Copyright © 1979 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 58, No. 1, January 1979
Printed in U.S.A.

Advanced Mobile Phone Service:

Development Support Systems

By S. H. TSIANG

(Manuscript received July 24, 1978)

The Advanced Mobile Phone Service system contains four major subsystems, each of which has its own processor and software. To support the development of such a complex system, several laboratory test systems have been developed. They are for simulation, software debugging, overall system testing, and performance monitoring. These test systems, briefly described in this article, have been invaluable tools to the AMPS development.

I. INTRODUCTION

The Advanced Mobile Phone Service (AMPS) system is composed of four major elements: the Electronic Switching System, the data link, the cell site, and the mobile unit. In the Chicago Developmental System, the switching functions are performed by a No. 1 Electronic Switching System (ESS)[1] in conjunction with a programmable controller (PROCON) residing in each of the cell sites. The data link equipment is an adjunct to ESS. It has its own processor and memory and runs autonomously. The mobile unit is controlled by a resident microprocessor. Thus, processing a mobile call involves four separate processors. Each of these must be tested individually and then as a system.

Several laboratory test systems have been built to support the AMPS development. This paper describes those that have been or are being used for simulation, software debugging, integration testing, and data gathering. They are:

(*i*) A Mobile Office Simulated Environment System (MOSES) that simulates the cell sites and mobile units for testing the ESS and data link software.

(*ii*) A Maintenance and Traffic Simulator (MATS) that simulates cell site peripherals, mobile units, and mobile telephone users to support cell site load testing and cell site controller software development.

(*iii*) A Mobile Call Generator (MCG) that produces mobile traffic and monitors results using real mobile units for overall end-to-end testing of the system.

(*iv*) A cell site response generator (a utility software package) that simulates cell site responses for ESS software testing.

(*v*) A Data Retrieval System (DRS) that collects data for monitoring performance in an operational switching system.

Figure 1 is a simplified block diagram of AMPS and the points at which each test system is connected physically or symbolically.

II. MOBILE OFFICE SIMULATED ENVIRONMENT SYSTEM

Planning for switching software testing started at the exploratory stage of the AMPS project. It was recognized then that the cell site hardware would not likely be available during the early stage of software testing. Therefore, as a part of the exploratory work, a simulator that would be able to simulate the external environment was designed; this was named the Mobile Office Simulated Environment System (MOSES).

MOSES can simulate any number of cell sites and mobile units and communicates with the Mobile Telephone Switching Office (MTSO) via data links and voice trunks. It assembles the messages it receives, analyzes them, generates appropriate responses, and sends them back to the MTSO.

2.1 Hardware

The MOSES controller consists of a minicomputer and a complement of general-purpose I/O devices. The computer system processes and simulates digital signals. For analog signal detection and simulation,

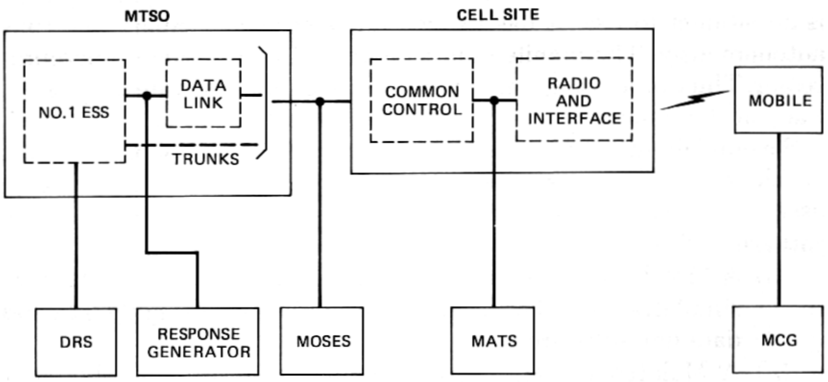

Fig. 1—Development support systems.

such as signaling on cell site voice trunks, standard ESS equipment is used.

2.2 Software

To make the simulator easier to use, MOSES is programmable in a high-level language, called ETSL (Extended Telephone Simulation Language). This greatly simplifies the design of users' test programs that specify a series of actions to be taken by MOSES; for example, "originate from a mobile and dial a certain number."

MOSES can process many independent test programs concurrently. It reads the users' inputs and converts them into execution blocks. Each block contains a word specifying the operation to be performed and a string of pointers that lead to the parameters necessary for performing the operation. The parameters are stored in a list built by MOSES. The word specifying the operation in each block identifies a function table. The function table contains pointers to routines that accomplish the required operation.

A MOSES run-time table-driven execution program performs sequential processing of execution blocks and their related tasks.

MOSES was fully implemented and used during initial software testing on the ESS test model. It was also employed extensively in debugging the software resident in the data link equipment. The first successful mobile call was placed through MOSES before the arrival of the cell site equipment in the test laboratory.

Because of the favorable experiences with MOSES, a second and expanded simulator, MATS (Maintenance And Traffic Simulator), was built for the Oak Park MTSO. The No. 1 ESS in that office was used, prior to the start of the Chicago Developmental System trial, as a major test facility for the AMPS software development. This included software residing in both ESS and the cell site controller.

III. THE MAINTENANCE AND TRAFFIC SIMULATOR

The MATS is a minicomputer-based system that simulates the actions of major cell site peripherals, mobile units, and mobile telephone users. It provides load testing and debugging support for the cell site controller (PROCON) software, and debugging support for cell site maintenance programs residing in the MTSO. MATS is capable of generating several thousand mobile calls per hour. It is intended to apply representative loads of various types to the system, including a load that approaches the design capacity of a single cell site. Since MATS is connected to the AMPS at input and output buses of a cell site controller, simulation is confined only to the cell site where the interface is made.

To provide the above functions and to simulate the cell site peripherals, MATS must:

(*i*) Monitor the cell site controller output for orders being sent to peripherals.

(*ii*) Simulate the actions caused by these orders, generate the appropriate response, and supply that response to the cell site controller at the proper time.

(*iii*) Provide on-hook, off-hook, and voice path verification on test trunks.

(*iv*) Accept user commands to control the actions of the simulated mobile units.

MATS employs the same high-level user input language as that used in the MOSES test system, ETSL. It has commands which allow the user to specify mobile unit actions (e.g., originate), check for mobile conditions (e.g., off-hook), and control program flow.

3.1 MATS hardware

The primary hardware components of the MATS hardware are the minicomputer and associated peripheral equipment, the cell site controller output bus monitor, the cell site controller input simulator, and the voice trunk control and monitoring equipment. These hardware components are shown in Fig. 2 and described in more detail in the following paragraphs.

3.1.1 Minicomputer

The minicomputer system consists of a processor with memory, a CRT display terminal, a magnetic tape unit, two cartridge disk drives, a card reader, and a high-speed printer.

3.1.2 Output monitor

The output bus monitor collects data being sent on the cell site controller output bus, as well as the peripheral address to which the data are being sent. It also monitors data being sent to the controller from a peripheral device and the sending device's address. The MATS software has the ability to selectively disable the monitoring of data to or from any peripheral. The information collected by the output monitor is buffered in a first-in/first-out memory and made available to the MATS software for analysis.

3.1.3 Input simulator

The input simulator allows MATS-generated data to be sent to the cell site controller as though they were coming from a real peripheral. Since MATS does not know at what time or in what order the cell site controller will read its peripherals, MATS software must be able to send the simulated data to the input simulator in advance and at any time. Therefore, an input simulation buffer memory is provided. Because the timing is critical between the read and the fetching of the simulated data, the speed of the buffer memory is a major consideration.

Simulated data may be sent to the cell site controller for any

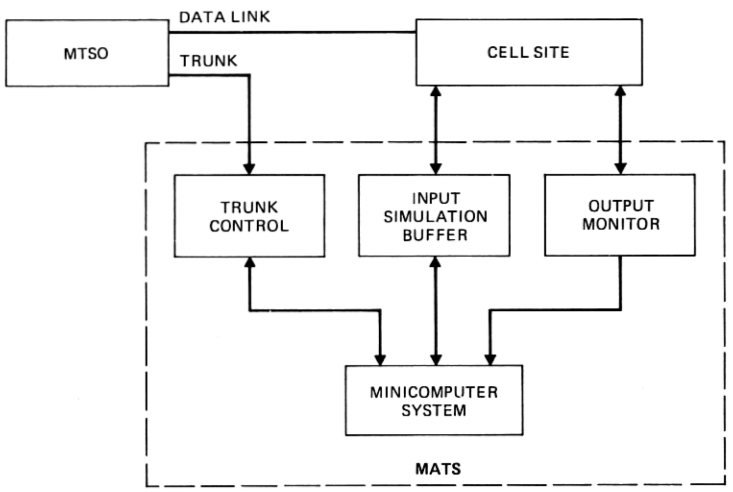

Fig. 2—Maintenance and traffic simulator.

peripheral device. A programmable bit mask selectively directs data from either the real device or from the input simulation memory to be sent to the cell site controller. As an example, this allows the data link to be specified as real, and to operate normally, while other peripherals are being simulated.

3.1.4 Trunk control

The trunk control hardware is used to terminate a set of test trunks from the MTSO. Calls to and from simulated mobile units use these test trunks instead of the regular trunks to the cell site. The trunk control allows MATS to put the trunks off-hook and to selectively connect tone generators and detectors for voice path continuity verification.

3.2 MATS software

MATS is implemented on the MERT[2] (Multi-Environment Real-Time) operating system, which was developed at Bell Laboratories. The MATS software can be functionally divided into four modules: a process controller, user interface, cell site peripheral simulation, and ETSL program executor. These four modules have access to data structures in a common memory area. The primary data structures are process records and message buffers. A process in MATS is a functional package of software, which usually consists of a set of subroutines and associated data structures that simulate one cell site peripheral. The process record is used to store current information about each process. The message buffer is used to pass data between processes, and routines are provided for administering the message buffers.

3.2.1 Process controller

The primary functions of this module are task-scheduling and timing. To provide as exact an emulation of the cell site peripherals as possible, the peripheral's timing has to be closely duplicated. Also, because the amount of inputs from the system is large, MATS must be able to handle them with a minimum amount of overhead. For these reasons, MATS has its own suboperating system for task dispensing, intertask communication, and interaction with hardware. However, MATS still depends on the MERT operating system for hardware interrupt handling, file system control, and user interface support.

The process controller maintains two separate timing lists. The first, called a time-out list, contains those tasks that must wait until a certain time has elapsed before being executed. The second, called a deadline list, contains those tasks that must be executed by a certain time but may be completed before then. A scheduler routine searches these timing lists and starts the appropriate tasks at their appointed times.

3.2.2 User interface

The main function of the user interface module is to compile the ESTL program specified by the user. In addition, it contains a number of support routines that provide print, display, and record facilities for the various computer I/O devices. This module also initializes the MATS hardware and other MATS software modules.

All transactions between the cell site controller and the simulated peripherals, as well as the time they occurred, are recorded. A detailed analysis of these interactions is made available to the user at the end of a simulation session.

3.2.3 Peripheral simulation

The peripheral simulation module contains one program for each peripheral device type simulated. It also contains a call controller that simulates the actions of the mobile user and a scenario progress controller that acts as a command interpreter and directs the actions of the call controller. Each device simulation program consists of a set of action routines and a function table that specifies the order in which the action routines are to be executed.

3.2.4 ETSL executor

The MATS user specifies in his program, written in ETSL, the simulation scenarios, such as type and quantity of calls, and number of mobiles involved. ETSL allows the user to control and monitor all mobile unit actions as well as to verify talking connections on simulated calls.

The ETSL program executor performs the commands specified by the user. It includes a table that contains the addresses of subroutines for each ETSL command implemented. Samples of ETSL commands are given in Table I.

IV. MOBILE CALL GENERATOR

The MCG provides an automated means for end-to-end testing of the entire Advanced Mobile Phone Service system. It employs a computer to generate calls and monitor the results on a small number of stationary mobile units. In essence, it emulates the mobile users. However, MCG, in addition, has the capability to monitor some of the internal behavior of the mobile unit and to force special events or abnormal mobile behavior to check potential trouble areas. Special events may include multiple seizures and critical interference, such as a fade, flash, or abandon, that occurs during call setup or while providing special service. Many of these events cannot be caused intentionally by a person because of lack of access to the internal actions of the mobile and because of the reaction times that are necessary. For testing purposes, it is desirable to be able to force abnormal mobile behavior to check for proper switching system and mobile reaction. This might be causing setup radio channel seizure or transmission failure, erroneous channel selection, or refusal of a mobile to release. The internal monitoring capability of MCG is used to gather data on internal mobile behavior, such as retry strategy.

In short, MCG is intended, in a step-by-step process, to verify that the system meets its requirements. It provides the desired monitor, control, and call-generating capabilities. It can supply all the customer's control stimuli and in addition can influence the internal behavior of the mobile. The test results consist of completion rates, specific errors, timing associated with calls, etc. The data are sufficiently detailed to allow further analysis and trouble identification.

4.1 Hardware

MCG is implemented on the same minicomputer as MATS. New hardware consists of a mobile unit interface for each mobile, and a

Table I—Samples of Extended Telephone Simulation Language (ETSL)

Command Name	Description
ORIG	Originate
CLEAR	Clear and reset mobile
MESG	Print message
ONHK	Go on hook
IF	Branch if equal
TALK	Verify talking path
PAGECHK	Check whether paging signal is received by mobile

multiplexer in the computer. Line drivers and receivers are provided both at the mobile and the computer ends (Fig. 3). The MCG hardware also includes a tone generator and detectors for each mobile for voice path verification. The interface at the mobile unit autonomously checks for status changes on those signal leads monitored by MCG and asynchronously reports these changes in serial form to the software in the computer. The commands from the computer to the mobile unit are also transmitted in serial bit streams. Therefore, between the computer and each controlled mobile, only two pairs of wires are needed, one for each direction.

The mobile unit consists of three functional units: a control unit, a transceiver, and a logic unit. The control unit, which includes the handset, controls, indicators, and associated tone circuit, is the interface between the user and the AMPS system. The transceiver provides duplex voice transmission and reception. The logic unit, which is built around a microprocessor, functions as the master control for the mobile unit, and encodes and decodes the digital signaling between the mobile and the cell site. The MCG hardware interfaces with the mobile units at the connector between the logic unit and the transceiver unit. Thus, no physical modification to the mobile units is required. A block diagram of the MCG hardware is shown in Fig. 3.

4.2 Software

MCG shares a significant portion of the MATS software, specifically the process controller and ETSL executor modules discussed in Sections 3.2.1 and 3.2.4. In addition, the MCG software contains three other modules: a hardware interface, a mobile monitor, and a mobile simulator.

The hardware interface module provides access to the MCG hardware. It includes an input and an output routine. The input routine receives mobile status change data through the multiplexer and passes them on to the mobile monitor. These input data include frequency change, carrier-on or-off, reorder, alerting, etc. The output routine enables the software to send orders (such as off-hook and dialed digits), to the mobile unit as if the orders were coming from the mobile's control unit.

The mobile monitor module receives status updates from the mobiles and maintains a status picture for each mobile. It also synthesizes multiple status changes on the monitored signal leads into a recognizable mobile order [such as Voice Channel Assignment (VCA) and release], which is then sent to the mobile simulator.

The mobile simulator module tracks the current state of the mobile. It also takes orders from the ETSL executor and translates them into the appropriate action instructions. These instructions then are sent

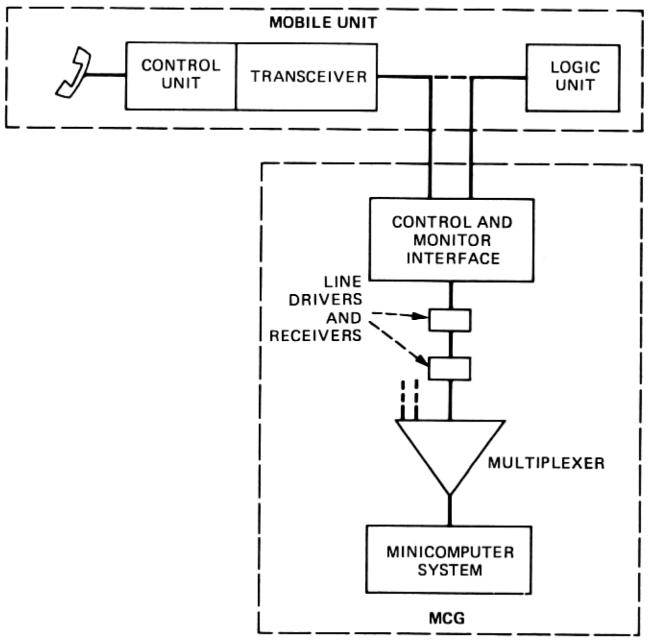

Fig. 3—Mobile call generator.

to the controlled mobile unit through the hardware interface output routine discussed earlier. For the status check orders, the mobile simulator sets the associated status check flags, so the mobile monitor will know which changes it is to report back.

V. CELL SITE RESPONSE GENERATION

In the AMPS project, the data link and the cell site hardware and ESS switching software were developed concurrently. Therefore, a way had to be found to facilitate initial program testing at the Oak Park MTSO independent of the data link and the cell site hardware. (The MOSES test system exists only at Bell Laboratories in Indian Hill and was not available at Oak Park.) To fulfill this need, a cell site response simulation program, known as the cell site response generator, was developed.

The response generator can provide most of the cell-site-to-MTSO messages: confirmation signals; and page, location, range, and signal strength replies. Thus, the response generator can be used to simulate mobile originating and terminating calls, and to test the locating and handoff software.

This simulation program is used in conjunction with the existing No. 1 ESS utility system. The utility system includes the AT function, with which the user can direct a utility function to be performed at a specific program address where an AT flag is planted. It is with this facility that a program tester activates the response generator. Prior to a test run, the tester uses utility commands to specify the predefined response message type to be generated, the parameters to be used (e.g., the digits dialed in a mobile origination), the required time delay for the response message to arrive, and the AT flag program address. Upon detection of the AT flag, the response generator builds the appropriate response message and places it in an internal buffer. When the specified time has elapsed, the response generator scanning program removes the message from the buffer and passes it to a data link I/O program as if the message had come from an actual cell site. It should be noted that the response generator produces cell site responses as specified explicitly by the user whereas other test systems generate the simulated messages in response to request messages sent to the cell site.

While most cell site responses are transmitted via the data link, the confirmation signals (such as voice channel, or answer) are transmitted as on-hook or off-hook transitions on the cell site voice trunks.

The response generator provides the appropriate trunk status transitions through a relay wired to each cell site trunk used in simulation. These relays can be controlled via ESS by software. When a trunk confirmation signal is requested by the user, the response generator software places a special code in an internal buffer. When the specified time has elapsed, the response generator scanning program, upon detecting this special code, will control the special relay to cause the corresponding trunk to go on-hook or off-hook as required.

The response generator provides a means by which nearly all call processing software can be functionally tested without the data link and the cell site hardware. This has been a very valuable tool in the initial development of the call processing software at the Oak Park MTSO. It is also useful in segmenting the source of system trouble in ESS, in data link, or in cell site equipment.

VI. DATA RETRIEVAL SYSTEM

The Data Retrieval System (DRS) is a minicomputer-based data-gathering system. It is designed to help evaluate the AMPS performance. The system can monitor, on a continuing basis, events occurring in the switching system. In addition, it can, in effect, access any data memory location without affecting ESS operation. An event is defined as some action or collection of actions in the ESS that is identifiable by program or data memory addresses. For example, suppose the program that handles mobile origination were loaded in the ESS program memory starting at address AAA. It can be reasonably assumed that, when the

ESS is executing the instruction at location AAA, a mobile origination is being processed. The execution of the instruction at location AAA is considered to be an event. The events to be monitored can be changed easily.

The principal advantage of DRS is that it is noninterfering and transparent to the No. 1 ESS. Since it is connected to the ESS through already existing test connectors, it requires neither hardware nor software changes in the switching system.

One design objective was to make DRS a system that could be easily transported to a field site. The hardware, therefore, was kept to a minimum. A minicomputer was chosen as the main processor. The heart of the DRS is a specially-designed Interface-Matcher (IM), which serves as the link between the No. 1 ESS and the DRS processor. The matcher has the ability to monitor all registers in the ESS control processor (see Fig. 4). From these registers, the location of the program instruction being executed and the location of the data being interrogated or changed can be obtained. These addresses are used to determine when an event has occurred in the system.

The matcher appears to the DRS processor as if it were two storage areas, matcher address area and snapshot buffer area. The matcher address area stores the specified program memory and data memory addresses. Each time one of these addresses is encountered during program execution, the values of all ESS registers existing at that time will be loaded into the snapshot buffer area. The matcher address area is divided into program memory locations and data memory locations.

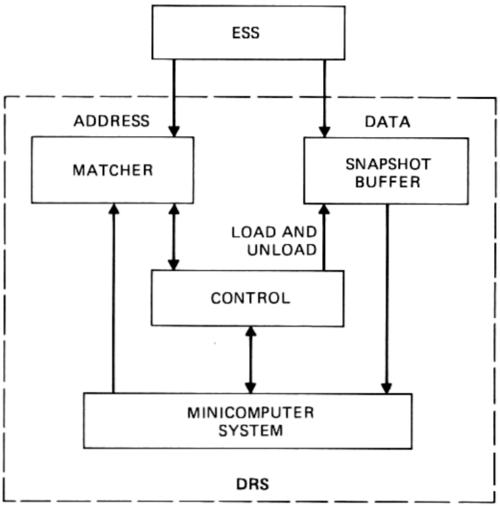

Fig. 4—Data retrieval systems.

Single addresses or pairs of addresses specifying address ranges can be used. The snapshot buffer area is a first-in/first-out buffer consisting of a fixed number of slots. Each slot is large enough to store values for all the ESS registers being monitored, plus additional information tags and time of day.

The main function of the matcher, therefore, is to provide snapshots of pertinent data whenever an address listed in the matcher address area is encountered. The data loaded in the snapshot buffer will remain there until processed.

All events to be monitored by DRS are defined prior to the actual data collection. Event definition consists of two parts, event attribute and event directive. Event attribute is used to identify a specific event, such as mobile origination or fade (loss of radio signal). It consists of an address or address range and, in the case of a data memory address, an optional action specification. The action specification allows the user to define an event as occurring during (*i*) a write into the location, or (*ii*) a read from the location, or (*iii*) either a read or a write. Event directive consists of a list of actions whenever the specified event is recognized. Generally, the actions are specifications of the data desired and the output medium to be used.

Because of its versatility and portability, DRS can also be an effective utility tool for troubleshooting subtle software problems in any working No. 1 ESS office in the field.

The DRS software is written in the high-level C programming language developed at Bell Laboratories. In addition to data-gathering functions, some of the more useful software debugging utility functions are also implemented.

VII. SUMMARY

The AMPS is a complex system. It contains several major subsystems, each of which has its own processor and software. The five laboratory test systems briefly described in this paper were designed to fulfill specific needs, either for the individual subsystems or for the entire system. The needs are simulation, software debugging, load test, performance monitoring, and overall system testing. These test systems have been invaluable tools to the AMPS development. They all share the common goal to ensure that AMPS meets its design requirements and provides a high quality of mobile service to the public.

VIII. ACKNOWLEDGMENTS

The development of the test support systems described in this article is the result of joint efforts of many people. F. W. Bowen and D. R. Fuller were responsible for the design of the MOSES system; L. D.

Mayfield and W. F. McMillan for the MATS and MCG systems; E. J. Fontenot for the Response Generator; and J. P. Lee and R. L. Lien for the Data Retrieval System. These people have made substantial contributions, and much credit is due to them.

REFERENCES

1. "No. 1 Electronic Switching System," B.S.T.J., *43,* No. 5, Parts 1 and 2 (September 1964).
2. D. L. Bayer and H. Lycklama, "MERT—A Multi-Environment Real-Time Operating System," ACM, Operating System Review, *9,* No. 5 (November 1975), pp. 33–42.