Elektrotehniški vestnik 69(5): 259–265, 2002
Electrotechnical Review, Ljubljana, Slovenija

Analog Subscriber Call Generator

Boštjan Vlaovič, Zmago Brezočnik

University of Maribor, Faculty of Electrical Engineering and Computer Science,
Smetanova ulica 17, 2000 Maribor, Slovenia
e-mail: {bostjan.vlaovic, brezocnik}@uni-mb.si

Abstract. The paper describes the development of the analog subscriber Call Generator (CG) for the SI2000 V5 switch node. It provides developers with the test environment without the use of external testing units. The proposed architecture assures effective tests of program layers above the driver of analog subscriber circuits. This was accomplished with minimal changes of the existing program code and application of the existing communication procedures. Compared to external testing units, the software Call Generator possesses some limitations. Because of the software nature of its operation, electrical interfaces on analog subscriber line circuits are not tested for proper operation. Also, usual audio checks of the connection are not performed. The call generator can be extended with this functionality, but this would demand greater changes of the existing program code and an additional use of digital signal processing resources. This would impose unnatural performance of the switch node under test. Testing without using external testing units represents the main achievement for the developer. For the CG operation, only an IP connection between the developer's workstation and the tested switch node is required. Developers can control and monitor the execution of tests from their development workstation with any WWW (World Wide Web) browser.

Key words: telecommunications, testing, SDL, call generator, switch node

Generator klicev analognega naročnika

Povzetek. Prispevek predstavlja razvoj programske opreme za telefonsko centralo SI2000 V5. Opisuje programski generator klicev analognega naročnika. Generator omogoča testiranje programske opreme brez uporabe zunanjih testirnih naprav. S primerno zasnovo smo zagotovili učinkovito testiranje programske opreme, ki se nahaja nad gonilnikom vezij analognih naročnikov. To smo dosegli z minimalnimi spremembami obstoječe programske opreme in ponovno uporabo obstoječih mehanizmov komunikacije med sloji programske opreme. V primerjavi z zunanjimi testirnimi napravami ima generator določene pomanjkljivosti. Zaradi programske zasnove se v času izvajanja testov ne preverjajo električni vmesniki na vtičnih ploščah analognih naročnikov. Dodatno se ne preverja t.i. slišnost povezave, ki ugotavlja pravilno povezavo dveh naročnikov. Z razširitvami bi se ta pomanjkljivost lahko odpravila, vendar dodatni programski posegi in zaseganje virov zmanjšujejo smiselnost takšne razširitve. Bistvena pridobitev s stališča razvijalca izvira iz samostojnega delovanja, ki ne predvideva uporabe zunanjih testirnih naprav. Poseben poudarek pri razvoju je imela prostorska neodvisnost uporabnika. Za normalno delovanje zadostuje povezljivost delovne postaje in testne centrale po protokolu IP. Za nadzor generatorja in prikaz rezultatov zadostuje katerikoli spletni brskalnik na razvijalčevi delovni postaji.

Ključne besede: telekomunikacije, testiranje, SDL, generator klicev, telefonska centrala

Received 29 July 2002
Accepted 13 November 2002

1 Introduction

The telecommunication market is changing rapidly. The manufacturers and carriers need to adapt quicker than ever. The software development cycle is getting shorter and shorter. Only a proper development methodology can ensure quality of the product. Specification languages, theorem provers, and model checkers are beginning to be used in industry. Testing of products takes a considerable part of the development process. Tests provide a systematic evaluation of the product's functionality against its specification through its development process. Reviews of the program code and tests should be performed at each development process phase.

In this paper a new method for testing the SI2000 V5 switch node is introduced. It cannot fully replace other testing methods, but provides a cheap, quick, and practical test environment for the developer.

The architecture of the mentioned digital switch node is presented in Section 2. A basic description of the switch node and its analog subscriber configuration is given. Next, a general description of the Specification and Description Language (SDL) is provided in Section 3. It is used for the specification and description of structural and behavioral aspects of the system. In Section 4, SDL architecture of the SI2000 V5 digital switch node is

presented. The design, development, and operation of the Call Generator are described in Section 5. In Section 6, we comment on the results and give directions for further work.

2 SI2000 V5 digital switch node

IskraTEL is the largest Slovenian telecommunications company for development, marketing, planning, manufacturing, installing, and servicing of telecommunication systems. It was established in 1989 with the capital of Slovene investors and the German company Siemens. The current generation of its digital switch node is called SI2000 V5. Its capacity spans from a few hundred to several thousand ports. It is an advanced modular system that offers basic functionality as well as a wide range of services including PSTN, ISDN, SS7, H.323, Centrex, IP Centrex, etc.

SI2000 V5 has different modules. The paper focuses only on MLC (Line Module Version C). The task of MLC is to connect analog subscribers, ISDN terminals, H.323 terminals, and network transmission paths. It can perform functions of an access node, PBX system, or small local exchange. The MLC's functionality is defined by its hardware configuration and loaded software. Sub-rack configuration of MLC incorporates a main control unit, power supply unit, and peripheral units (ISDN and analog subscribers). The system connects to the IP network via Ethernet adapter.

Peripheral units are inserted in the MLC sub-rack. The MLC module can hold 22 peripheral plug-in units. They are connected in a form of a star, thus minimizing the mutual influence among plug-in units and allowing plug-in unit replacement under voltage.

The SAC (Analog Subscriber Unit) plug-in unit serves for a two-wire (a/b) connection of analog terminals. There are 32 analog subscriber line circuits on the plug-in unit, thus providing MLC module with a maximum of 704 analog subscribers. Usually a duplication of the main unit and power supply unit is used, so the maximum number of analog subscribers drops to 640.

The core of the modern switch node is its software. It provides functionality, control, and management of the system. Most of the software is implemented with SDL. Various drivers for the peripheral devices and smaller parts of the protocol stacks are implemented with C and C++ programming languages.

3 Specification and Description Language

SDL was developed by the switching systems industry and was first standardized by CCITT (Comité Consultatif International de Télégraphique et Téléphonique) in 1976 [1]. It is based on finite state machines, but it uses graphical representation of flowcharts to show allowed transitions. In the development cycle, SDL is employed for the formal specification and design of the system. SDL supports specification and description of structural and behavioral aspects of the application under development. It is a formal description technique (FDT). Its dynamic semantic is formally defined with a combination of Meta-IV and CSP (Communicating Sequential Processes) [2].

SDL may serve a number of purposes, from reasoning about systems at an abstract level to the automatic derivation of implementations. Nowadays, several commercial and academic tools are available that support the development of systems with SDL. Tool support comprises graphical editing, validation, verification, simulation, animation, code generation, and testing. We use Telelogic's Geode and ObjectGeode.

At the highest level of the SDL hierarchical specification is an object called *system*. The system is an entry point to the SDL specification. It comprises a set of *blocks* and *channels*. Blocks are connected with each other and with the environment by channels. The hierarchy in SDL is a static structuring concept. Block is described by sub-blocks or set of processes. Blocks are static entities — they are created during the initialization of the system. Communication between blocks is only possible along the defined channels. Channels are asynchronous and can be unidirectional or bidirectional. Blocks are finally refined into processes.

A process is defined by a process graph. Each branch of the graph represents a possible execution of the process. Processes describe the system behaviour. Communication between different processes, and between processes and the block interface is done via signal routes. Signal routes are non-delaying. Signals are the primary communication mechanism in SDL. Each process has an unbounded queue at the input port. The port allows received signals to be queued until they are consumed or discarded by the process instance.

Process instances can be created either during the initialization of the system or dynamically during the execution of the system. Within the process declaration a dynamic range of the allowed number of instances can be set. Each process instance in an SDL system represents an independent asynchronously executing CEFSM (Communicating Extended Finite State Machine). A process instance may be executed as soon as one of its trigger conditions holds. Transitions of a single process instance are executed sequentially. Interleaving of different actions concurrently executed by different processes are eliminated in SI2000 V5 system by implementation of atomic transitions.

4 SI2000 V5 Software Architecture

The SI2000 V5 system software consists of operation and maintenance functions, control functions, and connection

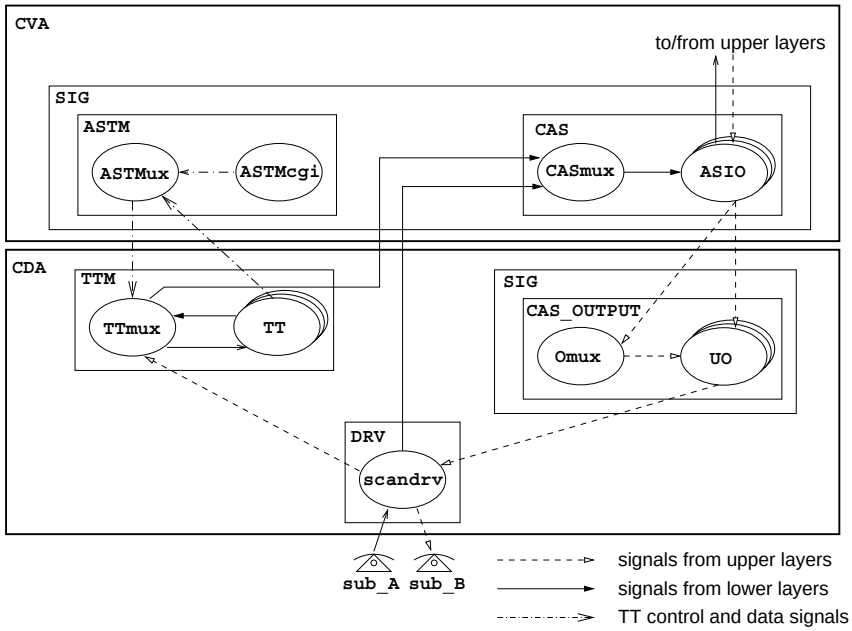

Figure 1. SDL architecture of Call Generator and its environment

functions [3, 4]. Operation and maintenance functions include real time operating system pSOS (plug-in Silicon Operating System), database support and diagnostic functions. Control functions are the core of the switch node. These functions are required to control services and connections, e.g. signalling, routing, and connection/resources handling functions. Connection functions are directly related to the connection path through the exchange, i.e. switching and transmission mechanism.

Figures 1 and 2 show the architecture of the SI2000 V5 SDL code. It consists of two main blocks. CDA (Communications Controller Version A) block includes drivers for peripheral units, scanner for events on analog subscriber lines, DSP (Digital Signal Processing) driver, diagnostic and test program blocks, and initialization procedures. CVA (Central VME Processor Unit Version A) block includes signalling protocols, signalling control, connection control, call control, services, tariff information, and management. We will focus only on parts relevant to the analog subscriber that uses pulse dialling. The SI2000 V5 signalling control layer unifies pulse and tone dialling for the call control layers.

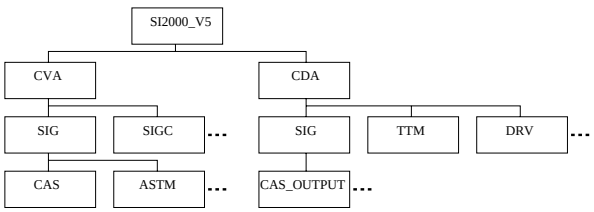

Figure 2. SI2000 V5 SDL program architecture

The program code is executed in real time. That means it responds to the stimuli from the environment at the predefined times. Because of the system architecture, a change on the analog subscriber line is detected in 4 ms at the latest. Changes on analog subscriber ports are recorded in a special table transfered with the SDL signal `grouped_status_change` (Fig. 3) from the `scandrv` process to the upper layers (`CASmux` process) (Fig. 1).

`CASmux` manages `ASIO` (Analog Subscriber Input/Output) processes and acts as a multiplexer for the communication with the lower layers. Each analog subscriber has its own `ASIO` process instance. `CASmux` parses the received signal from `scandrv` and forwards changes to the appropriate `ASIO` processes. Control of analog ports is managed by the `ASIO` process through the `Omux`, `UO`, and `scandrv` processes in the CDA block (Fig. 1). The partial analog subscriber signal path for a simple call from port A to port B is shown in the MSC (Message Sequence Charts)[5] diagram in Fig. 3.

5 Analog Subscriber Call Generator

The analog Subscriber Call Generator (CG) has been developed for the basic functionality testing during the program development [6]. Until now, developers had to use external testing equipment if they wanted to test their code against its specification. The main idea was to create software processes that would follow a predefined scenario and provide the switch nodes' existing program code with the same stimuli as would be received from the environment (analog subscriber port).

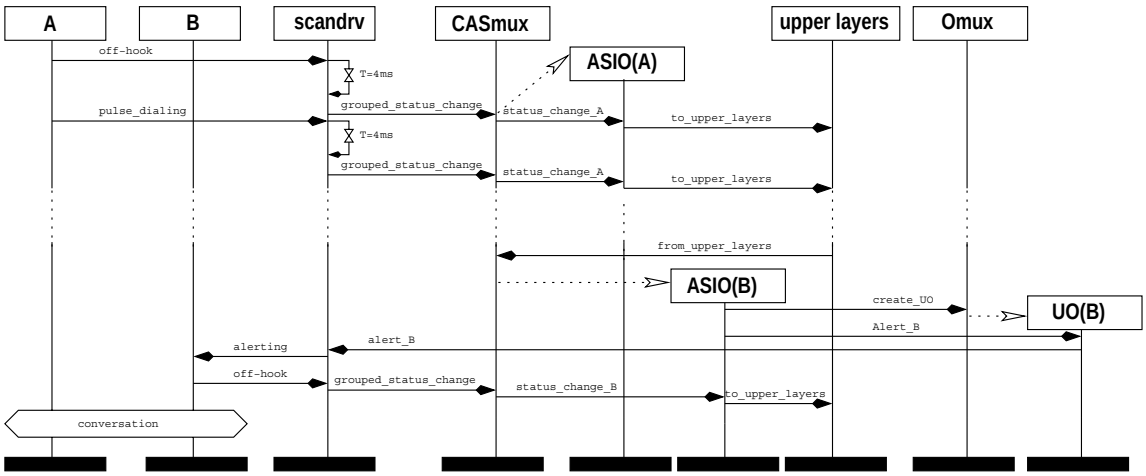

Figure 3. Simplified MSC diagram of a telephone call from port A to port B

The main objective was to provide every developer with the test environment that would require only IP connection between his/her workstation and tested switch node. SI2000 V5 switch nodes have installed a small HTTP (Hyper Text Transfer Protocol) server. User interface for the CG is provided through the WWW browser. Remote control of CG enables testing of already installed switch nodes in Russia, Turkey, etc. Additionally, developers are not limited to a testing site in the company, but can control and monitor the execution of tests from his/her development computer with any WWW browser. We will not deal with security questions and will assume that they have been solved at the IP level.

5.1 Program architecture and configuration

Tests should cover as much of the existing software as possible, so CG should introduce only small changes to the existing program code. Generation of stimuli from the virtual subscriber should be done as close to the usual signal origin as possible. The natural place for the virtual telephone functionality is within the CDA block. The core of the CG is in the `TTM` (Test Telephone Module) block (Fig. 1). It consists of two processes. Process `TTmux` (Test Telephone Multiplexer) manages `TT` (Test Telephone) processes and acts as a multiplexer for the communication with upper layers (`CASmux` process). It collects changes on ports and forwards them to the `CASmux` process with the same SDL signal as the `scandrv` process would. With this architecture, SDL signals reporting changes on the virtual subscriber ports traverse the same route as the original ones. Complete SDL architecture of CG and its environment can be seen in Fig. 1. Changes in the existing code have to be made only to the `scandrv` process. All control signals from upper layers for the analog ports are now forwarded also to the `TTmux` process. It checks if the control information is destined for one of the test telephones and forwards it to the appropriate `TT` process. All logic for test scenario execution resides in the `TT` process which is dynamically created upon the start of CG.

The management part of the CG is built in the CVA block. It consists of two processes: `ASTMux` (Analog Subscriber Test Module) and `ASTMcgi` dummy process. The latter includes operators that implement CGI (Common Gateway Interface) programs for interactive control and WWW user interface for CG. The dummy process has only one state with no actions:

```
PROCESS ASTMcgi(1,1);
  START ;
    NEXTSTATE S010_DUMMY;
  STATE S010_DUMMY;
    INPUT *;
      NEXTSTATE -;
  ENDSTATE;
ENDPROCESS ASTMcgi;
```

Its role is to define external operators and signals for interaction with the rest of the SDL program code. Operators are implemented in the C programming language. SDL signals are generated within the C program code. It is a one way communication. These signals are generated on the user's demand through the WWW interface (Fig. 4). The user can instruct `ASTMux` process to execute control actions over virtual telephones, refresh test scenario, or delete statistical data.

User interaction with the CG is possible, but it is not obligatory. It can operate in an interactive mode or standalone mode based on its configuration data. At the power up of the switch node, `ASTMux` process checks the database for CG configuration data. Separate records are used for the configuration of the CG, test telephones, and test scenarios. Additional records are available for the statistical data about scenario executions. Based on the configuration data, `ASTMux` process instructs the `TTmux` pro-

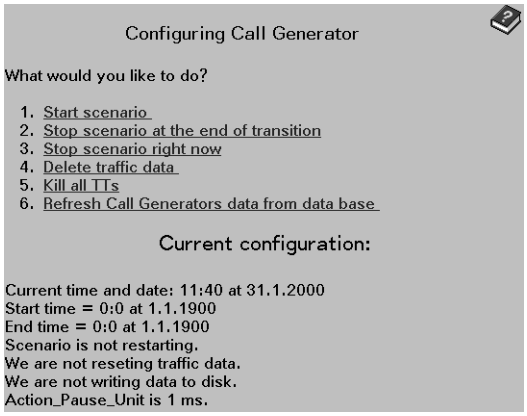

Figure 4. CG control interface

cess to create the expected number of `TT` processes with the prepared test scenarios. Each `TT` process executes its scenario independently. Behaviour of CG depends on its general configuration data.

5.2 General Configuration

General configuration data include the following information: start time, stop time, restart period, instructions for the statistical data management, and definition of the `action_pause_unit` parameter. The first two can be predefined for the automatic start and stop of CG at an appropriate date and time. They can be used to execute tests while test switch node is not occupied, i.e. during the night. Each test telephone (`TT` process) is executing its scenario independently (asynchronous operation). Restart period enables user to restart execution of all test telephones from the start of their scenarios. This is some sort of synchronisation that helps scenario inconsistency tracking. During the execution, each change in the test telephone state is regularly reported to the `ASTMux` process. Statistical data on test execution are based on these reports. For each test telephone several statistical data are collected: number of successful diallings, number of successful outgoing calls (the called party accepts the call), number of incoming calls, number of accepted calls, and current state (idle, dialling, ringing, talking). The current status of each test telephone can be presented through the WWW user interface (Fig. 5). The user can choose to reset or keep the statistical data upon the restart of CG. Additionally, data can be written to the database on a regular basis. This option is especially useful if tests crash the switch node. If the statistical data are written to the database, they can be analysed on the reboot of the switch node. The last option, `action_pause_unit` parameter, defines the time unit to be used in the test scenario specification. Test scenario specifications include a natural number which represents the multiplier of the `action_pause_unit`. It can be set in 1 ms resolution. Default value is 10 ms. With the change of the `action_pause_unit` parameter all test scenario timings are changed.

This concludes the description of the general configuration data. We continue with the discussion of the individual test telephone configuration data.

5.3 Test Telephone Configuration

The call generator assumes that no MLB will be equipped with more than 22 peripheral plug-in units. Each analog subscriber plug-in unit holds 32 analog subscriber circuits. Changes at ports are reported through STbus (Split Transaction Bus) serial connection to the main unit. Each STbus holds 32 channels. The analog subscriber port is uniquely defined with the STbus number and channel number. Each test telephone has to be assigned to the equipped analog subscriber port for proper operation.

Test telephone processes are created by `TTmux` process based on the configuration data. Beside STbus and channel pair, some other attributes are available. To avoid a concurrent start of all virtual telephones, which would be unrealistic, start delay can be set for each `TT` process. Other attributes define default time values for the execution of the test scenario.

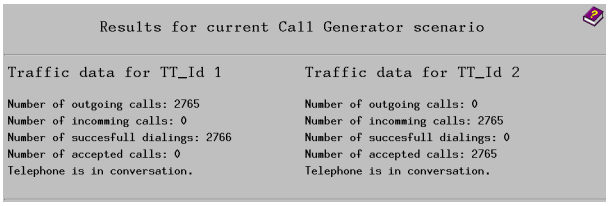

Figure 5. Test telephone statistical data

Test telephone behaviour is described with a test scenario. It consists of incoming and outgoing actions. Incoming actions define user's behaviour after alerting has been received. Outgoing actions define user's activities for outgoing calls. If there is no outgoing action, `TT` process only accepts incoming calls. Defined actions are shown in Tab. 1.

Each action is followed by a pause that can be explicitly defined or chosen between different random values. Additional diversities in the test telephone operation can be provided with the unique definitions of the random call duration time limits, short pause unit (100 ms resolution), long pause unit (1 s resolution), and random pause unit limits. With proper values different subscriber behaviours can be simulated — fast or slow typing, varying duration of conversation, unexpected call termination, etc. A practical demonstration of the test scenario for two virtual analog subscribers is given in the following subsection.

Code	Action
0-9	numbers from 1 to 10
11	*
12	#
33	*RR*
20	*off-hook*
30	*on-hook*
40	*GND_on*
50	*GND_off*
66	*End_of_Selection*
99	*End_of_Scenario*
80	*No_Operation (NOP)*
100	*Wait_for_Ring*

Table 1. Code for defined user actions

5.4 Practical example

For proper operation of the CG, the configuration data have to be set according to the switch node's hardware configuration and test scenario specification. CG can operate in the interactive or stand-alone mode. We will describe interactive operation with live database update, start, and stop of the test scenario execution. Default values assume interactive operation. The user has to change only data that are relevant to the chosen test scenario. We assume that the tested switch node is running and can be reached over the IP network.

Our example will include only two virtual analog subscribers. First, general CG configuration has to be set. No changes to the general configuration database records are necessary, unless we are not satisfied with the 10 ms time unit (`action_pause_unit` parameter), or specific instructions for the CG restarts and statistical data management are defined.

Next, test telephones have to be defined. If the test scenario does not require the use of the random time values, only STbus, port, and start delay attributes have to be set. Their values depend on the current switch node's hardware configuration. Access to the database is accomplished with the SQL (Structured Query Language) query and `irtsql` program which is part of the development environment. Setup of the two test telephones can be performed from the user's workstation with the following commands:

```
irtsql switch_node_IP INSERT INTO \
       test_telephone VALUES (1,14,1,1)

irtsql switch_node_IP INSERT INTO \
       test_telephone VALUES (2,14,11,2)}
```

The first value represents the `TT` process ID, the second holds the STbus number, the third is port number, and the last one defines start delay in 100 ms resolution. The second test telephone starts its test scenario execution 200 ms after the start of CG. Test scenarios are defined with the incoming and outgoing actions which are presented in Tab. 2.

tt_id	No.	in_a	in_p	out_a	out_p
1	1	99	NULL	20	300
1	2	NULL	NULL	2	100
1	3	NULL	NULL	3	100
1	4	NULL	NULL	5	100
1	5	NULL	NULL	4	100
1	6	NULL	NULL	66	1500
1	7	NULL	NULL	30	100
1	8	NULL	NULL	99	100
2	1	100	2	99	NULL
2	2	20	500	NULL	NULL
2	3	30	100	NULL	NULL
2	4	99	300	NULL	NULL

Table 2. Test scenarios for two test telephones

Test telephone with `tt_id=1` (TT1) will not accept calls since `End_of_Scenario` command is specified as its first input action (Tabs. 1 and 2). It will start executing the outgoing test scenario after the specified start delay (100 ms). `Off-hook` command (Tab. 2) is followed by `300 * action_pause_unit` seconds of pause. Next, telephone subscriber number 2354 is dialled (Fig. 6). Mapping between the subscriber number and STBus/port depends on the switch node's configuration and is not part of the CG. For proper test scenario preparation, user should be acquainted with the configuration of the tested switch node. We will assume that the subscriber number 2354 maps to STBus=14 and port=11 (TT2). Virtual subscriber waits for 1 second after each selected number. Successful dialling is reported to the `ASTMux` with the `End_of_Selection` command. TT1 terminates the call after 15 seconds.

The second test telephone (TT2) will not perform any outgoing actions. It will only accept incoming calls. When `Wait_for_Ring` command is issued, the input pause parameter (`in_p`) gets a special treatment. It represents the number of rings the TT process should wait before accepting the incoming call. In our example, the call will be accepted after two rings with the `off-hook` command. TT2 terminates the call after five seconds. This is the simplest possible scenario for non-problematic call establishment.

Table 2 with the scenario description is written to the database following the same procedure as described earlier. Next, the user should refresh CG's configuration. This is accomplished through the control interface (Fig. 4). Now, test scenario execution should be started. `ASTMux` process receives the start signal from the

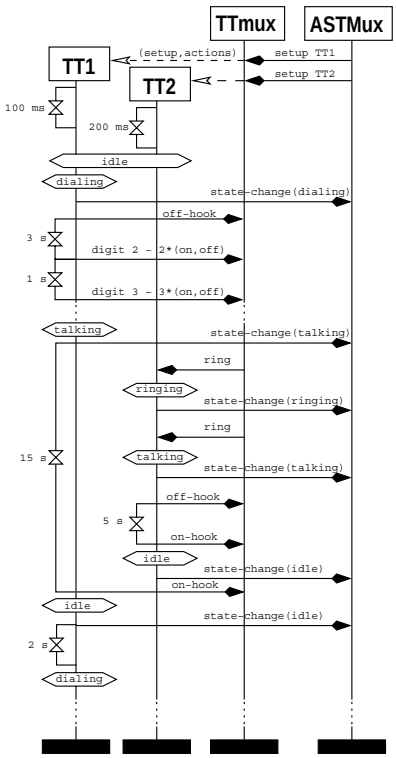

Figure 6. MSC diagram for the test scenario

`ASTMcgi` process. To be exact, the SDL signal is generated within external operators of the `ASTMcgi` process. `ASTMux` process instructs `TTmux` process to notify each `TT` process to start executing its scenario. The test telephone repeats the execution of its test scenario until it is instructed to stop.

Each change in the test telephone state is reported to the `ASTMux` process. It keeps track of all the changes for each `TT` process. This data can be accessed through the WWW interface (Fig. 5) or from the database — if we are using this feature. CG stops the execution of the test scenario after user's intervention or when stop time has been reached. Stop time has not been defined within the presented setup, so the user should stop the execution through the control interface (Fig. 4).

Users access switch node from their development workstation. Different scenarios can be quickly prepared with the help of the simple shell scripts. Access to the database is assured with `irtsql` program, while the control interface and statistical data presentation are provided through the WWW user interface.

6 Conclusion

The analog Subscriber Call Generator provides developers with the test environment without the use of external testing units. The proposed architecture assures effective tests of program layers above the driver of analog subscriber circuits. It is successfully used within the company.

Compared to external testing units, software Call Generator possesses some limitations. Due to the software nature of its operation, electrical interfaces on analog subscriber line circuits are not tested for proper operation. Additionally, usual audio checks of the connection are not performed. Consequently, tests cannot guarantee that connection between two subscribers has actually been performed in the multi-subscriber test environment. The call generator can be extended with this functionality, but it would demand greater changes in the existing program code and additional use of digital signal processing resources. This would impose unnatural performance of the switch node under test.

Further improvements are possible on the WWW user interface. It can be extended with software agents to present the current status and history of each test telephone in a more practical and graphically attractive way. A detailed history of the test telephone operation would enable additional off-line analyses of the switch node performance.

7 References

[1] ITU-T Recommendation Z.100, *"CCITT specification and description language (SDL)"*, Series Z: Programming Languages, ITU-T, 1993.

[2] A. Mitschele-Thiel, *"Systems Engineering with SDL"*, England, WILEY, 2001, pp. 141-150.

[3] R. Slatinek, *"Primerjava signalizacij v vmesnikih ISDN"*, Maribor, Faculty of EE & CS, University of Maribor, 2000, pp. 43-51.

[4] ITU-T Recommendation Q.521, *"Digital exchange functions"*, Series Q: Digital Exchanges, ITU-T, 1993.

[5] ITU-T Recommendation Z.120, *"CCITT Message Sequence Chart (MSC)"*, Series Z: Programming Languages, ITU-T, 1993.

[6] B. Vlaovič, *"Generator klicev za telefonsko centralo MLB SI2000 V5"*, diploma, Maribor, Faculty of EE & CS, University of Maribor, 1999.

Boštjan Vlaovič (Student Member, IEEE) received his diploma in Electrical Engineering from the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia, in 1999. He is in his third year of the Ph.D studies at the same faculty and works as a researcher in the field of telecommunications. His special interests cover voice communications over packet based networks and their integration with traditional PSTN. His current research interests include formal protocol verification, especially symbolic model checking.

Zmago Brezočnik (Member, IEEE) received his M.Sc. and Ph.D. degrees from the University of Maribor, Faculty of Electrical Engineering and Computer Science, in 1986 and 1992, respectively. He is currently associate professor, head of Laboratory for Microcomputer Systems, and deputy dean of education at the same faculty. His main research areas are formal hardware and protocol verification, especially symbolic model checking, and binary decision diagrams.