2.4 Test Manager. The uppermost layer, the test manager, represents level 7, the application layer. It is in control of the whole test procedure. It activates complex procedures, awaits their completion and evaluates the target system's reaction by means of its configuration and the context of the test procedure, including exception handling.
The results of the tests are written onto a log file and may be analysed later on.
Because the control level software in most of our target systems is written in the SDL language in the interpreter mode, it is also possible to produce a real-time trace from the target system.
The test manager is driven by test files dedicated to the target system. These test files are written in a specific pseudocode language. A tool translates the test files into data files of the programming language C. Another tool generates test flow documents from these history files.
To circumvent the limited mass storage capacity of UNES especially in the case of automatic long-time test runs, an on-line link between UNES and a host computer, e.g. VAX is available.

4.3 Testing Under Load Conditions

The aim of testing under load conditions is to evaluate the performance of the target system under a given subscriber load and a given subscriber behaviour. Moreover, the influence of the system's reaction to the subscribers' behaviour can be determined by the user.

The software structure of the simulator for testing under load conditions is depicted in fig. 5, where also the embedding of the modules into a hierarchical structure is shown, as already described in sect. 3.
Starting a simulation run by the user at the man machine interface, the operating system deactivates its overall system control and activates the measurement and statistics software inside the SCM. After an internal initialization phase, this task transfers the control via the system and messaging processes to SBM, thus activating the control units of THD and TSI. In the following, the interlevel communication is executed via message interchanging. This message interchanging is done by a dispatcher inside SBM.

The tasks of SCM during a simulation run are to monitor the whole system activities and to gather messages from SBM for measurement and statistics purposes. These messages can be stored in a trace buffer and saved simultaneously in the mass storage and the host computer.

The main function of the SBM is to model the designed number of subscribers and trunks connected to UNES via ADAPTER. It generates telephonic events, (e.g. off-hook, on-hook, digits, trunk seizure, trunk release, etc.) for the simulated subscribers or trunk groups according to the the subscriber or trunk behaviour models and the reactions of the target system. The subscriber or trunk processes which define the behaviour of subscribers or trunk groups, are described by means of finite state machines (FSM).
In order to model a subscriber (trunk) in conjunction with its individual behavioural timing properties (impatience interval, wait for reattempt event, interdigit intervals, etc.), a software timer is allocated to each simulated subscriber (trunk) process. This timer is controlled by THD.
Messages generated by a simulated subscriber or trunk are transmitted to TSI, and messages

received from TSI are routed to the addressed subscriber, that is the appriate FSM. Inside the FSM each subscriber is represented by an individual random number driven process.
The actual state of a process is located is located in the individual data area of the simulated subscriber or trunk associated to this process, where references to the specific data for this type of subscriber (e.g. behaviour-oriented time periods, probabilities for actions/reactions, facilities, etc.) are stored.

The behaviour of subscribers is modelled by a SDL diagram. An example is given in fig. 6, which shows the modelling anf specification of the subscriber's behaviour.
SBM reports all activities of its FSMs to SCM which in turn describe all activities between UNES and the target system.
This enables the user to observe in detail the target system's reactions under a given traffic load or to do performance evaluations for specific time intervals or events.

Based on the hardware configuration currently used, run time measurements of the software executed in the modules of UNES and performance investigations of the message transport delays between SBM and the interface to ADAPTER resulted in acceptable delay figures for up to 1000 simulated subscribers (2).

5 APPLICATION

UNES is used at first with a digital PABX, which is currently under development.
To gain experience, we realized a prototype of the functional test software, including only some of the usual telephony features.
This prototype was tested with an existing analogue PABX as target system.

6 CONCLUSIONS

The specification of the target system should be done in a formalized way rather than in a kind of prosaic description, and alterations should be communicated to the team developing the test software in parallel to the system's development.

Although using a formal method specifying and implementing the logical behaviour (control layer) of the target system it is not possible to derive from this the control layer of a test machine because of two reasons :

1. A formal translation method deriving the control layer of the test machine from the control layer of the target system would result, if possible, in testing erroneous software by a test software which contains the same errors.

2. The logical layer of the target system's software contains no information on the test strategy necessary to get a sufficient coverage of all possible test situations.

Already in the early design phase, we need a specification of the intended scope of the testing to be done.
This depends mainly on the 'customer' of the test software. Although the requirements of these different departments are rather different, it was possible to satisfy our 'customers' with a software differing only in the highest, the test manager level.

The handling of the layered test software requires the support by tools to keep the software system consistent and to generate different versions. This implies the use of a powerful support computer.

The use of an automatic test equipment resulted in some major improvements. We now can repeat a test exactly as we did it before. This enables us to check whether changes in the software have any unwanted side effects. Secondly, we are now forced to set up detailed test scenarios, making "wild", uncoordinated testing impossible. Furthermore, we are able to do more complex testing than without an automatic tester. This concerns the number of affected lines as well as interdependencies between features.

It has been shown, that by involving the maintenance and manufacturing staff in the test activities the total development time can be reduced.

The simulator also provides a universal performance evaluation tool for switching systems, where realistic phenomena, which strongly affect the switching performance, like subscriber impatience or repeated call attempts can be considered and investigated. The environment simulator allows us to investigate the process interference inside the software of a SPC Switching System under designed load and overload conditions. Furthermore, it can be applied to evaluate the effectivity of overload control strategies in switching systems, whereby detailed modelling of subscriber behaviour including considerations of the system feedback is arbitrarily programmable and adaptable. The concept can be extended to model subscribers and networks operating with new services, which will be provided in current and future system software developments, e.g., in ISDN-featured systems (ISDN: integrated services digital network).

REFERENCES

1. Gaissmaier, B., Schirmeier, H.: An integrated software development method for switching systems based on CCITT's SDL. Fifth International Conference on SETSS, Lund, Sweden, 1983

2. Lemppenau, W., Tran-Gia, P.: A universal environment simulator for SPC switching system testing. 11th Intern. Teletraffic Congress, Kyoto, Japan, 4.-11. Sept. 1985, Paper 5.1B-3

3. Metzger, R.M., Staber,E.: Anrufsimulator UCS für die Prüfung von Fernleitungsausrüstungen. Elektrisches Nachrichtenwesen 55, 1980, 210-216

4. Dael, G., Amarger, D., Marrois, C.: AROMAT telephone call monitoring and recording equipment. Communication and Transmission 1983, 39-52

5. Gruszecki, M.: ENTRASIM: Real time traffic environment simulator for SPC switching systems. Proc. 8th ITC, Melbourne, Nov. 1976

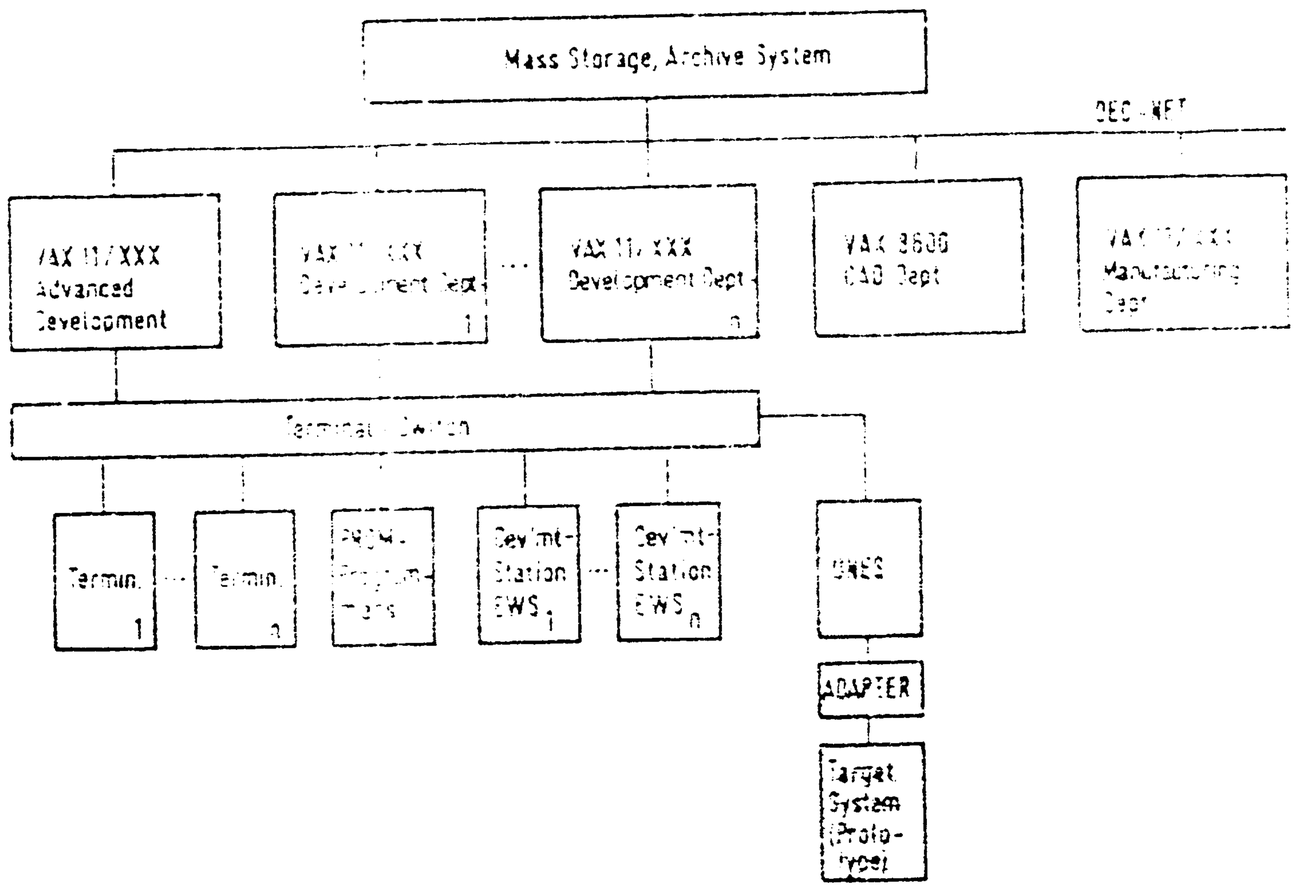

Figure 1 Software development environment

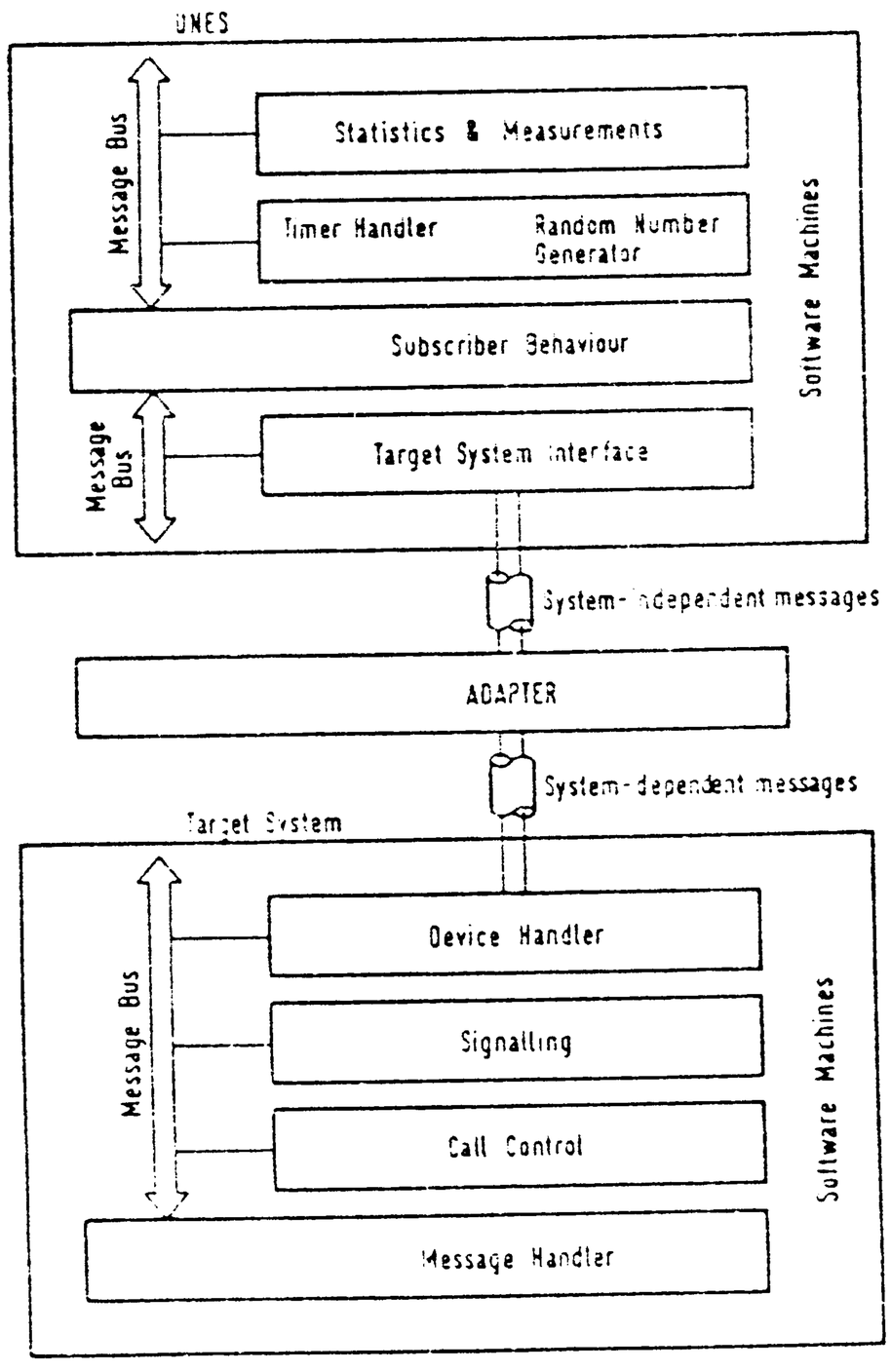

Figure 2 Functional interworking of UNES and target system

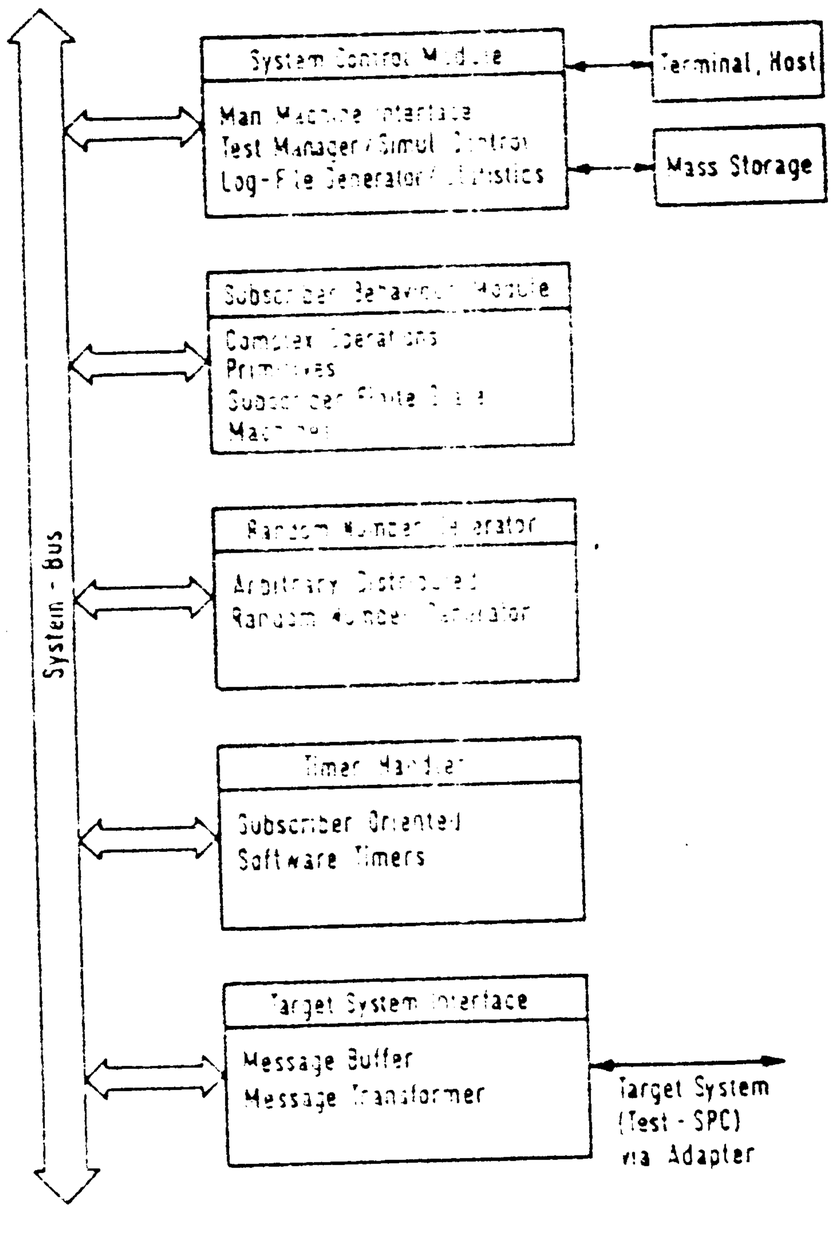

Figure 3 System structure of UNES

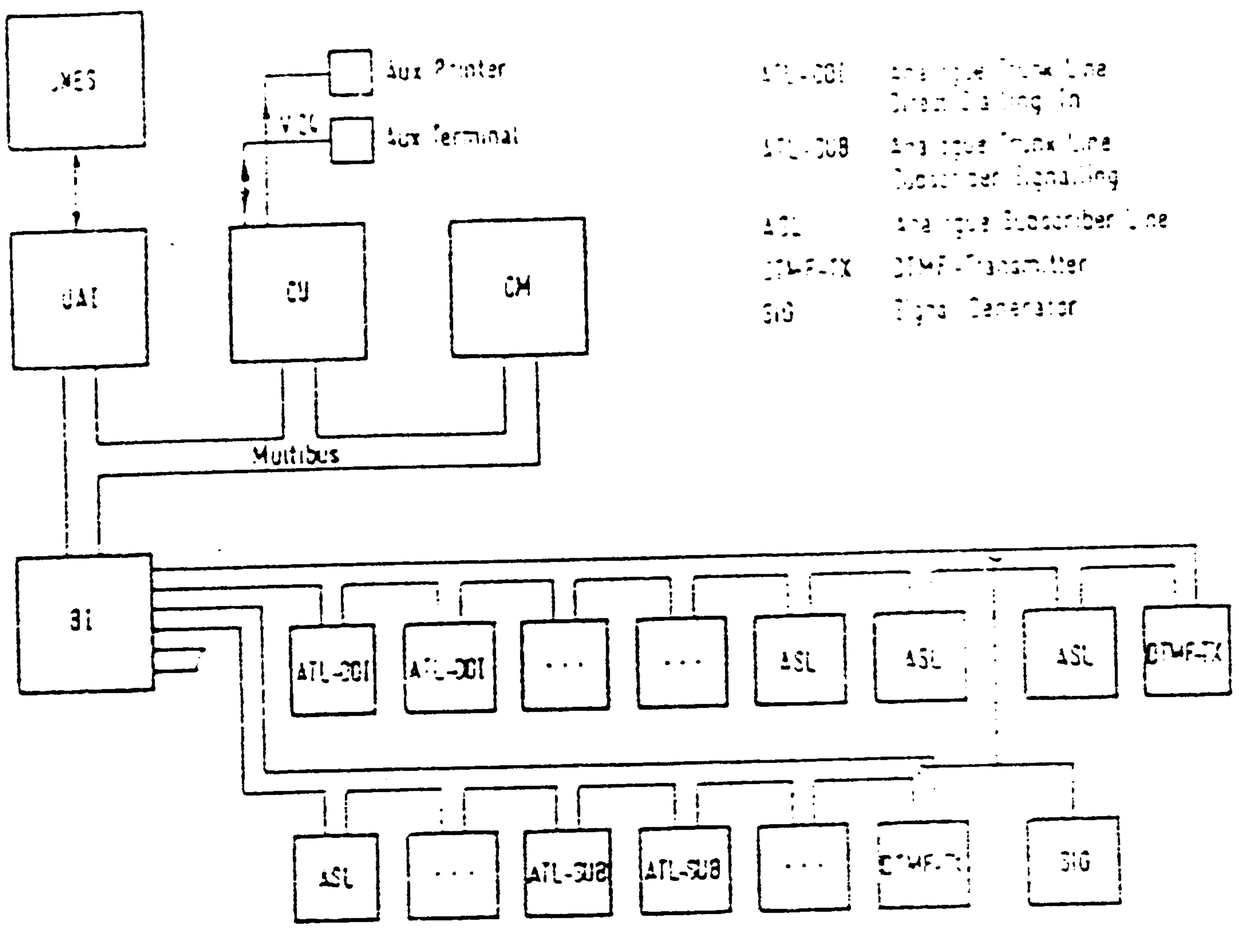

Figure 4 System structure of ADAPTER

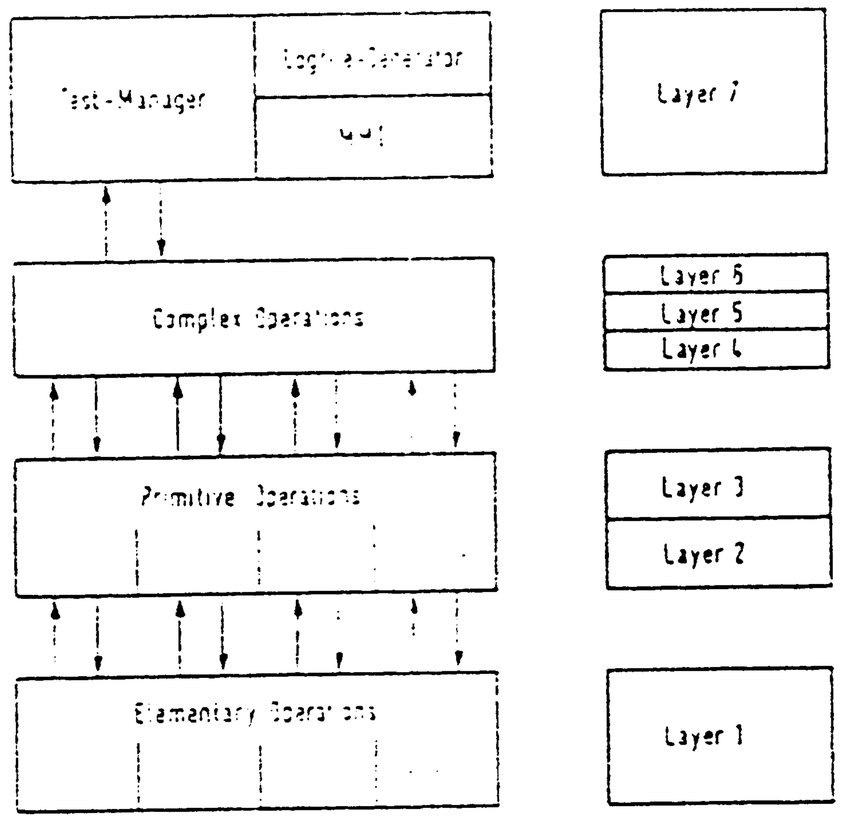

Figure 5 Layered software for functional t

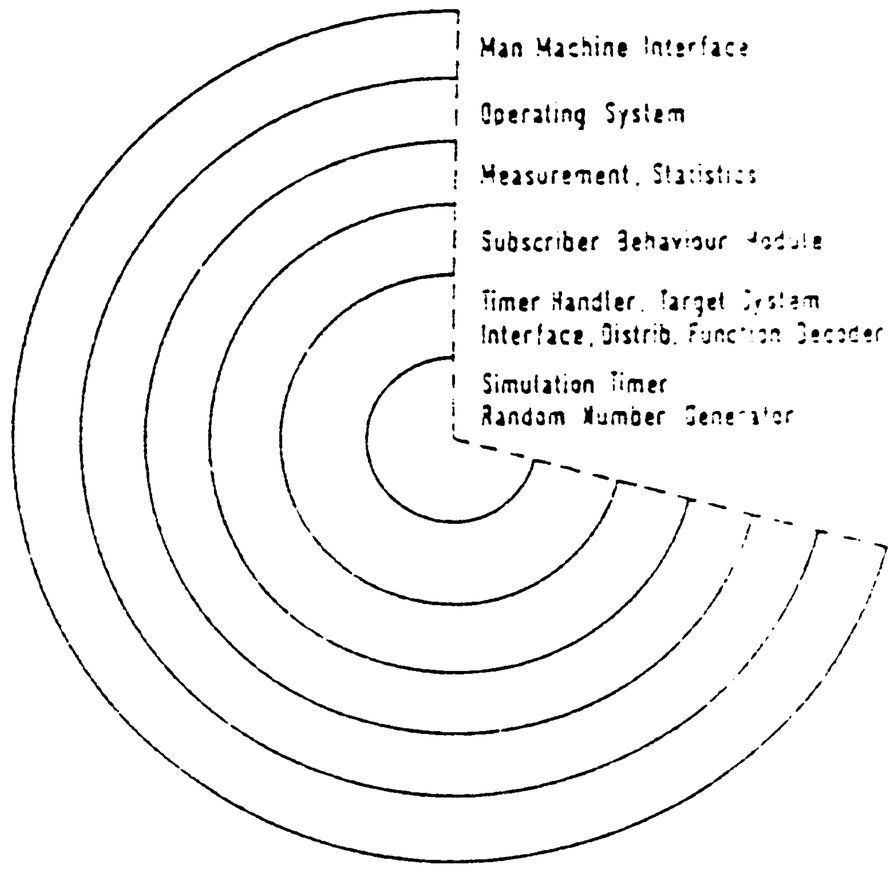

Figure 6 Software structure for testing under load conditions

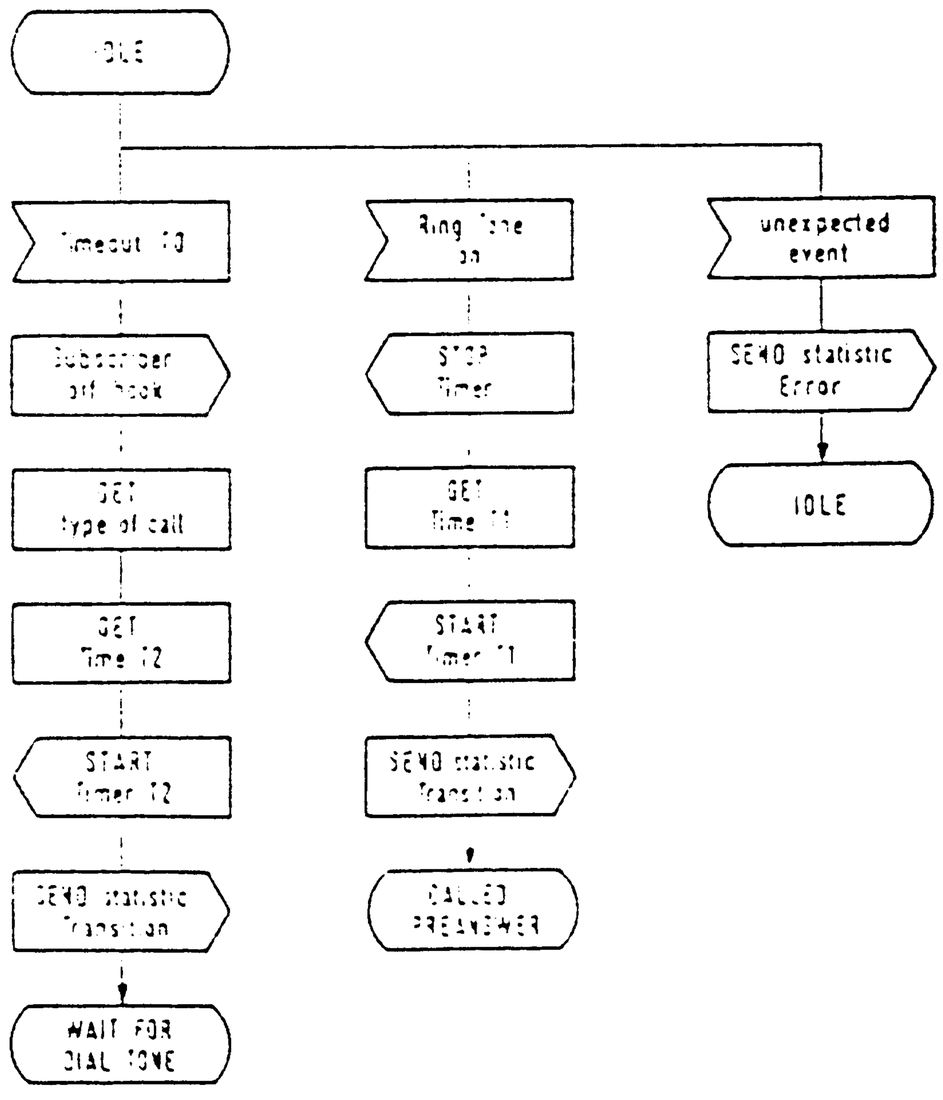

Figure 7 Example of the modelling and specification of a subscriber's behaviour