THE BELL SYSTEM TECHNICAL JOURNAL
JULY-AUGUST 1982
VOL. 61, NO. 6, PART 2

Automated Repair Service Bureau

THE BELL SYSTEM TECHNICAL JOURNAL

THE BELL SYSTEM TECHNICAL JOURNAL is published monthly, except for the May–June and July–August combined issues, by the American Telephone and Telegraph Company, C. L. Brown, Chairman and Chief Executive Officer; W. M. Ellinghaus, President; V. A. Dwyer, Vice President and Treasurer; T. O. Davis, Secretary. Editorial inquiries should be addressed to the Editor, The Bell System Technical Journal, Bell Laboratories, Room 1J-319, 101 J. F. Kennedy Parkway, Short Hills, N. J. 07078. Checks for subscriptions should be made payable to The Bell System Technical Journal and should be addressed to Bell Laboratories, Circulation Group, 101 J. F. Kennedy Parkway, Short Hills, N.J. 07078. Subscriptions $20.00 per year; single copies $2.00 each. Foreign postage $1.00 per year; 15 cents per copy. Printed in U.S.A. Second-class postage paid at New Providence, New Jersey 07974 and additional mailing offices.

ISSN0005-8580

Comments on the technical content of any article or brief are welcome. These and other editorial inquiries should be addressed to the Editor, The Bell System Technical Journal, Bell Laboratories, Room 1J-319, 101 J. F. Kennedy Parkway, Short Hills, N. J. 07078. Comments and inquiries, whether or not published, shall not be regarded as confidential or otherwise restricted in use and will become the property of the American Telephone and Telegraph Company. Comments selected for publication may be edited for brevity, subject to author approval.

THE BELL SYSTEM
TECHNICAL JOURNAL

Volume 61 July-August 1982 Number 6, Part 2

AUTOMATED REPAIR SERVICE BUREAU

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Preface

By L. SCHENKER and S. J. BARBERA

(Manuscript received June 22, 1981)

A family of computer-based operations support systems, the Automated Repair Service Bureau (ARSB), has been introduced at Bell Operating Companies. The ARSB includes the Loop Maintenance Operations System and the Mechanized Loop Testing System. A third component, the Loop Cable Maintenance Operations Support System, is in an earlier stage of introduction.

On May 3, 1971, at AT&T in Newark, New Jersey, representatives from AT&T, New York Telephone, and Bell Laboratories met to discuss repair service records. Although the operation of the telephone network had been highly automated for decades, the need for streamlining, perhaps mechanizing, support operations, such as repair service, was just becoming apparent. This first became an issue with those Bell operating companies (BOCs) serving densely populated metropolitan areas. There, the rapid growth of the telephone system, coupled with inexperienced repair service craft persons made rendering of quality service to customers increasingly difficult. New York Telephone was one of the first to experience this problem and to bring it to the attention of AT&T and Bell Laboratories. The meeting produced tangible results: The conception of the Loop Maintenance Operations System (LMOS), one of the most widely deployed operations support systems and the charter system of the Automated Repair Service Bureau (ARSB).

Fortunately, decreasing effort on government systems freed personnel at Bell Laboratories to study the technical feasibility and economics of mechanizing the repair service operations, which included both the

administration of customer trouble reports and testing. This effort resulted in the first mechanized repair service bureau—the New York City 73rd Street Bureau—in December, 1972.

The ARSB has now been introduced at every BOC. The LMOS will soon be serving virtually 100 percent of Bell System lines, and use of the second major component system, Mechanized Loop Testing (MLT), developed two years later, is following closely behind. The family now includes the Loop Cable Maintenance Operations System (LCAMOS), for predicting, tracking, and analyzing cable troubles. In many cases, the availability of the mechanized repair records has played a major role in the restoration of service following disasters, such as floods, hurricanes, and fire.

This special issue of *The Bell System Technical Journal* tells the complete story. It demonstrates that a successful system requires the contribution of people with different disciplines and organizational ties. Many times, seemingly insurmountable problems were overcome simply because someone from a different organization with a different point of view could see a solution that had escaped those who were too close to the problem.

Obviously, many people played a role in the conception, design, development, manufacture, implementation, and operation of ARSB. This issue is dedicated to them.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Evolution

By P. S. BOGGS, M. W. BOWKER, E. A. OVERSTREET, and R. W. VETTER, JR.

(Manuscript received June 16, 1981)

This paper describes the evolution of a family of integrated computer-based operations support systems, the Automated Repair Service Bureau (ARSB). The ARSB supports the maintenance of a telephone customer's service from the local switch to the subscriber's premises (i.e., the loop portion of the service). The automation of labor-intensive manual tasks has made it possible to provide better service at lower cost.

I. INTRODUCTION

Since 1971, Bell Telephone Laboratories has focused considerable attention on Repair Service Bureau (RSB) operations. Motivating factors for this effort included the increasing cost of trouble report processing, loop testing, and fault location; a decrease in the experience level of craft persons; increasing complexity of testing; and an increase in the number of customer trouble reports being processed. The result of this effort was the development of a family of integrated computer-based operations support systems, the Automated Repair Service Bureau (ARSB). The major reasons for automating repair service functions were to

(*i*) improve customer service by more rapid detection, location, and repair of troubles;

(*ii*) reduce the trouble report rate and outage time; and

(*iii*) improve the efficiency and reduce the cost of testing, dispatch and repair operations.

In addition, the system was designed to be compatible with existing and planned future Bell System standard equipment and, as far as

possible, with equipment manufactured by outside suppliers for use on Bell System lines.

Customer service has been improved by simplifying the customer trouble report-taking procedure, providing more reliable identification of the type and location of troubles, keeping the customer better informed of the status of testing and repair work during the trouble handling process, increasing the reliability of appointment times for repair visits, and reducing the total out-of-service interval.

The customer trouble report, subsequent report,* and repeat report† rates have been reduced by improved testing and analysis techniques. Outage time has been reduced by improving the efficiency of the trouble handling process, and by dispatching the right repair person to the right place at the right time. Trouble reports (both on found troubles and other reports) have been closed out with greater confidence that the trouble no longer exists. The use of mechanized testing and repair force administration has resulted in faster, more efficient processing and clearing of troubles.

Repair Service Bureau operations have been improved by mechanizing many clerical and analytical tasks, by making more effective use of available employee resources (e.g., by allowing people to make more important decisions), and by better managerial control over RSB operations through the use of an automated information system. The ARSB concept described in this special issue of *The Bell System Technical Journal* represents, in many respects, a departure from previous RSB operational procedures. As a result, particular attention was paid during the design process to the problems associated with personnel subsystems, especially the human-machine interface. (Refer to the paper entitled "Automated Repair Service Bureau: Human Performance Design Techniques," appearing in this issue.)

A flexible, modular design of both the hardware and software was used to provide for compatibility with both existing and future equipment.

The ARSB system was designed to use existing standard interfaces, such as no-test trunk‡ interfaces for testing and line insulation testing teletype channels§ for predicting incipient cable troubles, and to share these interfaces with other systems when possible. In addition, it

---

* A subsequent report is a customer trouble report on a line for which there is already a pending report.

† A repeat report is a customer trouble report received within 30 days of a previously cleared customer trouble report on the same line.

‡ No-test trunks are four wire metallic trunks used to gain metallic access, through the switch serving the subscriber, for testing. The term "no-test" is used because the switch performs no test for a busy line to determine if access should be provided.

§ The LIT channels are teletype channels over which maintenance information, including an identification of those lines with low insulation resistance, is transmitted to the RSB.

included manual and machine interfaces with related Bell System standard testing and records systems.

The remainder of this paper describes the loop maintenance process before automation and the evolution of the ARSB from an experimental system in the New York Telephone Company to its present configuration.

1.1 Historical background

In the late sixties, it was apparent that the RSB, where customer trouble reports were being received and processed, offered opportunities for improvement in both the efficiency and quality of loop maintenance. Let's see how the RSB handled a typical report. For example, a customer may have tried to use the telephone and heard no dial tone, and no side tone,* i.e., the phone was dead. (Assume that the reason was a broken wire in a cross-connecting terminal halfway between the customer and the central office.) Consequently, the customer used a neighbor's telephone to report the trouble. At the bureau, the call was received by a clerk who filled out a form, called a trouble report, with all the pertinent information including telephone number, name, address, reach number (in this case, the neighbor's), a description of the trouble condition, and the commitment time (the time by when the phone was to be fixed). After the trouble report was taken, the line card was retrieved from a large tube file and the trouble report and line card were clipped together and passed to a screener. The line card contained pertinent information on the customer's service including:

(*i*) the portion of the central office equipment dedicated to the customer,

(*ii*) the location on the central office main frame where the cable pair is terminated,

(*iii*) the designation of the cable(s) and pair(s), or the Pair Gain System and channel, used to connect the customer premises to the central office,

(*iv*) a full description of the customer premises equipment (e.g., one main green 500-type set with *TOUCH-TONE*® service and one extension served by a blue *PRINCESS*® telephone), and

(*v*) the location of the serving terminal where the customer's drop wire is connected to the cable facility.

This information was needed for testing and troubleshooting the fault. The screener then passed the trouble report and line card to a tester

* Side tone is the portion of the voice signal entering the transmitter of the telephone set that is coupled to the receiver of the same set. This allows the person speaking to hear both the transmitted and received signals.

who verified the trouble condition. The typical test position available was a test desk that consisted of a 100-volt, 1000-ohm/volt voltmeter and a talking set that could be bridged to the loop.

For the case being discussed, the 100-volt battery was applied to one side of the line, through the meter, and the other side of the line was grounded. The tester verified that the loop was open by observing the "capacitance kick" on the meter when the sides of the line to which the battery and ground were connected were reversed. The kick was lower in magnitude and shorter in time than would have been observed with a standard telephone set connected.* A more experienced tester might also have recognized that one side of the line was shorter than the other by measuring the capacitance kick from each side of the line to ground. This would have indicated that the open was not at the customer's premises but back towards the central office. However, since most opens were near the customer, a repair person responsible for station equipment and inside wiring up to the station protector (mounted on the side of the customer's premises) was dispatched. By testing from the protector, where the service entered the premises, and the serving terminal, the craft person concluded that the trouble was "in the cable plant" and turned the trouble over to "cable repair."

Cable repair† craft persons brought more sophisticated test equipment to bear and concluded that the open was halfway between the central office and the customer. Cable records were referred to and showed a cross-connecting terminal at the approximate fault location. Since most faults occurred where the cable was most easily accessible, such as in a cross-connecting terminal, the cable craft persons went to that location. An examination of the terminal and further testing revealed the trouble which was then repaired. The trouble found was reported by a call from the repair person to a tester at the RSB. The tester retested the line, found it normal, called the customer and "closed out" the trouble. Trouble time and disposition were added to both the trouble ticket, previously discussed, and to the cable trouble ticket. At still another position, various statistics were collected, summarized, and analyzed. Certain standard statistics were reported to AT&T and the state commissions.

Methods by which this approach could be improved are probably apparent to the reader, but the following will highlight the main ones:

(*i*) Mechanize the data on the line cards to

---

* A typical station set, when on-hook, has a ringer circuit (LC ringer) consisting of an inductor and a capacitor in series across the line. This circuit looks basically like a tip to ring capacitor when a "ballistic kick" type of measurement is performed.

† Cable repair craft are responsible for all cable facilities from the protector on the mainframe at the central office up to the station protector.

(a) allow multiple users to access records at the same time, and

(b) reduce the manual updating required, and the inherent inaccuracies associated with it.

(*ii*) Mechanize trouble report tracking to

(a) provide a way of determining the status of a particular trouble without hunting through the bureau to find the trouble report,

(b) allow the clerks to talk more intelligently to customers who call again while their trouble report is still open, and

(c) allow bureau personnel to obtain information on groups of open troubles (for example, how many troubles have not yet been repaired).

(*iii*) Mechanize testing and analysis of test results to

(a) eliminate the need for asking the customer to stay at home on almost every trouble by making test result summaries available to the person receiving the trouble report (note: there is no need for the customer to stay home unless the trouble is on the premises), and

(b) decrease dependence on the experience level of the tester.

(*iv*) Mechanize the analysis of "closed" troubles to

(a) eliminate the need for extensive manual analysis,

(b) significantly reduce the time delay associated with manual analysis that detracts from the usefulness of the results, and

(c) allow previously rigid fixed format reports to be changed to meet each particular area's needs.

By the late sixties, it was technically feasible to use a computer for the line record information, and to track a trouble through the process. There was also no question about the technical feasibility of being able to measure the three terminal (tip, ring, and ground) characteristics of a faulted or unfaulted loop accurately from the central office end with the telephone on hook. The real issues were operations, human-machine interfaces, and economics. The following sections provide a more detailed discussion of these issues.

1.1.1 Operations

The RSB is the nerve center for all loop maintenance and is busiest during a crisis (e.g., flood, fire, hurricane damage). A mechanized system must be sized for the so-called peak busy hour, but seldom can we afford to size it for the "once-every-few-years" crisis. However, the design of the total system must be such that it does not "choke" at some point causing processing capability to decrease beyond some load point. There also has to be some backup procedure when computers or data lines fail. The mechanization should make the average position or work station more efficient without making it boring. Care should be taken not to mechanize that which can be done better manually. For example, a decision was made with respect to the Basic Output

Report (BOR) (a computer printout to be described later) to pass it manually from work-station-to-work-station rather than retrieve it from a computer terminal each time.

1.1.2 Human-Machine Interface

In addition to the obvious human factors problems related to screen content, mask design I/O transactions, etc., there was the retraining problem for the people who had operated in the manual environment. The human mind allows tremendous flexibility. It enables us to change jobs in midstream and to be amazingly tolerant of errors because of our ability to grasp context and also remember valid sets of information. For a system to be accepted, it should not accept "foolish" data, provide impossible outputs, or take too long between input and output.

1.1.3 Economics

The only way computers can lead to net savings in the repair process is to reduce the size of the work force. Reducing, even by 80 percent, the load on a one-person work force is meaningless if that one person must remain to perform 20 percent of the original task.

Therefore, economics studies were aimed at identifying areas where mechanization would reduce the number of people in a particular work force. Specifically, it was anticipated that a work force of, for example, 7 testers could be reduced by 2 or more, and a work force of 8 repair answering clerks could be reduced by 4. The savings in labor associated with repair answering clerks results from the consolidation of many small work forces into a larger centralized one and the elimination of the manual line record file.

1.2 Opportunities for upkeep expense savings

To quantify the potential opportunity for ARSB benefits, certain expenses were identified as being prime candidates for significant reductions. Included were the RSB testing and clerical expenses, plus potentially unnecessary dispatch expenses. The RSB clerical expenses included taking trouble reports, maintaining the manual line record card file, and dispatching repair craft. It appeared that the RSB expenses could be significantly reduced by mechanizing the line record files, automating much of the testing function, and streamlining the trouble report flow in the RSB.

Although some repeat reports are unavoidable, it was felt that many repeat reports are a result of not properly repairing the first trouble. A test to quickly verify that a trouble had been repaired before the repair craft left the area was needed to help reduce repeat reports.

Work discontinued is due, in part, to the outside plant repair craft

being unable to locate a trouble within a reasonable amount of time and having to redispatch on the same trouble the following day. "Found OK" dispatches are those on which no trouble is found and the dispatch is terminated, resulting in a repeat report and/or wasted repair craft time. To reduce both the work discontinued and found OK dispatches, improved testing was required. Outside plant troubles were frequently dispatched first to station repair craft and then redispatched to cable repair craft because the manual testing methods could not identify the trouble as being in the outside plant. Again, it was felt that improved testing methods, by identifying the approximate location of the trouble, could avoid sending many of these dispatches to the wrong craft.

In summary, almost 20 percent of the total upkeep budget was a prime target for significant reductions because of an RSB mechanization system like ARSB.

II. THE EARLY DAYS—FIRST GENERATION

The manual RSB has just been described. We now focus on the early days of defining requirements for the new mechanized system and the need to understand the manual environment thoroughly; this is the reason for the extensive involvement of systems engineering personnel in RSB operations. This effort, which continues even today, was facilitated by New York Telephone Company's development of an experimental system covering ten thousand lines in its West 73rd Street RSB in 1970–1971. Careful observation of this experiment, as well as long hours within RSBs interviewing and observing personnel at work, led to the early definition of requirements for a system. The principal focus of this work was on the tracking of trouble reports and the mechanization of the customer line card, resulting in the development of a Mechanized Line Records (MLR) system at Bell Laboratories. Also, during this stage of definition and development, it was recognized that much of the environment would be changed significantly by the advent of a mechanized system and that it was difficult, if not impossible, to predict the RSB environment after the change. This realization dictated that a system be deployed as quickly as possible to discover operationally how the people and machines interacted. Thus, MLR was deployed quickly on a centralized processor for New York Telephone to better understand the needs of the bureau and to provide data for planning future development.

In addition to MLR, substantial development effort by Bell Laboratories resulted in the Line Status Verifier (LSV), designed jointly with Western Electric Company. The LSV, which was field-trialed in 1972, was an automatic line verification device applicable to one class of RSB testing problems. The system was especially useful when applied to

trouble reports which, when tested shortly after the trouble report was received, gave no indication of trouble (i.e., Test OK). The use of LSV allowed a large percentage of these trouble reports to be closed out during the initial customer contact, resulting in less handling and faster processing of those reports. The mechanized combination of MLR and LSV, as separate stand-alone systems, provided a prototype for the ARSB.

In the processing and clearing of trouble reports, the RSB can be envisioned as consisting of seven basic functional operations. These operations are trouble report taking, screening, testing, dispatching, clearing, closeout, and line record maintenance. The impact of MLR and LSV on these operations will now be described.

2.1 Trouble report taking

Incoming calls from customers were routed by a call distributor to a person available for taking trouble reports. Upon answering a call, the attendant obtained the telephone number and entered this information into MLR using a computer terminal equipped with a cathode-ray tube (CRT). The MLR system responded with a trouble report mask which served as a vehicle for entering the trouble report. The trouble report mask also contained line record data for use by the attendant in interaction with the customer. This information included customer name and address, class of service, and status of pending troubles, as well as historical data covering the last trouble cleared.

While talking with the customer, an LSV test was initiated from a separate LSV console on the customer's loop. The LSV system accessed the customer loop and ran an automated "GO/NO-GO" series of tests. The attendant was notified of the results. If the LSV did not detect a fault on the customer loop, the attendant would so notify the customer and attempt to close out the trouble report. If the customer refused to accept the closeout of the trouble, or if there was a positive indication of a trouble (either by test result or customer description), the attendant negotiated a commitment time with the customer. A suggested time, based on internally stored time intervals given to MLR by RSB managers, was displayed on the CRT. The actual commitment negotiated with the customer was entered into the system. The calculation of the suggested commitment time and intermediate jeopardy times for testing and dispatch was a major improvement over the manual RSB.

2.2 Screening

The purpose of the screening function was to provide a method of distributing trouble reports to the appropriate work stations. The

screener routed a BOR* to the appropriate person based on trouble description, LSV test results, and the type of service.

2.3 Testing

Under ARSB with MLR and LSV, the testing function remained relatively unchanged, except for a decrease in the test load at the local test desks. The decrease was due to the detection of some faults and the verification of Test-OK's† by LSV at the time of the customer contact.

2.4 Dispatching

The dispatching function was handled in a variety of ways. In some RSBs, testers of dispatchers with dedicated repair crews were responsible for dispatch of their individual repair people. In other RSBs, there was a group of dispatchers for the entire repair force, usually controlled by a first-level supervisor, who decided which trouble to dispatch next and to which repair person.

This function, too, remained relatively unchanged with MLR and LSV, with the exception of the use of LSV to verify that repairs had been made correctly before the craft left the trouble site.

2.5 Clearing and closeout

With mechanization, the recording clearing and closeout operations changed significantly. These operations involved repair of the trouble, customer notification and concurrence, and entry of final status into the MLR computer. In addition, MLR edited and validated the input and provided significantly more information about the customer's trouble and equipment involved to down stream analysis systems. (Refer to the paper entitled "Automated Repair Service Bureau: The Trouble Report Evaluation and Analysis Tool," appearing in this issue.)

2.6 Line record maintenance

With MLR, line record maintenance work was done manually by clerks with CRTs. This turned out to be a major expense for the Bell Operating Companies (BOCs), and mechanizing the records' update

---

* The BOR is a report automatically printed out as the result of entering a trouble into the system; it includes line record information, a description of the trouble, and LSV test results. As noted later in this article, the LSV results were eventually replaced by more sophisticated results.

† A "Test-OK" occurs when a trouble is caused by a transient condition that is no longer present when the test is performed. The test indicates that the condition of the loop is satisfactory.

process proved to be a major needed future enhancement as described below.

III. THE EARLY DAYS—SECOND GENERATION

The MLR and LSV systems were developed on short time schedules to solve some of the more pressing RSB problems in urban areas. Experience gained with MLR and LSV dictated changes needed in the operational procedures used with these systems. In addition, experience indicated that a single centralized computer would not handle the projected load economically in a wide variety of operating companies. Hence, the second generation of the system had a distributed architecture. This second system was applied to a second BOC (Southwestern Bell Telephone Company) to determine the generality of operating procedures and the viability of the system.

The need for an automated method of updating and maintaining the data base was recognized, and appropriate software was written to process service order information automatically from the service order systems of the BOCs.

Also, since the number of functions being performed by the system was expanding, MLR became the Loop Maintenance Operation System (LMOS).

To meet the objectives outlined previously, LMOS provides the following:

- Mechanized data base with automatic line record update
- Computerized trouble report processing
- Management and analysis reports on demand
- Mechanized aids for more efficient repair force deployment and more accurate commitment times.

In addition to LMOS, a second-generation automated testing system, the Mechanized Loop Testing (MLT) system, was designed and replaced LSV. The MLT system, unlike LSV, was not designed as a stand-alone system. It was designed as an addition to LMOS, and the two systems, LMOS and MLT, were fully integrated to form the basic ARSB system. As an interim step in the development process, LSV was partially integrated with LMOS to allow tests to be automatically initiated by the entry of a trouble report in LMOS and to provide for the automatic storage of LSV test results on LMOS. However, LSV was still basically a stand-alone system to be replaced by the integrated LMOS/MLT design. The MLT system provides the following:

(*i*) Improved testing of circuits under computer control using an adaptive series of tests, generated in real time, based on the electrical characteristics of the customers equipment in the idle state. (The data are available from LMOS.)

(*ii*) Test data interpretation based on the same customer equipment data.

(*iii*) Simplified test results to the Repair Service Attendant (RSA) as a basis for more Test-OK closeouts while the customer is on line.

(*iv*) Analyzed detailed test results in hard copy as a result of RSA initiated tests with indicated routing.

(*v*) Automated sequential testing of lines or equipment in a list.

3.1 Overview of LMOS

The LMOS is based on the MLR design, but has some significant system improvements that were added as a result of the MLR experience. Thus, LMOS represents an expansion of the MLR concept. For example, LMOS has the capability of automatic service order input that significantly reduces the manual effort for entering changes to the computerized customer line records. The system also has the operational report structure redesigned and expanded to reflect experience gained from the New York field test of MLR and tests of analysis systems in other BOCs.

In addition, several new features were added to LMOS. They allowed more effective management of the repair forces, as well as greater overall efficiency of the repair operation. These enhancements are described in the following paragraphs.

3.2 Major LMOS features

3.2.1 Trouble report and status entry

The trouble report and status entry feature was retained from MLR to provide fast and effective tracking of each trouble report through all intermediate statuses. In addition, the feature provides a major input to operational and administrative reports.

3.2.2 Bureau personnel and repair force administration

Bureau personnel and repair force administration is a new feature. It is a management tool for assessing, on a minute-by-minute basis, the utilization of bureau personnel, as well as outside repair forces.

The trouble report status feature and the mechanization of line records, combined, provide a capability that allows dispatching of trouble reports by any person using LMOS. The utilization of RSB personnel is increased, while MLT reduces tester/dispatcher requirements.

3.2.3 Dynamic bureau operational reports

The operational report structure of LMOS provides two types of reports: on-line bureau operational reports and the trouble report and data base analysis reports. The on-line bureau operational reports are

structured to predict and identify bottlenecks and problem areas in repair force operation and are based, in part, on the on-line report structure of MLR. An example of the use of this type of report is the detection of personnel work overloads in the functional areas of screening, testing, and dispatching.

3.2.4 Trouble report and data base analysis reports

These reports involve the statistical analysis of the repair operation for periods of from 1 to 40 days and are available on demand. Included are reports associated with employee work summaries, Customer Trouble Report Analysis Plan (CTRAP) output data, cable and originating equipment analysis reports, and the mechanization of the manual Repair Force Management (RFM) program. These reports will, for example, aid in pinpointing particular parts of the switching machine or particular work groups that need attention.

3.2.5 Line record administration

Storage of line records and recent trouble histories in the computer alleviate the problems of losing line cards, inaccuracy of records, and excessive clerical posting effort.

Automatic update of the line records from existing BOC service order networks resulted in significant clerical savings associated with line record upkeep. After initial conversion of line records, the only clerical forces required were those necessary to resolve conflicts between the BOC service order network and LMOS.

3.3 Mechanized Loop Testing (MLT) overview

The MLT system provides complete and accurate single-ended three terminal measurement parameters at dc and 24 Hz, which are analyzed by an adaptive test algorithm that is generated in real time. The selection of the tests to be performed and the analysis of the results are both based on data relating to the service and equipment of the particular customer's line being tested. These data are provided by LMOS. Two outputs are provided: one in text that tells the clerk, during the customer contact, if the line is good, faulted, or in use; the other output provides the values measured and the results of the analysis, e.g., one side open 18,400 ft from central office, etc. This analysis is enhanced by the on-line availability of records that provide data on the subscriber's service and equipment. The MLT system is also used for testing prior to dispatching a craft person, to verify that a fault is still present, and for post-dispatch testing to verify that a line fault has been corrected.

3.4 *Mechanized Loop Testing (MLT)* features

The MLT system provides expanded capability (beyond LSV) for verification of troubles on initial customer contact. The basic tests include the following:

(*i*) Hazardous EMF Check
(*ii*) Improved Busy Tests (including electronic speech detection and receiver-off-hook detection)
(*iii*) AC FEMF Measurement (each side-to-ground)
(*iv*) DC Leakage and FEMF Measurement (3 terminal)
(*v*) Longitudinal Balance Measurement
(*vi*) Termination Test (3 terminal)
(*vii*) "Open" Test
(*viii*) Ringer Counting
(*ix*) Rotary Dial Measurements
(*x*) Draw and Break Dial Tone (noncrossbar office).*

Comparison of measured analog values to thesholds occurs in the computer and allows Go/No-Go type results to be returned to the bureau attendant, while still in contact with the customer. Analog data are also retained for use by screeners and other bureau personnel as required.

3.5 *Operational strategy—ARSB*

With the advent of LMOS and MLT, the RSB was again significantly impacted. The operational strategy of this second-generation RSB will now be described.

The principal work functions of LMOS and MLT are associated with RSAS (who receive incoming trouble reports), the MLT tester (a screener/analyzer who may request additional automated tests), the regular tester (who performs interactive manual testing with craft personnel using a manual test desk and who also can access MLT for automated testing), and the dispatch controllers (who coordinate and manage the outside plant craft).

A simplified work flow diagram is shown in Fig. 1. The RSA, upon receiving the trouble report call, enters the telephone number of the reported line into LMOS via an interactive CRT. The LMOS responds with a display containing selected line record information, such as name, service address, and location, to the RSA. The RSA can then verify this information with the customer. If a previous trouble report is pending, then this information is also displayed to the RSA, so that the status of the pending report can be discussed with the customer.

---

* These two checks are not made during the initial contact, or on every reported trouble.

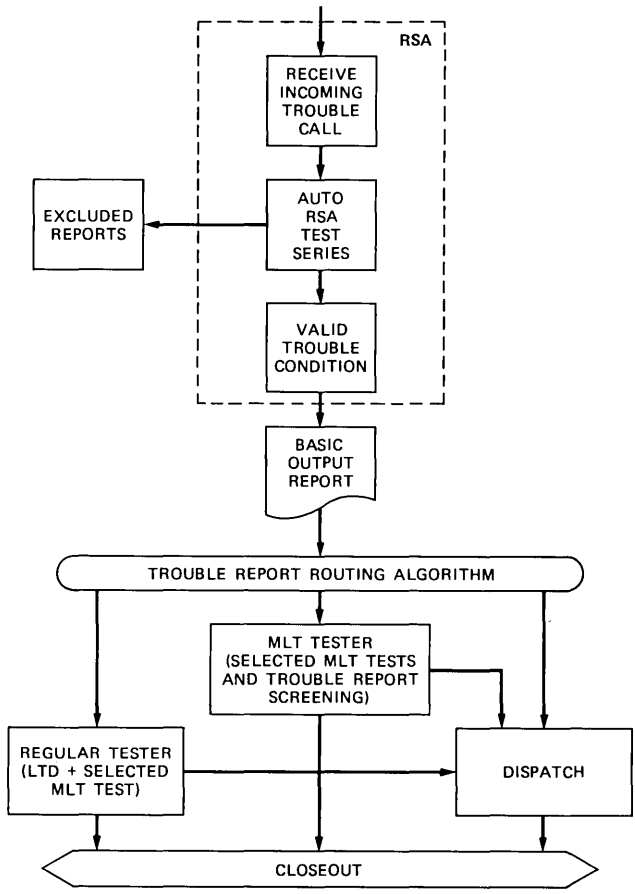


Fig. 1—Bureau work flow for ARSB (LMOS/MLT).

The RSA enters the trouble information (or any additional information for subsequent reports) gathered from the customer onto the CRT screen.

When the telephone number for a reported trouble is entered into the system, LMOS transmits the data required for testing to MLT and causes a test series to be initiated. This test series is performed concurrently with the RSA entering the trouble information (as described in the preceding paragraph). The results of the tests are displayed on the CRT, in a simplified format, while the RSA is still in contact with the customer. If the customer is calling from the same line on which they are reporting trouble, MLT will detect that the line is in use and display this result to the RSA. A test series, if required, can then be requested after the customer hangs up.

The RSA can use the test results to attempt a closeout of the trouble

with the customer (in the case of a "Test-OK" or Receiver-Off-Hook") or negotiate a short commitment (for trouble in the central office) or an appointment time (for troubles outside the central office) with the customer.

At this point, the RSA transmits the gathered data to LMOS, and LMOS responds with a CRT-formatted mask for the entry of the next trouble report.

Thus, the MLT test results, coupled with the line record data, assist in resolving certain reported troubles directly without further processing. Those reports which cannot be excluded* by the RSA are entered into the system, and a hard copy BOR† is sent to the responsible RSB.

Certain additional tests may be automatically suggested by the system when the initial RSA test result is returned, and such a suggestion is identified on the BOR. For example, a trouble may need testing with an MDF test clip.‡

Based on the results of the RSA test, the screener routes the BOR to the appropriate position. Troubles for which the test results clearly indicate the need for a station or central office dispatch are routed directly to dispatch controllers. Some trouble reports, involving facilities not testable by MLT, are routed to the regular tester at the local test desk (LTD). The objective is to route as many BORs as possible directly to the next probable function to be performed.

The MLT tester is responsible for coordinating the inside forces (i.e., central office and frame personnel) and, when necessary, passing the BOR to the regular tester or dispatch controller. To perform this function, the MLT tester may request retests (supplementary tests not made in the series automatically initiated when the RSA enters a trouble report into the system), tests covering groups of telephone numbers, cable pairs, or central office equipment, or retests to be performed over a period of time. These test capabilities provide a level of precision and thoroughness not available from the manual test desk. In addition, a number of LMOS equipment inventory and analysis reports are available to help RSB personnel localize troubles or analyze recurring troubles.

There are normally only a small number of troubles that cannot be resolved completely by MLT tests and, therefore, go to the regular tester. However, the regular tester has access to all MLT tests, as well as the LTD. The regular tester is responsible for lines that require line

* The RSB is allowed to exclude certain types of trouble reports, such as a report on a service that has been suspended for nonpayment.

† The BOR contains all of the available line record, trouble history, trouble report, and test result information for the reported line.

‡ An MDF test clip is used when tests requiring the switch and other central office equipment to be "bypassed" are performed.

conditioning prior to test application (e.g., on four-party lines with full selective ringing using three element gas tubes) and tests of other troubles that require interaction with other craft or the customer. The regular tester can coordinate the inside forces or give the trouble to the dispatch controller for outside troubles.

A cable tester/dispatcher (in the RSB or elsewhere) has access to the MLT tests and the LMOS data base to assist in cable trouble repair. The input of known cable failure data will allow new trouble reports within a cable failure to be automatically identified so that only minimal processing will be required in the RSB. Automatic tests covering groups of cable pairs will assist the cable tester/dispatcher in identifying the range of a suspected cable failure.

At each stage of the flow of a trouble report through the RSB—say, from MLT tester to regular tester to dispatcher—a report status should be entered into LMOS. The status information includes the employee code of the person doing the work, "work done" (e.g., tested), intermediate status (e.g., pending dispatch), routing information (i.e., next employee code), and test results (if any). This information allows anyone looking at the trouble report to determine exactly which functions have been performed and what the status is. It also allows the RSA to give accurate information on the status of the trouble to a customer calling in with a subsequent report.

As a final status, the report can be closed out from any position in the RSB when the trouble has been cleared or resolved. At this point, the trouble report and its related data become part of the trouble history for the appropriate RSB.

IV. INTRODUCTORY STRATEGY

The expectation, now essentially on target (see Figs. 2 and 3), was that LMOS would penetrate 100 percent of all Bell System lines and MLT, 80 percent. The objective was to penetrate as rapidly as possible. However, from the formation of a BOC study team, through planning, funding, procurement, data conversion, and cutover of the first bureau, would have required a minimum of three years. Therefore, a major cooperative effort of AT&T, Bell Laboratories, Western Electric Company, and the BOCs was launched. This effort involved the following:

(*i*) A joint steering committee, which included representatives of the early users, with enough authority to make the decisions required to clear road blocks and balance timeliness with feature enhancements.

(*ii*) Economic planning guidelines to help BOCs prepare estimates for approval.

(*iii*) Telephone company timetable guidelines for detailed planning, installation, data base conversion and a cutover schedule.

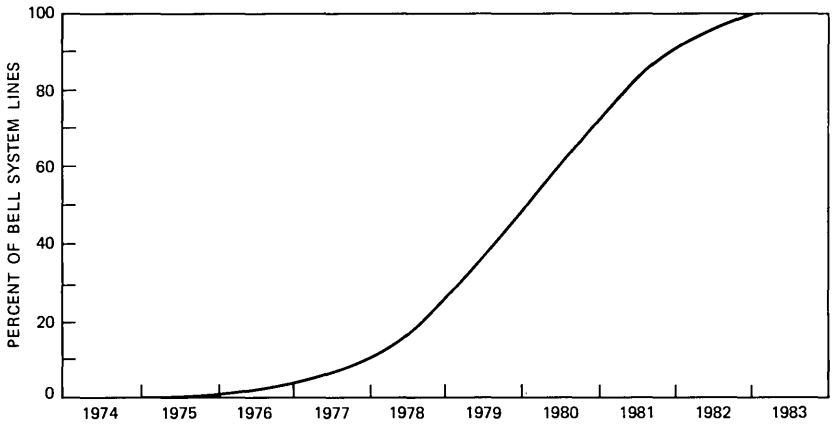


Fig. 2—Bell System implementation of LMOS (year-end 1980 estimate).

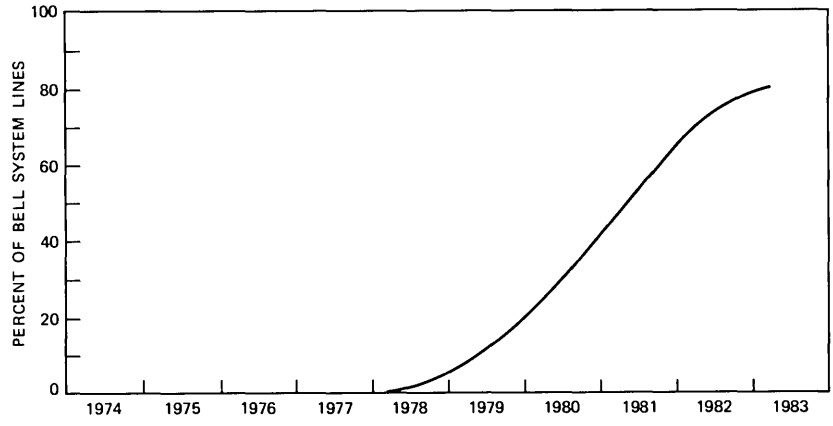


Fig. 3—Bell System implementation of MLTS (year-end 1980 estimate).

(*iv*) Telephone company letters of intent and joint AT&T/Western Electric scheduling of installations.

(*v*) A Western Electric planning and installation team to support the BOCs.

(*vi*) A Bell Laboratories "shock troop" to move in when tough problems were encountered.

(*vii*) A formal system change request process and committee to establish priorities for telephone company requests to tune the system to the user needs. (At one time these requests were being processed at the rate of one every other day, and 80 percent of the continuing development effort was involved.)

(*viii*) Operational reviews to ensure that the potential savings were being realized.

The net result was an overwhelming acceptance by the BOCs and the rapid introduction shown in Figs. 2 and 3. At one time, two million lines were being converted each month and, at another, one MLT frame was being installed every working day.

V. Continuing evolution

The ARSB is continuing to evolve. Some of the major changes, both completed and partially underway, since the development of the integrated LMOS/MLT system, are as follows:

- A context-sensitive data switch which facilitates large-scale centralization of the repair answering function.
- A Loop Cable Administration and Maintenance Operations System (LCAMOS), integrated with LMOS/MLT, which provides for the prediction, tracking, and analysis of cable troubles.
- A new generation of Mechanized Loop Testing System (MLT-2) designed to be cost-effective for very small population centers and functionally expanded to allow for the elimination of the need for local test desks.

The remaining papers in this special issue of *The Bell System Technical Journal* provide more detailed information on the design, implementation, operation, and evaluation of the ARSB and give descriptions of the improvements outlined above, as well as others.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

System Architecture

By R. L. MARTIN

(Manuscript received June 29, 1981)

The four main functions being served by the Automated Repair Service Bureau are (i) Customer Line Card Maintenance—a large, complex data base problem; (ii) Trouble-Taking and Tracking—a simple transaction problem; (iii) Loop Testing—a process-control problem; and (iv) Trouble-History Review—a large, filter, sort, and count report-generation problem. All of these functions were implemented among different computers, which were then networked together to form the full ARSB *architecture. This decomposition resulted in an architecture that has been adaptable to Bell Operating Companies' needs over a full decade.*

I. INTRODUCTION

The architecture of a system is the product of the history of the organization which builds it, the present and near-present technology, and the intended application. The architecture of the Automated Repair Service Bureau (ARSB) is the result of several iterations, and many decisions. In retrospect, many of the decisions which we tortured over seem, given that the system works, clear and obvious. Hopefully, our experiences will be useful lessons for other similar developments.

After describing the characteristics of the present ARSB architecture and the history which led to it, the major design decisions in its formation will be discussed. This will be followed by a slightly more detailed discussion of the architectures of the subcomponents of the system.

1.1 History

The Loop Maintenance Operations System (LMOS), as an operations system for the Repair Service Bureau, was first conceived as a system

in August, 1970, as part of a broad-reaching systems engineering study. Specific development work on what ended up being its prototype, the Mechanized Line Record System (MLR) started in April, 1971. The first MLR repair service bureau (RSB) was placed on-line in Manhattan, New York, on December 8, 1972, on an IBM 370/155 and DEC PDP* 11/20 computer. The 370/155 provided all the major user functions. The 11/20 initially provided simple backup when the 370/155 was down by recording trouble entry and status transactions for later submission to the 370 when it recovered; also, it provided loop-testing facilities via the Line Status Verifier (LSV), an early automated testing system. After going on-line, the next year and a half of MLR application development had two major activities:

(*i*) The transactions were restructured for ease of use and to utilize the IBM 3270 synchronous display terminals.

(*ii*) The data bases were restructured to improve computer performance and simplify program development by changing a combined line record/trouble data base to separate line record and trouble data bases.

As other Bell operating companies (BOCs) (Southwestern Bell Telephone Company and South Central Bell Telephone Company, in particular) expressed interest in MLR, we decided to totally rewrite the system because:

(*i*) The existing data base structure could not easily accommodate the new Universal Service Order (USO) requirements. (USO is the language used for the entry of customer service requests for new services or changes to existing services.)

(*ii*) An IBM 370-based system would not have been economical for the "small" towns (small, at that time, was anything less than 750,000 people.)

(*iii*) The IBM 370-based system's one-half hour or so mean-time-to-repair or daily availability of 16 hours was not satisfactory for a back-bone customer support system. It is important to note that given a fixed monthly availability, e.g., 98.5 percent, the user is more concerned with mean-time-to-repair than with mean-time-to-failure.

We initially planned to build two systems, an IBM 370 maxi-based system for the large cities and a separate DEC 11/45 mini-based system for the suburban/rural areas. They were to have identical user views to minimize development cost and time, and to facilitate moving RSBs and personnel from one system to the other. J. Cloutier of Bell Laboratories suggested the distributed architecture which we subsequently implemented as LMOS and which this paper addresses.

Though no MLR code was included in LMOS, the MLR system provided two major benefits to the LMOS project. First, it was a working model

* Registered trademark of Digital Equipment Corporation.

or pilot plant from which we could gather data and experience for the LMOS design decisions. Its role in this capacity cannot be over emphasized. Second, it acted as an existence proof that such a system could be used as part of an on-line customer interaction and could save the Bell System money.

II. ARCHITECTURE OVERVIEW

The hardware/software distribution of function and data is summarized in Fig. 1, while Table I summarizes the operational characteristics of the network.

The function/data distribution was such that the system could be partitioned into large complex-data "maxi" pieces and high-transaction-volume/small simple-data "mini" pieces. (See Fig. 1.) They were as follows:

(*i*) *Line card maintenance (a host maxifunction).* Copies of the customer's service and service history are maintained—a low-volume but large-complex data base function.

(*ii*) *Trouble taking and tracking (a front-end minifunction).* The customer's trouble is entered and subsequent repair tracked (via on-line status and reports)—a high-volume but simple data base function.

(*iii*) *Loop testing (a front-end minifunction).* The loop is tested for fault presence and location—a low-volume but complex algorithm function.

(*iv*) *Trouble history review (a host maxifunction).* A 40-day history of troubles is reviewed via "batch" reports for analysis of equipment and personnel performance trends—a large sort-and-count function.

The LMOS transaction load follows the classic 80/20 rule. That is, 80 percent of the total transaction load is generated by a few high-use

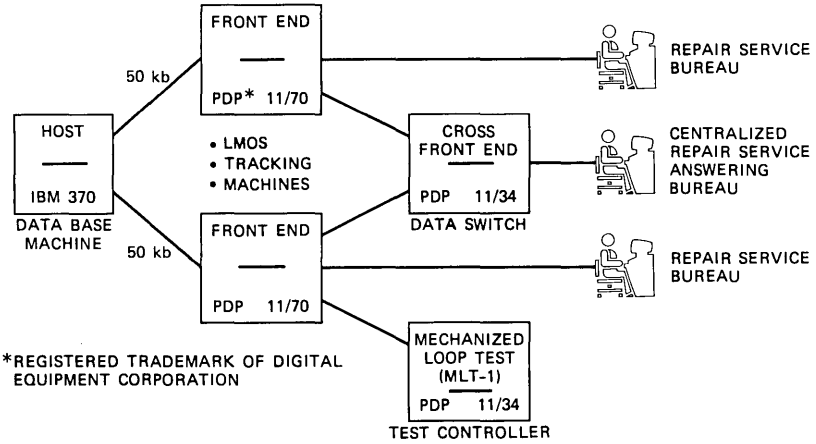


Fig. 1—Automated repair service bureau architecture.

Table I—ARSB data sheet: typical configuration serving five million customer lines

	Host	Front End	Cross Front End	MLT Controller
COMPUTERS				
Number in system	1	7	2	40
Processor type	IBM-370-303x	PDP* 11/70	PDP 11/34	PDP 11/34
Main memory (per machine)	4 megabytes	1 megabyte	1/4 megabyte	1/4 megabyte
Secondary storage (per machine)	6,000 to 8,000 megabytes	300 megabytes	0	0
TOTAL SYSTEM TRAFFIC				
Busy hour transactions	17,000	28,000	12,000	12,000
Daily transactions	145,000	175,000	80,000	60,000
OPERATIONAL PERFORMANCE				
Scheduled availability	24 hr (1)	22 hr	24 hr	24 hr
Operational	98.5%	99.5%	99.5%	99.5%
Mean time to recover	40 min	7 min	4 min	8 min
SIZE OF SOFTWARE (LINES)				
Batch	140,000	23,000	0	0
On-line	35,000	225,000	22,000	30,000
Operating system	(2)	34,000	12,000	12,000
Utilities	(2)	45,000	0	0
Total	175,000	327,000	34,000	42,000

* Registered trademark of Digital Equipment Corporation.
(1) On-Line 12 to 18 hours (depending on BOC). Available for batch processing all other times.
(2) IBM Multiple Virtual System and Information Management System.

transaction types (about 20 percent of all transaction types). These characteristics were first verified during the MLR pilot operation and justifies the distributed architecture of LMOS which places the high-use, simple data transactions on several front-end computers. The projected cost of the distributed architecture was 30 percent less than either an all-maxi or all-mini approach and allowed us to build one standard system rather than two.

The best way to understand the distributed architecture is to trace how a trouble is reported and repaired. This is as follows:

1. *Enter trouble description*—A customer calls a repair service attendant at the Centralized Repair Service Answering Bureau (CRSAB).[1] The attendant enters the telephone number of the line in trouble. The entered telephone number is switched by the cross front end[2] to the front end[3] having the copy of the customer's miniline card. (The miniline card is roughly 10 percent the size of the full line card which is kept in the host[4] IBM system; however, it contains enough "critical" data to support trouble entry, as well as limited operations when the host is "down.") In the case of numeric telephone numbers, a cross front end table is used to determine the proper front end. In the case of nonnumeric circuit identifiers, the cross front end interrogates all front ends to determine which has the miniline card. After

receiving the telephone number, the front end returns a description of the customer's line (and any existing trouble information) to the attendant via the cross front end.

2. *Test loop*—Simultaneous with trouble entry, the front end initiates a test of the line in trouble via the Mechanized Loop Testing (MLT)[5] controller computer. When the MLT controller returns the test results to the front end, roughly 25 seconds later, it routes them, again via the cross front end, to the attendant. The attendant, now armed with customer service information and test results, enters the trouble description and an agreed-upon repair commitment time.

3. *Route trouble description to RSB*—This final trouble description is sent via the cross front end back to the front end. The front end then ships this trouble description to the host computer for combination with the full customer line record and associated trouble history. This combined record is shipped via the front end, and perhaps cross front end, to the RSB having repair responsibility. (If the host computer is unavailable, the trouble description is combined with the miniline record for the RSB.)

4. *Repair trouble*—While being repaired, the status of the trouble, e.g., tested, is entered into the front end. In this way, trouble repair can be tracked by the bureau management using on-line reports, can be relayed to the customer, and can be analyzed after trouble clearance.

5. *Post repair analysis*—After repair, a description of the closed trouble is entered and sent to the host computer for a 40-day running historical file used for analysis of various components of the repair process. The resultant reports that are generated by the trouble report evaluation analysis tool (TREAT)[6] are routed from the host computer to the RSB (optionally via the front ends or cross front end).

While the repair process is taking place, the customer line records are being updated either manually or automatically from data received over the BOC service order distribution network on the host computer. Complete copies of the miniline record for all changed line records are sent to the front ends the night after the change is entered into the host computer. A night-time versus day-time update was chosen to off-load the front ends and the 50-kb lines connecting the host to the front ends (see Fig. 1).

III. MAJOR DESIGN DECISIONS

There were, in effect, two levels of ARSB architecture-design decisions: (*i*) those affecting the overall system (discussed in this section), and (*ii*) those affecting an individual component of the system, e.g., the front end (discussed in the later sections). The decisions that affected the overall system had one of two motivations—either trying

to satisfy the end users operational objectives or trying to satisfy our development objectives. The interplay between what *should* be done and how it *could* be done is the most exciting and yet exasperating experience for the designer. If either perspective—the user or the developer—gets out of hand, the end system will surely fail.

The major user and developer issues and lessons will be discussed in the following sections.

3.1 User issues—don't forget the end user will rely on and supplement your product

The major user issues which impacted the architecture evolved around the entry level skills of the end user, the desire to install the system quickly, and the high efficiency of the existing manual repair process. These factors all interrelated to lead to the following decisions:

(i) The System would not be based on a pure data base—don't make the machine enforce what the field won't.

After much interaction with the operating companies who wanted, initially, to have the software enforce a pure data base, i.e., reject a line record with machine detectable inconsistencies (e.g., a cable assigned to two telephone numbers), we decided that it was better to put the line record in the data base with its known errors than not to put it in until the errors were resolved. Known errors are, of course, flagged for future resolution. This decision was contrary to that being made by the facility assignment systems, e.g., COSMOS and BISCUS/FACS which assign loop facilities to provide service. We could not afford the difference in the cost of conversion—more than 6 to 1 for each customer record. Further, the end user of the system, repair people, historically had been able to use impure data for repair operations and would not tolerate the expense of trying to keep it purer than it had been in the past, though there would be tools to do this. (We have since found that repair does not in fact, keep the data pure, nor is it necessary.)

(ii) High-volume operations—do what you can, don't do everything.

The RSB operation is essentially a high-volume repetitive job wherein a 100,000-line RSB will process 500 troubles a day. The majority of these troubles is in residence and simple business services. This gave us the opportunity to save money by leaving the hard cases out of the system, both in the data base and in the automated testing. A human being handles these relatively low-frequency, very complex cases. Thus, the system could and can be viewed as the high-volume mechanized adjunct to the human being. The decision was: do not try to do everything; let the human augment the machine (and conversely!).

(iii) Part of the on-line customer flow—sit in the end user's job and view availability from that perspective, for example, 98.5 percent availability is 2½ hours down time a week!

The system is an adjunct to the daily customer interaction process. Thus, we had to ensure that high availability was provided for parts of the system that were used for daily interactions between BOC personnel and their customers. Further, these operations had to have a relatively predictable response time. Thus, we allowed no algorithm which had high run time variance, otherwise customer contact would be unpredictable. Also, we learned that the BOC user can manually augment missing features and, thus, will tolerate, though not happily, their absence; however, they will not, and should not, tolerate bad performance or availability.

As mentioned before, in the operations support world, mean-time-to-repair is much more crucial than average uptime. (Two hours down Monday morning is intolerable.) A rapid mean-time-to-repair allows less stable software to be introduced to the field. The result of this issue was a front end and cross front-end hardware configuration which had at least one passive backup unit for each two on-line units. This backup hardware could be switched on-line via a network switch and disk switches. Also, a very fast data base recovery system was built, (less than two minutes recovery time for all but the head crash).

3.2 Developer issues—manage complexity

The development decisions were all motivated toward reducing development complexity. All the major design decisions were oriented towards decoupling the development of the subcomponents. (An extension of this theme was not to make any changes in the IBM software for the front-end system to work.) We partitioned the software tasks into pieces which could be done by teams of three to five people for development efficiency and risk containment. Further, by uncoupling the development activities from each other, the companies could decide on the order of installation for the various pieces of the system; however, there was a natural and commonly followed installation sequence of TREAT,[6] host, front end, cross front end, and MLT. It is important to note, however, that the overall architecture and development approach was for an integrated system which could be developed in phases as compared to stand-alone pieces which could, somehow, be forced to fit together.

The resultant major decisions to allow this decoupling were as follows:

(i) Duplicate data to simplify distributed processing—reduce risk by using known technology.

We decided that no single computing activity would be dependent

on the on-line interaction between computers. That is, we did not want a single user transaction to have to get data from more than one machine. We did not know how to solve the data base locking, consistency, and data base recovery problems that accompanied that form of operation and so we avoided it. (For example, we did not want to solve the clean-up problem resulting from transaction A on machine 1 locking data base B on machine 2, updating it, going down, and simultaneously having the line go down.) Instead, we duplicated a small amount of data across machines in a master (host data base)/slave (front end) relationship to support the transaction activities.

(ii) Communications design—do the riskiest first.

The most complex design problem—though we did not know it at the time—was the distributed communications network. For MLR, the PDP 11/20 communications software we wrote handled solely the synchronous terminals (not the printers!) in a stand-alone fashion (i.e., we did not have a full pilot plant).

Our initial design was quite simple. We decided that to each sending computer, the receiving computer would look like a TELETYPE* teleprinter 40/4 controller. The IBM host computer thought that the DEC front-end computer was a pair of 3270 controllers, one with 32 printers and one with 32 CRTs. The CRTs were used as "virtual" terminals by the PDP computer and were assigned or coupled to a real terminal when it used an IBM resident transaction. In this way, the IBM transactions did not have to be tested for use with the front end and no special work had to be done in the front end when an IBM transaction changed or was added.

Two of the "virtual" CRTs were used to effect the batch data transfers between machines. That is, the miniline cards, for example, were blocked into CRT-like messages and were sent to one of these two CRTs.

The handling of printers and front-end-to-host messages almost killed the project. The early strategy had four parts:

(a) By mapping the printers on the front end to one of the "32 on the 50-kb line," the host system would do all the spooling for Host generated messages.

(b) To reduce spooling logic, the front end would separately spool its output for the printers. The front end would control actual printing by either "unspooling" its messages or control passing through of the host messages.

(c) To simplify the software, no messages from the front end to the host would be spooled to disk. Rather, as these messages were created, they were put in a core buffer for transmission to the host.

(d) When the host was "attached" to a printer or CRT, all terminal

* Registered trademark of Teletype Corporation.

and printer status messages would be passed directly from the front end to the host.

The first two decisions were good ones—the last two were somewhat less than optimal. By not spooling to disk, we implicitly counted on predictable bandwidth to the host, i.e., the core buffers could fill up on the front end and stop the system. Further, by sending the status directly to the host, sometimes it would take the whole 50-kb line down because, for example, it thought a bad printer was on the line, i.e., we confused network and terminal control. These two design problems took 3 to 6 months to fix, generated user dissatisfaction, and "work-arounds" that complicated the communications manager design. The major lesson was that we tried to schedule the development of something that neither we (nor anybody else) had done before.

(iii) Operational decoupling—reduce the need for organizations to communicate in the field.

We decided that the basic operations of the host and front end had to be viewed by the computer operations personnel as though, for all practical purposes, the other system did not exist. For example, if a data base recovery took place on one system, then an operator (or software) would not have to initiate one on the other. This decision actually simplified the overall final design but was hard to implement as we built a variety of ad hoc message synchronization schemes and (over the years!) found holes in them.

We shall next discuss the architectures of the individual subsystems or components.

IV. HOST DESIGN

The design of the host software was driven by two major issues—the use of the IBM support software and the largeness of the data bases (several billion bytes on line). The decisions were as follows:

(i) Use PL/I and only simple features of IMS—avoid local optimization and use of complex features.

PL/I was chosen for its relative development efficiency. The other feasible alternatives of COBOL and Assembly were rejected because of their awkwardness or resulting code complexity. Further, as far as we could tell, we suffered little to no performance impact by choosing PL/I over Assembly.

The Information Management System (IMS) was used to create simple hierarchical data bases. We avoided using any new IMS programming feature for approximately two years after it was announced. By placing these restrictions on the use of IMS, we made sure that the feature had been debugged and we substantially improved the overall system field reliability and performance, as well as easing our internal training problems.

(ii) Data base size—big data bases have lots of inertia.

The projected size of the data bases (many billion bytes) resulted in two major design decisions. First, a series of inverted files were created from the line card file so that data could be indexed and retrieved other than by telephone number. Without these indices, the query of telephone numbers by cable pair, for example, would take hours. These files included a cable file, office equipment file, as well as three other small files. Each of the programs which updated the line card file also updated these inverted files. (A common, but poor practice of the computer science community at that time was allowing individual programmer access to the indices. Data base structure and indices should be hidden from the programmer by access routines.)

Given the need to reorganize data bases, take image copies, and recover them, care had to be taken to split a potentially large single data base into several logically smaller parts. While we recognized this for the line card data bases, unfortunately we did not for the cable data base. The cable data base became exceedingly large in one of the "rural" BOCs, which had extensive cable networks, and several 3350 disk drives were required. Reorganizing, recovering, and loading this one data base soon became the pacing item of the host computer operations.

(iii) Performance prediction—"off-line" not "on-line" will get you.

The performance prediction for the host computer was simple for the real-time day and surprisingly complex for the night time, off-line batch operations. The transaction processing in an IMS system inevitably is CPU limited. (The majority of the CPU cost is due to I/O processing; however, the CPU, not the channels or disks, saturates first.) Thus, all we had to do was to estimate the CPU utilization of the on-line transactions. Since we had the MLR system in New York for a model, we estimated the performance of the distributed system by removing the load of those transactions which would migrate to the front end. The only problem that we ran into was that as the system matured at a BOC, terminals accessing the data base for its review spread like wild fire. As experience with the system grew at BOCs, this extra use added an additional 20 percent to 30 percent load to the repair-only predicted load.

The nighttime load prediction had two problems. First, our MLR model experience was in Manhattan which was, at the time, having very little, if not negative, growth. As the system spread to the expanding areas, the need to do massive rearrangements of the data base for area transfers, major cable throws, etc., added what was to us an unexpected load. Most importantly however, we did not leave enough time for operator or machine error. In effect, we had planned

the operational evening too tightly. To meet the original load projections, we had to redo some of the programs so that they could run during the real-time day and in effect cut down the need for "off-line" time. The only real lesson here is that one really needs to understand all the environments of the end system, including the variants of the offered load and the realities of how big machines operate and are operated.

V. FRONT-END DESIGN

5.1 The operating system—there is no such thing as a simple operating system

The front-end system was to be, in effect, a transaction processor. The first design decision was related to whether we should use what at that time was a rather new operating system from Murray Hill—UNIX* program operating system—or build our own. Because of the need for a robust file system, high-performance interprocess communications, and the need to handle forty-eight 4.8-kb synchronous communications lines, we decided to build our own. It is still not clear whether this was our best or worst decision. It was best from the viewpoint that subsequently evolving the UNIX program operating system to handle the high-performance, high-availability, synchronous-terminal-driven, transaction-processing application has been most complex. It was the worst in that we might have been able to substantially modify the UNIX program operating system to satisfy our needs and in the process would have had earlier access to C language and the UNIX software development environment. (Two years into the development, we moved our development environment to the UNIX program operating system and started to use C as the programming language of choice. Later, (see Ref. 3), we decided to modify UNIX software, given our experience with the special-purpose operating system.) The other impact was that, as all operating system developers are, we were plagued with the never-ending minor utilities for operational and maintenance ease, e.g., the utility to do system file transfer.

The other major design features of the front end were all oriented towards high predictable performance and low mean-time-to-repair.

5.2 Design for availability—(It's easy if you include it in the design from the beginning.)

The low mean-time-to-repair was achieved by providing a standby

* UNIX is a trademark of Bell Laboratories.

processor, extra peripherals, and a rapid software recovery system. The recovery system was based on

(a) Using the communications terminals for message recovery—a message was processed in its entirety, one in and one out. The terminal held the input message until completion. If the system went down, the operator would re-enter the message.

(b) Keeping preupdate copies of a transaction's intended updates on disk to roll back the data base in case of error.

(c) Keeping a log tape of all before and after data base copies which were used to reconstruct the data base in case of a head crash.

This rather simple design resulted in a mean-time-to-repair of approximately two minutes for all but the head crash which would take approximately an hour. During the hour recovery, a simple back-up system was given to the user so that troubles and statuses could be entered for journaling on tape. This journaled information would then be read into the system when it was brought on-line so that the bureaus would not have to do any special manual catch-up work.

5.3 Performance—(Keep the model simple and worry about it from the start.)

The performance design was based on assuming the system would be designed to be single-threaded, i.e., once a transaction started, no other would run. This solved the data set locking problem, simplified recovery, and seemed like a reasonable approach given that at that time we only had two disks. Given this and using an application load model from the New York Telephone Company (which remained relatively invariant across other companies), we projected what the single thread load would be at peak busy hour. *Before announcing system capacity*, we designed each transaction and counted their disk accesses. Since the transactions were all simple, their elapsed time was directly related to the number of disk accesses. We then sized every other component of the system, the 4.8-kb lines, the 50-kb line, and the core buffers for holding transaction input, so that they would not be a limiting resource. Simple single server and multiserver queuing equations were used for the sizing.

The system algorithms were then designed to drive this single-server queue, i.e., the transaction stream. For example, the priorities for polling the terminals were such that once work came into the system, the polling priority would be lowered so that the transaction would run to completion. The major performance tuning was in handling the 50-kb line to the host and the communications buffers in the front end. These tuning needs were driven by all the idiosyncrasies of handling the bisynchronous protocol in an environment with noisy and failing lines.

VI. 11/34 MLT CONTROLLER DESIGN—Reexamine your architectural decisions under change.

The 11/34 MLT controller originally supported a special-purpose terminal which is no longer used in the system. This terminal, the Status Entry Device, was specially designed and built to be used to enter numeric status information into the system. Originally, it was used in the MLR system and was viewed by the computer as a TELETYPE CRT. On changing to the IBM 3270, we decided to use an 11/10 to simulate an IBM 3270 controller and to co-locate it with the Status Entry Devices. Once the 11/10 was at the bureau, it seemed natural to use it for controlling the loop tests. (The 11/10 was replaced later with the 11/34.) The building of the special terminals was a well-motivated detour. They were built because normal terminals were too big to use with the test desk and were too expensive. However, their production volume was never enough to allow cost reductions; thus, they did not track the cost reductions in the terminal field, but more importantly, the tester ultimately needed a full-feature terminal as the ARSB system grew in feature.

Once we decided to co-locate the 11/34 in the bureau, its design was actually very simple. For reliability, it had to have no moving peripherals. However, while it was appropriate to have a separate computer for driving the loop testing system, the 11/34 was probably not a good design choice. A better choice, both in cost and later development and deployment flexibility might have been to have a separate PDP 11/70 computer for testing as compared to several 11/34's. We recognized this too late in the development/deployment cycle.

VII. CROSS FRONT END—Geography is a bad division.

The cross front end was the last major addition to the system. Though we had recognized the potential need for such a system in the large cities, it represented a nicety and not a necessity. The design guidelines for the cross front end were identical to that for the MLT controller, with one addition—all cross front end tables had to be automatically derivable from the front ends. We could not stand the thought of the operator difficulty, then error, and then our repair, of having to keep the systems in constant synchronization. The performance approaches for the cross front end were almost identical to the front ends. The only exception was in assuring that one heavily loaded, or failing, front end did not use up all the buffers in the cross front end and, thus, jeopardize access to the other front ends.

Though the cross front end did the job, it was the early warning signal for the need to redo the ARSB architecture. Until the cross front end arrived, the system was completely hierarchical serving a smaller, self-contained geographic entity. The cross front end, in effect, re-

sponded to the desire of the BOCs to avoid the need to organize their operations according to this rather rigid computer-imposed geographic view. In several installations, the cross front end use grew beyond its original design intent of serving just the CRSABs to serving coin bureaus, and special business bureaus. This growing need led to the second architecture for the ARSB which is described in Ref. 3.

VIII. MAJOR LESSONS

Some of the lessons already described and one or two others deserve special mention as follows:

8.1 Time scales and tolerance to change

The LMOS system, from its inception to its full Bell System penetration, will take 13 years to complete. There is no way that any operations systems designer is 13 years smart. Thus, the early architecture must be very carefully examined to determine impact of BOC organizational change, changing data needs, etc. The issue of performance was very significant to the first generation of the ARSB. This issue of flexibility versus performance will have to be balanced exceedingly carefully in succeeding generations of the system.

8.2 Data view

The host system was designed from a functional view point, i.e., a conversion system, an automated line record update system, an on-line system etc. This view did not recognize that the most complex and unyielding issue was the large and unwieldy data base. Future systems should take a more data-centered view and build for example, a cable system, a customer service record system, etc. This should be done because it is hard to evolve mechanisms which allow easy minor changes of the basic view of the data base.

8.3 Data synchronization

The growing use of the LMOS data base resulted in our having to synchronize it with many other systems. We found this to be an almost impossible task unless the synchronization was done using self-contained groups of information, e.g., a full miniline record. Even then, minor differences in item definition (it took 6 to 9 months to resolve cable/pair status definition between two systems) would lead to terrible confusion. The only lesson is that if at all possible, do not duplicate data; however, if you have to, synchronize it in very big blocks.

8.4 Modularity—planned and achieved

Perhaps the major advantage of the distributed architecture was that it enforced modularity on the system. It seems that only physical

boundaries, either different machines or 16-bit address space or time slots, enforce that modularity after the early designers leave.

8.5 Models and pilot plants

The major lesson towards building a predictable architecture and system is to build a pilot plant first. Make that pilot plant exercise the technical risk and use it to gather data for the performance versus flexibility equation.

IX. ACKNOWLEDGMENTS

A system can have an architecture only after someone conceives the need for it. M. W. Bowker of Bell Laboratories deserves credit for that vision. Major contributors to the architecture were L. S. Dickert, J. M. Hunt, III, D. S. Watson, D. L. Kessell, S. G. Glover, D. Lloyd, S. Hensdale, and E. Mays.

REFERENCES

1. M. W. Bowker et al., "Automated Repair Service Bureau: Evolution," B.S.T.J., this issue.
2. J. P. Holtman, "Automated Repair Service Bureau: The Cross Front End: The Context-Sensitive Switch," B.S.T.J., this issue.
3. S. G. Chappell, F. H. Henig, and D. S. Watson, "Automated Repair Service Bureau: The Front-End System," B.S.T.J., this issue.
4. C. M. Franklin and J. F. Vogler, "Automated Repair Service Bureau: Data Base System," B.S.T.J., this issue.
5. O. B. Dale, T. W. Robinson, and E. J. Theriot, "Automated Repair Service Bureau: Mechanized Loop Testing Design," B.S.T.J., this issue.
6. S. P. Rhodes and L. S. Dickert, "Automated Repair Service Bureau: The Trouble Report Evaluation and Analysis Tool," B.S.T.J., this issue.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6 July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Data Base System

By C. M. FRANKLIN and J. F. VOGLER

(Manuscript received June 11, 1981)

The role of the data base system of the Loop Maintenance Operations System (LMOS) is to maintain up-to-date information about the customer's telephone service and trouble history to facilitate customer trouble repair. The discussion covers data base content, rationale for the data base system architecture, and methods for keeping the data current.

I. OVERVIEW—LMOS

To provide an understanding of the data base system issues of LMOS, this overview shows functional linkage between the various system parts and briefly discusses the host architecture design, data base conversion, and data base update. The remaining sections of the paper provide detailed discussions of the evolution of the host data base system architecture, data base conversion strategy, and methods and types of data base update required to keep the operational data base current.

The Automated Repair Service Bureau (ARSB), described in this issue of *The Bell System Technical Journal*,[1,2] consists of two major functions: mechanized management of customer repair data and mechanized testing. The LMOS, a distributed system, is the customer repair data management system. The LMOS host maintains customer line record and trouble data so that repair personnel have up-to-date information about the facilities being repaired. The LMOS front-end transaction processors record and track troubles on telephone equipment from the time the troubles are reported until they are repaired.

For simplicity, discussion of other ARSB systems, such as Mechanized Loop Testing (MLT), Loop Cable Administration and Maintenance

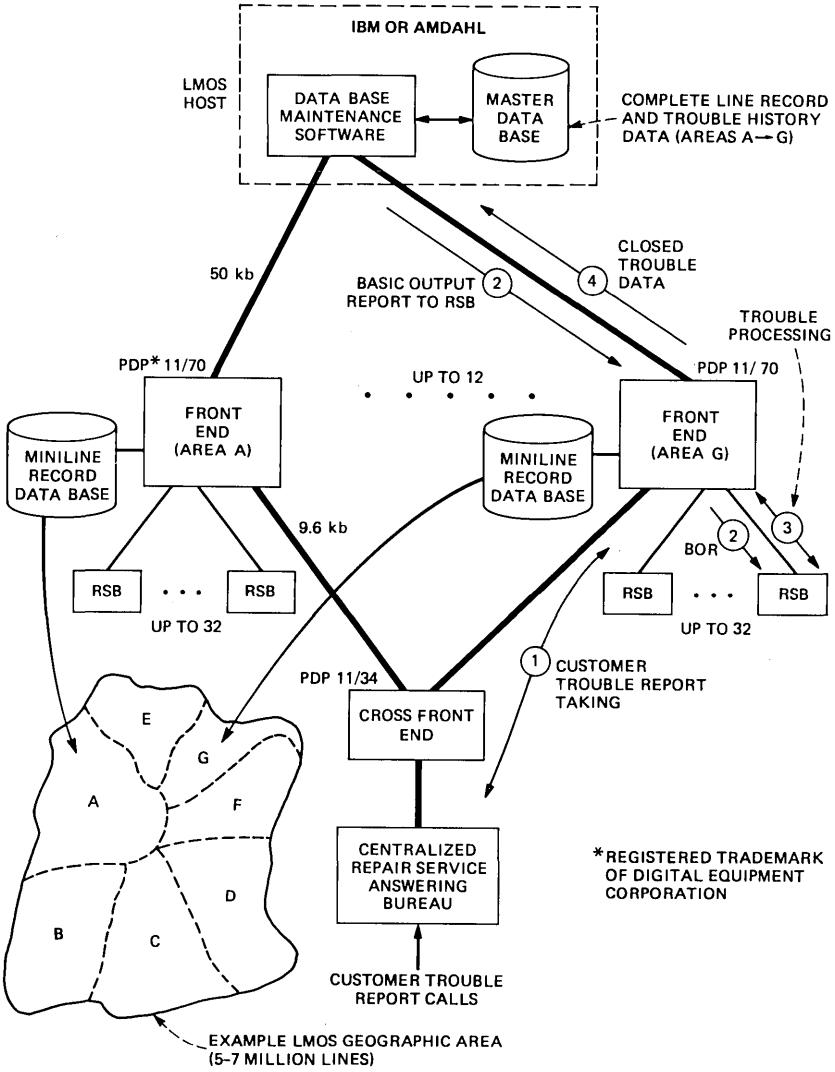


Fig. 1—Loop maintenance operations system (LMOS).

Operations System (LCAMOS), and Trouble Report Evaluation and Analysis Tool (TREAT) are not covered in this paper. See Refs. 3, 4, and 5 appearing in this issue.

1.2 System organization

Figure 1 illustrates geographic overlay of the system on a typical Bell operating company's (BOC's) area of application (from five to seven million lines) and the trouble processing functions supported by major elements of the system.

The LMOS host data base stores in a large "master data base" complete information on the customer's telephone service including premise equipment data; class of service; network facilities assigned to the customer's circuit; and trouble history. In the example LMOS geographic area of Fig. 1, complete "line card" and trouble history files for all customer lines in the five-to-seven-million-line area of application exist in the host data base.

The distributed front-end transaction processors form the real-time interface with customers via the Centralized Repair Service Answering Bureau.[6] An LMOS installation can have up to 12 front-end transaction processors. The maximum capacity of a single transaction processor is approximately one million lines. The actual number of transaction processors installed will depend on considerations such as transaction rates and area boundaries.

In the example shown, we assume seven transaction processors, each mapping into one of the subareas A through G. The data base for each front-end processor contains a subset of the line record data in the host and is called a miniline record. The miniline record contains essential information for repair and is used principally for trouble report taking. If the host is down, the miniline record can also provide basic information for processing troubles to a "closed" status. It is about one-seventh the size of the full line record on the host. In addition, a given transaction processor contains miniline records only for customers in its subarea.

1.3 Trouble processing

Figure 1 also illustrates how the distributed architecture of LMOS supports customer report processing. Assume the customer's phone service is in subarea G and is out-of-order. The customer may report the trouble from any location within the total area; however, the cross front end links the Centralized Repair Service Answering Bureau with the front-end data base serving subarea G, while the trouble report is being taken (event 1).[7] When the trouble report is forwarded to transaction processor G, it requests a Basic Output Report (BOR) from the host. The BOR, containing complete line record data, test results, and trouble history, is transmitted to a printer in the RSB serving that customer (event 2). The BOR is screened for the appropriate next step, which may include further tests, a craft dispatch or other activity as required to repair the trouble (event 3). When the telephone circuit is repaired, trouble report closeout information is transmitted to the host (event 4).

1.4 Host data base architecture

Figure 2 shows the architecture of the LMOS host data base system.

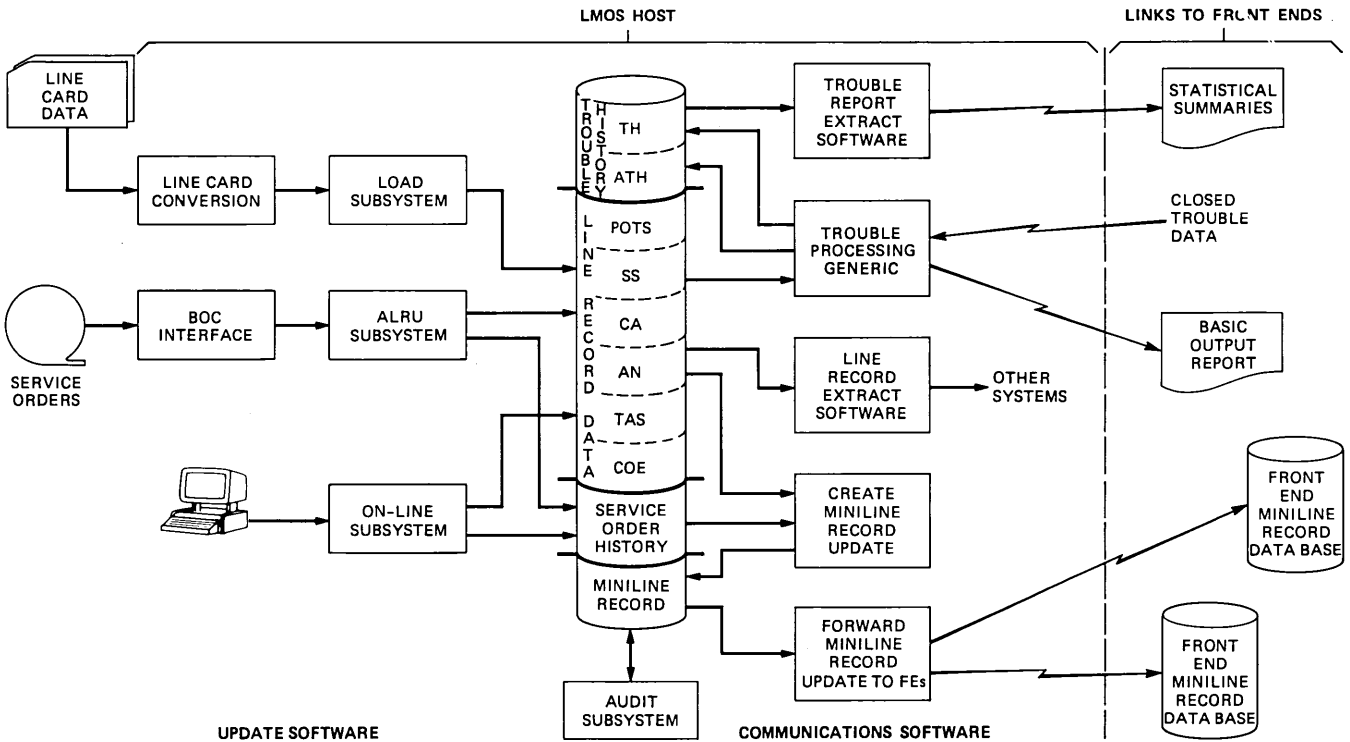


Fig. 2—LMOS data base systems architecture.

The three host software generics on the left keep the data bases current. To the right are data base interfaces with the front-end transaction processors and the statistical report generation software, Trouble Report Evaluation and Analysis Tool (TREAT).[5] Major divisions of data are as follows:

Past trouble history—The Abbreviated Trouble History (ATH) data base contains, as a minimum, the most recent 40 days of history. The Trouble History (TH) data base contains histories of troubles closed during the day and is used to support TREAT statistical reports.

Line record—These data bases contain information about the customer's telephone circuit. The Plain Old Telephone Service (POTS) and Special Services (SS) data bases are identical structures, except the POTS key is the 10-digit telephone number, while the SS key can be any alpha-numeric up to 16 characters, plus number plan area (NPA). These two data bases form the basic line record information. (Note that for LMOS convenience, the LMOS definition of an SS is any circuit having an identifier that is other than 10-digit numeric with a real NPA.) The Cable (CA), Associated Number (AN), Telephone Answering Service (TAS) and Central Office Equipment (COE) data bases contain data common to the line record file but have been "inverted" for access by cable and pair number, telephone number associated with a main account, TAS number associated with a customer's telephone service, and central office exchange key and switching equipment number, respectively.

Miniline record—These are reduced versions of the POTS and SS line record data bases described above. There is one miniline record data base for each front-end transaction processor; the miniline record provides the mechanism for transferring changes that have occurred in the host to the front end.[6]

Service order history—This data base contains a list of all line records changed during the day. The list is used for constructing miniline records to be sent to the front ends.

1.5 Data base conversion and update

The three modes for processing changes to the host data base are LOAD, Automatic Line Record Update (ALRU), and ON-LINE. The LOAD subsystem is used to initially create the line record data base from existing paper records or from other mechanized sources. The ALRU automatically performs the bulk of day-to-day changes to the records because of service order activity. The ON-LINE subsystem provides a means for manual inspection and/or change of line record information via a CRT; the principal uses of this subsystem are error correction, input of nonstandard service orders, and input of information as a result of plant rearrangements and changes (work orders).

Audits provide for internal consistency between common data items in the various data bases. Accuracy checks usually require data comparison to physical circuits.

1.6 Data base decisions in retrospect

The data base system of LMOS had to accommodate the following major repair functions:

(*i*) *Taking trouble reports*—The operational objective was to display, in five seconds or less, information about a customer's telephone service when the customer contacted the Centralized Repair Service Answering Bureau.

(*ii*) *Tracking open troubles*—The system had to provide a capability for accepting new trouble reports and maintaining status information about the troubles until closed out.

(*iii*) *Maintaining trouble history*—The most recent forty days of closed trouble information had to be maintained in the data base for summary and review purposes. Afterwards, the trouble history data could be transferred to microfilm storage.

(*iv*) *Maintaining up-to-date line record data*—Changes made to the customer's telephone service had to be reflected (typically within 24 hours) in the LMOS data bases.

While the above list is not exhaustive, it does show a requirements pattern for the data base system. Functions (*i*) and (*ii*) require the data base system to provide rapid access to data and to manage volatile trouble report information while the trouble is being processed. Functions (*iii*) and (*iv*) are characterized by long-term storage of large amounts of data that change relatively slowly (i.e., 1/3 to 1/2 percent of the data changes every working day).

It was decided that functions (*i*) and (*ii*) could best be met with small data bases distributed across several front ends. These data bases would contain copies of essential line record data (miniline record) obtained from a large master data base. The master data base (referred to in this paper as the LMOS host data base) would be the focal point for all updating and distribution of line record changes throughout the LMOS system.

Using redundant storage to meet response time requirements somewhat complicates the data base update process, but time constraints on update are much less severe and the penalty was considered worth paying. In addition, the power of a large main frame machine (host machine) could be effectively used to update the master data base and to propagate changed miniline records to the front-end data bases.

Regarding data base conversion, another judgment made early in the program was not to require BOC's to purify records prior to LMOS

load. The rationale was twofold. First, the repair operations suffered mostly from lost records as opposed to inaccurate records. With LMOS, the line record would always be available and at a quality level with which the BOC chose to operate. This philosophy significantly reduced data base conversion expense. Secondly, tools were provided in LMOS to permit gradual record quality improvement if desired by the BOC. This basic decision is one of the main reasons that LMOS has gained a rapid market penetration.

1.7 Summary

This overview has summarized the LMOS data base system from the viewpoints of trouble processing flow and data base architecture to support this flow. Advantages being realized by the distributed data base architecture described include the following:

(*i*) Use of inexpensive minicomputers as transaction processors with small data bases while taking advantage of the Information Management System (IMS) data management system for manipulating large data bases on the host.

(*ii*) Availability of several highly reliable transaction processor configurations for real-time customer interaction.

(*iii*) Ability to locate the transaction processor near the RSBs being served to minimize communications costs.

To date, the data base design has served the loop repair process well. At the end of 1980, approximately 50 million customer line records were resident in LMOS installations throughout the Bell System. This huge reservoir of data (60 billion bytes) is now being viewed as a system resource that will undoubtedly be tapped to support other loop operations in addition to repair.

II. DATA BASE ARCHITECTURE OF LMOS

2.1 Evolution of architecture

Architecture of the data base system of LMOS has evolved from the centralized data base design of the prototype system installed in the first trial company in 1972.[1,2] Experiences with that system and additional data needs of the second and third BOC customers forced changes in the data base structure.

The prototype system divided the line record data between two data bases, POTS and SS, to allow use of IMS's fastest access method for the POTS data base. The prototype system installed in December, 1972, contained data that today is kept on the front end. At that time, the line record data bases also contained data about open troubles on the circuit. The trouble history (TH) data base contained both trouble history and history of changes to the line record data bases, called service order history.

During the early months of the first trial company's conversion, projections indicated that a POTS data base for a 2.5-million-line system would span twenty 3330 Model 1 disk packs, making the necessary backup and recovery processes intolerably slow and unmanageable. The two line record data bases were split into seven POTS data bases and three SS data bases, and a "WHICH" table was added to tell which data base contained a line record, given the exchange (NNX) of the line record key.

Data base lockout problems were experienced because, at that time, IMS prevented access to the entire data base while updating a record. To reduce the incidences of lockout, two new data bases were created: one for the open trouble data that resided in the line record data base and one for the service order history data that was in the TH data base.

The open trouble data base was moved to the front-end system with the introduction of the distributed LMOS in the second BOC in 1974. (The TH data base remained on the host because the data base was so large.) Since the lines covered by the second installation included different area codes (NPAs), NPA was added to the line record key and the WHICH table was expanded to include NPA with the NNX.

The structure of the POTS data bases changed again before installation in a third BOC in 1975. This BOC has many NNXs in which the assigned telephone numbers fall predominantly in certain thousands (last 4 digits) groups. For instance, there might be 800–900 numbers of the form 8611xxx, but less than a hundred of the form 8612xxx. Since the data base design at that time allocated space on a switching entity basis (10,000 records), this would have resulted in very large POTS data bases with many records having unassigned numbers. The data base was redesigned to allocate space on one-thousand group entities rather than ten-thousand group entities, thus, reducing wasted storage space.

The principal lesson learned from the introductory experience is that the repair operation environment varies widely from company to company. It is strongly recommended, prior to the introduction of major mechanized systems, that prototype "soaks" in at least two companies having widely different geographic environments be performed.

2.2 Current architecture

The LMOS host data base structures are hierarchical. The structure of the line record data base serves as an example. The line record data base contains a root segment for data that is constant in length and almost always present. Examples are the line record key (telephone number or circuit number), central office equipment, listed name, repair route, indicator for special service protection, essential line

number, and class of service. Variable length data items are kept in child segments having the following structure names:

LCLOC	Line Card Location—Contains additional listed name, service address, and location information. (The first 55 bytes only are stored in the root segment.)
LCRMKR	Line Card Remarks, Retained—Contains remarks to inform the repair technician about access and equipment information.
LCSE	Line Card Service and Equipment—Contains codes for customer's service and equipment.
LCSEN	Line Card Service and Equipment Narrative—A child segment of LCSE that contains narrative about the service and equipment.
LCCL	Line Card Cable—Contains cables and pairs assigned to the circuit.
LCCLN	Line Card Cable Narrative—A child segment of the LCCL that contains cable narrative, binding post, and terminal address data.
LCISG	Line Card Incoming Service Group—Contains hunting data.

Figure 3 shows the line record structure. Any of the child segments may have multiple occurrences. The customer trouble processing operation frequently results in having to access line record data when the telephone number or circuit number is not available, but, one of the following is:

(*i*) Central office equipment number
(*ii*) Cable pairs

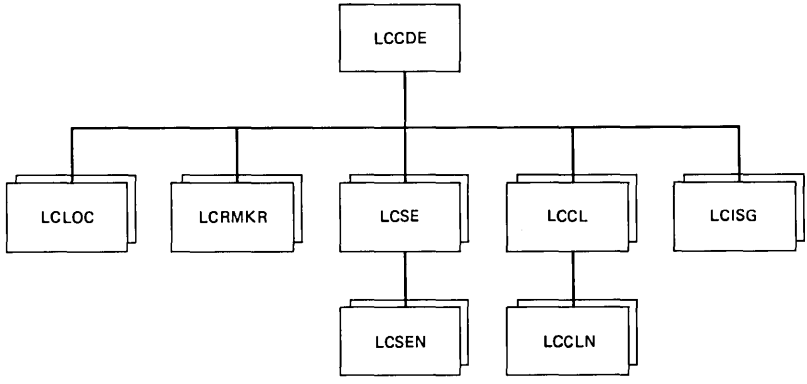


Fig. 3—Line record structure for POTS and SS data bases.

(*iii*) Main telephone number if the circuit is part of a multiline account

(*iv*) Board, position, and jack if the circuit has telephone answering service.

This access capability was implemented by building four inverted data bases rather than using the secondary indexing provided by IMS. Access through the secondary index would have been too slow and data base reorganizations would have been required too frequently. The price for the inverted data bases is that the update programs must consistently update two data bases every time one of the data items is added or deleted; when programs have bugs, inconsistencies between the line record data base and the inverted data bases result. This points to the need for a common data base access routine (which has not yet been implemented).

The four inverted data bases are described below:

(*i*) *Cable (CA)*—The CA data base contains a segment for each 25-pair complement within the cable. Associated with each pair is a pair status, taper code, pair use, and telephone number if the pair is working. Multiple telephone numbers working on the same pair are listed in separate child segments.

(*ii*) *Associated number (AN)*—Large business accounts have a main account telephone number that is used for billing and other central functions. All other telephone numbers assigned to that business are known as associated numbers. The AN data base contains a root segment for a main account telephone number and a child segment for each associated number. A program performing a disconnect of a large business account would use this data base to find all the line records to update.

(*iii*) *Central office equipment (COE)*—The COE data base contains a segment for each piece of COE. Multiple telephone numbers working on the same equipment are kept in a separate child segment, one for each additional number. A separate data base, the COE parameter data base, contains the range of allowable COE numbers. This data was separated from the COE data to reduce the disk space needed for the COE data.

(*iv*) *Telephone answering service (TAS)*—The TAS data base contains a root segment for each telephone answering service board and position number. Two occurrences of a child segment exist to associate the telephone number of TAS customers with their jack numbers, one for jacks 1–49 and one for jacks 50–99. Multiple circuits working off the same jack are kept in a third-level segment.

Brief descriptions of the other LMOS data bases follow.

(*i*) *Miniline card (MLC)*—The MLC data bases provide the mechanism for transferring changes that have occurred in the host data

base to the front-end data bases. There are up to 12 MLC data bases in the host, one for each front-end system. At the end of every day, a miniline record is constructed from each line record that was changed during the day. These miniline records are placed in the MLC data bases. During the following day, the front-end system asks for the new miniline records to refresh its miniline record data base, which is used for trouble report processing.

(*ii*) *Service order history (SOH)*—The SOH data base contains a list of line records that were changed. After the miniline records are built and placed in the MLC files, the SOH data base is reinitialized.

(*iii*) *ALRU messages (ALRUM)* and ALRU recovery monitoring *(ARM)*—The Automatic Line Record Update (ALRU) system (described below) uses these two data bases. The ALRUM data base collects error messages during the ALRU run. When ALRU finishes, the messages are sorted and distributed, and the data base is reinitialized. The ARM data base contains data useful for ALRU recovery and monitoring.

(*iv*) *Cable fail (CF)*—The CF data base contains a list of all cables for which a known cable failure exists.

(*v*) *Trouble history (TH)*—The TH data base contains trouble history for all troubles closed during the day. At the end of the day, the front-end system sends trouble history data to the host to populate this data base, which is the primary input to the Trouble Report Evaluation and Analysis Tool (TREAT). The TH data base is reinitialized daily.

(*vi*) *Abbreviated trouble history (ATH)*—The ATH data base contains a subset of the data in the TH, but it keeps, as a minimum, the most recent 40 days of trouble history.

(*vii*) *Pending service order (PSO)*—The PSO data base contains the text of pending service orders by main account telephone number. The BOR checks this data base to warn the repair technician of any pending work on the telephone number. About half of the LMOS companies have implemented this data base since its utility varies from company to company.

(*viii*) *Completed service order (CSO)*—The CSO data base contains the text of completed service orders by main account telephone number. These orders are used primarily for reference when correcting errors generated by ALRU. As above, about half of the LMOS companies have implemented this data base.

2.3 Data base sizing for a typical BOC

For a five-million-working-line system, the LMOS host data bases will contain about six billion bytes of data, and the total of all LMOS front-end data bases will contain about 1.3 billion bytes of data. The "per line" average is shown in Table I.

Table I—Average bytes of data per line record

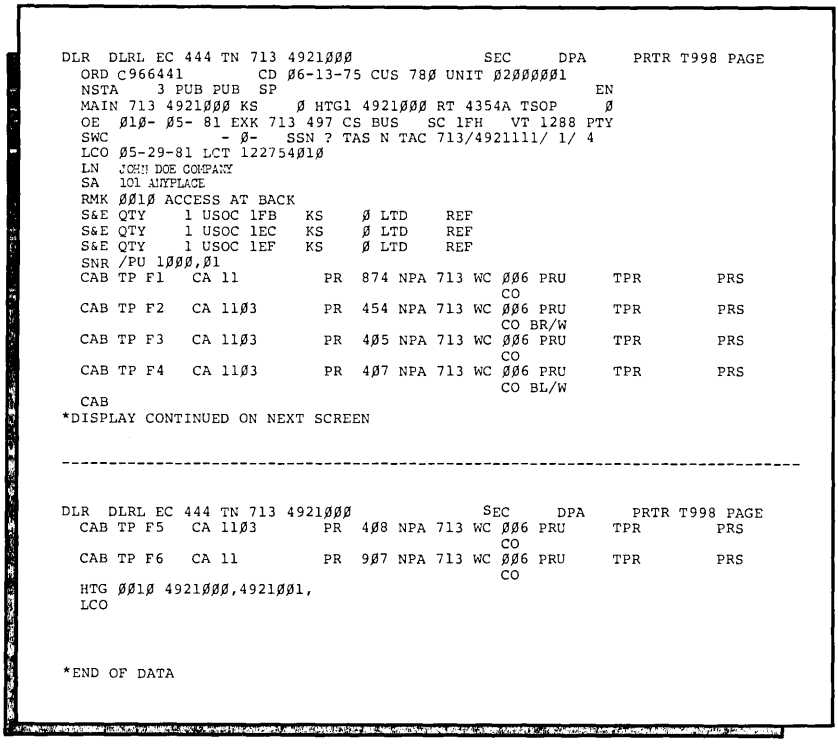
Host Data Base	Bytes	Front-End Data Base	Bytes
Line record	790	Miniline record	130
Cable	250	Open troubles	30
Office equipment	45	Testing, other	30
Associated number Telephone answering	(negligible)	Index to open troubles and line records	70
Closed troubles	60		
All other	75		
Total (per line)	1220	Total (per line)	260

Note that the front-end data base miniline record contains only 15 percent of the host line record data, since the front end must contain only that data subset required for on-line customer trouble report processing when the host is down.

Figure 4 is a display of a miniline record. The corresponding host line record is shown in Fig. 5. Note the additional information the host line record contains. The line record lists customer service and equipment codes and accompanying narrative (S&E), retained remarks (RMK), which may contain premises access information, the vertical termination of the originating equipment (VT), party position number

```
DMLR  TN- 713 4921000        PG- 1    PRTR- W998    REQ BY- JFH

LN-  JOHN DOE COMPANY                  WKG- YES          UNAS- NO
SA-  101 ANYPLACE                      DISC- NO          TEMP NWKG- NO
LOC-

OE- 010- 05- 81   RPT- 4354A    UNIT- 02000001
SC- 1FH    CS- BUS

CAB1 NPA- 713  WC- 006   CA-       11 PP- 0847
                                      COLR-
CAB2 NPA- 713  WC- 006   CA-     1108 PR- 0454
                                      COLR- BR/W
CAB3 NPA- 713  WC- 006   CA-     1108  PR- 0405
                                      COLR-
CAB4 NPA- 713  WC- 006   CA-     1108 PR- 0407
                                      COLR- BL/W
CAB5 NPA- 318  WC- 006   CA-       11 PR- 0907



CO-                  BSP-              TERM- BUSINESS
FEATURES-
NBR OF RINGERS-
DMLR  TN- 713 4921000            PG- 2  PRTR- W998   REQ BY- JFH


**LINE CONDITIONS**

  TAS- NO       DPA- NO            SEC- NO
  MAIN NBR- NO  HAS A MAIN- YES
DELETE LINE REQ- NO

**MLT INFORMATION**
  ASSOC TEST TN- -000000     DIFFERENT NNX- NO

DIRECTORY RECORD NBR- 0000217168     LRF RECORD NBR- 022348
```

Fig. 4—Miniline record.



```
DLR  DLRL EC 444 TN 713 4921ØØØ              SEC     DPA     PRTR T998 PAGE
  ORD C966441        CD Ø6-13-75 CUS 78Ø UNIT Ø2ØØØØØ1
  NSTA    3 PUB PUB  SP                                  EN
  MAIN 713 4921ØØØ KS    Ø HTG1 4921ØØØ RT 4354A TSOP     Ø
  OE  Ø1Ø- Ø5- 81 EXK 713 497 CS BUS   SC 1FH   VT 1288 PTY
  SWC          - Ø-   SSN ? TAS N TAC 713/4921111/ 1/ 4
  LCO Ø5-29-81 LCT 122754Ø1Ø
  LN  JOHN DOE COMPANY
  SA  101 ANYPLACE
  RMK ØØ1Ø ACCESS AT BACK
  S&E QTY    1 USOC 1FB   KS    Ø LTD    REF
  S&E QTY    1 USOC 1EC   KS    Ø LTD    REF
  S&E QTY    1 USOC 1EF   KS    Ø LTD    REF
  SNR /PU 1ØØØ,Ø1
  CAB TP F1   CA 11         PR  874 NPA 713 WC ØØ6 PRU     TPR         PRS
                                               CO
  CAB TP F2   CA 11Ø3       PR  454 NPA 713 WC ØØ6 PRU     TPR         PRS
                                               CO BR/W
  CAB TP F3   CA 11Ø3       PR  4Ø5 NPA 713 WC ØØ6 PRU     TPR         PRS
                                               CO
  CAB TP F4   CA 11Ø3       PR  4Ø7 NPA 713 WC ØØ6 PRU     TPR         PRS
                                               CO BL/W
  CAB
*DISPLAY CONTINUED ON NEXT SCREEN

--------------------------------------------------------------------------------

DLR  DLRL EC 444 TN 713 4921ØØØ              SEC     DPA     PRTR T998 PAGE
  CAB TP F5   CA 11Ø3       PR  4Ø8 NPA 713 WC ØØ6 PRU     TPR         PRS
                                               CO
  CAB TP F6   CA 11         PR  9Ø7 NPA 713 WC ØØ6 PRU     TPR         PRS
                                               CO
  HTG ØØ1Ø 4921ØØØ,4921ØØ1,
  LCO



*END OF DATA
```

Fig. 5—Full line record.

if the customer has party service, and a complete list of other circuits participating in a sequential hunt group (HTG). The example indicates that the customer is a telephone answering customer (TAC) and is connected to a telephone answering service (TAS) board with telephone number 7134921111 at position 1 on jack 4. While the miniline record has a limit of 55 characters for name, address, and location, the host line record can contain up to 823 characters. Finally, the miniline record has a limit of five cable pairs, but the host line record contains the complete cable pair list and any accompanying cable narrative. Considering the total "per line average" storage requirements per customer, the front-end storage space per customer is about 20 percent of the host storage space per customer. Assuming a five-million-line system using five front ends, the storage space per front end is met by two DEC RP06 disc storage units (1200 cylinders). The host storage requirements are met by approximately 30 IBM 3350 direct access storage devices (DASDs).

III. BUILDING THE OPERATIONAL DATA BASES

This section describes the LMOS host procedures to

(*i*) initially load the data bases when LMOS is first installed, and
(*ii*) merge additional data into the data bases when repair entities are added.

3.1 Initial load

In preparation for data base loading, BOC personnel must assemble the paper records and other data sources currently used in the repair operations and convert these data to machine readable form. Accuracy checks on the data prior to loading LMOS are not required since LMOS provides internal consistency checks and data correction capabilities based on conflicts and errors observed by the field craft.

A total of nineteen jobs must be run to complete the initial host data base load. Figure 6 summarizes the flow of these nineteen steps and partitions the steps into four major loading functions:

(*i*) Build skeleton history and message data bases
(*ii*) Initialize data base to accept only that data falling in specified ranges.
(*iii*) Perform validation checks on line record data and partition data by principal data bases
(*iv*) Load the principal data bases (line record data base and inverted data bases).

After the principal data bases are loaded, audits of data validity and consistency must be performed. Audit programs fall into two categories. The "self check" class of audit programs scan the line card, central office equipment, and cable files independently, looking for load errors, such as two or more telephone numbers connected to the same central office equipment terminal. The "cross-check" class of audits compares common data in the inverted files (AN, COE, TAS, and CA) to the line record files (POTS and SS). These data must be consistent; therefore, if a discrepancy is found, the line record file is assumed to be accurate and the cross-audit program automatically updates the inverted file to agree.

3.2 Merge load

The LMOS data base is typically loaded in phases (by RSB) until the entire locality served by the host is populated. For example, customer data for additional switching machine entities or RSBs can be added to the data base by performing steps 7 through 19 (using merge options) of Fig. 6.

IV. KEEPING THE DATA BASE CURRENT

4.1 Data input sources

For our purposes, the telephone network can be divided into two

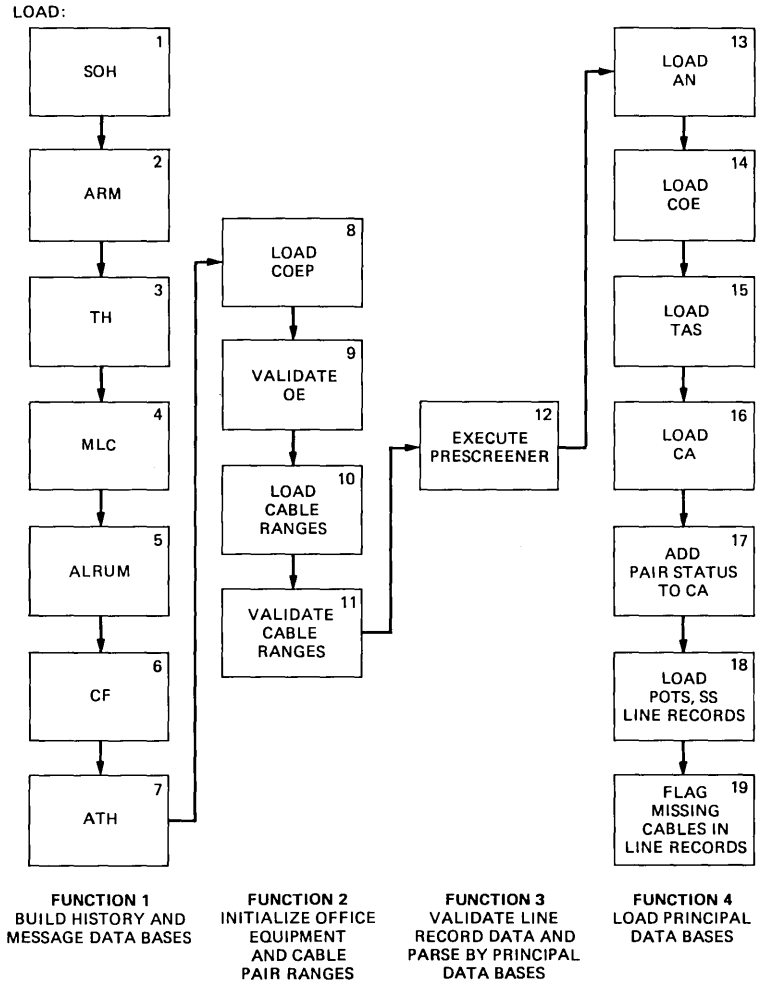


Fig. 6—Steps required for initial LMOS host data base load.

major pieces: the loop portion (i.e., from the end office to customer premises) and the toll portion (the remaining network that interconnects that national long-distance facilities). While the toll portion of the network is relatively stable, the loop portion undergoes constant change because it is the customer interface with the total network.

Since the LMOS data base is customer (and, thus, loop) oriented, these changes must be tracked. Activities that generate data base changes fall into two basic categories: (*i*) *customer* initiated service requests, and (*ii*) *Bell operating company* initiated plant changes. These categories are described below:

4.1.1 Customer initiated service requests

These requests are typified by a customer calling the local Residence Service Center/Business Service Center (RSC/BSC), or stopping by a phone center store, to request telephone service for a newly completed home or business. Other types of requests include rearrangement or addition of station equipment for an existing service, household moves, and changes to class of service.

The fundamental loop network record of these requests, and subsequent changes made to the customer's service and facilities, is the Universal Service Order (USO). Figure 7 provides an example of a completed service order (i.e., all work to implement the customer's request has been completed) for a simple POTS service.

The service order *header* is the only "fielded" portion of the order and contains record identification information, such as the telephone number for the account and service order number.

The *listing section* contains customer's name and location information; the *billing section* is of minor interest to LMOS; the *service and equipment section* identifies service features, quantity, and circuit arrangements for station equipment; and the *assignment section* identifies the central office and outside plant facilities.

Standards for the machine-readable USO are documented by AT&T.

4.1.2 Bell operating company initiated plant changes

The BOC's engineering and construction forces are charged with having adequate facilities in place at the right locations to meet customer service requirements. The requests that stimulate additions and rearrangements to loop plant are called work orders (sometimes called job orders).

Examples of work orders include the following:

(*i*) *Cable throw*—A new cable may be installed to augment an existing cable feeding a high growth area. To achieve desirable cable pair utilization levels (fill level), a range of cable pairs in the old cable can be freed up and reassigned to the new cable. This involves a change to the customer's cable and pair number.

(*ii*) *Area transfer*—It is occasionally necessary to do wire center load balancing for growth. One method is to reassign customers in a geographic serving area from one wire center to another, or to a newly installed central office switch. This usually involves customer feeder cable changes and frequently involves change of customer telephone number.

The above are only two examples of a variety of work orders. The record format and content of completed work order forms depend on the type of order worked.

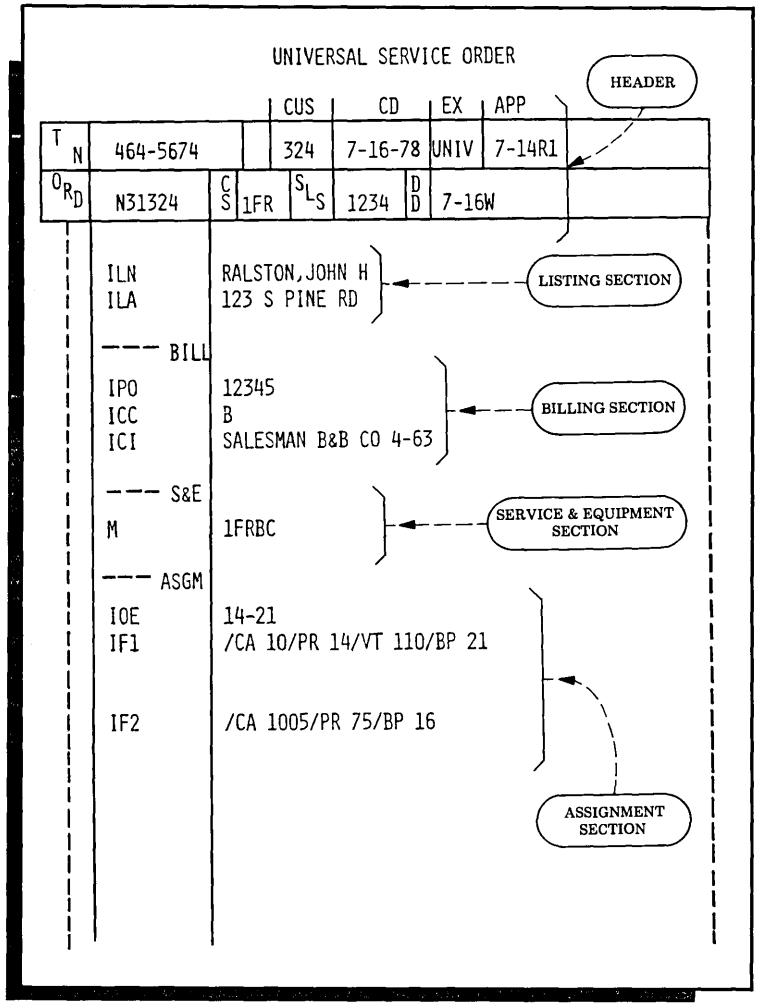


Fig. 7—Example service order.

4.2 Service order processing

Because service orders are written in USO language, they can be processed by machine. Each BOC has a mechanized service order network that produces a daily tape of completed service orders for updating the LMOS host data bases.

Before LMOS programs can read these service orders to update the data bases, the orders must pass through a BOC written interface program to add RSB identifiers (repair unit numbers) needed by LMOS and to translate BOC unique data to the standard USO format.

The LMOS programs that update the data bases from service order

input form the ALRU system. The ALRU comprises two program functions, the service order reader and the packet processor. The service order reader parses service orders, extracts data of interest to LMOS, and produces "packets," which are groups of data that correspond to the LMOS data base structure. Among the packets produced from the service order in Fig. 7 would be a packet to

(*i*) create a new line record,

(*ii*) install the listed name and address,

(*iii*) install the service and equipment data on the line record,

(*iv*) install the repair route on the line record,

(*v*) install the office equipment on the line record and update the COE data base,

(*vi*) install the cable data on the line record and update the CA data base,

(*vii*) install COE remarks on the line record, and

(*viii*) install assignment remarks on the line record.

The packet processor reads the packets and updates the data base. The number of packets produced for each service order will vary, depending on service order type and complexity. Based on field experience, a typical LMOS installation may process 20,000 orders, or about 200,000 packets a day.

4.3 Work order processing

Unlike the service order, the work orders today are not written in a uniformly structured language. Hence, the "load" generic and the "on-line" generic are used to input work order data to the LMOS host (see Fig. 2). If the work order involves a bulk change of data, such as throwing 400 pairs from cable 102 to cable 109, the bulk cable throw batch program of the load generic accomplishes this very efficiently. If, on the other hand, only a few pairs are to be "thrown," say five pairs, the enter cable change (ECC) transaction of the on-line generic would be the best choice to use. Changes made to the host data base by using batch programs of the load generic, or by using the various transactions available in the on-line generic, are propagated to the front-end data bases in the same way that service order changes are.

4.4 Data base update summary

Figure 8 provides an encapsulated view of the LMOS data base update processes described in this paper, and how those processes interface with the BOC's service order and work order flow.

First a summary of the service order flow as shown by the solid lines in Fig. 8:

1. The customer requests new or changed telephone service.

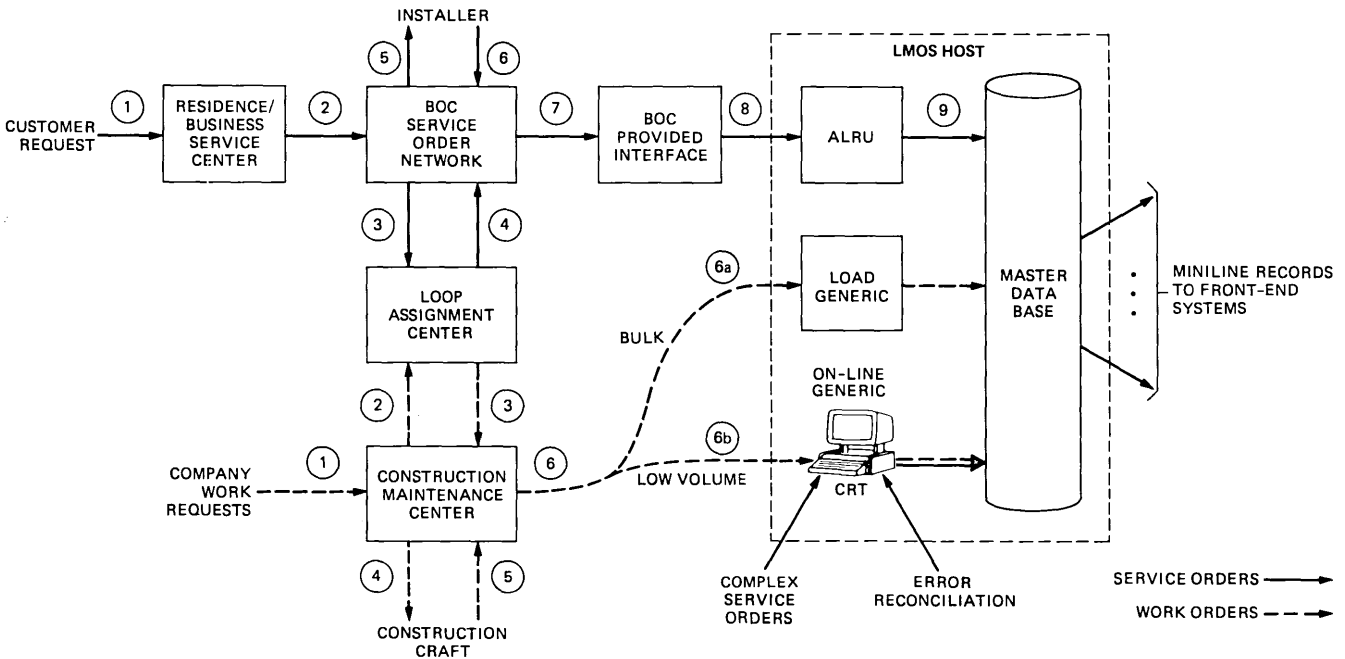


Fig. 8—Flow paths for LMOS data base update.

2. The request is entered into the BOC's service order network to be "worked."

3. A request is made to assign facilities necessary to install or modify the customer service.

4. Facilities are assigned and information is forwarded to the service order network.

5. Service order network forwards information to do work to the installer.

6. Installer completes work, returns notice to service order distribution network that service order has been completed.

7. Completed service order goes to BOC interface program to perform selected data translations for "standard" ALRU input.

8. A day's worth of service orders are accumulated and read into ALRU.

9. Automatic Line Record Update automatically updates the host data base.

On the average, there is one service order processed per line per year. A typical 5-million-line LMOS installation will process about 20,000 service orders per working day.

A summary of the work order flow as shown by the dashed lines of Fig. 8 follows.

1. The Distribution Services Design Center forwards requests for loop facility additions or rearrangements to the Construction Maintenance Center to be worked. In addition, other work centers may request work to be performed. For example, the repair forces may request work to be performed to repair a trouble (maintenance change request).

2. If the request for work involves existing facilities, facility assignment information is requested.

3. Facilities assigned to the work order are forwarded to the Construction Maintenance Center.

4. The construction craft receives the complete work instructions.

5. Work is completed and notices sent to the Construction Maintenance Center.

6. A paper record of the completed work order is distributed to LMOS. Either the load generic or the on-line generic is used to input the data, depending on the magnitude and type of change.

Field experience indicates that, on the average, work order activity accounts for about one-fourth the data base change activity incurred by service orders (in terms of customer lines affected per year).

When service order and work order activities are combined, it is estimated that 20 megabytes of data in the typical LMOS host data base is modified in some way every working day.

V. SUMMARY

The data base system of LMOS has evolved over several years. Among the lessons learned are that it pays to install a working prototype system in more than one BOC and to plan to make revisions based on field experience. An interface with the BOC service order process is a critical interface, and a mechanized work order interface would have been beneficial. In addition, the fixed length data base fields and storage attributes (e.g., packed binary) for selected data items have resulted in a more rigid data base design than we now find desirable. It is suggested that future data base systems be designed with the view that data items can change in format, length, and attributes.

VI. ACKNOWLEDGMENTS

The data base system architecture of LMOS has evolved from close interaction with personnel in the New York Telephone Company, South Central Bell Telephone Company, and Southwestern Bell Telephone Company during the years 1972 through 1975. We gratefully acknowledge the efforts of R. L. Martin, who led the design of the overall LMOS system, and of E. H. Mays, who was responsible for the detailed design work that led to the current IMS host data base structure.

REFERENCES

1. R. L. Martin, "Automated Repair Service Bureau: The System Architecture," B.S.T.J., this issue.
2. M. W. Bowker et al., "Automated Repair Service Bureau: Evolution," B.S.T.J., this issue.
3. O. B. Dale, T. W. Robinson, and E. J. Theriot, "Automated Repair Service Bureau: Mechanized Loop Testing Design," B.S.T.J., this issue.
4. P. S. Boggs and J. R. Mashey, "Automated Repair Service Bureau: Cable Repair Administrative System," B.S.T.J., this issue.
5. L. S. Dickert and S. P. Rhodes, "Automated Repair Service Bureau: The Trouble Report Evaluation and Analysis Tool," B.S.T.J., this issue.
6. S. G. Chappell, F. H. Henig, and D. S. Watson, "Automated Repair Service Bureau: The Front-End System," B.S.T.J., this issue.
7. J. P. Holtman, "Automated Repair Service Bureau: The Cross Front End: Context-Sensitive Switch," B.S.T.J., this issue.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

The Trouble Report Evaluation and Analysis Tool

By S. P. RHODES and L. S. DICKERT

(Manuscript received June 2, 1981)

The Trouble Report Evaluation and Analysis Tool (TREAT) is a system that provides Repair Service Bureau (RSB) personnel with an effective analysis tool for trouble reports that have been repaired (closed). TREAT consists of a set of computer-produced reports that can be tailored to the user's needs through the use of a simple report generator language. The user is supplied with approximately 50 reports written in the report generator language and can build new ones or change copies of the standard ones. With the aid of a large collection of written documentation and the methodology supplied with the software, users can investigate certain problem areas and determine possible solutions.

I. INTRODUCTION

The Trouble Report Evaluation and Analysis Tool (TREAT) is a system that provides Repair Service Bureau (RSB) personnel with an effective analysis tool for closed trouble reports. TREAT consists of a series of computer-generated reports. The user is able to obtain reports either in a standard format or in a high-level report language. The user's guide (*Bell System Practices*) is extensive and contains both report documentation and methodology. Through proper use of the guide and the reports, certain problem areas may be detected and corrected.

II. OVERVIEW OF TREAT FEATURES

The objectives of this paper are to provide the reader with an overview of TREAT's features and a description of its development. The

remaining sections of this paper describe analyses of TREAT's output, give more details on its usage, and cover its developmental history and future use.

With the use of TREAT, some of the primary areas of investigation are the following:

(*i*) Common equipment troubles, e.g., disproportionally large number of trouble reports on one switching machine.

(*ii*) Not-found troubles (trouble reports where the line either tested okay or was found in the field to be okay), e.g., transient transmission noise problems.

(*iii*) Customer service problems, e.g., an abnormally large rate of missed appointments.

(*iv*) Repair personnel productivity and performance, e.g., exceptionally long average repair time for outside craft.

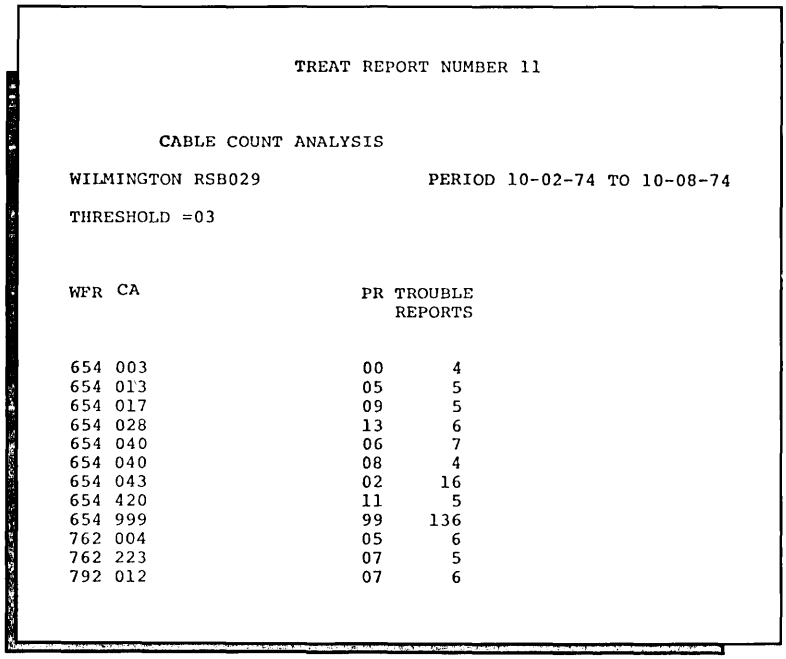
The TREAT reports are generated from a data base of closed trouble reports that are submitted to TREAT daily from the Loop Maintenance Operations System (LMOS) or from a Bell Operating Company (BOC) manual trouble report collection system. Typical RSBs process 100 to 500 trouble reports a day.

The TREAT analysis strategy is to use automatically generated reports to detect a problem area, then isolate the source of the problem through use of the requestable reports. These reports may be either prestructured or user defined. For this to be effective, the requestable reports must be easily and quickly obtainable. This was one of the basic design goals of TREAT.

The automatic reports are set up for each RSB and sent at a predetermined frequency—usually daily, weekly, or monthly. Most daily reports employ a threshold mechanism whereby certain lines are printed only if the value exceeds the preset threshold. These reports are transmitted to the RSB by using LMOS printers or teletypewriter terminals.

The requestable reports are sent to the RSB or to a staff analyst only when requested. There are about 50 standard reports that may be requested by entering the report number and a few other parameters. Others may be constructed entirely by the requester through a high-level report generation language. The turnaround time to obtain the output can vary from a few minutes to a few hours, depending on performance options specified by the local companies.

The TREAT reports are simple in format, being either the tabular or list type. The tabular type provides a matrix of counts that satisfies the selected criteria (Fig. 1). The list type provides selected fields from the trouble reports that satisfy the selected criteria (Fig. 2). The tabular type aids in global problem identification, whereas the list type permits investigation on a more detailed level.



TREAT REPORT NUMBER 11

CABLE COUNT ANALYSIS

WILMINGTON RSB029 PERIOD 10-02-74 TO 10-08-74

THRESHOLD =03

WFR	CA	PR	TROUBLE REPORTS
654	003	00	4
654	013	05	5
654	017	09	5
654	028	13	6
654	040	06	7
654	040	08	4
654	043	02	16
654	420	11	5
654	999	99	136
762	004	05	6
762	223	07	5
792	012	07	6

Fig. 1—Cable count analysis—tabular version.

TREAT REPORT NUMBER 12

CABLE COUNT ANALYSIS

LOCUST RSB002 PERIOD 09-10-74 TO 10-08-74

WFR	CA	PRNO	TN	DATE RECD	TIME RECD	T	DISP
110	234	02	215-5617364	100474	1632	3	0700
110	234	11	215-5615879	100274	2200	2	0900
110	243	11	215-5687731	100374	1540	1	0700
110	243	11	215-6658989	093074	1626	4	0700
110	248	12	215-5612520	100374	1540	1	0700
110	248	12	215-5688059	100973	1003	3	0700

Fig. 2—Cable count analysis—list version.

III. ANALYSIS

The bureau's trouble reports provide information useful in detecting areas for improvement. Some of the data items in the trouble report stored for analysis by TREAT are as follows:

(*i*) Telephone number
(*ii*) Data and time reported, tested, dispatched, and cleared
(*iii*) Category of report (customer direct, customer relayed, or employee report, etc.)
(*iv*) Class of service (business, residence, PBX, coin, etc.)
(*v*) Type of report—what the customer told the repair service attendant, e.g., no dial tone
(*vi*) Disposition—what was done to fix the trouble, e.g., repaired station set
(*vii*) Cause—what caused the trouble, e.g., weather
(*viii*) Central office line equipment
(*ix*) Cable and pair
(*x*) Repair personnel, e.g., RSB tester or outside craft
(*xi*) Elapsed time for each step in the repair process
(*xii*) Miscellaneous other items.

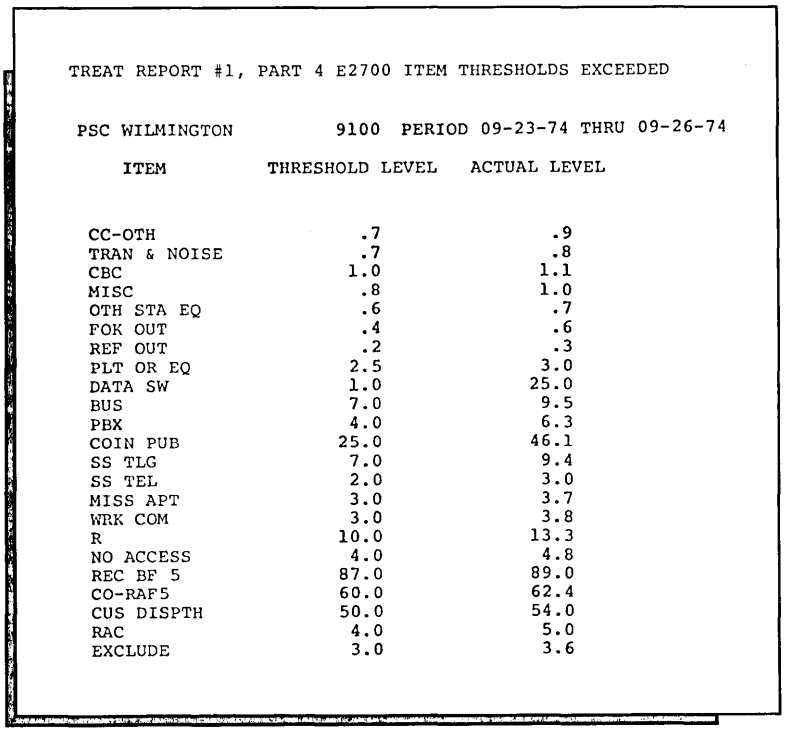
This ability to specify trouble report selection criteria allows the requester to narrow the scope of the problem and to reduce the amount of printout obtained. This is a crucial feature because the RSB users may not have high-speed printers.

As an example of TREAT usage, one of the morning reports (Fig. 3) summarizes the trouble report activity of the previous day and accumulates the activity for the report month. The report month runs from the 23rd of one calendar month to the 22nd of the next. In the example provided, the threshold for item R is exceeded. This indicates that repeated reports (i.e., troubles where the customer calls back within 30 days) are still too high. TREAT report number 17 (Fig. 4) shows, by tester, how many tested reports later had another "repeated" report. This example shows that tester "1" had about 16 percent (92/593) repeated reports, higher than the user's objective of 10 percent. A detailed listing of all of these reports can be requested and reviewed for possible improvements in the testing procedure or additional tester training.

In addition to the analysis-oriented reports, TREAT provides a series of reports and interfaces for AT&T, public utility commissions, and other centrally developed systems.

IV. USAGE

TREAT executes on an IBM 370 compatible computer. The requestable reports are obtained through IMS requests that feed a batch reporting system (Fig. 5). The automatic reports come from a strictly batch system. The IBM 370 was chosen because it was the same machine supporting LMOS. This choice also made it possible to deploy TREAT in advance of LMOS since all BOCs had data centers with 370 capability.



TREAT REPORT #1, PART 4 E2700 ITEM THRESHOLDS EXCEEDED

PSC WILMINGTON 9100 PERIOD 09-23-74 THRU 09-26-74

ITEM	THRESHOLD LEVEL	ACTUAL LEVEL
CC-OTH	.7	.9
TRAN & NOISE	.7	.8
CBC	1.0	1.1
MISC	.8	1.0
OTH STA EQ	.6	.7
FOK OUT	.4	.6
REF OUT	.2	.3
PLT OR EQ	2.5	3.0
DATA SW	1.0	25.0
BUS	7.0	9.5
PBX	4.0	6.3
COIN PUB	25.0	46.1
SS TLG	7.0	9.4
SS TEL	2.0	3.0
MISS APT	3.0	3.7
WRK COM	3.0	3.8
R	10.0	13.3
NO ACCESS	4.0	4.8
REC BF 5	87.0	89.0
CO-RAF5	60.0	62.4
CUS DISPTH	50.0	54.0
RAC	4.0	5.0
EXCLUDE	3.0	3.6

Fig. 3—Morning report.

The TREAT data base consists of closed trouble reports for the most recent 40 days. The 40-day period was selected to provide a monthly data base, with enough extra days to be certain that end-of-the-month reports can be generated before first-of-the-month data are deleted. The trouble reports are entered into TREAT nightly from either LMOS or from a BOC-supplied collection system. This is a batch operation in which trouble reports are edited and either accepted or rejected, certain fields are automatically scored, and data are entered into the data base. The 41st day's data are removed with each update. The data are organized by RSB, with the oldest data at the end of the files.

All TREAT reports are generated either from this 40-day data base, known as the Cumulative Abbreviated Trouble (CAT) file or from summary files derived from the CAT file. Daily reports are generated after the update is made.

In addition to the CAT file, there are several auxiliary files used to define RSB hierarchy up to the company level (six levels), and other RSB parameters used to run TREAT. These files are updated rather infrequently as needed.

The heart of the report-generating capability of TREAT is the report

TREAT REPORT NUMBER 17

REPEATED REPORT ANALYSIS CRAFTSPERSON - TESTER OR VERIFIER

LOCUST RSB002 PERIOD 09-10-74 TO 10-08-74

TESTER OR VERIFIER	ORIG REPEATER	TOTAL TESTS
--	24	105
A	1	7
D	0	1
H	0	2
J	0	3
M	0	1
P	0	1
Q	1	12
R	3	13
O	14	71
1	92	593
5	45	345
8	33	342

Fig. 4—Repeated report analysis for the craft person.

compiler. This report compiler is used to generate most of the reports. The report compiler can generate both tabular- and list-type reports. The command language was designed to be easy to use at the expense of extremely complex reports. For example:

TITLE	Starts the report and gives a report title.
RSB(rsb#)	Selects the RSB for the report.
DAY(day1,day2)	Selects the span of days to be included in the report.
CONDITION(expr)	Gives the trouble report selection criteria. Simple Boolean expressions are allowed, but not arithmetic expressions.
SORT	Sorts by fields on the trouble report.
PRINT	Gives the fields to print on a list-type report,

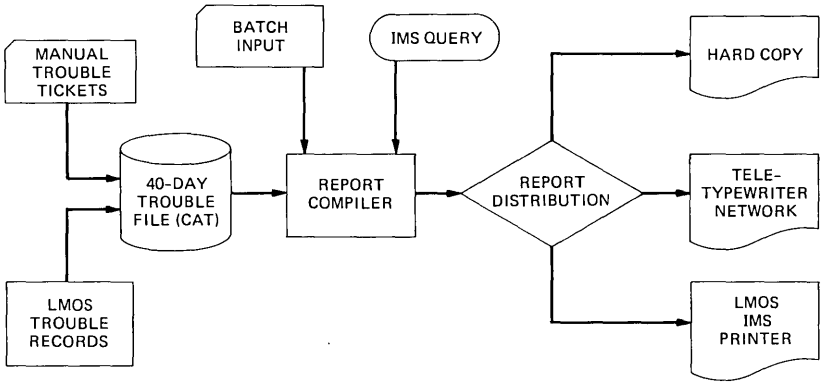


Fig. 5—TREAT report generation.

or the horizontal and vertical fields to use for a tabular report.

A sample list-type report is as follows:

```
TITLE (This is a sample TREAT report request)

RSB(020)

DAY(1,10)

CONDITION(CAT=1&CS=04)

SORT(RECD,TN)

PRINT(RECD,TN,TYPE,DISP,CAUSE)
```

Although there are currently about 25 commands, most reports can be obtained with six to eight commands. In actual practice, the commands used are fairly simple.

The automatic reports are run off-hours through the report compiler as batch programs. The output is usually placed on a file to be transmitted back to the user at the RSB rather than printed at the data center.

Other than an older Time Sharing Option (TSO) of TREAT, there are two report request modes:

(*i*) *ONLINE*—The report request is entered by requesting an Information Management System (IMS) report request mask at an LMOS on-line terminal. The mask is filled in with TREAT commands as

illustrated above and transmitted. The request is queued up and the actual report is prepared in a deferred batch mode, along with those from other requesters. Depending on the company's choice, this is done from a few times a day to several dozen times a day. The output is returned to the RSB on an LMOS on-line printer. Turnaround in this mode varies from 10–15 minutes to a few hours depending on the BOC's environment.

(*ii*) *BATCH*—The report requests are gathered manually, and the inputs are prepared in punched-card format. The reports are generated in a batch mode, with the printout usually being delivered to the requester by mail. Fortunately, this is not a mode that is used very much, but was useful prior to converting to LMOS.

The report compiler runs the same way in all modes, which makes for an easy user transition from batch to on-line because the reports are identical—only the method of requesting changes.

TREAT has about 50 standard reports that were designed to provide a BOC with a basic set of reports. About 40 of these reports are generated by the report compiler, and are delivered as source commands. An analysis plan is also furnished to the users as a guide to help them in determining the order of report analysis and to aid them in ascertaining the actual problem. This approach allows BOC personnel to make local changes to standard reports and also learn how to set up additional reports. Most BOCs have set up many additional standard reports. In the BOCs today, the majority of the reports are requested directly with the report compiler language rather than with the standard prestructured reports.

V. HISTORY

TREAT had its beginning in January, 1973, as the reports portion of the MLR (Mechanized Line Record system) trial at New York Telephone Company. It was designed to provide some of the standard required reports for the RSB. Fortunately, it was early recognized that several reports were similar in format and could be produced by a simple general-purpose report compiler. These reports were the list type, and the first report compiler only had about eight commands (it now has over 25). TREAT, in this MLR environment, was well-accepted by RSB personnel, who especially liked getting reports the next day. There were no requestable reports as such, but the number of "standard" automatic reports grew rapidly. This system eventually grew to support the 12 RSBs on the MLR system.

In 1974, AT&T began looking for a replacement for its Mechanized Customer Trouble Report Analysis Program (MCTRAP), which was over ten years old, and was becoming difficult to change. There were three choices:

(*i*) Build a new MCTRAP II by using specifications prepared by Bell Laboratories from a Chesapeake & Potomac Telephone Company study.

(*ii*) Expand the programs used for the MCTRAP II study to a working production system.

(*iii*) Separate TREAT (not called TREAT at the time but, rather, the MLR off-line system) from MLR, and offer it as a stand-alone system.

TREAT was chosen because it was the only one in a production mode, it already had a working report compiler, and it was supported by a Bell Laboratories development staff committed to LMOS/TREAT for years to come. The MLR department head, R. L. Martin, completed the decision-making process by coining the acronym "TREAT."

AT&T chose The Bell Telephone Company of Pennsylvania (PA) as the trial company, and a trial start date of June 23, 1974 was set. It was decided to offer requestable reports through TSO and to design a new set of standard reports. A task force met at PA and designed the first set of standard reports. Interestingly enough, these reports have remained virtually unchanged to this date. The major work needed in TREAT for the trial was to provide a manual trouble report interface, implement the 40 standard reports, add the tabular report feature to the report compiler, and provide a TSO report requesting interface.

TREAT, operating out of a Bell Laboratories data center, was put on-line for four trial RSBs in June, 1974. It was run for four months in this fashion, then it was moved to PA's data center. During this period, most work centered on cleaning up and changing the standard requestable reports. In addition, the TREAT User's Guide was developed by PA.

TREAT was then installed in the Houston area of Southwestern Bell Telephone Company (SW) to precede the LMOS installation. Although TREAT was to be run in batch mode only at SW the real test was that it would have to support 65 RSBs, a much larger number than before. This installation was completed in November 1974, and the worst fears were realized. Some of the TREAT update jobs had run times that appeared to increase with the square of the number of RSBs. Jobs that took minutes at PA and New York Telephone Company (NY) ran for almost two hours at SW. Otherwise the system worked well.

The first release of TREAT to Western Electric Company (Issue 1) was in February, 1975, with some of the performance problems corrected. Issue 1 was installed at South Central Bell Telephone Company (SO CN) and expanded to 135 RSBs (still the largest TREAT installation). South Central Bell Telephone Company personnel commented that "some part of TREAT was running 24 hours a day." Issue 2 was installed at SW (which had now become a trial company) in May, 1975, with still more performance improvements.

A redesign of TREAT was undertaken with an objective of significantly reducing the run times. One way this was accomplished was through a more optimized file structure. This was to be TREAT, Issue 3, nicknamed "speedy TREAT." "Speedy TREAT" was first installed at NY to replace the MLR version in December, 1976. First measurements were very encouraging. Tests were then run at SW in January, 1977, and the results were outstanding. "Speedy TREAT" met all expected performance objectives, and some that had not been planned. South Central Bell Telephone Company, which was limping along with TREAT Issue 2, quickly converted to "Speedy TREAT" in February, 1977, and promptly announced that they were so pleased with TREAT that they had no more requests for changes.

Deployment of TREAT then began in earnest, and companies cut at a rate of one per month through 1977 and 1978. At this writing, all BOCs are using TREAT and there are 32 data centers involved, making TREAT the most widely deployed centrally developed system.

As an indicator of the acceptance of TREAT, over 6000 TREAT User's Guides have been sold.

Finally, Issue 4 of TREAT was developed to change one of the major reports to reflect the business/residence/coin split of the Bell System, and to provide a few new features; New Jersey Bell Telephone Company tested it in the summer of 1978.

VI. TREAT'S FUTURE

As a part of the second major version of the Automated Repair Service Bureau (ARSB-2), the architecture and the implementation of TREAT were reexamined by Bell Laboratories. Two significant areas were explored.

The first area was the user and the data system oriented requirement—both current and future. A particular area of concern here was the degree to which Issue 4 served the needs of a vastly restructured corporation.

The second area involved the internal implementation. The software was getting old and had been patched frequently. The global system design, on the other hand, was still judged to be sound.

After the investigation of several alternatives, it was determined that a two-pronged approach was appropriate. The code is being redesigned and improved using newer software technologies. At the same time, we are incorporating several enhancements to the system to respond to the changing corporate requirements. As an example of these changes, all of the reports and the data bases in the next issue will be segmented. This will permit the user to retrieve all these reports collected by business, residence, public services, or network. This can

be done either on an individual bureau basis or on any level in the management hierarchy.

VII. ACKNOWLEDGMENTS

We would like to acknowledge all of the efforts by the original developers of TREAT, E. H. Moody, N. R. Petree, and G. G. Sutton; the subsequent developers, M. Baade, B. E. Cruse, S. D. Rhodes, and M. B. Wicker; and the more recent staff, L. LeBlanc, C. A. Daye, L. A. DiBella, L. Krantz, K. E. Lewis, K. A. Rosenberg, S. M. Tovar, and R. H. Witt.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

The Front-End System

By S. G. CHAPPELL, F. H. HENIG, and D. S. WATSON

(Manuscript received June 17, 1981)

The primary role of the Loop Maintenance Operations System front-end computer is to help the Repair Service Bureau personnel track and repair troubles reported on telephone services by our customers. Each customer trouble report is entered into the system and its status is updated at each step toward completing the repair. Management reports are generated that warn of overload conditions and potential degradation of repair service.

I. INTRODUCTION

The components of the Automated Repair Service Bureau (ARSB) described throughout this volume serve four major functions. These are:

(*i*) maintaining a customer line record data base so that repair personnel have up-to-date information about the facilities being repaired,

(*ii*) recording and tracking troubles reported on telephone equipment from the time the trouble is reported until the time it is cleared and closed out,

(*iii*) testing and analyzing the condition of customer loops, and

(*iv*) analyzing closed trouble report data to aid in managing the repair process.

The first and last of the above functions are handled by the Loop Maintenance Operations System (LMOS) host,[1,2,3] where the power of a large main frame computer and the availability of large amounts of disk storage can be used to advantage. The third function—automated loop testing—is performed by the Mechanized Loop Test (MLT) system.[4]

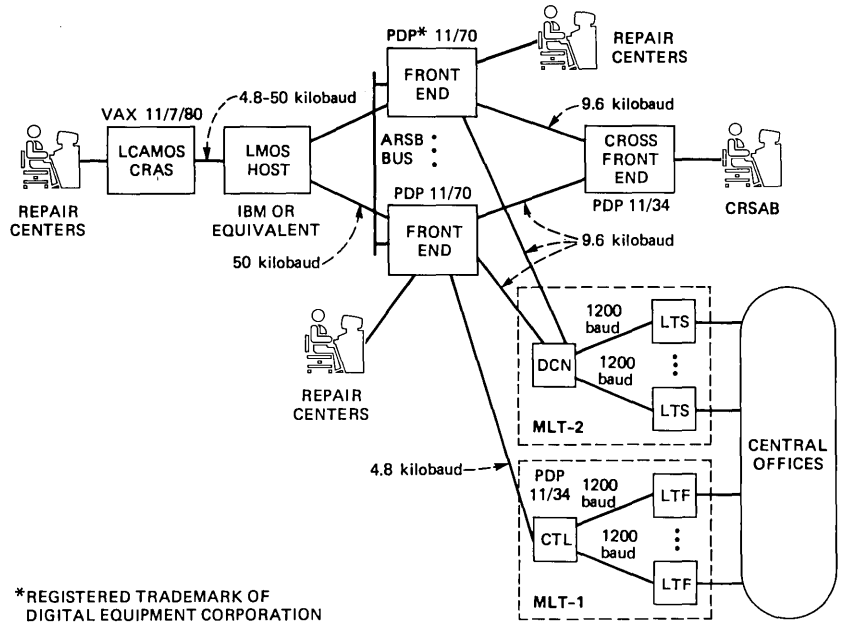


Fig. 1—Automated Repair Service Bureau—an example.

The second function is the topic of this paper and is the role of the LMOS front end (FE), a transaction-oriented tracking system designed to record troubles and maintain information on their status until the customer is satisfied that the problem has been corrected.

In addition, the LMOS FE is a communications handler. It serves as the primary user interface to the ARSB, providing access to both the host and MLT, in addition to the FE itself.

This paper describes the major capabilities of the FE, its role in the distributed ARSB, and its continuing evolution.

II. HARDWARE OVERVIEW

A typical LMOS system configuration is shown in Fig. 1. Figure 1 also depicts LMOS interfaces to other systems included in the ARSB and described elsewhere in this issue.[4,5] The LMOS system consists of a large IBM or IBM-compatible host (370 or 303X class) connected to as many as ten PDP* 11/70 LMOS FEs by 50-kilobaud data links. The interface between the two types of computers is well defined: the FE looks like a terminal controller to the host.

Access to FEs is provided via synchronous display terminals and

* Registered trademark of Digital Equipment Corporation.

printers (e.g., Teletype* 40/4 keyboard displays and printers). Each FE can support up to 512 such devices on up to forty-eight 4.8- or 9.6-kilobaud data links. These terminals and printers provide access to the system from the Repair Service Bureaus (RSBs), the Centralized Repair Service Answering Bureau (CRSAB), and various staff and data systems organizations. Typically, the terminals in the repair bureaus are connected directly to the FE, while those in the CRSAB, requiring access to multiple FEs are connected to a cross FE (XFE) which acts as a context switch to the FEs it serves.[6]

To achieve high availability, both the FE and the XFE systems are configured with backup systems. The FE systems are configured with a backup PDP 11/70 for every two FEs, and the Western Electric Company has developed a switch to allow the communications lines to be switched quickly from either FE system to the backup.

The FEs are also connected via data links to MLT. Up to 16 MLT controllers (DEC PDP 11/34s) can be handled by a single LMOS FE.

With LMOS-2 (the second generation of LMOS), a high-speed bus has been added to the system architecture. This 300-foot, 3.2-megabaud bus connects up to 12 FE systems (including backups) and provides the hardware base for the inter-FE communication described below.

III. THE FRONT-END TRANSACTIONS

Although the FE software includes some 50-odd transactions, the workhorses of the system are the four trouble processing transactions and the management report transactions. They comprise approximately 85 percent of the user transactions entered into the FE.

A typical trouble processing sequence is shown in Fig. 2, with the masks simplified somewhat for illustrative purposes. The full repair process is described in more detail in Ref. 7.

The Trouble Entry (TE) transaction is normally entered by a Repair Service Attendant (RSA) in the CRSAB when the customer calls to report a trouble. The attendant enters the customer's telephone number and the transaction returns a Trouble Report (TR) mask partially filled in with information from the customer's line record, information on any outstanding troubles associated with the telephone number, and the time when the repair bureau is able to have the trouble fixed. The TE transaction also initiates an MLT test on the customer's line.

When the partially filled-in TR mask is displayed, the attendant enters the trouble description provided by the customer. The attendant then negotiates a time with the customer (called the commitment time) when the trouble will be repaired and enters this information on

* Registered trademark of Teletype Corporation.

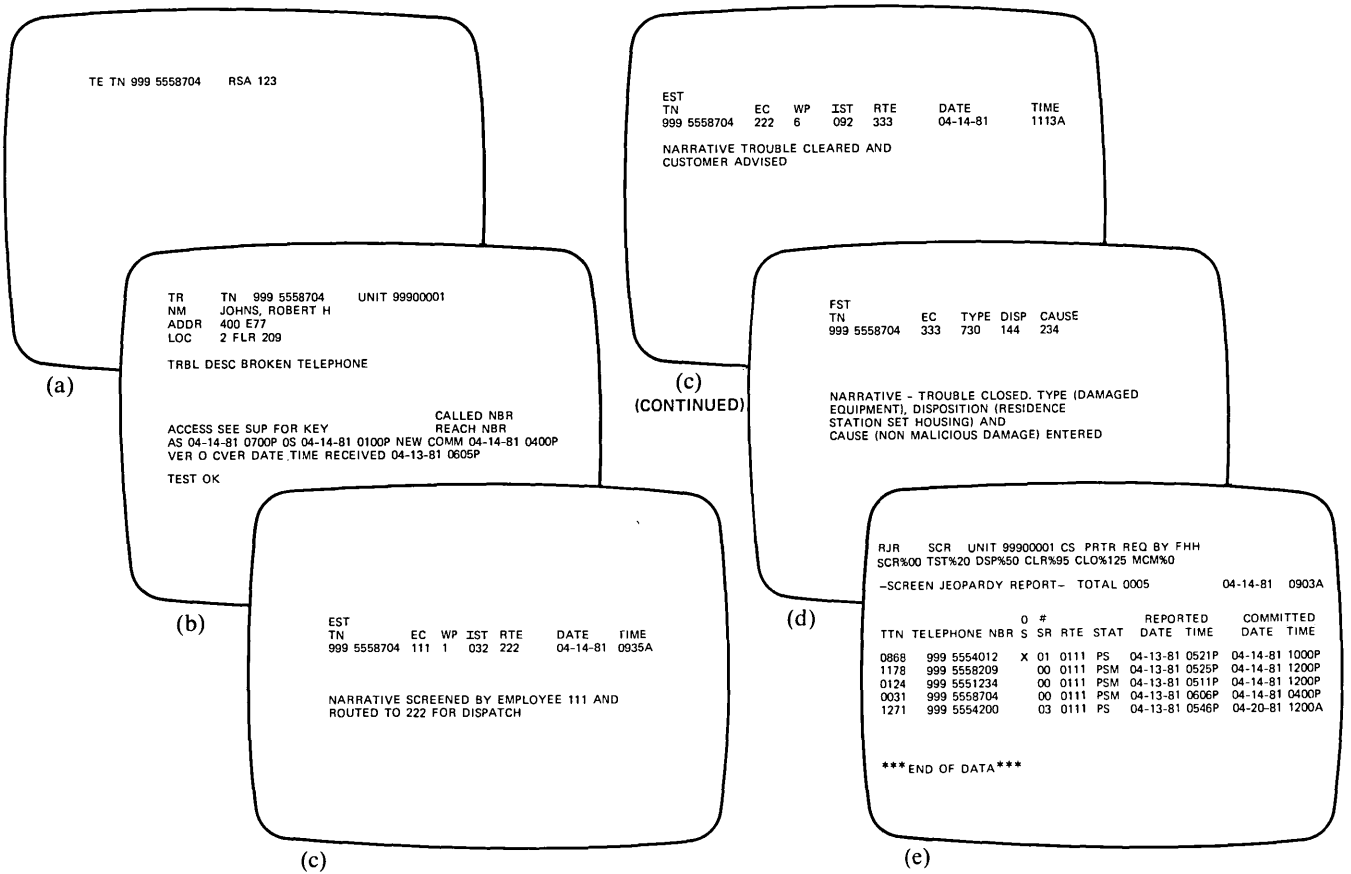


Fig. 2—Typical trouble processing sequence. (a) Completed trouble entry (TE) mask. (b) Completed trouble report (TR) mask. (c) Completed enter status (EST) masks. (d) Completed final status (FST) mask. (e) Completed jeopardy mask.

the mask. The TR transaction records the trouble in the FE trouble data base and generates a Basic Output Report (BOR). This output report contains the trouble description, MLT test results, and information from the customer's line record. The report is printed at the repair bureau assigned to fix the problem.

At the repair bureau, repair personnel screen the trouble, re-test it under special circumstances, dispatch someone to fix it, and once it has been cleared, notify the customer and close it. As each of these steps is taken, repair personnel enter current status information into the trouble data base using the Enter Status (EST) transaction. The status information includes the work performed on the trouble, who performed the work, and to whom the trouble is being routed next.

When the trouble is cleared and the customer advised, a final status is entered using the Final Status (FST) transaction. The FST records information on the cause and disposition of the trouble, and marks the closed trouble ready for transfer to the LMOS host (for later analysis[3]) and for deletion from the FE open trouble data base.

This status information allows the repair bureau management to track the trouble as it is being repaired. The Request Jeopardy Report (RJR) transaction prints out all troubles for which the bureau is in danger of missing the commitment time negotiated with the customer.

These five FE transactions—TE, TR, EST, FST, RJR—comprise the basic trouble processing sequence. Other FE transactions perform additional functions, allowing management and bureau personnel to get various reports on the number and status of outstanding troubles, to enter company and bureau-related information (e.g., the hours each bureau is open) and to administer the FE system.

IV. FRONT-END COMMUNICATIONS

In addition to its trouble tracking functions, the FE serves as a communications handler for the entire ARSB system (Fig. 1). Front-end communications software handles the interfaces to the synchronous terminals and printers on the FE and the links between the FE and the host, and the FE and the MLT.

From the users' standpoint, three types of access are provided:

(*i*) Terminals in the CRSAB and the RSB access the FE itself, either directly or through a XFE. These terminals are used to enter and track troubles, as described above.

(*ii*) Terminals connected to the FE also have access to the MLT test systems (PDP 11/34's) connected to the FE. Front-end applications and communications software enables repair bureau personnel to use FE transactions to initiate tests on a loop or a series of loops and to display or print the results.

The FE software also includes transactions to administer the MLT

system and a download facility so that MLT controller software can be downloaded to the 11/34s from the FE.

(iii) Terminals attached to the FE have a switch-through interface to the LMOS host. Transactions not recognized by the communications software on the FE as FE transactions are automatically passed on to the LMOS host where they are treated as normal Information Management System (IMS) transactions. The terminal output from these transactions is routed back through the FE to the user's terminal. This interface allows a single LMOS terminal to access both the FE and host systems. Terminals connected directly to the host are normally provided for data base update groups, since these groups require access only to the host.

V. DESIGN CONSIDERATIONS

The principal design requirements for the FE system are high availability, data integrity, and performance.

5.1 Availability

The requirement for high availability stems from the critical role of the FE; it serves the trouble taking and tracking function central to repair bureau operations and is the gateway to the other ARSB components. This availability is provided by using backup hardware and by reducing the interdependency of the various system components. There is one spare FE for every two active FEs, plus the associated hardware necessary to manually switch from the active to the spare FE. This process normally takes 5 to 10 minutes during which time the FE is unavailable. The rapid recovery time was very important to our early success since it allowed us to crash relatively often without enraging our terminal users. We found several short outages to be much more acceptable to our users than one long outage. Measured availability in the field is normally over 99.5 percent.

The system is designed (see below) so that the repair bureaus and centralized answering bureaus can continue to operate efficiently even though the host is down. Thus, the host is not duplicated.

5.2 Data integrity

The requirement for data integrity is an operational requirement: customer troubles must not be lost, either while they are open or after they have been closed out but not yet moved to the host. This means that data base integrity must be provided when the system crashes. When the crash does not involve damage to the physical disk, this data integrity is provided by FE software that backs out any partially completed transaction. When a data base is damaged (e.g., from a head

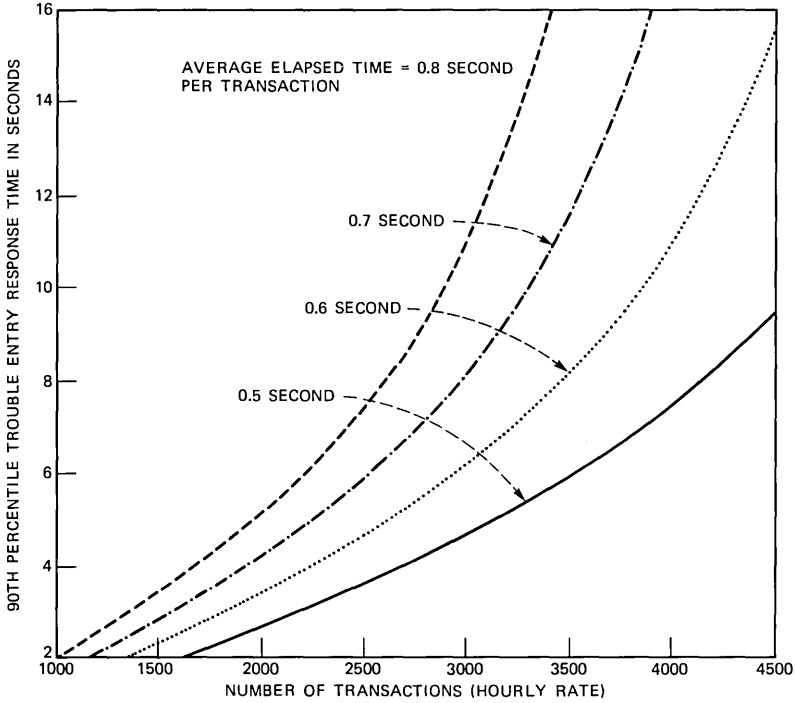


Fig. 3—Loop maintenance operations system FE average transaction response times.

crash), recovery is provided by using journal log tapes to update the latest (typically the previous night's) backup copy of the data base.

Data integrity must also be preserved when data are being transferred between the host and the FE. Data integrity is achieved here using standard acknowledgment techniques. The sending system looks for acknowledgment that its transmission has been received and correctly processed. If it does not receive that acknowledgment, the transmission is re-initiated. In addition, the receiving system recognizes duplicate transmissions and takes appropriate action to protect against system failures during final acknowledgment processing.

5.3 Performance

The curves in Fig. 3 depict FE performance as a function of the number of transactions handled per hour and the average elapsed time per transaction. The elapsed time for a transaction is the time the transaction spent in the FE from the time the transaction task is initiated until the time it is terminated. The "average elapsed time per transaction" is a measure of the average service time per transaction, and varies primarily with the mix of transactions running on the

system (some transactions consume significantly more resources than others) and the amount of background work running.

The transaction mix varies from FE to FE because of local operational practices, the nature of the customer base (e.g., whether it is primarily business or residential) and the availability of other ARSB components (e.g., MLT). Transaction mixes will also vary on a single FE in the course of a day, normally with a higher percentage of TE and TR transactions appearing in the morning and a higher percentage of statusing transactions appearing later in the day as troubles are cleared and closed out. System capacity is determined on the basis of the transaction mix at the peak trouble reporting hour.

Front-end system performance in Fig. 3 is shown in terms of the 90th percentile response to the TE transaction since that is the most critical response time measurement. After initiating a TE transaction, the attendant in the answering bureau must wait for a TR transaction mask to be returned—with the customer on the line—before taking information on the reported trouble or negotiating a commitment time. The actual response time the attendant sees is the front end TE response time, plus approximately two seconds for XFE and transmission time.

The load on an FE can vary greatly as a function of the day of the week (Monday morning is traditionally the busiest trouble reporting period) and weather conditions. Most companies size their systems for a "normal busy hour," that is, the load they expect to see on a rainy Monday morning. As the system becomes more heavily loaded, the operating companies use system tuning parameters and administrative procedures to maintain response times to the answering bureau attendants at the cost of less critical functions. Catastrophic conditions (e.g., hurricanes) can, however, throw the system into an overload condition. In this case, a portion of the troubles will be taken manually (i.e., written down on paper) for later entry into the system.

VI. THE DISTRIBUTED ARCHITECTURE

The interface between the LMOS host and FE is a classic example of a technically conservative distributed system. The distribution of line record and trouble data on the FE and the host illustrate the design principles applied in the system:

(*i*) Where possible, data and functions are partitioned, so that they are required on only one system.

(*iv*) Where partitioning is not feasible, duplication of the data or function is provided to de-couple the system components and optimize performance and availability.

There are three major functional links between the host and the FE: processing closed customer troubles, updating customer line record, and generating the BOR for the repair bureau.

6.1 Processing closed troubles

The trouble data base on the FE is completely partitioned from the trouble information contained on the host. The FE knows only about open troubles; it has no knowledge of a trouble once it has been closed. The host has no knowledge of open troubles, but maintains a trouble history data base containing 40 days of closed trouble information.

The trouble processing functions are similarly partitioned. The FE transactions deal only with taking and tracking open troubles; the host transactions and the Trouble Report Evaluation Analysis Tool (TREAT) display and analyze historical trouble data.

Troubles marked for closed trouble processing by the Final Status (FST) transactions are batched for transmission to the host. The sending system (in this case the FE) waits for acknowledgment that the batch of troubles has been successfully received and processed by the host. If such acknowledgment is not received, the FE will reinitiate the transmission. This acknowledgment process is required because it is critical that closed troubles are transmitted to the host for measurement and reporting purposes. The transmission process is expensive in terms of system resources, and system administration facilities are provided so that these functions can be turned off during prime shift busy hours.

6.2 Customer line record updates

In contrast, the customer line record information on the FE is a duplicated subset of the line record information on the host. The data is duplicated to minimize the interdependence of the FE and the host for both performance and availability reasons. The data on the FE "miniline record" data base is approximately 10 percent of the data stored in the host's line record. It is the subset that is critical to the repair process. (One of the more predictable evolutionary phenomena of the system has been an increase in the data which is seen as "critical.")

Updating the line record data bases is done using a strict master-slave relationship, with the LMOS host being the master. The process begins when the host data base is updated by the Automatic Line Record Update[2] programs processing service orders, or by data base personnel issuing host transactions to change the line record.

Nightly, host line records that have changed are batched for transmission to the FE. The FE initiates the "change miniline record" process in which the host transmits a group of new line records, awaits acknowledgment from the FE, then sends the next group, until all the updates are complete. The FE line records can only be updated as a result of the host's updates or from off-line bulk load programs. Theoretically, this should keep the data bases synchronized once all

the host change requests have been processed, although the two data bases may be out of step for a considerable period of time until the FE "catches up." However, the data bases do get out of synchronization and cross-audit programs are provided to detect discrepancies and reload the FE "miniline record." More importantly, none of the software of the host or the FE depends on the two data bases being synchronized.

6.3 The output report

Output report processing is an example of an area where a *function* has been duplicated on the host and the FE to avoid system interdependence. Normally, when a trouble is entered into the system, the FE sends the trouble description and the test results to the host. The host takes this information, adds the full line record information and an abbreviated history of the troubles taken against this telephone number over the last 40 days, and formats the BOR, which it sends back to the FE to be printed at a repair bureau.

If the host is unavailable, an output report must still be sent to the bureau, since this piece of paper informs the bureau that they have a trouble to be worked. This output report is a "Mini Output Report" (MOR), generated by the LMOS FE. The MOR has only a subset of the line record information and it does not have any trouble history. It does, however, have most of the information critical to repairing the trouble.

VII. SOFTWARE EVOLUTION

The original LMOS FE software used an operating system called Bell Operating System (BOS) which was developed for, and tailored to, the LMOS application. The system was written in assembler code and Digital Equipment Corporation's MACRO-11* language and included a file system with logging and recovery procedures, plus a sophisticated communications handler which allowed us to support synchronous terminals and interfaces to the host, the XFE, and the MLT.

In 1978, a decision was made to redesign and reimplement the FE software. This new system (LMOS-2), which is currently being tested at Michigan Bell Telephone Company, uses the *UNIX*† program operating system.

The applications software is written in C (a high-level language) and a superset of C called Transaction Specification Language (TSL).[8] The motivation for redesigning the software was threefold:

(*i*) The success of the LMOS system generated many requests for additional functions. Change requests started coming at the rate of one

* Registered trademark of Digital Equipment Corporation.
† Trademark of Bell Laboratories.

a week. Being responsive to such requests with minimal ripple effects was clearly a growing requirement, and "change tolerant" software became a new objective. Most of the requests required changing the FE data bases or the FE transaction masks. This motivated the development of a data base management system, a generalized mask handler, and an internal data structure designed to protect the transactions themselves from knowledge of the physical layout of data on the screen or in the data base.[8]

(*ii*) The original LMOS system was designed assuming that repair bureaus were geographically based. Thus, an FE could be expected to service a number of repair bureaus within a geographical area, and, conversely, a given repair bureau was expected to need access to only one FE. In most cases, when a trouble had to be referred from one repair bureau to another, the two repair bureaus were both based on the same FE, and software was provided so that both could obtain up-to-date information on the status of the trouble. In those rare cases where a trouble needed to be handled by a repair bureau not based on the same FE, trouble referral and any status information had to be handled manually.

The addition of a high-speed bus (Fig. 1) to the ARSB architecture allowed us to transcend some of the geographical constraints in the original LMOS system by providing multi-FE transactions. Thus, in LMOS-2, a trouble can be referred to a repair service on any FE on the bus, and its status will be known to both the original and the currently responsible bureau.

The multi-FE features are still evolving, but have been useful in meeting the requirements of a changing Bell System organization. Shortly after LMOS-2 development started, the Bell System reorganized along market segments. Instead of having a repair bureau serving a geographical area, new business, residence, and network repair bureaus were planned. The more recent reorganization of the Bell System into regulated and unregulated companies has imposed yet another set of requirements on LMOS. These changes will also make use of the new ARSB architecture.

The evolving requirements resulting from the changing organizational environment have reinforced our commitment to producing "change tolerant" software.

(*iii*) Software technology had developed to the point where the idea of building a transaction system from a number of re-usable software tools or components appeared feasible. This tool-oriented approach is described in Ref. 8.

VIII. SUMMARY AND CONCLUSION

The LMOS FE computer is the "trouble tracking" component of the

ARSB. Customer reported troubles are entered into the system by attendants in a CRSAB, automatically tested using MLT, and routed to the appropriate RSB. At the repair bureau, status information is updated at each step toward the completion of the repair, and management reports are generated to warn of overload conditions and potential missed commitments.

The first LMOS FE was installed at Southwestern Bell Telephone Company in June, 1975. As of year-end 1981, approximately 230 FEs were deployed in 18 operating telephone companies covering approximately 65 million customer lines.

REFERENCES

1. R. L. Martin, "Automated Repair Service Bureau: The System Architecture," B.S.T.J., this issue.
2. C. M. Franklin and J. F. Vogler, "Automated Repair Service Bureau: Data Base System," B.S.T.J., this issue.
3. S. P. Rhodes and L. S. Dickert, "Automated Repair Service Bureau: The Trouble Report Evaluation and Analysis Tool," B.S.T.J., this issue.
4. O. B. Dale, T. W. Robinson, and E. J. Theriot, "Automated Repair Service Bureau: Mechanized Loop Testing Design," B.S.T.J., this issue.
5. P. S. Boggs and J. R. Mashey, "Automated Repair Service Bureau: Cable Repair Administrative System," B.S.T.J., this issue.
6. J. P. Holtman, "Automated Repair Service Bureau: The Context-Sensitive Switch," B.S.T.J., this issue.
7. M. W. Bowker et al., "Automated Repair Service Bureau: Evolution," B.S.T.J., this issue.
8. R. F. Bergeron and M. J. Rochkind , "Automated Repair Service Bureau: Software Tools and Components," B.S.T.J., this issue.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July–August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Software Tools and Components

By R. F. BERGERON and M. J. ROCHKIND

(Manuscript received July 15, 1981)

Complex software systems can be built from components, where a component is a software entity that performs one type of function in a general, usually table-driven, way. If the components are defined well and a standard interface is provided between them, the resulting product is flexible. It is easy to understand and easy to change. The performance of the system is a function of that of the components. The new Loop Maintenance Operations System Front End (LMOS-2) is a transaction processing system developed in this way. The components, described in this article, are a mask handler, a data validator, a database management system and a transaction-oriented filing system. Transactions are written in a high-level language designed for the purpose. The LMOS-2 system runs on the UNIX operating system. The desired flexibility properties have been realized and have produced important benefits in the course of LMOS-2 development. Moreover, with considerable attention to the performance characteristics of the components, a system has been produced which is as fast and efficient as its predecessor, coded mostly in assembly language.*

I. INTRODUCTION

The software systems developed in the past for the Automated Repair Service Bureau (ARSB) have been well suited to the application. By almost any measure—performance, availability, protection of the database—they are excellent systems. Their major drawback is inflex-

* *UNIX* is a trademark of Bell Laboratories.

ibility—a fault shared with many systems of their generation, particularly production systems built for major applications.

The Loop Maintenance Operations System (LMOS) front end (FE) was handcrafted for the ARSB application. The Bell Operating System (BOS) was built to the requirements of LMOS, as was the BOS Filing System (FS). The original application code was written in PDP* MACRO-11 assembly language. Application programs access and manage data files themselves, calling on FS only for direct operations on data blocks. As a result, database changes ripple through the application code, making change expensive and error prone. Another consequence of making LMOS very application-specific is that there has been little spin-off of LMOS technology to other projects. This is unfortunate, for LMOS has been an extremely successful product.

During the past decade (but for practical purposes after the LMOS FE system was built and deployed), the *UNIX* operating system was developed, bringing with it new software development tools and new notions of software flexibility and transferability.[1] Several years ago, the *UNIX* operating system supplanted DEC's DOS operating system in our development environment and we began to write new programs in C. The BOS continued to be our production operating system.

It was not enough to change the development environment, though it helped. Costs incurred in modifying the FE, extrapolated into an environment of increasingly rapid change, indicated that an entirely new approach would be necessary. The course we followed in LMOS-2 was inspired in part by the *UNIX* operating system and represents a step forward in software development technology. It is expected to be cost-effective for LMOS and, unlike previous LMOS development work, should have significant side benefits for other Bell System software projects.

The approach, described more fully in the next section, is to build the software system from components, taking care that the components are sufficiently general in function and that they interface neatly together. We avoid the temptation to handcraft another LMOS. Not that the temptation isn't there, for our objective is to buy flexibility without paying in performance or in any other way.

We think we have partitioned the system into components in such a way that the partitioning itself causes no loss of efficiency. We then have only to worry about the performance of the individual components. The challenge, for all components, is to be both general and efficient (either property by itself is easy). This was relatively straightforward for some components (the mask handler and the data valida-

* Registered trademark of Digital Equipment Corporation.

tor), and extremely difficult for others (the database management system and the filing system).

II. DEVELOPING SOFTWARE FROM COMPONENTS

The ability to adapt a system to a rapidly changing environment is related to an ability to build totally new, though similar, systems with relative ease. The difference in capabilities required of an LMOS system at points in time ten years apart might be quite significant. If we can really make LMOS tolerant to change, we should be able to use the same approaches to build new systems.

Our approach is to develop the capability to build "LMOS-like" systems and, in the process, make LMOS itself flexible.

One of the keys to this approach is *modularity*. However, we seek not just to achieve modularity for LMOS, but to identify for a class of systems a set of basic, application-independent, functions. For each function a software component is built, if it does not already exist somewhere.

Another significant factor is *generality*. Our components are script-driven. Each is configured for a specific application by coding the specifications for that application in a special-purpose language. For example, the data validator is configured by coding the validation conditions (which look much like expressions where the variables are data field names) in a validation language.

Finally, we have a standardized format for passing data between components. This guarantees that the output of any component is in a form suitable for input to another component. Hence, many different arrangements of components are possible. Once the software components are on the shelf, it should be possible to build new systems, or change an existing one, rather easily.

Ultimately, we hope that building a software system will be similar to building a house: you start not by designing everything from scratch, but by looking through catalogs of windows, kitchen cabinets, heating systems, etc. You keep costs down, improve quality, and shorten the completion time by limiting on-site construction to building what you can't find ready-made, and to installing what you got from the catalogs.

III. COMPONENTS OF A TRANSACTION PROCESSING MACHINE

In our model of a transaction processor, there is a collection of input cathode ray tube (CRT) terminals at which operators enter transactions. The processor supports a predefined set of transaction types. For each transaction type there is input data, entered on a mask by the operator, and processing by the system, which involves validation, database access, and (sometimes) updates. The output may be a report, directed

to a printer or to a CRT, or just an indication to the operator that the desired function has been performed.

We can now identify some of the general-purpose components for transaction processing systems: a mask handler, which is easily programmable to produce input and output masks of arbitrary design, and which manages data flow to and from the CRTs; a data validator, which is capable of taking a set of data (e.g., input data for a transaction) and testing it against a corresponding set of validation conditions; and a database manager, which provides flexible yet efficient access to the system's database. With these components, and another component specific to the transaction to provide control and do any special processing required, we have nearly enough to implement many simple transaction types. The transaction model is this: the named mask, with input data, is on the screen when the SEND key is pushed, initiating processing. The mask handler routes the data to the validator for input validation. If valid, the data are sent to the transaction-specific component, which determines the rest of the execution sequence. That may involve local processing, as well as data transfer to and from the database manager and further validation. Finally, output data will be returned to the mask handler to be put up on the operator's screen or at a printer.

Note that the transaction is served by a number of components, including one that is tailored to it. The standardized component-to-component interface completes the component model of transactions.

IV. COMPONENT INTERFACE

The flow of data between components varies with each transaction. To allow flexibility in the order in which components are invoked and in the passing of data between them, there is a standard mechanism for invoking components, embodied in a subroutine called *invoke*, and a standard representation for data, called a *packet.*

4.1 Packets

A packet is an abstract data type that can represent a data record in which the fields are referenced by their names (as opposed to their positions within the record). Conceptually, a packet is a set of pairs of field names and field values; for example,

TN	201-3862773
NAME	Marc
ADDRESS	1D 301A Whippany.

Operations are provided to initialize a packet, change the value for a named field, and get the value of a named field, among other things. Field values are null-terminated strings. The data structures underly-

ing the packet implementation are not accessible to the user and do not need to be. All manipulation of data in packets is done using the packet operations. Packets provide a natural, easy-to-use mechanism for expressing and interpreting fielded data.

By standardizing on a data representation for all components, we allow the output from one component to be fed into another component without translation or reformatting. Also, common utilities, such as trace routines, can be used on any component.

4.2 Component invocation

All components have the same interface, a subroutine call with three arguments: the name of the component to be invoked, the packet to be operated on, and a flag indicating whether the caller is to wait for completion of the operation or not. The input data and the output data are in the same packet. This permits the use of a particularly efficient data transmission technique using memory shared between the calling and called components. For example, the Database Management System (DBMS) component is called from another component by

invoke ("DBMS", pkt, WAIT);.

This sends the packet pointed to by *pkt* to the component named DBMS. The packet contains fields that indicate to DBMS the operation, file name, key, and also the name-value pairs for the database fields involved in the access. For a "retrieve," for example, the names of fields for which data are desired would be present in the offered packet. The DBMS returns the packet with the values filled in.

V. TRANSACTION SPECIFICATION LANGUAGE*

The Transaction Specification Language (TSL) is a high-level programming language that includes the C language as a proper subset. The "non-C" features of TSL are intended to facilitate writing the transactions of LMOS-2 and similar applications. In particular, transactions written in TSL are easier to write, modify, maintain, and understand than the same programs written in C.

The key features of TSL are as follows:

- A *packet* data type, and operators for manipulating packets.
- String operators (e.g., concatenation).
- An operator for pattern matching.
- Statements to facilitate the use of the DBMS.

In the following, we describe the "non-C" features of this version of TSL. We assume that the reader is familiar with C and its conventions.[2]

---

* The Transaction Specification Language (TSL) was designed and implemented by J. W. Hunt. This section was adopted from his TSL user's manual.

5.1 Declaration types

In addition to the basic C-declaration types, TSL provides three new declaration types: *packet*, *field*, and *view*. Variables using the *packet* declaration type name instances of the abstract data type discussed in Section IV. The *field* declaration type is used to declare the names of accessed packet fields. Aggregates of field names are declared using the *view* declaration type.

5.2 Operators

Two binary operators are available in TSL that are not available in C: the concatenation operator (//) and the regular expression operator (*matches*). The language also provides an additional interpretation of the "." operator which is used in C to reference members of a structure.

5.2.1 Concatenation

The concatenation operator requires both operands to be pointers to a null-terminated sequence of characters. The value of the expression

$$a \,//\, b$$

is a pointer to a null-terminated sequence of characters consisting of the nonnull characters pointed to by a followed by the characters pointed to by b. This operator is simply an abbreviation for the *strcat* function of C.

5.2.2 Regular expression matching

Transaction programmers are often interested in finding out whether a string is a representative of some set of strings. The *matches* operator provides a general facility for doing just that. As with the concatenation operator, the *matches* operator requires both operands to be pointers to a null-terminated sequence of characters. The regular expression (the second operand of the *matches* operator) is a shorthand for specifying a set of strings. The value of the expression

$$a \textit{ matches } \text{regx}$$

is nonzero if the string pointed to by a is a member of the set of strings specified by the regular expression *regx*.

The regular expressions recognized by TSL are similar to those used in the *UNIX* text editor. That is, "." matches any character, "$" matches the end of a string, "[xyz]" matches "*x*", "*y*", or "*z*", and so on.

5.2.3 Referencing fields of a packet

Suppose *pkt* is a pointer to a packet and LRTN is a declared field

name. The term

pkt.LRTN

represents the value of the LRTN field in the packet *pkt.* If the term appears to the left of the assignment operator, the result is to change the value of the LRTN field in *pkt.* For example,

pkt.LRTN = "210" // "3863000";

changes the value of the LRTN field to "2013863000." If the term *pkt.LRTN* appears to the right of the assignment operator, the result is the value of LRTN, which is a pointer to the null-terminated sequence of characters stored as the field value.

5.3 Statements

The data-type *packet* plays a central role in the code written by a transaction programmer. For example, every transaction in LMOS-2 accepts a single input, a pointer to a *packet.* This packet is created by the mask handler (see next section). Transaction programmers also use packets to send information back and forth between other processes in the system. Thus, most of the "non-C" TSL statements provide convenient mechanisms for manipulating the values stored in a packet. In the following, examples from LMOS are used to help describe TSL statements, but keep in mind that these statements are useful in any application that must manipulate packets.

5.3.1 Creating a view for a database call

When retrieving information from a file using the DBMS (see Section VIII), one is required to send DBMS a packet that has all of the fields of the view to be retrieved in the packet. The DBMS forms the intersection of the fields of the view corresponding to a record of the file being queried and the fields in the packet, and returns the values of the fields in this intersection. Therefore, the first time that a retrieve request is made for any particular file, the transaction programmer must make sure that all the relevant fields are in the packet that was sent to DBMS. At worst, this would entail a separate statement to initialize each field of the view to be retrieved.
The TSL statement

make pkt a uif view

changes the value of each field of the view *uif* in packet *pkt* to the null string. If the packet is then used in a retrieve request to the DBMS, the fields that were listed in the definition of view *uif* will be retrieved.

5.3.2 Copying the fields of a view

There are statements that permit transfer of fields of a given view

from one packet to another and from a packet to and from temporary storage.

5.3.3 A variant of the "switch" statement

In addition to the *switch* statement of C, TSL provides a similar statement called *rswtch* which allows the programmer to affect flow of control based on the value of a null-terminated character string.

5.4 Experience with TSL

Transaction Specific Language has been used to code all of the transactions for LMOS-2. There are 53 transactions, averaging 425 lines of TSL code. While our evaluation is subjective, we feel that, in comparison to the old assembly language transaction programs, and even in comparison to a few transactions that had been written in straight C, the TSL programs are smaller, more easily written, faster to debug, and more readable. Furthermore, since the facilities of TSL are so well suited to the programming of transactions, there has been no significant loss of efficiency.

VI. MASK HANDLER*

The Loop Maintenance Operations System uses a fill-in-the-blank approach for transaction entry. The terminal operator initiates a transaction by first requesting the appropriate mask (for example, "/FOR TE" requests the trouble-entry mask). The mask that is displayed resembles a paper form: there are blanks to be filled in, and "preprinted" labels. The labels are protected; that is, the operator cannot overtype them. After the operator fills in the blanks, he or she submits the mask by pressing the SEND key. If the transaction has a result to be displayed, another mask (possibly with more blanks to be filled in) is displayed. Otherwise, the operator can clear the original mask and reuse it.

As we said in Section IV, LMOS-2 uses a standard data format, the packet, internally. The mask handler, thus, has two translation duties: When a mask is entered, it gets the data from the terminal in a format native to the terminal (a Teletype† TTY 40/4 or an IBM 3270 terminal) and builds a packet. On output it takes a packet and builds a mask, again in the format native to the terminal. Not only does this translation hide messy details of the terminal from the rest of the system, but it also allows different terminals to be used without affecting any module other than the mask handler itself. In fact, we have a version of the mask handler that works on simple time-sharing terminals.

---

* The mask handler was designed and implemented by D. A. Rosenthal.
† Registered trademark of Teletype Corporation.

These terminals are inefficient for production use, but they are invaluable for debugging. Once a transaction works with the time-sharing terminal, it also works with the Teletype 40/4 terminal, because its interface with the mask handler is absolutely identical in both cases.

Recall that a packet consists of *named* fields. When the data comes in from the terminal, the fields are not named—they are simply sequenced in the same order that they were in on the mask. To build a packet, then, the mask handler must know what transaction's mask is on the screen, and, for that mask, what the names of the fields are. This information appears in a *mask table.* The mask table also specifies display attributes for each field that determine how the mask is to appear on the screen. This information is used when the mask is output.

The mask table contains one specification for each mask. The following is a sample mask specification:

```
_TE  CLEAR
     1        1      pam   "TE"
                     skip  "TN"
              {16}   unm   TN
     1        30     skip  "RSA
              {3}    uam   RSA
     4        1      skip  "DESCRIPTION"
              {30}   uam   DESCP.
```

The first line names the mask "_TE" and specifies that the screen is to be cleared before the mask is displayed. The underscore before "TE" means that this mask is to be used in response to a form request ("/FOR TE") rather than for entry of the TE transaction (which probably will follow the form request, as soon as the form has been filled out).

Each line of the rest of the specification describes a mask field (not to be confused with a packet field). Columns 2 and 3 are used for x- and y-coordinates or for horizontal distances. Column 4 contains screen attributes. Column 5 contains either a literal to be displayed or the name of a *packet* field to receive some data.

The second line, then, specifies that the letters "TE" are to appear in row 1 of the mask, in positions 2 and 3. Position 1 is reserved for the attribute, "pam," which specifies that the letters "TE" are to be protected and returned when the mask is sent. The third line has no coordinates, so the letters "TN" go in the next available positions, which are 5 and 6. Position 4 holds the attribute, "skip" which means that the letters are to be displayed but not returned to the computer. The fourth line specifies an unprotected field 16 positions long, so it occupies positions 8 through 23. When the mask is sent, data typed

into this field by the operator will be inserted into the packet under the name TN. The remaining lines specify labels and unprotected fields for RSA and DESCP. When displayed, the "__TE" mask would appear like this:

```
TE TN                    RSA

                DESCRIPTION.
```

After the operator fills out the mask, it might look like this:

```
TE TN 2013861234    RSA 26

                DESCRIPTION.
```

When the SEND key is pressed, the screen is read and a buffer consisting of terminal-specific control codes and data is sent to the mask handler. It sees that the first field is "TE" and looks in the mask table for a specification of the TE mask (which is similar to the "__TE" mask described earlier). It builds a packet that looks (conceptually) like this:

```
XACT   TE
DVC    T16
TN     2013861234
RSA    26.
```

The first two fields, XACT and DVC, were not specified in the mask table, but are standard fields used by the mask handler for the transaction name and terminal device name, respectively. The device name is used later to send data back to the same terminal that entered this mask.

Since the value of XACT field is "TE," the mask handler sends the packet to the TE program (written in TSL), where the processing specific to the TE transaction is performed.

It so happens that the TE transaction ends by displaying a TR mask with information about the customer whose telephone number is 201-386-1234. To display this mask, the TSL program changes the value of the XACT field to "TR" in the packet:

```
XACT   TR
DVC    T16
TN     2013861234
RSA    26.
```

It then reinvokes the mask handler by calling "invoke": (see Section IV.)

invoke("MASK", pkt, NOWAIT);.

The mask handler finds the specification for the TR mask in the mask table, uses the data in the packet to build a buffer containing the appropriate terminal-specific control codes, and sends the buffer to device T16. Processing of this transaction then terminates.

The operator, meanwhile, has observed a delay of a few seconds after pressing SEND, and then sees the TR mask with the customer's information that he or she requested. After adding some information by filling in blanks on the TR form (description of the trouble, time when the customer will be at home, etc.), the operator presses SEND again, and the cycle repeats—this time for a TR transaction.

VII. VALIDATOR

The data validator[5] takes a single packet, subjects it to various tests specified in a special-purpose validation language, and returns the same packet augmented with an error code for each test that failed. In LMOS, the validator is called by the mask handler once for each incoming mask. If an error is detected, the code is translated to a short English message by table lookup, the message is displayed at the bottom of the CRT screen, and the transaction is terminated. If the data has no errors, the packet is passed on to the appropriate transaction-processing component. Thus, each transaction programmer may assume that the input is a valid packet.

By convention, the field VALCODE is used by the validator to return error codes. So, for example, if the validator receives the packet

NAME	Mary Smith
ADDR	123 Main St
TN	2913861234

it might return the packet

NAME	Mary Smith
ADDR	123 Main St
TN	2913861234
VALCODE	tnerr.

The code "tnerr" indicates that the value of the TN field failed its validation test (291 is an invalid area code).

The mask handler, upon receiving the packet returned by the validator, determines whether any errors were detected by checking for the presence of the VALCODE field. In this example, it translates the code "tnerr" to, say, "Invalid area code," and displays the message.

Presumably, the operator will correct the TN field on the mask (the old mask values would still be on the screen) and then press the SEND key. This re-entered transaction would be an entirely new transaction; the system would not, and need not, remember the previous, failed attempt to enter the transaction.

The validation criteria for a particular application (e.g., LMOS) are specified in a table which configures the validator for that application. The configuration process involves compiling the validation table with the validation compiler, which produces intermediate code that is interpreted by the validation machine at execution time. Fig. 1 illustrates this architecture.

What makes the validator general-purpose is that all application knowledge is contained in the validation table, so the validator may be used for different applications just by changing the table.

The validation table consists of two columns. The first is an expression, involving field names, number and string constants, and operators, that defines a validation condition. The second column contains an error code that is generated if the validation condition is false. For example:

NAME % "[A-Za-z]{4,25}"	badname
ADDR != ""	badaddr
SAL > 10000 & SAL < 75000	badsal.

The first condition requires the value of the NAME field to match the pattern within quotes. The pattern matches if the value consists of between 4 and 25 alphabetic characters. If it does not match, the error code "badname" is generated. The second condition just requires the ADDR field to be present (that is, not equal to the null string). The third condition requires the value of the SAL field to be in the range 10,000 to 75,000—supposedly, anything else is an unreasonable salary.

In the case of the last condition, it would be better to check that

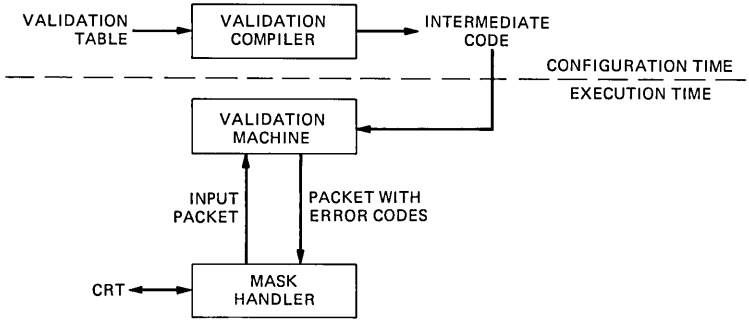


Fig. 1—Validator architecture.

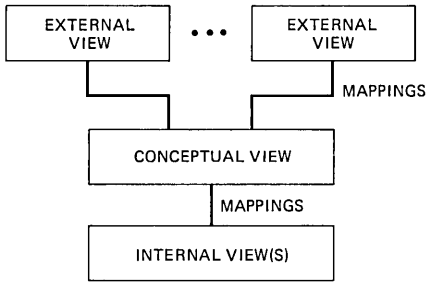


Fig. 2—ANSI/X3/SPARC database architecture.

SAL is numeric before checking its range. To do this, one may indent a condition under another condition:

```
SAL % "[0-9]{1,6}"                badsal1
  SAL > 10000 & SAL < 75000       badsal2.
```

Now error code "badsal1" is generated if SAL is not from 1 to 6 digits. The indented range check is performed only if the first test passes; if SAL is out of range, error code "badsal2" is generated.

About two dozen operators and built-in functions are provided to code validation conditions. The reader is referred to Ref. 3 for further details.

VIII. DATABASE MANAGEMENT SYSTEM*

The DBMS is packaged as a component and communicates with other components via packets, as illustrated in the example of Section IV. The packet interface is natural, as nearly all the data in the transaction processor's database are fielded. Other properties of the DBMS that are critical to its usefulness are its flexibility and efficiency. These aspects are treated in this section. The DBMS design is described in detail in Ref. 4.

8.1 Structural model

The DBMS is an implementation of the three-level ANSI/X3/SPARC model.[5] (See Fig. 2.) There is an *internal view* of the data, expressed in terms of constructs that naturally describe the layout of data on the storage medium: hash files, linked lists, trees, etc. The *conceptual view* provides a canonical application and an implementation-independent model of the data of the enterprise. The *external views* provide the interfaces to the application programs. They may be structurally different from each other and from the conceptual view, and they may

* The DBMS was designed by T. C. Chiang and implemented by Chiang and A. Weinstein.

change over time with the applications. The only requirement is that it must be possible to map them all from the conceptual view. Of course, one of the external views may coincide with the conceptual view.

The ability to define multiple external views is nice—and in one case we found it essential. The first version of the DBMS was retrofitted into existing front-end software that had had only a low-level filing system. That DBMS provided a relational "new view" of a major file, with additional fields, for several new transactions. It was necessary to do this without changing the preponderance of existing software. Therefore, an "old view" was provided for the old transactions. The old view duplicates the data format presented by the filing system. The retrofit was a success. The system is now running at more than a hundred LMOS sites.

The LMOS-2 DBMS presents several external views as a matter of convenience for different types of transactions. Another advantage of the multilevel model is that the internal representation of data can be changed relatively easily without impacting the external views. It is just a matter of changing mappings. In fact, fairly extensive internal database redesign will be done eventually for LMOS-2.

8.2 The entity-relationship model

At each level in the ANSI/X3/SPARC structure there is a data model. The external views may support several data models, as required by the application: relational, hierarchical, ad hoc, or whatever. We use the Entity-Relationship (E-R) model at the conceptual level.

In the E-R model there are "things," and relationships between things. The entities are represented by entries in a file. Files are connected by named "associations" in our version of E-R. Each instance of an association is a relationship (implemented via pointers or physical contiguity) between one or more items in one file and one or more in the associated file. The associations have properties, including cardinality (1-to-1, one-to-many, many-to-many), and update constraints.

The associations provide an easy way to express the update relationships that are to be maintained automatically by the DBMS. They provide a mechanism for navigation through the database. What is of most benefit is that they allow information to be reflected to the user about relationships between items in separate files. This makes significant performance gains possible. Data Manipulation Language commands take association name as an argument, and allow the DBMS to use information picked up in previous database accesses to eliminate index searches and, in some cases, to satisfy a data request without going to disk.

In a system like LMOS, where there are many interfile relationships,

the savings made possible by proper use of the associations may be significant. Analysis of busy-hour data for LMOS shows that nearly 50 percent fewer disk accesses will be required using the association data than would be necessary for a pure relational model in which each file were accessed independently. (In the old LMOS, transactions had direct access to pointers everywhere, the system was tuned for performance over a number of years, and there was no data model. Our problem has been to fit a semantically consistent data model to LMOS *without* sacrificing performance. Our E-R-based system and the old LMOS are equivalent in the number of disk accesses required.)

8.2 Scope of application

The LMOS-2 front end is a medium-sized record-based transaction processing system. The database is in excess of 300 megabytes, organized into 14 external files with 10 associations. External records range in size from a few tens to a few hundred bytes. Internal records may be thousands of bytes long.

The DBMS and the filing system (described below) run on version 4.0 of the *UNIX* operating system, modified somewhat to meet the performance objectives.

IX. THE C FILING SYSTEM*

The *UNIX* file system is a flexible and useful tool. It is not particularly well suited, however, to production transaction processing systems for which efficiency, protection of database consistency, and quick recovery from system crashes are critically important.

Therefore, we have developed a new filing system, the C filing system (CFS). This is a component, interfacing with DBMS and other subsystems requiring access to data in secondary storage. The CFS internal files are stored contiguously, minimizing the number of disk accesses required. Data are transferred across the CFS interface using our version 4.0 *UNIX* operating system's shared-memory facilities, another factor that improves efficiency. Multiple copies of CFS may be run, so that disk I/O may be overlapped.

9.1 The role of CFS in protecting database consistency

Database consistency is defined by a possibly large set of integrity constraints that often, in a complex system, are not formally expressed. The closest we get to a comprehensive statement of them is the set of audit/edit programs that is run against the database to find and correct inconsistencies. Inconsistencies may take the form of incorrect pointer

* The C Filing System was designed and implemented by D. H. Carter.

relationships, incorrect assignment status, failure of sums to balance, etc.

Because database updates are done one at a time, and because typically there are relationships between updates, the database cannot be consistent at every point in time. We can and do, however, identify groups of updates such that if the database is consistent before the group, it will also be consistent after all the updates have been made. These groups correspond to the transactions processed by the system. We can define the transaction, in fact, as the unit of database consistency. Care must be taken by the transaction designer to ensure that this is indeed the case.

Database inconsistency can arise in the following ways:

(*i*) In the event of a software crash, transactions may be left partly done. Inconsistency results if some, but not all, of the updates for a transaction are implemented.

(*ii*) If a database must be reconstructed, updates associated with transactions that had not been completed as of the catastrophe cannot be redone, for the same reason.

(*iii*) If transactions are run in parallel, with their database accesses interleaved, database states that could not have been developed by any serial execution of transactions may arise. These may be inconsistent. If we can guarantee that all database states are equivalent to states that would be produced by some serial execution of transactions, we achieve database consistency.

The CFS provides capability to recover from hardware and software failures and preserves consistency in doing so. It also provides concurrency control: data accesses of transactions executing in parallel are interleaved in such a way that serializability is guaranteed.

9.1.1 Concurrency control mechanism

Locks are applied at the block level by CFS. (The block is the unit of physical data access.) The CFS has "share" locks, which allow other readers, and "exclusive" locks, used when writing. Locks are applied when needed, on behalf of a transaction, and are held until the transaction is complete. This locking policy guarantees serializable transactions.

9.1.2 Logging and recovery

To protect database consistency in the face of system hardware and software failures, CFS delays updating the "real" copy of the file until the transaction signals, via a "commit" message, that it has issued all its update requests and is ready to have them implemented. Until the commit signal is received, updates are held in CFS buffers, tagged by transaction ID. Upon receiving "commit," the logging subsystem gathers all updates for the transaction into a large buffer and writes the

buffer to the logging area on disk. If this atomic write is successful, the transaction is considered completed. Writes to the individual files must still be made, and locks are held until these are complete.

The overhead of this logging approach is minimal: one additional write per transaction. Recovery is simple if there has been no database damage. Transactions which have not committed have written nothing to the database. They simply disappear. Transactions which have committed but have not completed disk writes must have them re-applied from the logging area. The recovery process takes less than five minutes.

The strategy for recovering damaged databases involves periodic full copy of the database—daily for LMOS—and logging to tape of all committed updates. The database is reconstructed by applying the log tapes to the most recent backup copy. This may take an hour or two depending upon time elapsed since the backup copy was made.

X. EARLY FIELD EXPERIENCE

The LMOS-2 system, built with the components described in the preceding sections, went into field trial on October 16, 1980, with Michigan Bell Telephone Company. Development of the application and the components had proceeded in parallel over the preceding 15 months. The staffing was split roughly two-to-one between component and application development.

The development experience had met our expectations. With validation, mask handling, and database services provided by special components, and with the high-level TSL language and the simple packet interface, transactions were small, easy to write, and easy to debug. The reduction in lines of source was 7 to 1. Programmers who had experience implementing and maintaining the LMOS-1 transactions found the difference particularly striking. Even inexperienced programmers were able to build working transactions in a very short period of time. One, in his first programming assignment, wrote six transactions in four months.

Success or failure, however, would be determined by behavior of the system in the field. How would LMOS-2 compare in reliability and performance with its predecessor?

The reliability question was answered quickly and positively. On the first day of operation the system suffered five "fatals" (i.e., errors causing interruption of processing.) By the end of the first week, it was getting through an entire day without a fatal; by the end of the first month, its availability surpassed that of some of Michigan Bell Telephone Company's LMOS-1 systems. This compared favorably with the LMOS-1 trial experience, in which months passed before the system could process through the day without interruption.

Problems that did surface in LMOS-2 were almost always fixed within a day, sometimes within hours. The ability to find and fix problems quickly is due at least in part to the component-based system design.

As we indicated earlier in the article, performance of LMOS-2 was a major concern. Our objective was to add flexibility to the system with minimal sacrifice of performance. At the time the trial began, the load presented by the application—a new system serving a single repair bureau—was small: on the order of 300 to 500 trouble processing transactions in the busy hour. Laboratory load testing showed the system to be capable of processing 800 to 1000 transactions per hour at that time. The objective—the standard set by LMOS-1—was 4000 transactions per hour. In the seven months that have elapsed between the field trial cutover and the time of this writing, significant efficiency improvements have been made. The LMOS-2 system now processes more than 4000 transactions per hour in load tests. (The trial application has grown during the same interval and now presents a busy-hour load of 2800 transactions per hour.) Performance of LMOS-2 is approximately as good now as that of its predecessor.

The flexibility of LMOS-2 has served us well in management of the project and of the trial. One example: Much care was taken to ensure that in the early days of the trial the LMOS-2 and LMOS-1 databases were compatible, so that in the event of user-affecting problems at the trial site it would be possible to convert back to an LMOS-1 system within minutes. (This option was used several times: it maintained LMOS availability and gave us additional time to solve problems.)

To add new features to LMOS-2, a database conversion was performed in May and new versions were introduced for about 30 modules. Quick convertibility was given up at this point.

It is obviously desirable to give new standard installations of LMOS-2 a similar initial period during which reversion to LMOS-1 is possible. One way of doing that is to maintain the first versions of all the software used in the trial and use them in initial LMOS-2 installations. Fortunately, the generality of our component interface allows us to install all of the latest versions, except for the DBMS, atop the LMOS-1 database. The resulting system is functionally equivalent to LMOS-1 and can be backed out to it quickly. The new LMOS-2 features become available when the data conversion is performed and the DBMS replaced. We have the best of both worlds: easy convertibility in the early days of an installation, without the need to maintain two versions of the software.

XI. CONCLUSION

We have the beginnings of a catalog of components out of which "LMOS-like" transaction processing systems can be built. The compo-

nents include a mask handler, a data validator, a database management system, and a filing system. There is a standard interface—packets—between components, and a language for programming transactions that fits the component model. Because the set of systems which we wish to build includes some (like LMOS) that place a premium on performance, availability, and integrity of the database, the database manager and filing system are designed with efficiency and robustness, as well as flexibility, in mind.

The first system to be built from our components is the LMOS-2 FE, which has been field tested since October, 1980, with very promising results.

XII. ACKNOWLEDGMENTS

Development and application of the LMOS software components has been a cooperative effort involving many individuals. Special recognition goes to J. W. Hunt for TSL, D. A. Rosenthal for the mask handler, T. C. Chiang for the DBMS, and D. H. Carter for the C Filing System.

REFERENCES

1. B. W. Kernighan and J. R. Mashey, "The UNIX Programming Environment," COMPUTER *14* (April 1981), pp. 12–24.
2. B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Englewood Cliffs, N.J.: Prentice-Hall, 1978.
3. M. J. Rochkind, "A Table-Driven Data Validator," Proc. of COMPCON FALL 1980, IEEE Catalog No. 80CH1598-2C.
4. T. C. Chiang and R. F. Bergeron, "A Data Base Management System With an E-R Conceptual Model," Proc. of Int. Conf. on the Entity-Relationship Approach to Systems Analysis and Design (December 1979), pp. 540–9.
5. D. Tsichritzis and A. Klug, "The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems," Oxford: Pergamon Press, 1978.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

The Context-Sensitive Switch of the Loop Maintenance Operations System

By J. P. HOLTMAN

(Manuscript received June 5, 1981)

In a distributed system where the data base is partitioned, e.g., on a geographical basis, incoming transactions must be routed to the correct computer based on information in the data. In some instances, this is performed manually by the attendant entering the data, but as the switching criteria becomes more complex and the transaction load increases, a mechanized system which could examine the data and route automatically would decrease the time required for an attendant to handle an individual input and, thereby, increase the productivity of the attendant. This paper describes the design and implementation of a "context-sensitive" switch that is used to route transactions in the Loop Maintenance Operations System.

I. INTRODUCTION

The Loop Maintenance Operations System (LMOS)[1] is a hierarchical network composed of a host computer[2] which is a large data base machine, and a number of transaction processing front-end (FE) computers.[3] The entire data base is contained on the host computer, but this data is distributed to each of the FE computers based on the geography of the system. The geographical boundaries that are defined by an FE are those areas served by a Repair Service Bureau (RSB).

Since an RSB typically is associated with several wire centers, the customer lines that are served by that bureau correspond to the geographical area served by those wire centers. These lines are typically grouped into what are referred to as NNXs. An NNX is the first three digits of a telephone number, commonly referred to as an

exchange. When LMOS was first designed, all trouble reports for a given geographical area were routed to the appropriate RSB automatically by the telephone switching equipment. There, the attendant would input the trouble report to the FE that served that RSB. In this case, everything worked fine as long as the calls were routed to the correct RSBs.

As the Bell operating companies (BOCs) consolidated work centers to give better service, and reduced the size of the staff necessary to do that function, they created the Centralized Repair Service Attendant Bureau (CRSAB). This is the central site that handles all the trouble report calls for a very large area; this area now spans many FEs. Since the calls are answered in the CRSAB on a random basis, there is no longer the guarantee that the calls will be directed to an attendant who has a terminal that is directly attached to the proper FE. Therefore, the attendant has to have the capability of dynamic switching his/her terminal between any of the FEs to input the trouble reports.

In an existing BOC, there were six FEs servicing the entire metropolitan area. When the CRSAB was created in this BOC, it necessitated the capability for each answering position in the CRSAB to be able to access any one of the six FEs. This implied that the attendant, upon taking the trouble report, had to determine which of those six FEs contained the telephone number referred to in the trouble report. Therefore, the attendant had to be able to determine for a given NNX (one out of 600 in this area) which FE contained that NNX (one out of six).

This was exactly the manual solution that the BOC was forced to take. At each of the answering positions in the CRSAB, there was located a six-position switch. This allowed the attendant to select one of the six FEs in which to input the trouble. The appropriate FE was selected on the basis of the telephone number for which the trouble was being reported. A table was provided giving the FE associated with an NNX and the attendant would consult this to determine the FE.

If the NNX was not in the table, the attendant would try to enter the transaction on each of the FEs until the transaction was accepted or negative responses were obtained from all the systems. If all the responses were negative, the attendant would change the transaction name to route this trouble for special handling. This manual switching arrangement was expensive and complex. It was also dependent upon a specific type of terminal and, therefore, was not usable by other BOCs.

To provide a system that was usable by all the BOCs and to provide for future extensions to the LMOS system, the design of the Cross Front End (XFE) was undertaken. The main function of the XFE is to automatically determine the FE that contains a particular phone number. When this determination is made, it routes the input trans-

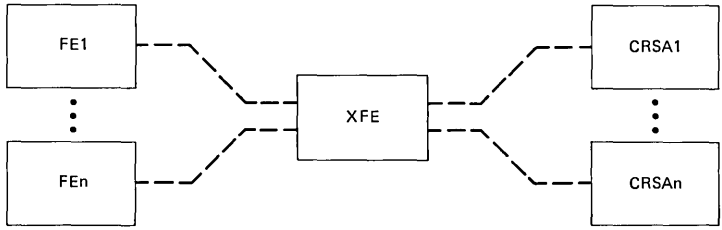


Fig. 1—Cross front end in the LMOS network.

action to that FE, and when the response to that transaction is obtained, it routes the response back to the appropriate terminal. The XFE is situated in the network in such a way that all the terminals in the CRSAB are connected to the XFE, and the XFE, in turn, is connected to all the FEs. Therefore, the XFE appears in the same location as the manual switch did in the previous discussion and serves the same purpose, except now the function is being done automatically.

II. BASIC DESIGN OF THE CROSS FRONT END

The basic design for the XFE was relatively simple. Its function was to be a "context-sensitive" switch. That is, it would look at the phone number on a transaction, look that phone number up in an internal table which gave the mapping between phone number and FE, and then route the transaction to the FE indicated. When the response to that transaction was obtained from the FE, its function was then to route that response back to the terminal that originated the transaction.

Since a BOC could optionally install an XFE as part of an LMOS system, it was designed so that it did not affect the operation of the FE software. That is, it had to be completely transparent to any existing software in the LMOS system. Therefore, the XFE was designed to appear as a terminal controller to the FE, making it transparent to the FE, whether there was a real terminal controller at the other end of the line, or an XFE simulating the terminal controller. (See Fig. 1.)

The hardware configuration for the XFE consisted of a PDP* 11/34 computer with 256 kilobytes of memory, a cache memory to increase the speed of the processor, a maximum of 32-line controllers, and a floppy disk for loading the system software and for recording system dumps on abnormal terminations. For reliability reasons, the XFE was configured in either a duplex or a triplex configuration. A duplex consisted of a live XFE and a backup XFE. A triplex configuration consisted of two live XFEs and a backup which could be configured as

* Registered trademark of Digital Equipment Corporation.

either one of the other XFEs. Switching between a live and the backup was accomplished with a manual switch and the entire operation was initiated by the operators in the computer center.

III. STRUCTURE OF THE XFE SOFTWARE

The XFE software consisted of three major components:

(*i*) The operating system

(*ii*) The communications management software

(*iii*) The application code.

The operating system used for the XFE was the Bell Operating System (BOS11); it is the same one that was used on the FE computers. The BOS11 was field proven, reliable, and very few changes were made in it except to include the new device drivers for the communication lines.

The Communications Manager on the XFE was completely rewritten from the Communications Manager that existed on the FEs. The reason for this is that the Communications Manager on the FEs was not general enough to handle the capabilities required in the XFE.

The Application Code (APL) is the software that implements the functions of the XFE. When a transaction is input, APL uses information contained within the input data to switch that transaction to the appropriate FE. The APL divided into three modules.

(*i*) A module to handle the interactions with a terminal

(*ii*) A module to handle the interactions with a FE.

(*iii*) A module to handle the interactions with the XFE console terminal.

The rest of this section will discuss the design of the XFE in more detail.

3.1 System configuration

An XFE can handle a maximum of 150 terminals, while an FE has the capacity for 512 terminals. If the XFE were configured in the simplest fashion, then each terminal on the XFE would have an appearance on an FE. This would allocate 150 of the 512 terminals on an FE to the lines serving the XFE. But, at any one time, only a small fraction of the terminals on the XFE would be in communication with a specific FE. Also, since the customer contact time (120 s) is long compared to the response time (10 s) of a transaction, the XFE was designed to dynamically couple a terminal to an FE only for the duration of the transaction. A transaction is defined as the input from a terminal and the output response returned to that specific terminal. Therefore, after the output has been sent back to the specific terminal, the XFE can free up that coupling for use by another terminal. This allowed for the optimal use of the resources both by the XFE and the FEs.

When the XFE was put into the network, there were two physical communication lines from each FE to the XFE. These communication lines were duplexed for reliability. A single communication line was capable of handling the necessary transaction load between the FE and the XFE. On each of these communication lines, there was the appearance of 16 pseudo terminals (PTERM). When the XFE receives an input transaction from a real terminal, it dynamically allocates an available PTERM on the FE to which the transaction is to be routed. The XFE then transmits the transaction so that it appears, to the FE, as if it were entered from the PTERM. When the response is received at the XFE from this PTERM, it is routed back to the original input terminal and the PTERM is deallocated, freeing it for another conversation between the XFE and the FE.

3.2 Routing definition tables

The routing tables provide the XFE with the information necessary to determine how to route the transactions to the appropriate FEs. These routing tables consist of two basic sets of tables. The first set of tables contains the transaction name and the location of the field that the XFE is to use to obtain its routing information. The other set of routing tables contains the routing criteria; i.e., a list of NNXs versus FEs that contain those NNXs.

The information for these routing tables is obtained from two sources. The table that contains the information on the transactions is prepared manually by the operations personnel at the BOC from information provided by the Western Electric Company. The information contained in the routing criteria tables is created automatically from information on the FE computers. An off-line program is run on each of the FE computers in an LMOS installation that reads the data base on the FE computers to determine which NNXs are loaded on that data base. The results of this program are recorded on a tape, and the tapes from all the FEs are merged on to one single disk from which this table is generated. Once these tables have been generated, they are loaded on the XFE floppy disk and automatically loaded into memory when the XFE system is booted.

3.3 Network definition tables

Each XFE installation has its own unique requirements in the way that the physical communication lines are configured. As the system grows, an XFE may initially be connected to two FEs and finally to six FEs. At each stage, the XFE must know the actual communication network configuration attached to it. To do this, a facility is provided so the personnel in the computer center can describe the configuration to the XFE.

The description of the network configuration is prepared on a card deck that is read by an off-line utility program which stores the resultant tables on a floppy disk file that the XFE can read when the system is initialized.

3.4 Processing special service numbers

So far, it has been assumed that the XFE uses the telephone number that was input to determine its routing criteria. This routing was done by looking up the first three digits of this phone number to determine to which FE to route transaction. For most POTS ("plain old telephone service") numbers, this assumption is true. And since POTS numbers make up a large percentage of the subscriber lines on which troubles are reported, then this represents normal processing required on the XFE.

But, customer circuits, called Special Service Circuits, exist that do not have the normal seven-digit phone number that is typically associated with a telephone. For example, if a customer has a dedicated data line connecting two computers in different locations, this line may be designated with a phone number that would look like "96PL1234A." If one tried to use the first three digits of this phone number, "96L," to determine the routing criteria, the XFE would determine that this NNX does not exist in any of its tables.

If this were the manual system, the attendant would first try to input this transaction to FE number one, get the response, and if the response said that this number did not exist on FE number one, the attendant would switch to FE number two and repeat this scenario until either the FE was located that contained this number, or all the FEs had been tried to no avail. Therefore, it was required that the XFEs somehow implement this same capability.

To handle Special Service numbers, an "inquiry" transaction was built into the XFE. In the case where the XFE could not determine the routing criteria by doing a table look-up on the data, the XFE would initiate an "inquiry" transaction simultaneously to each FE to which it was attached. This inquiry transaction consisted of the phone number for which the XFE was trying to determine the routing. On the FE, the inquiry transaction would attempt to access the data base with this phone number. If the data did exist on the FE, the transaction responded with a positive acknowledgment. If the data did not exist on that FE, the FE responded with a negative acknowledgment.

When a positive acknowledgment is received, the XFE would route the transaction to that FE. If no positive acknowledgments were received from any of the FEs, the transaction would be routed to the last FE to respond. This was done so that the XFE did not have to generate any error message on its own. It would route the transaction

to an FE and the application program on the FE would be written such that if it did not find the data it needed, it responded with a known error message with which attendants were familiar.

One by-product of this capability of making inquiries took care of the case where a BOC did not update the XFE routing tables when a new RSB was added, or when a new set of phone numbers were added to an FE. Since this information would not have been contained in the XFE routing tables, when one of these new phone numbers was entered on a transaction, the XFE would automatically generate an inquiry transaction and the FE containing that phone number would send a positive acknowledgment.

3.5 Operator control

The XFE was designed as a standalone system; i.e., it did not require any operator interaction. If a system failure occurred, a dump was taken automatically to the floppy disk, and the system rebooted itself and started up again. Because of the limited resources on the XFE (memory and off-line storage), the console terminal was used both as an interface to the operator and as a logging device. The information logged on the console terminal consisted of the number of transactions entered, average response time for those transactions, and other miscellaneous data that could be used both to determine the performance of the system, and to analyze any strange behavior of the system.

At the console terminal, the operator could control which lines were active, which terminals were active, and change some of the system parameters to tune the response of the system. The XFE also allowed any operator command to be input from the system control terminal on the FE. The operator would input a special transaction on the FE and the application program that was executed as the result of this transaction would output the data (command) on the communication line between the FE and the XFE. When the input was received at the XFE, it was interpreted as if it came from the console terminal. Responses to that command appeared both at the XFE console terminal and as a response sent back from the XFE to the FE so that it appeared at the user's terminal. This allowed the operators to control and monitor the XFE performance without having to use the terminal at the XFE.

IV. PUTTING THE SYSTEM INTO THE FIELD

There were adequate test facilities in the laboratory for both the development and for the testing of the software. But the real test came in trying to put this software into an actual installation in the field. It must be remembered that CRSAB is a very critical work center in the LMOS system; it takes the initial trouble report and initiates all the

basic functions that LMOS performs. Therefore, if service is discontinued for any reason to the CRSAB, there is a severe impact on the LMOS system.

As was previously mentioned, the BOC where the XFE field trial was conducted had a manual system for switching the terminals between the various FE computers. It turned out that the manual switch that was used at each of the operator positions actually was an eight-position switch. When the field trial started, the XFE was attached to one of these extra two positions on the switch. Therefore, the operator could manually select one of the six FEs as they originally did, or for the field trial, select the XFE. This allowed for parallel operations during the field trial.

Initially, 16 CRSAB positions were attached to the XFE so that system response could be gauged to a small number of users (the XFE was designed to handle 150 terminals). Since the attendants could switch back to the manual system at any time, this provided feedback on the system performance by monitoring how often these positions had to resort to manual operations because of response time problems.

4.1 Handling slow responses from a front-end

The normal response time at a terminal not connected to the XFE was 3 to 5 seconds, and it was anticipated that the XFE would add a second to this response, bringing the expected response with the XFE to the 4- to 6-s range. Early results in the field trial indicated that response time for terminals connected to the XFE was in the 10- to 13-s range. These results were not satisfactory and, therefore, an effort was undertaken to determine the bottlenecks in the system.

At this point, statistics built into the code came in very handy. By analyzing the output and displaying some internal tables on the console terminal, the problems causing this slow response time were identified.

One of the major problems causing slow response time had not been anticipated in the design phase. This had to do with detecting when one of the FEs connected to the XFE was "down"; i.e., the FE was unable to accept the transactions that the XFE was attempting to send it. The reasons for this could be either

(*i*) The FE was physically down because of some hardware or software problem, or

(*ii*) The FE was heavily loaded and responding very slowly to transactions that were sent to it.

Whatever the reason for this slow response, it caused a longer holding time for the system buffers on the XFE. In the initial design phase of the XFE, it was anticipated that a buffer would be held for a relatively short period of time—2 to 4 seconds. This was the time it took to send the transaction over to the FE and get a positive acknowl-

edgment that the transaction was received. When an FE goes down, it is not sending back this positive acknowledgment and therefore is tying up the system buffer on the XFE.

Now the XFE had a time-out such that after approximately 30 s it would release this buffer and send a message back to the user's terminal indicating that the transaction could not be completed. But, if a large number of transactions had been input which were destined to this FE, then a large number of buffers would be tied up for a long period of time. Since buffers are one of the scarce resources on the XFE end, this would lead to a bottleneck in the system, preventing the transactions going to other FEs.

To overcome this problem, a system parameter was defined which specified the maximum percentage of the available buffer space that would be allocated for transactions that were destined for a specific FE. By default, this value was set at 30 percent, so if an FE was slow in responding, only 30 percent of the available buffer space would be allocated for use by that FE. If a subsequent transaction were entered that was to be routed to that FE an error message would be returned to the user indicating that no buffers were available and that the user should wait a few seconds and then reenter the transaction.

If it was determined that an FE was physically unavailable, then the percentage of available buffer space that it was allowed would be dynamically changed to zero, preventing any transactions that would be routed to that FE from being entered into the system. When the FE came on-line again, its percentage would be changed back to the system default. With these changes, the desired response of 4 to 6 seconds was achieved.

4.2 Buffer management

The scarcest resource on the XFE is the memory space available for buffers. The XFE places a large demand on this resource because of the number of conversations that the XFE can have in progress at any one time, and because of the requirements of the bisynchronous protocol.[4] The limited address space on the PDP-11 (a program can have a total of 64 kilobytes of text and data under BOS11) required that the buffers be kept in a separate address space from the programs using them. System calls were provided to allocate/free the buffers, but because of the heavy usage of this resource, 10 percent of the CPU was used to service these requests.

To improve the performance of the system, a "cache" of buffers (actually the address descriptors of the buffers in the separate address space) was maintained in the application program. Therefore, if the application program required a buffer, it would first look in this cache to see if there was an available buffer descriptor. If there was, this

buffer descriptor was used, saving a system call. When a buffer was freed, the application program would look for an empty slot in the cache, and if there was an empty slot, place this buffer descriptor in it.

If there was no available buffer descriptor in this cache when the application code requested one, a system call was made to allocate this buffer. But instead of allocating a single buffer, the system would allocate five buffers. One was returned to the application program and the other four were used to populate the cache. This was done in anticipation of future calls requesting the allocation of buffers. This cache algorithm allowed over 80 percent of the requests to be satisfied without requiring a system call.

The bisynchronous protocol required that a "large" buffer (3000 bytes) be allocated any time data was expected from an FE. Since there was a limited number of these "large buffers," and because the average size of the response from an FE was on the order of 200 to 300 bytes, a function was added to the system to "shrink" the data from a "large" buffer into a "small" buffer of the correct size; thus, freeing up this large buffer for subsequent input.

Detailed statistics were maintained on the buffer utilization and these statistics were printed out periodically on the console terminal. This allowed the developers to monitor the buffer usage of the system, and it also gave information necessary to track down any anomalies that may have occurred.

4.3 The communications manager

The communications network managed by the XFE is referred to as a "multipoint bisynch"; i.e., the lines in this network must be polled to see if a terminal on a line has data for input. To give reasonable response time to the terminal, the line must be polled approximately once every half second. On a typical XFE, there are 10 to 18 bisynchronous lines.

Each poll requires two I/O operations, and a system call is required to initiate an I/O operations. This system call requires approximately 5 ms. Calculating the time required to handle 18 synchronous lines, there are:

$$18 \text{ lines} * 2 \text{ I/Os/poll} * 2 \text{ polls/second} * 5 \text{ ms/I/O}.$$

This equals 360 ms/s or approximately 36 percent of the CPU just to support the communication lines with no other activity in the system. Under load with all the other functions in the XFE must handle this overhead is intolerable.

When an analysis was done of what happens during the polling, it was determined that 60 to 80 percent were "idle" polls; i.e., there was no data input from the line. Therefore, a method had to be found to reduce the overhead of these nonproductive polls.

One way of doing this was to put the polling capability at the lowest level in the system; i.e., at the device driver level in the operating system. Also, the driver can determine if it was an "idle" poll and if so, just initiate the next poll 0.5 s later. By putting this "auto-poll" capability in the device driver, the overhead associated with the poll was reduced from 10 ms (two I/Os times 5 ms) to 1 ms. This was a substantial reduction in the overhead required for the communications manager.

A similar capability was also required on the lines connecting the FE and the XFE. Because the XFE appears as a terminal controller to the FE, the FE polls the XFE to determine if there are any data to be transmitted. If the XFE does not have any data ready for transmission to the FE, it must respond with an "eot" and set-up to receive the next poll. If this I/O is being done from the user program, this would require two I/Os (or a total of 10 ms) to answer each of these polls. By moving this function down into the device driver again, the overhead is significantly reduced to 1 ms to answer a poll from the FE. This capability is referred to as the "auto-answer" capability.

V. PERFORMANCE

The XFE is designed to handle a maximum of thirty-two 9.6-kilobaud communication lines. In its maximum configuration, the lines would be allocated as follows:

Lines	Function
2	LMOS host computer
20	Two lines each to a maximum of 10 FEs
10	Terminal controllers (typically 16 terminals)

There are currently over 30 XFE systems deployed (and more being added) with each XFE switching a maximum of 5000 transactions per hour.

VI. CONCLUSION

The XFE was initially developed to handle the switching of transactions in a CRSAB. It was thought that only a few of the BOCs would select this optional capability because many of the smaller BOCs could probably get by with a manual switch. But as LMOS evolved and the XFE proved to be a reliable system in the field, there was a need for this switching function in other work centers besides the CRSAB.

As the BOCs desired more monitoring capabilities, some of the operations staff terminals were attached to an XFE. Also, since the XFE acts as a terminal concentrator, and in many BOCs there is a shortage of lines on the FEs, the XFE can be used to increase the available

number of ports to the LMOS system. This is especially true when the BOC wishes to add some low usage terminals that do not require fast access to the system.

VII. ACKNOWLEDGMENTS

I would like to thank the people who are responsible for the software in the XFE. B. B. Bittner and W. F. Hoyt designed the application code that implemented the switching functions of the XFE. J. P. Haggerty, S. Myer, and R. W. Underwood developed the communications management software that allowed the XFE to "talk" to the outside world. J. Cloutier made the changes in the BOS11 to support the XFE and helped to tune the system. Without their help, and the long hours spent in the development and field trial phases, this paper would not have been written.

REFERENCES

1. R. L. Martin, "Automated Repair Service Bureau: System Architecture," B.S.T.J., this issue.
2. C. M. Franklin and J. F. Vogler, "Automated Repair Service Bureau: Data Base System," B.S.T.J., this issue.
3. S. G. Chappell, F. H. Henig, and D. S. Watson, "Automated Repair Service Bureau: The Front-End System," B.S.T.J., this issue.
4. "IBM 3270 Information Display System Component Description," IBM Publications GA27-2749-9, August 1979.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July–August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Mechanized Loop Testing Strategies and Techniques

By F. J. UHRHANE

(Manuscript received December 2, 1980)

The Mechanized Loop Testing (MLT) system overcomes two long-standing inadequacies in the administration of loop repair: the lack of a single comprehensive automated testing device capable of analyzing the problems of loops having a wide variety of central office, outside plant, and terminating equipment installed; and the lack of a device of that type that could be integrated into the repair process in such a way as to provide for efficient administrative procedures. The MLT system meets these needs by means of a novel linkage with the Loop Maintenance Operations System (LMOS), which is the backbone support system for the repair process. The MLT system is unique in the way it makes use of information in the customer line records to configure tests and analyze their results. This paper outlines the means by which customer line record information is selected and processed for use in the test algorithms of MLT. It then discusses the general strategies used in the algorithms described, with some attention to the problems posed by inaccurate or incomplete records.

I. INTRODUCTION

The testing of a subscriber loop is presently undergoing a fundamental and dramatic improvement because of the introduction of the Mechanized Loop Testing (MLT) system. The MLT system is a recently integrated part of the Automated Repair Service Bureau (ARSB) described in detail in other articles in this issue. This article briefly outlines the operational changes in the ARSB attributable to the introduction of MLT and then goes into further depth in describing the

advances in loop testing made possible by MLT's unique coupling to other ARSB systems and by its adaptive testing algorithms.

II. BACKGROUND

The backbone system of the ARSB is the Loop Maintenance Operations Systems, known as LMOS. The basic LMOS consists of a large host computer, such as an IBM 370, and several front-end (FE) minicomputers, PDP* 11/70s.

The host computer receives completed customer service orders for installing, deleting, or changing telephone services, creates a data base of records of those services ("line records") for maintenance purposes, and processes the line records into subsets ("miniline records") for download to the appropriate FE computer. In performing these functions, the host computer manages a large data base (typically millions of line records) composed of relatively slowly changing data. A typical rate of line record change is of the order of one or two changes per line per year.

The FE computers, on the other hand, process relatively fast-moving data. Each FE computer is the access point for a large group of users; it supports CRT terminals with the transaction programs necessary for tracking the progress of reported troubles. When a customer calls the ARSB to report a problem, a Repair Service Attendant (RSA) enters a trouble report via the FE CRT. Front-end programs provide line record and time commitment information for dealing with the customer, output paper trouble tickets that repair craft can take with them, and track the status of the repair process. The personnel of ARSBs can, for example, make use of FE transactions to find out which reported troubles have less than a specified percentage of their repair commitment time remaining.

The MLT system is an additional software and hardware system for which the FE computer provides the user interface. The MLT system consists of a PDP 11/34 computer controlling Loop Test Frames which comprise the loop access and test hardware. The bureau personnel enter test requests at FE cathode ray tubes (CRTs); the FE in turn sends the appropriate test and access commands to the MLT controller. The MLT controller causes the hardware to dial the desired line, and then the algorithm programmed in the controller controls the hardware test devices and analyzes their results. The results are then passed back to the FE for formatting and display for the user at the CRT or at a line printer.

Prior to the introduction of testing integrated with LMOS, the ARSB

* Registered trademark of Digital Equipment Corporation.

procedures had to be structured around a certain delay between the customer's call reporting the trouble and the actual tests performed on the affected loop. The usual procedure was as follows: The RSA would receive the report from the customer and enter it into LMOS. The resulting trouble ticket (known as the Basic Output Report or BOR) would be examined by a screener, whose function was to determine whether the trouble should be tested or sent directly to the dispatcher. The large majority of the troubles (about two-thirds) were sent to the tester. If the tester was unable to detect a problem with the loop, he or she would call the customer to negotiate a close-out of the trouble. If, however, a problem was found, then the tester would write the test results on the BOR and pass it on to the dispatcher for the necessary action. The time between the trouble entry by the repair service attendant (RSA) and the completion of testing was typically a half hour to two hours, depending upon the volume of reports.

In the procedure, useful diagnosis of the problem was made long after the customer's call, so that a call back to the customer was often required to provide a proper time commitment and to arrange for access to the customer's premises, if necessary.

Frequently, the tester would call the customer immediately after testing the line to confirm or clarify the problem; then, somewhat later the dispatcher would also call to finalize access arrangements. This succession of calls prior to accomplishing anything toward the actual repair tended to aggravate the already annoyed customer. Furthermore, many times the customer was simply not available after the initial call, so that any information that was not received from the customer at the first contact was not obtainable at all.

Integrating MLT with LMOS makes possible a streamlined procedure and allows the contact with the customer to be much more efficient (and more satisfying to the customer). In the combined LMOS/MLT system the trouble entry transaction is used to supply the troubled telephone number along with commands necessary to start tests to the MLT controller. Now, as soon as the customer gives the RSA the telephone number of the troubled line (the first thing the RSA asks for), MLT starts a test on the line. The MLT tests and analyses are usually completed in about thirty seconds, and a simple summary result is provided to the RSA *while the customer is still speaking to the RSA.* The RSA can then make an appropriate commitment to the customer.

For example, the RSA might get "C. O. FAULT" as an MLT result. This means that the tests show that the portion of the customer's loop that is outside the central office is good, but there is some malfunction that was detected with the central office wiring or switching machine functions. Such problems are normally quickly fixed, and because the

maintenance force is already at the scene of the problem there is no delay because of travel. In this case, the RSA can inform the customer that no visit to the customer's premises is required, and that the probable time to complete the repair is approximately two hours.

If the MLT tests result in the message "CABLE FAULT" to the RSA, the customer will be given a much longer commitment time than in the previous example, because cable troubles are typically more difficult and time consuming to repair. "CABLE FAULT" also implies that the trouble is not likely to require entry to the customer's premises. This information is helpful to the RSA dealing with the customer, and it permits the customer to plan accordingly.

Both of these examples illustrate improvements brought about by integrating the testing system (MLT) with the administrative system (LMOS). Troubles are tested as soon as they are input into LMOS, instead of delaying for an hour or so in a paper queue at a screening or manual testing position. Furthermore, the RSA is given test results useful in dealing with the customer, while the customer is reporting the trouble, obviating the need to call the customer back in most cases until the trouble is repaired.

The function of the screener is also affected by the fact that tests are made at the time the customer reports the trouble. Within minutes after the RSA has completed the trouble report entry, the paper BOR is printed out. This report now has as an integral part a test summary and detailed results of the measurements and analyses performed by the MLT system. The screener can use them to decide whether or not to dispatch, and if dispatched, to which repair force. The screener may also decide that further MLT testing is desirable; this is immediately available via the CRT.

If the MLT results are not conclusive, or if the line is of a type too complex for present MLT tests (four- and eight-party lines, for example), the screener may refer the trouble for manual testing. However, manual testing is now required for only a very small fraction of the troubles reported.

The procedural improvements just described would have been possible with any test system triggered by trouble entries and automated to perform tests equivalent to the usual manual ones. The MLT system, however, is much more than an equivalent automation of a manual test system. MLT integrates the customer line record information into the testing process and makes use of the information in adaptive test algorithms to configure tests and analyze results. In this respect, the MLT system is unique among test systems, past and present. Furthermore, MLT commands an additional array of tests, not possible from the conventional, manual local test desk, to define precisely the condition of the tested loop.

Because the local test desk is a familiar and nearly universal equipment (at least within Bell System telephone RSBs), it may be instructive to compare its operation to that of MLT.

III. INSTRUMENTATION: THE LOCAL TEST DESK VERSUS MLT

The local test desk is basically a dc instrument; it is essentially a battery, a galvanometer, and a set of switches. Tests which involve ac characteristics of the subscriber loop are done by watching the "kick" of the galvanometer needle when a switch is thrown to produce a transient effect. Thus, while dc measurements are reasonably accurate, ac measurements are dependent upon the tester's experience and ability to interpret the transient response of the meter.

This is not to say that a tester and the local test desk are totally inadequate for the function. A tester is typically a highly skilled, experienced person who is able to detect quite a variety of loop problems with the relatively simple testing device. However, modern test hardware such as that included in MLT can provide precise measurements to replace the largely subjective ones of the tester. In addition, MLT measurements are repeatable, and this means that analysis and trend detection can be done.

The combining of the computer with the test hardware enables the use of test series, in which the results of several physically different tests may be combined to produce a diagnosis not possible from the isolated separate tests. In this way, the computer can supply the logical processing which formerly had to be done by the tester.

IV. LOOP INFORMATION NEEDED FOR PROPER TESTING

The tester provided another important ingredient in the loop fault analysis process, and that was to supply the knowledge of what equipment the customer's circuit had on it. From reading the paper line card or the LMOS computer line record, the tester determined the sort of termination and transmission equipment installed. He or she was assumed to know the expected electrical behavior of the desk meter for each type of loop equipment tested. The MLT system must also know the anticipated results of the measurements it makes. This information is supplied by programs which specially process the necessary LMOS line record information into the MLT system. The advantages of having the computer do this processing will become obvious later in the discussion.

Loop equipment which affects the results of manual or MLT testing may be divided into three categories:

(*i*) terminating equipment,
(*ii*) outside plant equipment, and
(*iii*) central office equipment.

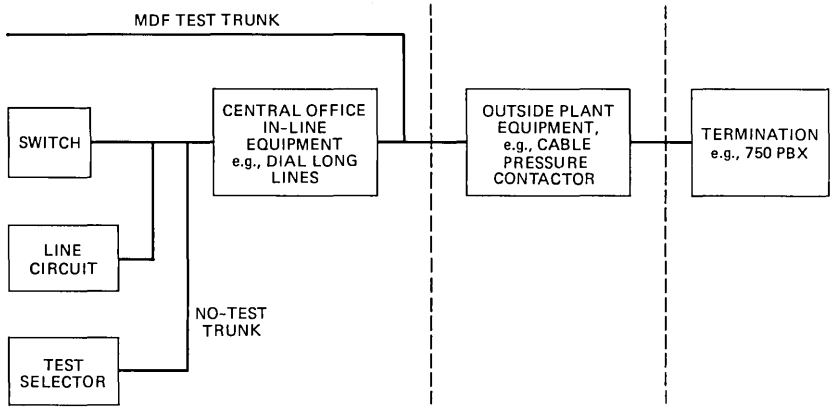


Fig. 1—Equipment affecting testing.

(In addition to equipment which is physically a part of the loop, there are sometimes other circuit features which affect test results. An example of such a "feature" is inward-only service in which dial tone is not provided, even if a tester were to try to draw it. The effects of a circuit feature are handled in essentially the same way as those of loop equipment as discussed in following sections.) Figure 1 is a block diagram of the equipment arrangement in a typical subscriber loop, showing the location of each category. By far the large majority of subscriber loops have only terminating equipment because no outside plant or central office equipment other than the switching machine is needed. The Test Selector with its No-Test Trunk* is the means of access to the circuit for most testing, although the Main Distributing Frame (MDF) test trunk may also be used for certain purposes.

If a test performed produces results characteristic of the loop plus the termination known to be at the end of the loop, then the loop is in good repair or has only minor faults. If the test fails to show results corresponding to the known termination, then most likely there is a fault to be evaluated. Recognition of the expected termination is, therefore, the key to loop testing, and this is true regardless of whether the testing is manual or performed by MLT. The presence of outside plant or central office equipment is important to the extent that it affects the test system's ability to recognize the termination or to detect and analyze a loop fault condition. The MLT system's means of acquiring the necessary loop equipment information will now be discussed.

* Reputedly, the "No-Test" Trunk is called that because the test probe is bridged onto the loop with "no test" performed by the switching machine to see if the line is in use by the customer. This is in contrast to "Regular" Test Trunks which will not bridge onto a busy line.

V. THE FLOW OF LOOP EQUIPMENT INFORMATION INTO THE MLT SYSTEM

The ultimate source of the loop equipment data is the Universal Service Order (USO) which provides information for installing the telephone service. The USO uses Field Identifiers (FIDS), FID code sets and Uniform Service Order Codes (USOCs), which are all codes from two to five characters long, to abbreviate the information about a customer's service and equipment required for the order. Field identifiers and code sets and USOCs indicate such things as the customer's name to be listed, the address, the cable and pair of wires to which the termination is to be made, the particular switching machine to be used, the type of termination, and so forth. There are many thousands of FIDS and USOCs which may be selected to make up a service order, and in fact there are several thousands of these which may be of significance to the MLT tests. Obviously, the customer's name is not required for analyzing test results, but the type of termination certainly is, and the USO is the vehicle for supplying that information.

It is important to recognize, however, that the MLT testing hardware is controlled by a small minicomputer which is intended to accept only a minimum of information about the line to be tested. The controller minicomputer cannot practically sort through all the data contained in a service order (or even the LMOS line record made from it) to glean the desired information. The process by which LMOS/MLT distills the USO information down to data useful for the testing algorithm and hardware is sketched in the following paragraphs.

The LMOS has a program called Automatic Line Record Update (ALRU). The function of ALRU is to read the service order and change or construct line records in the LMOS host computer. Because the service order is restricted to a certain format by the USO procedures, the resulting LMOS line record is highly formatted and amenable to automated searching. The data needed for MLT are retrieved by matching line record segments with a table of all the relevant FIDS, FID code sets, and USOCs. Each telephone company must maintain this table, because there is no one standard set of FIDS and USOCs that is applicable to all companies. (The USOCs, among other functions, are billing codes, and as such are subject to requirements placed by local public utility commissions.)

The host computer program which contains this table is called BINS (for binary search). When the host computer constructs a subset line record for downloading to the FE computers, the BINS program is run. The BINS program selects the FIDS and USOCs that were matched to its table, and assigns to each a simple numeric code which classifies the corresponding equipment. For example, all of the USOCs which correspond to a 701-type PBX may be given the code "20." There are

hundreds of USOCs which apply to various ways of billing a 701 PBX, but because they all mean the (electrically) identical termination, the single code "20" is sufficient. Every terminating-, outside-plant-, and central-office-type of equipment has its own code assigned in the BINS table.

The FE line record (miniline record) thus has incorporated within it a set of codes corresponding to the three types of loop equipment shown in Fig. 1. Presence of this set of codes provides for two functions to be performed during FE transactions. First, the codes, translated into English by a simple table in the FE software, give the tester or screener a concise description of the testing obstacles. The user does not have to memorize or look up any of the thousands of applicable USOCs or FIDs to know what is installed. Second, and really their primary function, the codes list the equipment information needed to facilitate testing with MLT.

Merely knowing which equipment is present is not sufficient to run an intelligent test algorithm. It is necessary to know what *electrical effect* that equipment will have on a test result. Many types of equipment have identical electrical effects; for example, the 701-type PBX looks electrically identical to the 740, 756, 757, and 770 PBXs. Specifically, the 701 PBX may be installed so as to show one of two idle terminations, either a characteristic dc "signature" or a thermistor termination identifiable by a special test built into the MLT hardware. Electrical characteristics defined in this way are directly usable by a test algorithm, whereas the identifying code "20" is one step removed from usability. It is possible to resolve the several hundred equipment types or codes into about 30 distinct electrical effects. These electrical effects are called the "attributes" of the equipment, and it is these attributes which are used by the test algorithm to configure and analyze the test series.

The test algorithm resides in the MLT controller, while the equipment codes are stored in the FE. When the FE processes a request for test, it translates the equipment codes from the miniline record into the attributes appropriate for all the equipment on the subscriber loop. It passes those attributes along with the test access instructions to the MLT controller. In this way, the controller algorithm is kept independent of changes in the types of loop equipment or in their observed electrical characteristics. Similarly, the individual FE line records, with correct listings of the installed equipment codes, need not be changed if it is found that the attributes of the equipment are not as was expected. It is necessary only to maintain the small FE table of equipment code-attribute mappings, and this is efficiently done on a Bell System-wide basis.

VI. THE TEST ALGORITHMS

To appreciate the sophistication of the MLT system as compared to other means of testing loops, it is necessary to describe the function of attributes more fully. Attributes are the electrical characteristics of the loop that indicate the expected results of measurements made on the loop. They are used by the controller to configure all tests, analyze the test results, and even to decide whether or not to run a given test. This ability to configure tests according to the expected attributes of the line is very important. It ensures that every line undergoes a set of tests custom-tailored to produce the most useful and accurate information for the maintenance center.

As an example, consider the attribute assigned to any PBX which may be installed in a ground start mode. Presence of this attribute causes the controller algorithm to select a dc test specifically designed to avoid alerting the attendant as an incoming call would. To keep from alerting the attendant, signals of certain voltages and of only one polarity may be used. In contrast, if the ground start PBX attribute is not present, the controller is free to make use of results from any previous tests in the series it is running. It will choose the optimum voltages and polarities for the most accurate measurements under the observed conditions.

Another benefit of a computer-driven test series is that tests which would produce confusing or misleading results can be avoided. To accomplish this, the test algorithm will check intermediate results where prudent. An example of such a process occurs when attempting to count the number of ringers on a residence line. This cannot be done accurately if there is a resistive fault worse than a certain threshold value on the line. Therefore, the MLT system will run a dc test first and examine its results to see if the ringer count test should be attempted. If the ringer count test is not done, the user will be informed of that and of the reason for not doing it.

VII. AVAILABLE ALGORITHM CHOICES

The MLT system is intended for users of varying knowledge and responsibilities within the maintenance operation. In the normal course of events, it may be used by a near-entry level clerk who receives reports of troubled lines from customers, and it may be used by highly experienced testers to diagnose complicated loop problems. Therefore, there are different capabilities provided for the different users.

The test performed when a trouble report is being received by a clerk (the RSA) is the most restricted one, in the sense that the clerk has no control over what type of algorithm is used. The algorithm is

a very general one, using the line record information as described above, and all the tests that are possible are actually performed. The objectives of this algorithm are to provide a simple statement of the loop problem for the RSA to use while making a commitment to the customer (while the customer is still on the line) and to provide for the screener, dispatcher, or any later user a comprehensive summary and detailed results of tests. This particular algorithm is known as the RSA/BOR series, because it is the series of tests that is initiated by the RSA and it produces the detailed test results on the BOR. The BOR is a paper record of all the data concerning a reported trouble, including much more than just test results, and it is produced *only* when a trouble is entered into the LMOS system.

A screener may wish to run another test after the initial trouble report processing. This is useful because customers frequently report trouble using the actual troubled lines. (Very little useful testing can be done on a line while it is in use because of the very low impedance of the station set during that time. The BOR test result summary for the line would be "SPEECH," and the detailed test results would be only "BUSY—SPEECH.") The screener would not want to use the trouble report transaction to initiate a test because to do so would cause another trouble report to be recorded by the LMOS trouble tracking system. Telephone company maintenance is evaluated in part by counting the number of trouble reports received, so to use the trouble report transaction would erroneously penalize the company rating. Therefore the MLT system is provided with an additional set of transaction options purely for testing under the TEST transaction. TEST transaction options do not generate trouble reports; they simply perform and analyze tests.

The "TEST" transaction includes the following options: FULL, VER, LOOP, RINGER, DIAL, MDFIN, and MDFOUT. FULL runs the RSA/BOR series to get a general diagnosis with output of detailed results. VER runs the same algorithm as FULL to output only the test summary or verification. LOOP is a fast algorithm intended to test only the loop part of the customer's circuit, omitting tests on such things as the line circuit. RINGER is a test to count the number of ringer-like terminations on the subscriber's loop. DIAL is a series of tests on the functioning of the rotary dial of the customer's station set. MDFIN and MDFOUT are algorithms used when manual test access is made at the main distributing frame (MDF), using access jacks wired for the purpose. To do these tests, the customer's line is actually opened at the point of access, so that the loop or switching machine part, respectively, of the line is physically disconnected. The disconnection may be necessary to isolate the cause of a particularly difficult problem. MDFIN is for testing in toward the switching machine from

the customer side of the MDF, and MDFOUT is for tests outward from the MDF. Of course, the line record information used in the test algorithm is altered to suit the conditions; e.g., for MDFOUT information about range extension devices in the central office is not applicable because the devices are disconnected by the access jack; therefore, the algorithm excludes that information.

VIII. BASIC ALGORITHM STRUCTURE

The more general algorithms, such as FULL and LOOP, have a co nmon basic structure. This structure has the following steps:

(*i*) Provide access to the loop
(*ii*) Do preliminary screening tests
(*iii*) Do 3-terminal tests and analysis
(*iv*) Select further tests and analyze.

It should be noted that there is some analysis performed *during* virtually every test, as well as afterward, to ensure that each measurement is, in fact, performed successfully and with optimum parameters.

8.1 Access

The MLT system provides access by dialing the telephone number of the line to be tested. Normally, this is adequate for any type of testing, MLT or otherwise, but there is a significant number of cases where it is not. Typically, these are for circuits which either do not have a directory number or do have one, but it does not give physical access to the loop. The MLT system, however, is set up to provide for access in these cases without requiring the user to do cross-referencing for an access number or to provide any special manual signalling which may be required by the switching machine.

An example of a nondirectory number line is the so-called "extra number" provided in crossbar switching machines for use in hunt groups. Hunt groups are series of lines in which only the first line need be dialed from another telephone. If the first line is busy, the switching machine automatically connects the caller to the second line. To conserve directory numbers, the second line is not listed in a directory, hence the term "extra number." Extra numbers, in fact, have more than the usual seven digits. A caller cannot directly access the extra number by dialing. Special signaling to the switching machine is required before it will accept more than seven digits, even from a test-access device. The MLT system does this signalling automatically. If, for example, the first number were 555-3000, and the second number were 555-3000-0001, test access signals to the switching machine for the second number would be automatically provided for the MLT user who typed in 555-3000XN0001.

Another case in which the telephone number is not directly usable

for test access is that of the local loop of an incoming Wide Area Telephone Service (WATS) line. In this case, the directory number might look like this: 800-555-3000. The 800-area code tells the switching machine that special logic is to be used to connect the caller. The actual number for test access, commonly called a plant test number, is likely to be quite different from 800-555-3000. (It could be, for example, 201-123-4567.) Because test access requests are treated differently from calls by the switching machine, MLT (or a manual test desk) must use the plant test number instead of the 800-number that was probably reported by the customer as having a problem. This is handled automatically by the MLT system because the cross-referencing of numbers is taken care of in setting up the MLT data base. The user types in the 800-number, and the MLT system automatically accesses via the plant test number.

Another access problem that is handled smoothly by the MLT system is the one in which it is necessary to ensure that the loop is not accessed accidentally. Some telephone companies have lines for which special permission from the customer is required for test access because of the extremely sensitive nature of the traffic on those lines. An example of a line like this might be an air controller circuit for a major airport; such a line might well be a dialable line which could be accessed if someone were to type in its number by mistake. In this case, the FE line record contains an entry which prevents the system from even accessing the line without the user's intentionally overriding the line record. Accidental access attempts cause a warning message to be displayed to the user, "TEST NOT MADE—PROTECTED SERVICE" for the RSA or "PROTECTED LINE—CUST PERMISSION REQ" for other users. The RSA cannot override the line record in the process of entering a trouble report; this capability is restricted to an option of the TEST transaction.

8.2 Preliminary tests

Preliminary tests perform several functions, the first of which is to determine if the line has a foreign voltage which is so high as to be hazardous to the test equipment or to personnel. If that is found, the algorithm terminates testing immediately with an explanatory message to the user. If no hazardous voltage is detected, the algorithm directs the hardware to check for a busy condition on the line. Busy lines should show a tip-to-ring voltage greater than about five volts. If a busy condition is found, a speech detection circuit is brought in to determine if the customer is talking on the line. This additional test is necessary because a dc voltage from an extraneous source or a short circuit on the pair could simulate the busy condition. If the line is found to have speech, the algorithm terminates the tests. At this point,

no signals that are detectable by the customer will have been put on the line, so the service is not impaired in any way.

A busy line which does not show detectable speech may simply be the result of the customer's leaving an extension off the hook. Such a condition is electrically detectable by MLT for the station set terminations found on the vast majority of residence and small business lines. If the line record indicates an appropriate termination type, the Receiver-Off-Hook test is performed. Successful detection of a receiver left off hook offers an immediate savings to the telephone company, which can simply inform the customer reporting the trouble, rather than make an expensive dispatch of a repair person to put the receiver back on the hook. A receiver-off-hook indication causes the algorithm to terminate testing.

Lines with a busy condition which show neither speech nor a detectable receiver off hook probably have a foreign dc voltage on them. Alternatively, they could have terminations like ground start PBXs, which have a voltage supply at the termination. These lines are candidates for further testing to define the problem. In addition, of course, the lines which do *not* show a busy condition (i.e., they are in a normal idle state) will receive further tests.

At this point it is possible that there is a voltage at a lower than hazardous level but still high enough to render the more sensitive diagnostic MLT tests inaccurate. Two further preliminary tests are done to check for this condition. The MLT hardware is directed by the algorithm to measure the open-circuit voltage of the line and then to measure the short-circuit noise current. Either measurement has an upper threshold which will cause the algorithm to terminate testing.

8.3 Three-terminal tests

When the test algorithm arrives at the point of doing the 3-terminal tests, the really sophisticated diagnosis of the customer's line problem begins. Three-terminal tests (the "3 terminals" are tip, ring, and ground) are performed by the test hardware in such a way[1] that the customer's line and its fault, if any, can be modeled as three Thevenin-equivalent circuit elements.

The Thevenin-equivalent circuit elements assumed for modeling the dc measurements are arranged in a delta configuration as shown in Fig. 2. The dc "signature" of a fault condition or of a piece of loop equipment is defined by a set of limits on the values of the circuit elements. For example, the 800-type PBX would have a signature as illustrated by Fig. 3. Direct current results that fit this signature would indicate that the loop was certainly continuous out to the PBX termination and that no dc faults worse than about 120,000 ohms existed at the time of the test. (It is possible for a fault of higher

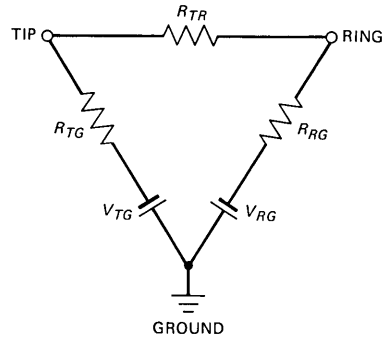


Fig. 2—Thevenin-equivalent circuit elements assumed for modeling dc measurements. Any or all of the five quantities R_{TR}, R_{TG}, R_{RG}, V_{TG}, and V_{RG} may be used to identify an equipment signature or fault condition.

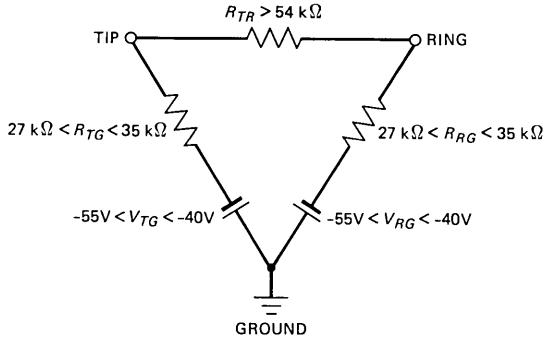


Fig. 3—Direct current Thevenin-equivalent circuit signature for the 800-type PBX idle termination.

resistance than 120,000 ohms to go undetected with this signature because there is a range of acceptable values necessary to allow for different loop resistances.) Note that five quantities must match the software values in order for the signature to be valid. This makes it very unlikely that a fault condition can simulate a good dc termination.

We can see the power of the 3-terminal model with the aid of a couple of examples. Referring again to Fig. 3, note that the circuit model shows a −48 volt supply in the PBX termination, just as it actually is. A 2-terminal measurement, such as that made from a test desk, would show *some fraction* of the −48 volt, depending upon the resistance of the loop and the termination itself. There are many different PBXs, with quite a range of resistances, hence widely varied 2-terminal signatures for the same voltage supply. However, there are only −48 and −24 volt supplies in PBX terminations; and because these show up explicitly in a 3-terminal Thevenin model of the measurements, they are well defined and extremely easy for the test algorithm (and the user) to recognize.

An even more obvious example of the advantages of the 3-terminal model is that of a customer's line which is in electrical contact with a second customer's line. This is known as a "cross" fault. It is characterized by the presence of the central office battery potential of the second customer on the first customer's tip or ring or both. The contact between the two lines may range in resistance from a "dead short" to a megohm or more. In a 2-terminal (unmodeled) measurement, this contact resistance and the resistance of the measuring device form a voltage divider, so that the resulting reading may be any fraction of the battery potential (−48 volt). High-resistance faults will, therefore, produce a negligible reading on a 2-terminal measuring instrument. The 3-terminal *model* produced by controller's computation, in contrast, separates the contact resistance of the fault from its foreign battery potential, displaying each explicitly, regardless of the value of the resistance. For example, the MLT results for a high-resistance cross might include "3 TERMINAL DC RESISTANCE = 600.00 KILOHMS T-G" and "3 TERMINAL DC VOLTAGE = −48.00 VOLTS T-G." The presence of the undesirable central office battery is immediately obvious. Lower computed voltages can result if more than two pairs are involved in the fault, but the cross is always more obvious using the 3-terminal model, because the voltage divider effect of the instrument is removed.

The Thevenin-equivalent circuit elements assumed for modeling the 24-Hz ac measurements are shown in Fig. 4. In similar fashion to the dc case, the ac "signature" of a fault condition or of a piece of loop equipment is defined by a set of limits on the values of the circuit elements. The signature for a typical 500-type set (the ordinary residence telephone) is illustrated in Fig. 5. Alternating current signatures are necessary because many terminations simply have no dc signature, i.e., they are open to dc. The 500-type set is one such termination. When it is on-hook, there is no dc path through the set between the

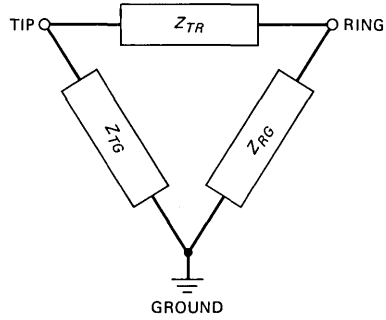


Fig. 4—Thevenin-equivalent circuit elements assumed for modeling ac measurements. Any or all of the six quantities (real and imaginary parts of the *Zs*) may be used to identify an equipment signature or fault condition.

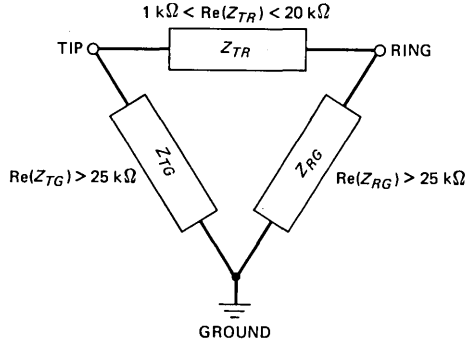


Fig. 5—Thevenin-equivalent circuit signature at 24 Hz for the 500-type residence telephone set. (The signature allows for several extensions.)

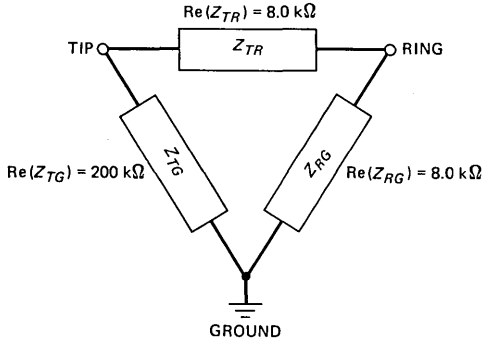


Fig. 6—Possible 24-Hz Thevenin-equivalent circuit signature for an improperly installed two-party line. The tip-to-ring and tip-to-ground impedances would be exchanged in a proper installation.

tip and ring conductors. However, because it is a resonant circuit at ringing frequencies (about 20 Hz), it has a readily detectable and characteristic real part impedance at the MLT measurement frequency of 24 Hz.

Here again, there are immediate advantages to the use of the 3-terminal model. Suppose an installer mistakenly connected the second party of a party line like a single-party service. The impedance of the first party would appear from ring to ground just as it was supposed to (see Fig. 6). The second party would appear on the tip-to-ring leg, cleanly separated from the other termination and obviously in the single party configuration. In addition to the measurement results, MLT analysis would provide the message "INVALID AC SIGNATURE" to highlight the improper termination configuration.

There is an additional type of termination which is not adequately defined by either the dc or ac 3-terminal tests. This is the termination in which thermistors are used as the sensing device for ringing signals.

The thermistors are capacitatively coupled to the tip and ring, hence appear open to the dc test. They present a rather high real-part impedance (25 to 100 kilohm) to the previously mentioned 24-Hz measurement technique, and this could be mistaken for a truly open circuit. Thermistors do, however, have a temperature dependent change of impedance when enough power is dissipated in them. The MLT system includes a special test capability to exploit this characteristic.

When the customer's line record indicates a type of termination which may have a thermistor in it, or when the dc and ac measurements show the possibility of a thermistor termination, the thermistor heating test is run. This test takes significantly longer than the dc and ac 3-terminal tests (which are run simultaneously), so it is not run by the algorithm unless it is needed. In this test, enough energy is supplied to the line to cause a measurable change of impedance in a thermistor station set, but the duration and magnitude of the signal is controlled so that it does not cause the set to ring. Thermistor terminations are found in many older PBXs and key systems, so it is particularly important that the ringing not be triggered by the test signals.

It will probably be noticed that no mention has been made of the use of the imaginary part of the 24-Hz measurement. The imaginary part of the measurement is particularly useful when an open circuit has been detected. Bell System cable leading from the central office to the customer's termination has a tightly controlled capacitance per unit length. The measured capacitance of the tip-ring, tip-ground, and ring-ground conductor pairs can be computed from the imaginary parts of the ac measurement results. Therefore, it is possible to determine the distance to the point where a wire may be broken from the ac measurement results. An obvious use of this measurement is to detect lines that are open in the central office by their very small computed length. A repair referred to the central office is much less expensive than a repair dispatched to the outside plant. The ability to distinguish open in (central office) from open out (a break outside the central office) gives the repair bureau direct guidance in the crucial dispatch decision, as well as assistance to the RSA in making a commitment to the customer. The MLT system has a built-in calibration method to ensure the accuracy of the open-in/open-out determination.

In cases of open out, the MLT system provides additional sophistication in the results reported to the user, again with the objective of facilitating accurate dispatch decisions. The imaginary parts of the tip-ground and the ring-ground measurements are compared for capacitive balance. If the balance is better than 99 percent (the two capacitances differ by less than 1 percent), then both conductors are probably broken at the same point. The messages to the user would be "OPEN

OUT" and "DISTANCE FROM C. O. = 10,400 FT." Experienced repair bureau screeners would know the approximate geographical location to dispatch outside plant repair craft to find the open at 10,400 ft from the central office.

If, however, the capacitive balance is between 95 percent and 99 percent, it is likely that only one wire in the pair is broken, and also that the break is quite close to the station set. In this case, the MLT results would say "OPEN OUT: NEAR DROP," and would in addition supply the information, "DISTANCE FROM C. O. = 10,400 FT" and "DISTANCE FROM STATION = 300 FT." If the capacitance of the tip conductor varies by more than about 5 percent from that of the ring, the MLT system will indicate (in addition to the distances to the open and station set in feet) that the open is in the cable, as opposed to the drop wire or station set. The user message would be "OPEN OUT: IN CABLE," enabling the repair bureau to assign responsibility for the repair directly to the cable repair craft, instead of station repair.

Similarly, MLT calculates the distance to a station set on a good line. This is useful as a sort of benchmark or as an aid to an inexperienced user. As mentioned above, MLT may detect an opened line for which the tip-ground and ring-ground capacitances are very nearly the same. Unless the total length of the line (hence its total capacitances) is known, it is not obvious from the measurement whether the line is broken at the end near the station set or somewhere back along the cable. Customer line record information does not include the length of the line to the station set. If the repair service does not have a good idea how long the line *should* be, the accurate distance to the open calculated by MLT may not be sufficient to determine whether cable or station repair craft should be dispatched. However, the repair service can test a good line to a neighboring telephone to get a length measurement for comparison.

The 3-terminal method of analysis has still another advantage over conventional techniques. This is the capability of doing ac analyses on lines which have dc faults. It frequently occurs that a dc-detectable fault (shorted or grounded conductors) is present along with an ac-detectable fault (open). This might be the case, for example, if the customer's drop wire were cut while digging for landscaping was being done. Here the effects of the resistive fault tend to overlay and mask the ac effects, making the open fault less detectable. However, the analysis provided in the MLT test algorithms is able to *isolate* the effects due to the simultaneously occurring but different faults. To the author's knowledge there is no test system, other than MLT, which provides this analysis. The MLT user is given a message like "RESISTIVE FAULT AND OPEN." This is a diagnosis which simply cannot be made by a tester using the local test desk. The local test desk can

detect only the short or ground in this situation, and this limitation makes the information given to the dispatched repair craft less helpful.

At this point in the series of tests, most of the measurements necessary for the final analysis of the loop's problems may have been done. Intermediate analysis will have provided the algorithm with the reasons to terminate testing or to select other tests to refine the analysis. In the case that further tests are needed, the data already at hand allow the algorithm to decide if a test that is otherwise desirable can produce useful results in the face of the known line conditions. One such useful test is the longitudinal balance test* used to detect line conditions which are conducive to induced noise.

The test algorithm will examine the results of both the dc and ac 3-terminal tests. If the ac test has detected a possible open circuit or a termination type that will have an intrinsically bad balance in the on-hook condition (e.g., two-party service with only one party assigned), the balance test would produce no useful information and would not be done. Similarly, a previously detected dc fault would almost certainly guarantee a balance problem, hence the balance test would add no information to the analysis. Finally, the balance test might not be appropriate, even if the measurement data permit it, because of loop equipment known to be on the customer's line. Bridge lifters, indicated in the LMOS line record, will inhibit the balance test because of their known deleterious effects on that particular test.

This rather rigorous screening process, using both measurement and line record data, is typical for all MLT tests. The process has two significant effects: first, it causes the test series to be kept to the efficient minimum necessary for complete analysis; and second and very important, it ensures that all tests are run under conditions which deliver reliable and definitive results for the user.

The 3-terminal tests mentioned so far have all been performed with the customer's central office line circuit removed from the loop. This permitted an effective analysis of any problems associated with the portion of the line from the central office out to the station set. The next step in the general purpose algorithms is to reconnect the line circuit to the loop and test for central office problems. The MLT system automatically signals the switching machine to reconnect the line circuit.

The first test done with the line circuit present is a dc 3-terminal test, modified to make accurate measurements on the low resistance (about 200 ohm) elements of the line circuit. Line record information is needed here because the results for loop start and ground start line

* The test is performed by applying a 200-Hz signal simultaneously from tip-to-ground and ring-to-ground (longitudinally) and measuring the signal produced tip-to-ring. The test hardware is described in some detail in Reference 1.

circuits are different. (A faulty loop start line circuit may look like a good ground start line circuit.)

Successful completion of the line circuit test leads to the dial-tone analysis test. This is a test in which the MLT hardware simulates the drawing of loop current by a customer's station set going off hook. The time it takes the switching machine to respond with dial tone is measured, and the MLT user is given a message indicating whether or not the time is within specifications. Upon detecting dial tone, MLT attempts to break it like the customer would while dialing, and reports the result of that test also. Here again, the line record data are required, because not all customer loops are supplied with dial tone. Some loops, so-called "inward only," permit the customer to receive calls but not to originate them. An example of this type would be a business telephone used only to receive purchase orders; the business might well want to proscribe outgoing calls which would interfere with the order-taking process. This type of service obviously does not need the switching functions associated with dialing. Inward-only service is specified in the service order, hence in the LMOS line record; consequently, the correct MLT attribute can be set for the line in question.

8.4 Additional tests in the general algorithms

The above general algorithm path is essentially complete with respect to establishing whether or not there is a fault, and if so, of what kind. There are two more tests which may be run. One serves to assist in the dispatch decision for a dc fault by examining its behavior with time. The other is intended to help the telephone company detect unauthorized additional terminating equipment the customer may have attached to the loop.

The refinement on the dc fault measurements aims at coping with a frequently observed physical phenomenon where a resistive fault is detected, but by the time repair craft arrive to fix the fault, it has disappeared. This commonly occurs when pulp-insulated cable has a small sheath leak, so that a rainy night may produce the detectable resistive ground or short. As the cable gets warmer during the day, however, it dries out and the craft dispatched to repair the fault can no longer detect the problem.

The MLT system includes a test, called the "Soak" test, which applies a potential of approximately 80 volts for a period of about 2.5 seconds. While the potential is being applied, the leakage resistance of the loop is sampled periodically to determine its time behavior. If the resistance goes up by more than a threshold amount, then it is likely to indicate one of the "disappearing" type faults. The MLT user is given a message which indicates the result: "SWINGING RESISTANCE—DRIED OUT." The dispatcher can then make an intelligent decision based on

the test results, the time of day, and local weather conditions. The expense of fruitless repair trips can be saved. Incidentally, the fact that a resistive fault is time-varying but does *not* dry out is also of aid to the dispatched craft, who can look for tree branches brushing drop wires or other conditions conducive to swinging faults. The MLT results would omit the "DRIED OUT" part of the results messages in this case.

Unauthorized terminal equipment is detected by the "Ringer count" test performed at the end of the test series on those customer lines which have ringer-like terminations, according to the LMOS line record. The number of station sets the customer is being billed for is counted in the line record and passed to the MLT algorithm for comparison with the measurement results. The discrepancy, if any, is noted in the detailed results for whatever action the telephone company policy requires. (In addition, the Ringer count test results are useful in detecting the type of line record error in which range extension equipment is installed but missing from the line record. Certain equipment types appear from the measurements to be thirty or forty ringers, an obvious impossibility.)

IX. ALGORITHM STRATEGIES FOR LINE RECORD INADEQUACIES

The preceding description has frequently pointed out the use of the LMOS line record information in the test algorithms. The line record is, of course, not perfect. It may be in error in data which affect the test algorithm; it may not be up to date because of the administrative delays inherent in service order processing; and in fact, it may sometimes simply not exist. Obviously, a practical test system must take account of these possibilities. The MLT system treats such situations by means of special sets of attributes supplied to the algorithms and by judicious use of the measurements obtained throughout the test series.

The most important principle in dealing with line record inadequacies is the fact that most fault conditions do not simulate valid termination signatures. This is so because the signatures tabulated in the MLT software always consist of at least three measurable parameters with severely restricted threshold values. It is simply very unlikely for a fault to fit all the requirements for a valid termination signature. The preceding paragraphs have shown how this fact facilitates the construction of an efficient and definitive algorithm. There is, in addition, a useful implication; that is, the algorithm becomes only slightly less efficient and precise if an additional valid termination signature which is *not* expected is allowed to be used. The algorithm still runs very little risk of a fault duplicating the additional termination signature, and the results supplied to the user should be the same as

with the more restricted signature set in a correspondingly high percentage of cases.

9.1 The case of a nonexistent line record

When the line record is nonexistent, for whatever reason, the logical extreme of the above principle is applied. The MLT algorithms are supplied with the attributes appropriate for every known standard termination tabulated in its software. In addition to these "dummy" attributes, the system also includes an attribute to let the algorithm (and eventually the user) know that line record information was not used. This provides in a very straightforward fashion for the same test algorithms to be used regardless of the situation.

The ability of the algorithms to detect all possible faults is degraded somewhat, but not as much as might at first be suspected. A known possibility is that the customer's termination equipment might be a type of PBX for which the valid signature is identical to that for another type with an open fault on one wire. The MLT results for this case would indicate a good line, because *one* of the valid termination signatures would have been matched. In practice, this sort of problem is quite rare; furthermore, if it occurs, the customer reporting the trouble is very unlikely to be convinced by the RSA that the line is good.

Because a multiplicity of signatures are accepted in the case of nonexistent records, the number of paths through the test algorithms is greatly expanded. The attribute which signals the lack of a line record is continually used to avoid closing off possibilities before the fault diagnosis is absolutely conclusive. Nonetheless, the analysis of measurements provided at every step of the test series soon narrows the possibilities to a tractable minimum.

One reason for a lack of a line record is the day or so delay in completing service order processing. This is the situation just described, in which the dummy attributes provide for correct algorithm analysis. The customer's line is installed and should be working, but the record system has not yet received the notification of completion. There is, however, another frequently occurring case where no line record would be available. This is the situation in which the line has been disconnected or was never installed. The test is attempted on a telephone number which is not at the time serving a customer. Such lines are said to be "on intercept." The incoming calls are intercepted and transferred to a recording which tells the caller that the line is not in use. Up to 30 percent of the available telephone numbers in a switching machine are commonly on intercept, so that if there has been an installation error, it is possible that a test may be run on an intercepted line.

To handle this case, the MLT system includes in the algorithms a signature test for intercept in all Bell System switching machines. Electromechanical switching machines require a minor modification to provide a recognizable dc signature. In electronic switching machines, a combination of the dc signature and a tone detection is used. This capability completely removes the line record dependence from the identication of an intercepted line.

The MLT intercept detection capability has become very useful in dealing with the type of installation problem in which, for example, a translation error has been input to an ESS machine. The customer's telephone will not supply dial tone, and no incoming calls will be received because they will be intercepted. The customer may very well not know that incoming calls are getting the intercept recording. Nonetheless, the RSA, while taking the customer's trouble report, will be informed by MLT that the line is on intercept. Thus, either the RSA or the screener can deal effectively with the problem without causing a futile outside dispatch. The RSB can offer a much more palatable commitment time to the customer, secure in the knowledge that the trouble can be resolved on the telephone company premises.

9.2 The case of an erroneous line record

It is possible for a telephone number to have associated with it the wrong line record, as sometimes happens when the line record for a previous customer still exists when a new customer on the same number is installed, but before the updated line record is available to the computer. In this case, the line record data are not valid, including those of concern to the testing system. A more likely case is that where the proper line record exists, but some of its data are incorrect because of formatting or typographical errors propagated from the service order.

It would be difficult to give a concise description of the strategies used by the MLT algorithms to deal with these problems. Perhaps an example will serve to illustrate a typical approach.

Here the line record is present and must be assumed to be correct by the algorithms at the beginning of the tests. However, the algorithms are adaptive, and this means that data can be in effect discarded during the running of the algorithms if the data are clearly erroneous.

Assume that the line record shows an ordinary residence telephone, but the telephone number has been newly assigned to a business with a small ground start PBX. Assume the test series are performed with the line idle.

Upon accessing the line, a busy indication would be detected because of the battery voltage in the PBX termination unit. The speech detection circuit would find no speech, so the algorithm would proceed

to call for the 3-terminal dc test. The results would be analyzed for the central office line-in-use signature because of the unexpected voltage. Since this would not match, and because the voltage is not a part of a residence telephone signature, the signature would be checked against the ground start PBX signatures, where a match would be found. Matching a valid dc signature would obviate the need to match the ac signature expected for the residence telephone. It would also preclude various other tests (longitudinal balance, for one) by means of their associated screening routines. Therefore, the algorithm would skip to the line circuit and dial tone tests and, upon completing them, terminate.

If the line were good, the results of the test would indicate that, but the discrepancy between the line record information and the measurements would be pointed out to the user. The test summary message would be "UNEXPECTED PBX TERMINATION," indicating the discovery of a good line with a valid termination and a discrepancy in the line record data. If there were a fault on the line, it would be detected via any of the possible paths in the algorithm, the path chosen depending upon the type of fault. A completely severed loop would obviously give no indication of the termination-type discrepancy, but the open fault would be correctly diagnosed. Simple resistive faults would be detected, and neither the dc nor ac termination signatures would match. The line record error is not so clear for these faults, so the message to the user would include the less specific statement "INVALID AC SIGNATURE."

9.3 The case of a line record with vague information

It is not always possible to abstract well-defined equipment information from the LMOS line record. Sometimes the customer has purchased a non-Bell System station set which is not in the software catalogue of terminations tabulated in the MLT system. Sometimes the termination USOC used in the service order is not yet included in the BINS table, so that no match can be found when the MLT information is being culled from the line record.

To cope with these situations, the BINS program includes a number of default procedures, so that the best available information is passed to the MLT algorithms. If the line record USOCS do not match anything in the BINS table, there still may be useful information in the record. Each line record is classified by its "class of service," which is a broad general category like "Residence," "PBX" or "Coin-Public." Some of these categories give no information about the terminating equipment. Residence, for example, covers a broad spectrum of possible equipment. Others, such as "PBX" and "Coin-public," do limit the possible terminations to more-or-less restrictive classes. The default proce-

dures, therefore, examine the class of service to provide the most likely termination data.

For a Residence line record, the default is to specify an "Uncatalogued Termination" code. This code, just like the more specific "Single Party" code, causes certain attributes to be supplied to the algorithms. The attributes include signatures for thermistors and the ringer count test and also an indicator for the presence of uncatalogued equipment. Similarly, for a "PBX" line record, the default code is "Uncatalogued PBX," causing all PBX-related attributes to be supplied to the algorithms. The uncatalogued equipment attribute is included here also. Presence of this attribute always causes the message "UNCATALOGUED EQUIPMENT" to be displayed, so that the user is aware that unusual results may be expected.

9.4 Overriding the line record

Section 8.1 mentioned overriding the line record where test access was prevented by an entry in the FE line record. The override capability provides an important flexibility to the MLT system. Previous sections have illustrated the intimate connection of the line record data and the testing process. Several problems associated with erroneous or missing line record information have been discussed. In many cases the test algorithms themselves are comprehensive enough to handle the problems. Nonetheless, it sometimes occurs that a particular line record problem has a drastic effect on the analysis of a trouble. When this happens, it is important to be able to "get rid" of the offending information to make better use of the MLT testing capabilities. This ability is provided by the "override line record" option of the TEST transaction.

The implementation of the override option is quite simple; when the option is chosen on the TEST mask by a screener or tester, the algorithms are provided with the same dummy attributes as if there were no line record at all. The line record equipment codes are ignored. The dummy attributes are "permissive" in that they cause the algorithm to accept any valid tabulated termination type on a good line. A further latitude is provided in that the set of dummy attributes contains *no* outside plant or central office equipment attributes. The reason for purposely omitting outside plant and central office equipment attributes is that those kinds of equipment always represent a testing *obstacle* to some degree. They are always in the loop "in the way" of the terminating equipment and thus tend to restrict the types of tests which can be successfully run to diagnose loop problems. A well-known example is the Dial Long Lines (DLL) equipment used in the central office to boost the ringing range of the switching machine. The DLLs are electrically opaque to MLT tests, so the algorithms

attempt to test only the line circuit and dial tone functions of DLL loop. A curtailment of the test process as severe as this is obviously to be avoided if useful information is to be gained from an override capability.

The DLL example also makes it clear why the override option is useful. Suppose the line record mistakenly showed a DLL, thus producing a very truncated MLT test. Simply choosing the override option of the TEST transaction allows the full gamut of the test series to be applied.

The override capability is also the means for accessing those previously mentioned very sensitive lines which have the line record "barrier" preventing accidental testing. Once the customer's permission has been received for testing, the powerful measurement and diagnostic capabilities of MLT are available through the simple device of overriding the line record. It should be noted that even if the test algorithms are not comprehensive enough to include every circuit configuration, they always provide the measurement results. The skilled user can, therefore, make use of MLT as a "meter" regardless of the type of circuit to which it is applied.

X. CONCLUSION

The MLT system is a comprehensive and sophisticated means of diagnosing loop problems in today's environment. Its improvements over the local test desk stem from the use of modern hardware and adaptive software. It is unique among loop testing devices in its integration of line record information with the testing algorithms. Extensive laboratory and field trial testing have helped to make it an extremely useful aid in the repair and maintenance of customer lines serviced by the Bell System.

XI. ACKNOWLEDGMENTS

I would like to acknowledge the fundamental work of Mr. C. E. Bowen, now with Mitre Corporation, who laid the foundation for the MLT algorithms. I would further like to thank Mr. J. E. Marowitz for guidance in structuring the algorithms and Mr. M. A. Botsakos for many helpful discussions and for keen and timely detection of flaws in the field application of the system. I have also drawn much on Mr. R. W. Vetter's knowledge of repair service operations and problems.

REFERENCES

1. O. B. Dale, T. W. Robinson, and E. J. Theriot, "Automated Repair Service Bureau: Mechanized Loop Testing Design," B.S.T.J., this issue.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Mechanized Loop Testing Design

By O. B. DALE, T. W. ROBINSON, and E. J. THERIOT

(Manuscript received December 2, 1980)

The Mechanized Loop Testing (MLT) system is a functional unit of the Automated Repair Service Bureau (ARSB) which tests and analyzes the condition of customer loops. The test results are used to verify trouble conditions, assist telephone company personnel in providing repair-commitment information to the customer, dispatch the appropriate repair craft, and reduce manual testing requirements. The MLT system design is distributed over three processing levels of the ARSB tree structure (host, front end, controller) to provide the necessary record utilization, data processing, loop analysis, and control of test equipment. The automatic access, monitoring, and testing of loops is performed by specially designed test equipment under the direct control of the controller. The MLT system provides a set of test series, each designed for specific applications. The test series are composed in real time as a function of the equipment thought to be present on the loop and the results of tests already performed to that point in the test sequence. This adaptive testing process has been implemented in hardware and software to provide an effective loop-testing system for most applications in the Bell System.

I. INTRODUCTION

The Mechanized Loop Testing (MLT) system is a major functional unit of the Automated Repair Service Bureau (ARSB) which automatically tests and analyzes the condition of customer loops. The tests are run at the time the customer reports a trouble, or at any time it is necessary to check the condition of the loop. Results of the tests and

a detailed loop analysis are generally available within 30 seconds from the time the request is initiated. The results are used to provide the customer with an accurate assessment of the trouble and to assist in establishing an appropriate repair commitment time. In addition, the results can be used to efficiently dispatch repair craft and reduce the manual testing requirements of the repair operation.[1,2]

The MLT system had its origins in a testing system developed in the early 1970s, known as the Line Status Verifier (LSV).[3] This system was a threshold-based testing system used by repair-center personnel to test a customer line in a rapid, automatic fashion with a simple console input request. Tests initiated at the time of customer contact reduced substantially subsequent testing by skilled repair personnel. The LSV was later integrated with the Loop Maintenance Operations System (LMOS) so that a test could be initiated from an LMOS computer terminal and the result made part of the LMOS trouble report record. While this generation of the ARSB provided significant economies, it was clear that LMOS, by virtue of its comprehensive data base and computing power, provided the potential for much more sophisticated loop testing. For example, the electrical characteristics of the customer's station, loop, and central office equipment can be derived from the LMOS data and used to direct tests in an adaptive fashion and provide a comprehensive analysis in real time.[4] It was also clear that the LSV was not an appropriate testing vehicle since its hard-wired logic and threshold testing capabilities were not well-matched to the adaptive testing techniques and sophisticated analyses envisioned. Hence, a new testing system was necessary to take full advantage of the LMOS potential and provide additional benefits.

The development of these expanded testing capabilities occurred in two logical post-LSV phases. The first phase, representing the second generation of automated testing and known as the Automatic Line Verification system, developed the data-communication structure, modular physical design, loop access and automatic monitoring capabilities, computer-driven self-maintenance features, and the basic LMOS record-utilization techniques. The third-generation testing system, known as MLT, added a more comprehensive testing package and associated control software which used records more extensively. In this paper, we shall discuss only the resulting MLT system.

The MLT system was intended to significantly reduce but not eliminate the need for manual testing facilities such as local test desks.[1] Since manual testing facilities were already present in repair bureaus, they could be used as backup testing devices when the MLT system experienced temporary outages. In addition, some testing needs, such as interactive testing between field and inside repair craft, coin station testing, and four- and eight-party line testing seemed to be best left to

the manual testing facilities because they were not needed frequently and, therefore, mechanizing them did not appear to be cost effective.

The purpose of this article is to describe the MLT system design. Most of the article is devoted to the hardware/software design, but it should be noted that an equally important piece of the design, namely, the personnel subsystem, is covered, at least sparingly, elsewhere in this issue.[5,6] The application of human factors engineering for the MLT project by a talented group of psychologists was a significant and highly valued contribution to the design and introduction of the system. Similarly, we felt it somehow unfair to pay only brief attention to other areas of the project including the elaborate self-maintenance design of the MLT system, but, again, brevity won out.

II. BACKGROUND

A number of requirements and assumptions influenced the design of the MLT system and led to an architecture that distributes the MLT software functions over three processors. Two of the processors have as their primary function the implementation of the LMOS system with which MLT must interact; the third processor is dedicated to the MLT hardware control task.

2.1 Operational users

The MLT system provides operational data to two types of users: Repair Service Attendants (RSA), who are in contact with the customer, and Repair Service Bureau (RSB) personnel, who analyze the trouble and dispatch repair craft. The needs of these two users are similar, but not identical.

The RSA needs a test summary that provides insight to the reported problem in a global way. Is a trouble confirmed? Is it a central office trouble, a loop trouble, a station trouble? The test has to be performed automatically when the trouble report is taken and the results are needed promptly so that an appropriate repair commitment can be given to the customer.

The RSB needs complete test results, preferably included with the trouble ticket that is automatically produced when the RSA takes the trouble report. The RSB also needs the ability to perform tests upon demand, sometimes while the repair craft is at the location of the trouble. Thus, the RSB needs a list of the available tests, some designed to duplicate the comprehensive tests performed when a trouble is taken, some tailored to providing data on only a subset of all possible problems but at smaller costs of system resources and with shorter run times. A total of 11 different series of tests were determined to be required. Examples of these limited test series include: ROTARYDIAL to test the speed and make-break ratio of a rotary dial, RINGER to

simply count the number of ringers, LINECKT to test only the office line circuit, and LOOP to test only the loop and on-hook station.

For both users, it is necessary to interpret the test results in light of expected office, line, and station equipment (as gleaned from LMOS line records) and to be tolerant of incorrect or absent equipment records.[4] These requirements dictate a close tie to LMOS in the initiating of tests automatically, in getting test results onto the trouble ticket, and in the use of equipment data to define expected test results.

2.2 Support users

The MLT system has to provide for two additional types of users concerned with support functions. The MLT Administrator is concerned with MLT maintenance and the MLT Data Manager is concerned with MLT software data initialization and integrity.

The MLT Administrator requires tools to perform maintenance functions on the MLT system and to change various system parameters and thresholds so that the performance can be tailored to a particular test environment. Among the maintenance functions required is the ability to calibrate both MLT equipment and test trunks so that systematic errors can be subtracted from test results. Other functions provide the ability to perform tests designed to verify the general "sanity" of the system as a preventative maintenance tool or to perform detailed diagnostic testing of circuitry when specific hardware faults are known or suspected. Control functions include the ability to change the test-result decision thresholds of acceptable levels of foreign voltage and loop unbalance and the ability to take specific equipment out of service for maintenance. Eleven different transactions or commands were identified for maintenance and control functions.

The MLT Data Manager requires tools to create and maintain the various data bases related to MLT. In particular, a data base is used to define the equipment configuration of each set of MLT test hardware. This data base includes not only the specific quantities of optional testing hardware and test trunks that are present, but it also provides information concerning which test trunks are used to test which lines, the type of switching machine involved (step-by-step, crossbar, ESS), and the calibration constants for test trunks and MLT testing hardware. Some of these data are considered static, that is, seldom changed and then only by an appropriate user. The configuration and status data are of this type. Other data are considered dynamic, that is, changed by software; calibration data are of this type.

The requirements for the two support users are not served by a single software structure. The needs of the Administrator must be met by real-time software that can interact with the testing process when necessary.

Since the MLT files overlap the LMOS files, and since similar file creation and maintenance problems are solved by LMOS via off-line processes, the same approach was used for MLT. We will not discuss these off-line processes further except to note that one of the considerations was to design procedures for the creation and maintenance of the MLT files to be similar to the procedures used for LMOS files.

2.3 Hardware imposed requirements

Since the test hardware contained no processing capability, the software had to control the hardware on a step-by-step basis. The software had to identify the proper hardware resources for a particular use, allocate the resources, and cause the equipment to perform a particular sequence of steps on a time-critical schedule. This suggested that the hardware controller tasks should be on a dedicated machine to meet the real-time constraints.

III. ARCHITECTURE AND FUNCTIONAL DISTRIBUTION

The MLT system can be thought of as having five fundamental tasks: accessing the loop, monitoring the loop to ensure that testing can proceed, testing the loop, analyzing the test results, and presenting the results in an easily understood fashion. Loops are accessed via "no-test" trunks which connect the test system to the switching machine. Test trunks are switched onto the desired loop by commands sent from the test system to the switching machine. The test system then automatically monitors the loop for hazardous and/or busy conditions to determine if the testing process can proceed. The access and monitoring processes are performed by the Loop Testing Frame (LTF) under the direction of the MLT Controller (see Fig. 1). Assuming that testing can proceed, the Controller directs the LTF to connect an MLT Measurement Module (MMM) to the test trunk (and hence the loop) and then commands the MMM to perform a series of tests on the loop based on the central office, loop, and station equipment indicated by the LMOS data base. Tests proceed in an adaptive fashion, taking advantage of the increased knowledge of the loop as each test is run.[4] The Controller analyzes the test results and decides when to terminate the testing process and communicate the results to the front-end processor, where they are presented to the user. Typical results might look like those shown in Fig. 2.

The provision of these five basic tasks is accomplished by functions distributed throughout the ARSB system.

3.1 ARSB host processor

The host processor is used by the ARSB to maintain historical data on closed trouble reports and to maintain extensive line record infor-

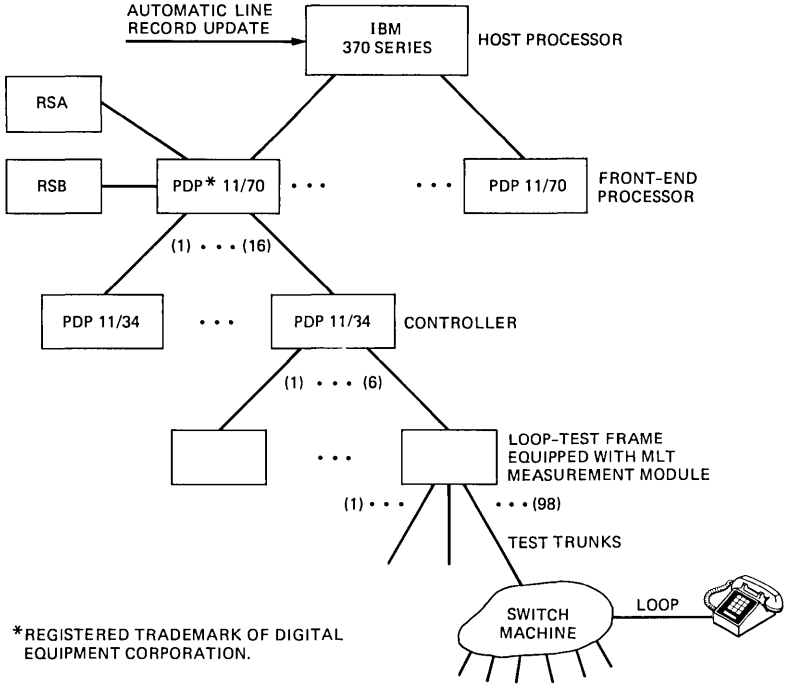


Fig. 1—Automated Repair Service Bureau architecture.

mation on each line. A subset of this line record (called a mini-line record) is duplicated in a front-end (FE) processor for fast access by RSB craft and to provide the ability to support basic operations when the link to the host fails. For MLT, the host is used to scan line records, at the time they are established, for the presence of installed equipment that could influence testability or test results. These data are formatted into a six-byte field for the mini-line record. Also, the host has to expand coded test results in real time when they are received from the FE into an english narrative in accordance with a predefined algorithm.

3.2 ARBS front-end processor

The FE is used by the ARSB to process trouble reports and to interface with RSA and RSB craft in real time.[8] The LMOS software on the FE maintains the mini-line record information. It also supports the numerous CRT terminals for trouble entry and printers to provide trouble tickets—called Basic Output Reports (BOR)—to the RSB craft.

Certain files maintained by the LMOS software had to be expanded to provide MLT information. For example, the file which provided data on the NPA-NNX (area code-exchange code) combinations adminis-

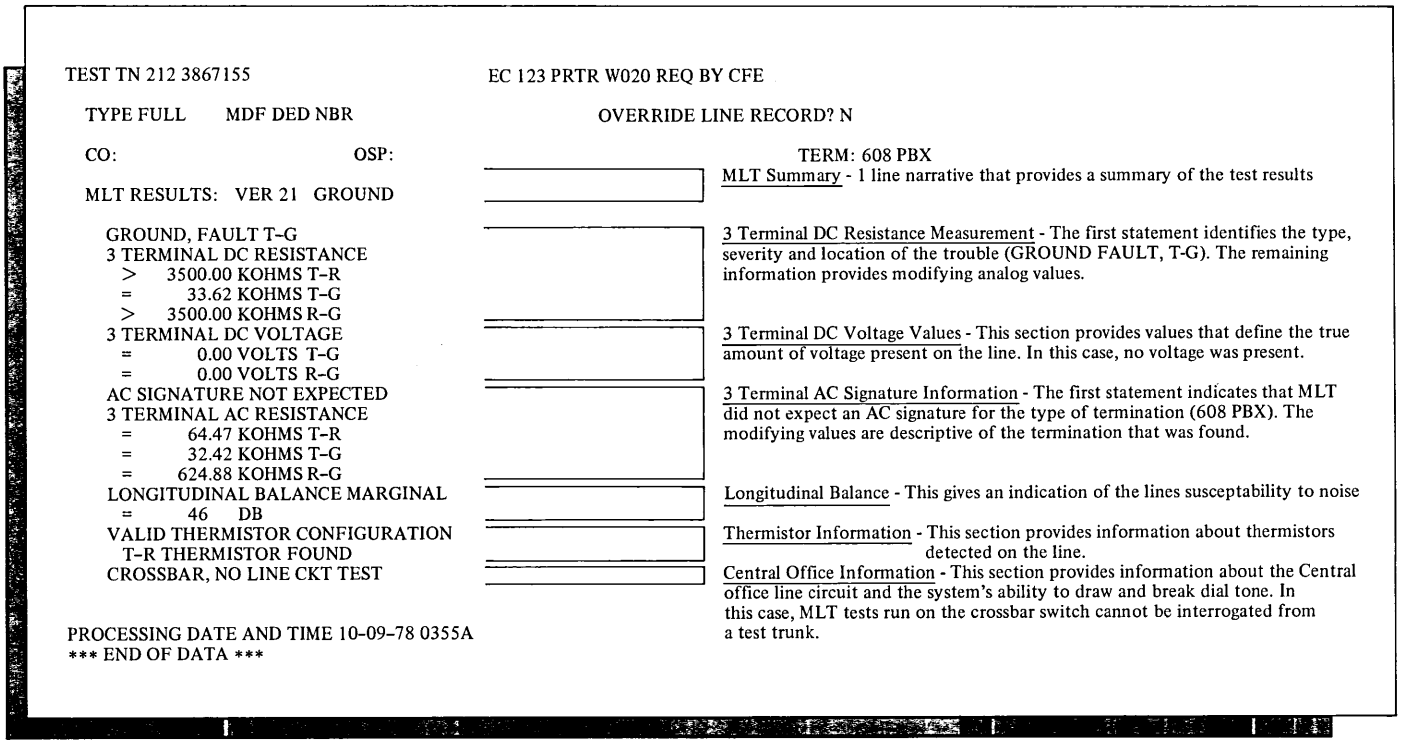


Fig. 2—Typical MLT test results.

tered by the FE had to be enhanced to include an indication of MLT testability for the NPA-NNX and which MLT equipment could be used. The LMOS software also had to be expanded to include a provision to schedule the MLT process each time a trouble was taken on a line that was testable by MLT. The scheduling procedure included the passing of the identification of the terminal on which the trouble was taken, pertinent information from the mini-line record (such as the six bytes of test-affecting data), and information from the NPA-NNX file.

The MLT software on the FE provides several functions:

(*i*) It provides an interface to the LMOS software so that testing can be scheduled when a trouble report is being taken. It also translates the MLT bytes into attributes that describe the electrical characteristics of the loop, termination, and central office equipment for use by the Controller.

(*ii*) It provides an interface to RSB craft so that subsequent or alternate testing of a line can be performed when requested.

(*iii*) It provides an interface to the MLT administrator so that MLT maintenance requests can be serviced.

(*iv*) It manages the testing process by controlling the throughput to the various MLT Controllers and by preventing lost or delayed requests. In doing so, it resolves conflicts between the automatic tests requested as a result of trouble entry and tests requested by the RSB or MLT Administrator.

(*v*) It returns data to the correct requester in a format appropriate to the request.

(*vi*) It maintains data files unique to MLT needs. For example, one file contains the particular LTF configurations served by the MLT Controller as well as related calibration data.

(*vii*) It stores the Controller software and associated tables, and loads the Controllers when the system is initialized or when certain trouble conditions arise at the Controllers.

3.3 MLT Controller

The MLT Controller is given as large a part of the MLT software task as possible. This is done partly to minimize the load on the FE and partly to ensure that the LMOS software is isolated from MLT software changes. The Controller software is responsible for allocating MLT hardware resources, controlling the hardware to perform a specific test, interpreting test results in light of expected results, adjusting the sequence of tests in accordance with expected test results, and determining the format of the report.

Because it was originally expected that the Controller would be placed in non-EDP environments, the Controller hardware was kept simple. No colocated off-line storage devices or peripherals are used

and all software is kept in 256 Kbytes of main memory. The Controller software and associated tables are stored in the FE and down-loaded over a data link to the Controller at the time of system initialization. Although subsequent deployment strategies have tended to centralize Controllers in controlled environments, the simplicity of the Controller hardware configuration has provided very high reliability and allowed low-cost backup schemes.

The operating system used in the Controller is the same Bell Operating System (BOS) used in the FE, although only a subset of the BOS features are provided in the Controller.[7] In particular, BOS provides a software driver for interfacing to multiple loop testing frames. The driver links the testing hardware and application software in real time so that data can be sent to and from an LTF in a serial format.

Similarly, communication between the FE and Controller is managed by the Communication Control Manager (CCM)[8] software on the FE and a subset of CCM, known as MLTCCM, on the Controller. The communication protocol used was chosen to be a subset of the IBM bisync protocol used between the host and FE processors to avoid the creation of a new protocol.

3.4 Loop Testing Frame

The Loop Testing Frame (LTF), equipped with MLT measurement modules, carries out the access, monitoring, and test functions under the command of the Controller (Figs. 3 and 4). A data link between the Controller and the Communication Control Circuit (CCC) of the LTF provides the data communication facility. No-test trunks between the Trunk Access Switch (TAS) of the LTF and the switching machines provide the metallic test path to the loop. Commands from the Controller are decoded by the CCC, and control information is passed to the appropriate LTF subsystem. Similarly, data from each subsystem are passed to the CCC which appends the output address and transmits the data to the Controller for interpretation.

The LTF was envisioned as an area-based test unit, i.e., a single system deployed on a wide-area basis and serving many central office switching machines, to share the common testing facilities over as many lines as possible. Since interactive tests were to be relegated to existing manual testing facilities, the most lengthy test series run by MLT on a loop was estimated to take less than 15 seconds of actual testing time. Hence, an area-based, high-usage, low-holding-time system was conceived. The LTF was designed to serve up to a maximum of 98 no-test trunks with nine test ports (or nine simultaneous access/monitor/test operations) for a concentration ratio of about 11:1. Traffic estimates suggested that this was adequate to cover anticipated testing

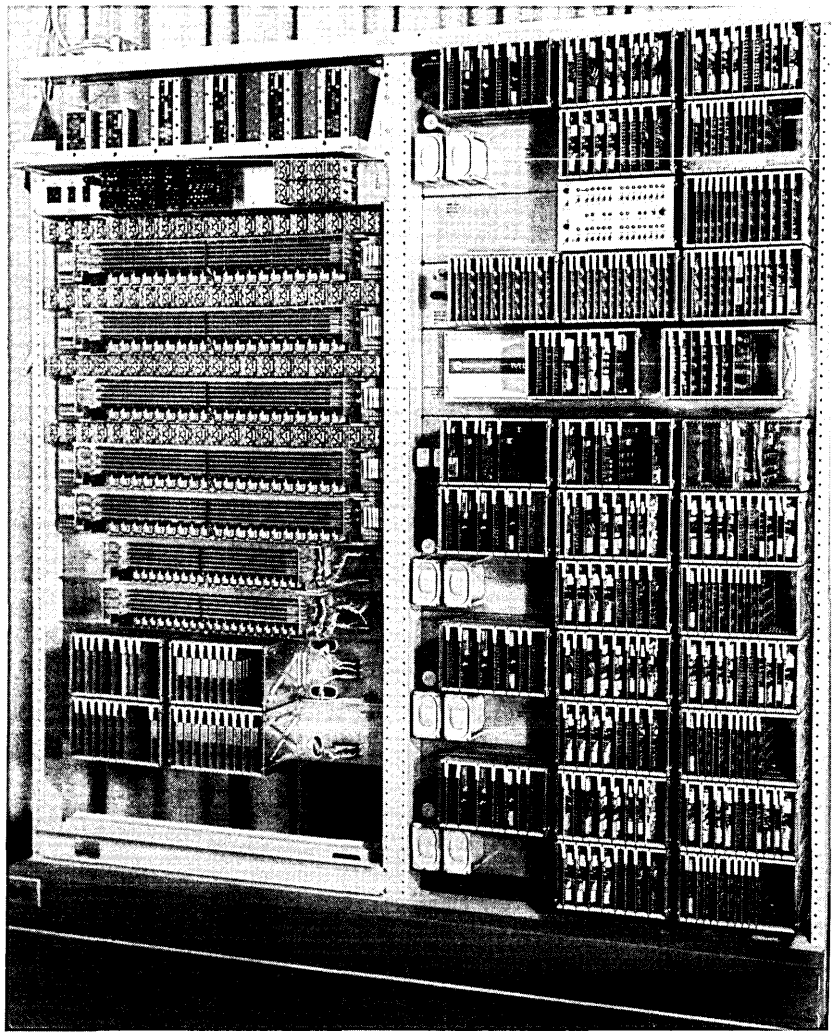
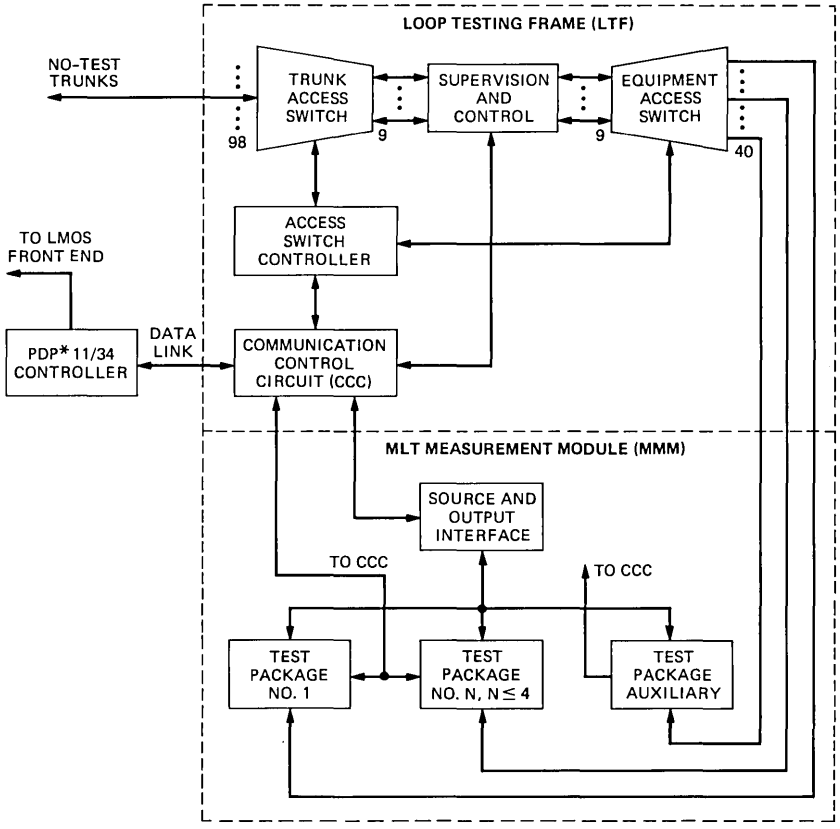


Fig. 3—Loop Testing Frame equipped with four MLT measurement modules.

traffic for more than 250,000 lines, which was expected to at least equal the capacity of the Controller.

The LTF serving area was determined not only by the demographics but also by the physical limitations of metallic testing. Beyond about 3,000 ohms of loop resistance or 100,000 feet if 19-gauge cable is used, it becomes increasingly difficult to differentiate between open loops and loops terminated with station set ringers or the equivalent. Since differentiation between open and terminated loops is essential, 3,000 ohms essentially defined the LTF serving area. This was a reasonable



*REGISTERED TRADEMARK OF DIGITAL EQUIPMENT CORPORATION.

Fig. 4—Schematic of loop testing frame equipped with an MLT measurement module.

limitation since the signaling range for most no-test trunks is in the neighborhood of 1,500 ohms, and most loops are also less than 1,500 ohms. Hence, the LTF serving area accommodated most loop lengths with maximum-length no-test trunks, thereby supporting rather neatly the area-based testing concept.

3.5 Configuration

Each FE can serve up to 16 Controllers. Each Controller can serve up to six LTFs with no more than 40 NNXs or 120 no-test trunks. The Controller limits are primarily based on table limitations, but a ratio of about three no-test trunks per NNX has proven to be the proper average for MLT testing traffic. The LTFs are connected by data links to the Controller and therefore can be located in central offices in a

pattern that ensures optimum coverage relative to the 3,000-ohm LTF serving area.

IV. SOFTWARE DESIGN

In this section the design of the software that executes on the Controller is described in some detail. First, we define three terms:

(*i*) LTF Communication—The protocol used to communicate between the Controller and the LTF was referred to briefly in an earlier section. In particular, a given transmission "packet" can be a variable number of words in length. Each word consists of two parts, an address and data. In the direction from the Controller to the LTF, the protocol includes a bit to indicate if the word is the last one of a "packet." One or more packets of data interchange may be necessary to set up or perform the most basic hardware action. In the direction to the Controller, the data is completely asynchronous. The Controller must be capable of accepting data at the peak rate determined by the serial line speed and must buffer these data until they can be processed.

(*ii*) Test Sequence—A set of hardware/software interactions that accomplish a single characterization, such as a count of ringers, a dc or ac Thevenin equivalent circuit, etc. Twenty-three separate test sequences are provided by MLT.[4]

(*iii*) Test Series—A set of test sequences that provide a complete characterization of a line. Obviously, the requested type of series influences the specific sequences that are used; for example, the RINGER series (designed to be a high-speed, low-cost test to count ringers) does not include sequences that make measurements on the switching office line circuits. But, the particular sequences that are used—even the order of the sequences—can be influenced both by the results of previous test sequences and the expected results from examination of the records.[4]

The operating system and MLT software processes are organized as depicted in Fig. 5. Their functions are described below.

The *Driver*, which is part of the operating system, interfaces with the Frame Communication Manager (FCM) and to the hardware circuitry that implements the serial interface to the LTFs. The driver is provided data for transmission that has been formatted into the proper protocol. It collects data from the various hardware circuits on a character-interrupt basis and inserts the data into buffers that are emptied by the FCM process when cpu time permits. The driver serves the data distribution and collection function to the LTFs and permits the MLT software to be largely unaware of the existence of more than one LTF.

The *Frame Communication Manager* interfaces to the driver and to the TSP and SUP (supervisor) processes that contain application

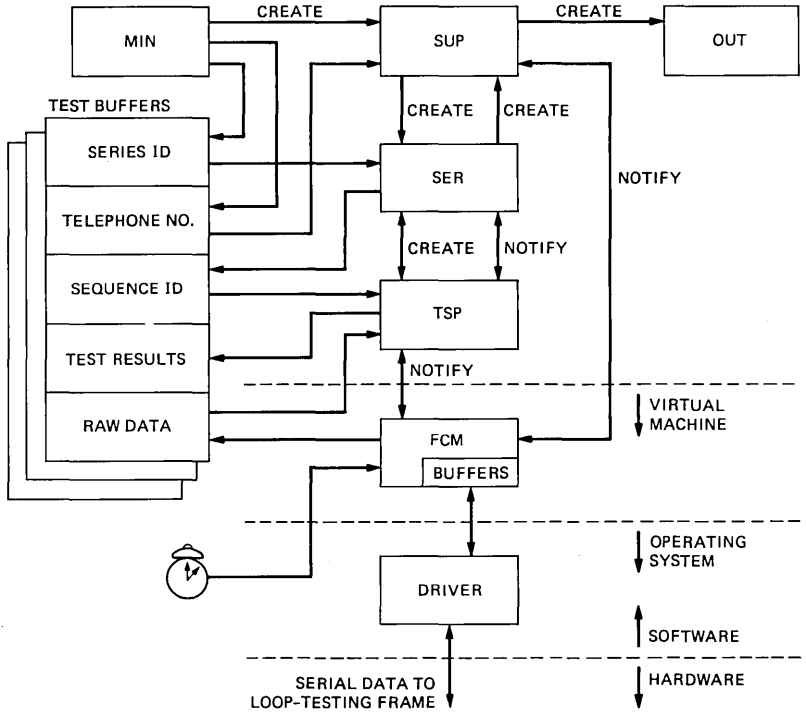


Fig. 5—MLT controller software structure.

software. The FCM process performs a service that is similar to the CCM process discussed earlier. The FCM can be thought of as an extension of the operating system in that it provides a virtual machine environment for the application software. With FCM, the processes that control a single test series have no knowledge of the existence of other processes executing other tests concurrently. The FCM accepts primitive commands to be sent to the LTFs, implements the protocol, collects data and sends it to the proper user process, and generally gives each user process the appearance of being the only process requesting data from the LTF.

The FCM process is created at initialization time, and it never terminates. It is awakened by either of two types of events. User processes, knowing that they have data to send to the LTF, can awaken the FCM process through BOS via a "notify" mechanism. FCM can also be awakened by an alarm mechanism, also managed by BOS. Each time that FCM is awakened, it checks whether data are available in the buffers that are filled by the driver. If so, it collects the data into complete messages and provides them to the user processes by writing the data into a previously agreed upon area of the test buffer. Normally,

the user process is waiting for either these data or for a timeout time. The FCM, seeing that the data are available (or that the time-out occurred), awakens the user process that owns the test buffer (again via BOS). Also, each time that the FCM is awakened, it searches through the test buffers to see if any process has data ready to be sent to an LTF. Data to be sent are left in a previously agreed upon area in the test buffer and a ready flag is set in the buffer. Finally, FCM cancels any previous alarm and sets a new alarm call to be awakened within a short time.

The selection of the sleep time was an interesting problem. At one extreme, FCM could set the alarm for such a short time that little processing resources are available to other processes. At the other extreme, FCM could set the alarm for such a long time that the data collected by the driver would be utilized too slowly and would overflow the existing buffers. An aspect of this design is that the alarm feature is used less frequently when the Controller is heavily loaded with tests since the notify mechanism awakens FCM more frequently. The alarm feature is required to collect data when only one test is underway in the Controller. Under this condition, it is desirable not to wait too long before looking for returned data as this would unnecessarily delay the testing process. The optimum sleep time was determined experimentally to be in the range of 100 milliseconds.

The *Test Supervisor* (TSP) process is a child process to the Series Control (SER) and implements the test sequences. That is, SER, when it first determines that testing is required, creates a child TSP process (via a call to BOS) and passes to TSP the address of the test buffer that is owned by SER. At this point, the child TSP process inherits the test buffer. When a requested test sequence is completed, TSP notifies its parent process and asks to have its execution suspended. Each time that SER wants another test sequence executed, the name of the sequence is placed in an agreed-upon place in the test buffer and TSP is notified. Each time that TSP performs a sequence, it puts the accumulated data in the test buffer in two areas. Binary results (yes-no) are indicated by setting bits in a bit-field.* Analog results are placed in the test buffer in a format appropriate to the result. For example, voltages or currents are kept in floating-point format, whereas the number of ringers is stored as an integer. The TSP is finally terminated by SER when no additional testing is required.

The SER process interfaces to TSP and SUP, and implements the test series. SER is a child process to SUP, but SUP terminates after SER is created and gains control of the test buffer. (In principle, it would not

* Examples of binary results are: bit 66, if set, means that the dc signature looks like a PBX; bit 25, if set, means BUSY SPEECH; etc.

be necessary for SUP to terminate after creating process SER, but the arrangement described is used to minimize the number of processes that must be active simultaneously, since each active process entails additional overhead.)

The SER creates child process TSP; SER and TSP take turns being active as discussed above. When SER has implemented enough of a test series to determine that "early" results to the RSA are required, process SUP is scheduled to send the results to the FE. The SER waits for SUP to terminate and then continues to the point that testing is completed. Then SER terminates the child process TSP, schedules process SUP, waits for SUP to gain control of the buffer, and then terminates.

The SUP process interfaces to MLTCCM and to SER. SUP provides general control over the test function, gains access to the line under test, and formats reports to the FE.

V. THE TESTING HARDWARE DESIGN

As previously mentioned, the testing hardware (Figs. 3 and 4) can be divided into two major subsystems: the LTF and the MMM. The LTF controls access to customer loops via no-test trunks and determines the busy/idle status of the loop. The MMM performs tests on idle loops for fault characterization.

Two interfaces to the LTF exist. The CCC provides the interface to the Controller. The TAS provides the interface to customer loops via no-test trunks to the central office switch.

The CCC serves as a multiplexer and demultiplexer for data to and from the Controller. Transmission is via a four-wire circuit using full duplex asynchronous serial data rates of 1200 or 2400 baud. The 1200-baud rate uses 202-type data sets for installations where the Controller and LTF are separated by more than 1500 feet. Within 1500 ft the transmisssion can be set to 2400 baud using optically isolated current loops.

Data from the Controller consist of an address field and a control-signal field. The CCC decodes the address and passes the control signal to the appropriate subassembly of the LTF/MMM. Data from each subassembly are digitally encoded by the subassembly and passed to the CCC which adds the output address to the data and transmits them to the Controller for interpretation.

The Access Switch Controller (ASC) receives control signals from the Controller via the CCC and provides signals to the TAS and the Equipment Access Switch (EAS). These signals select, hold, and release the crosspoints of these two switching matrices.

The TAS serves as a concentrator and permits the connection of up to 98 no-test trunks to one of nine test ports. The test ports are routed

through the Supervisor and Control Circuit (SCC) to the EAS. The EAS permits the connection of any test port to either a dialer, busy detector, or MMM.

The SCC provides the control and monitor functions required to access a customer loop via a no-test trunk. Loop access includes the functions of sharing trunks with the local test desk, sleeve-lead supervision, ring-tip supervision, dialing, and busy detection. In addition to access, the circuit provides for hazardous potential detection and interruption of the test path to prevent equipment damage. Dialing circuits are provided for both dial pulse (DP) or multifrequency (MF) signaling.

Once the loop has been dialed and the sleeve lead manipulated to effect a cut through to the loop, a busy test is made. The busy test provides outputs of idle, switching-machine overflow, dc busy (battery ring to tip), and speech busy. If the loop is idle, the test port, and therefore the loop under test, is transferred to the MMM for testing. If the loop is other than idle, access to the loop is dropped and the loop status is reported to the Controller.

5.1 MLT measurement module description and operation

The MMM provides the loop-testing capability for the MLT hardware and consists of three major sections: the Source and Output Interface (SOI), the Test Package (TP) and the Test Package Auxiliary (TPA).

The SOI performs a service function for the TPs and TPA by providing clock frequencies for the digital circuits, precision leveled frequencies for analog sources, and a precision dc reference for analog sources. The SOI also contains output encoding circuits for the analog measured voltages generated by TPs and the TPA during loop tests. The SOI pools the TPs and TPA for outputs and, if present, they are connected to the SOI encoding circuits. The encoding circuits include a 12-bit analog-to-digital (A/D) converter, an auto ranging amplifier required to present properly leveled signals to the A/D, and a polarity sensor. The digitally encoded output measurement is temporarily stored in a RAM buffer along with its unique output address. The address is determined by which TP or TPA circuits are being serviced by the A/D. Upon receipt of a poll, the output data is passed to the CCC for transmission to the Controller.

The SOI also contains sources and terminations which are used to isolate hardware failures to particular circuit packs. This capability is controlled by the Sanity and Diagnostic software modules in the Controller (Section 2.3).

The TP (Fig. 6) is a multipurpose test circuit which is configured for a particular test by commands from the Controller. The TP is the heart

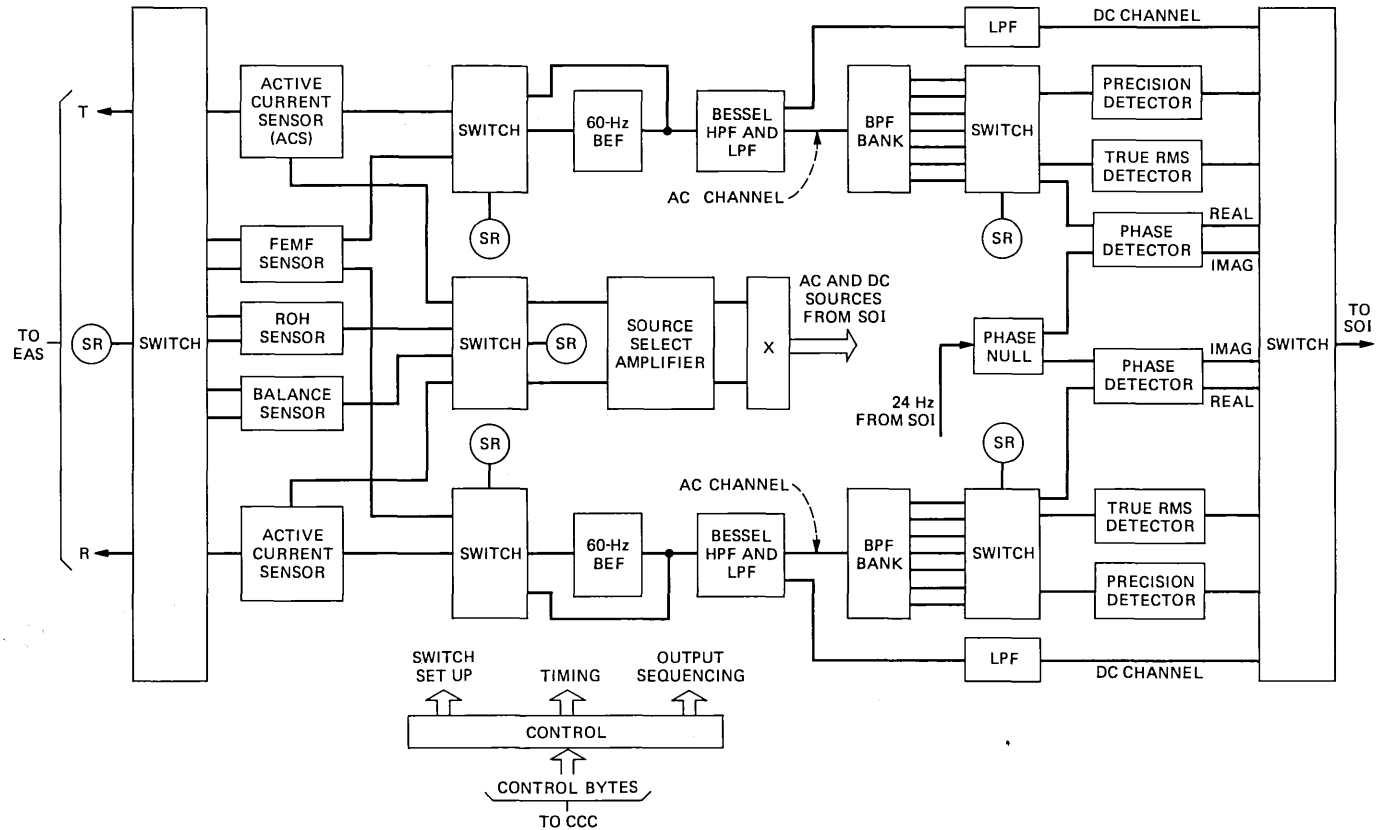


Fig. 6—Schematic of MLT measurement module.

of the MLT hardware. It can be divided into a test section, through which all analog test signals flow, and a control section.

The test section has as its input, the tip and ring of the loop under test. The output of the test section is a dc voltage proportional to the quantity being measured. The dc voltages are passed to the SOI for processing as previously described.

The test section has two essentially identical measurement channels which allow simultaneous testing of both tip and ring conductors. Each measurement channel consists of a number of analog signal-processing blocks which can be connected in various combinations, via switches, to set up the desired measuring circuit.

The analog signal processing blocks can be separated into four main categories: sensors, filters, detectors, and miscellaneous. The TP employs several types of sensors for the tests it performs.

The Active Current Sensor (ACS) senses the current flowing in the conductor under test. The dc and ac voltages ranging from 0 to 80 volts can be applied to tip and/or ring through the ACS, and the resultant current measured to give a measure of circuit impedance. Other sensors are provided for detecting foreign voltage, measuring the longitudinal balance of the loop, and identifying Receiver-Off-Hook (ROH) conditions.

The test package filters provide for rejection of 60-Hz noise by a 60-Hz band-elimination filter, separation of ac and dc signals by Bessel high-pass and low-pass filters, and selection of individual test frequencies by a bank of bandpass filters.

Three types of signal detectors are incorporated into the design. The phase detector produces a pair of voltages that are proportional to the real and imaginary components of the impedance at 24 Hz of the circuit under test. The RMS detector produces a dc voltage output equal to the true RMS of any ac signal or noise appearing at its input. The precision detector produces an output that is the average value of the ac signal appearing at its input.

Miscellaneous TP circuits are the Source Select Amplifer and the Phase Null circuit. The Source Select Amplifier selects the ac and/or dc voltage to be applied for a test and sets their levels dependent upon control signals received from the Controller. The phase null circuits automatically compensate for phase shift through the analog signal processing blocks and null the output of the phase detector during periods of idle time (no-test request).

Sanity Reference points (SR points in Fig. 6) are provided for signal injection or passive termination connection by the SOI during self-check (known as Sanity and Diagnostics) testing.

The control section of the TP receives instructions from the Controller via the CCC, operates the test package switches which control

test configuration, times the set up and progress of all tests, and controls the sequencing of connection of the outputs to the SOI encoding circuits.

The third major section of the MMM is the TPA. Unlike the TP, the TPA is not a configurable test circuit. It is designed to perform specific tests which, due to their length, would be an inefficient use of the TP resources. Two test functions are provided: Rotary Dial Analyzer and Dial Tone Analyzer.

The Rotary Dial Analyzer contains a signal-conditioning circuit which performs dc restoration and signal clamping required to interface with logic measuring circuits. The dial pulses gate a clock signal to two digital counter circuits. The output registers of the two counters are used to determine the number of dial pulses received and the speed and percent break of the rotary dial being tested.

The Dial Tone Analyzer presents a constant-current sink for the switching-machine dial-tone generator. This sink provides a 20-mA load to the generator regardless of the length of the no-test trunk, thereby simulating worst-case loop conditions. The sink can be configured for loop-start, ground start, or ground start reverse line circuits.

The constant-current sink is followed by a band-pass filter to separate the dial-tone frequencies from noise signals. A precision detector converts the received dial-tone voltage to a dc level compatible with the encoding circuits of the SOI.

5.2 Test capabilities

A brief description of the test capabilities follows:

DC Thevenin—This test measures parameters required to form a three-terminal dc Thevenin equivalent circuit looking into tip, ring, and ground. It identifies resistive faults, dc foreign voltage, crosses to working pairs, ground start PBX signatures, central office line-circuit faults, and lines on intercept.

Short Circuit Current—This test measures the short circuit noise current tip to ground and ring to ground.

Three Terminal Admittance—This test uses the phase detector in the TP to determine the real (resistive) and imaginary (capacitive) components at 24 Hz of the loop between tip, ring, and ground. This measurement is used to determine the length of no-test trunks, the length of loops, to detect POTS ringers, to identify whether a pair is open on tip, ring, or both sides and whether the open is in or out of the central office, and to measure the distance to the open if it is out of the central office.

Open Circuit ac and dc Foreign Voltage—This test measures the open circuit dc foreign voltage and the ac voltage appearing tip to ground and ring to ground.

Receiver-Off-Hook (ROH)—This test uses the nonlinearity of the off-hook station set to discriminate between a tip-to-ring resistive short and an off-hook set.

Thermistor Heating—This test provides a means of detecting the presence of thermistors in the alerting circuits of PBXs and Key Telephone Sets by detecting the change in the real part of the admittance at 24 Hz as a voltage is applied by the TP.

Ringer Counting—This test provides a means of counting the number of ringers connected tip-to-ring, tip-to-ground, or ring-to-ground. Cable capacitance is differentiated from ringer capacitance by an algorithm that compares loop admittance at several frequencies.

Longitudinal Balance—This test provides measurements for determining loop balance to 65 dB at 200 and 800 Hz.

Soak—This test measures the time variance of a resistive fault with an applied dc voltage.

Dial Tone Analyzer—This test measures whether dial tone can be drawn and broken. It indicates whether the dial tone is drawn normally (within 3 seconds) or slowly (3 to 6 seconds).

Rotary Dial Analyzer—This test counts the number of pulses from a rotary dial, and measures the dial speed and percent break.

5.3 Calibration

To assist in improving measurement accuracy, the ability is provided to perform no-test trunk calibration. Dedicated telephone numbers, which are open circuited on the main distributing frame, are accessed and tested to determine whether trunk faults exist and to determine the length of the trunk. The length measurements are used in making decisions on the locations of open faults.

Component aging may cause a degradation of the measurement accuracy. To compensate for this effect, the TP is designed to permit measurement of critical ac and dc gains and offset voltages. These parameters are stored in the Controller and used in the associated software test algorithms. The calibration is initiated via a system request at the FE.

As previously mentioned, test nodes are provided in the TP and TPA where sources and/or passive terminations in the SOI can be applied, under software control, to isolate component failures to particular circuit packs. The SOI also cotains circuits that permit testing of the A/D encoding circuits and the RAM buffer.

5.4 Packaging

The LTF is mounted in a standard 7-ft UNIFRAME consisting of dual 38-in. bays (Fig. 3). The design is modular and connectorized, which permits growth from a minimum to a maximum system by the

addition of circuit packs of plug-in subassemblies. The minimum LTF/MMM serves up to approximately 15,000 lines and the maximum LTF/MMM serves approximately 250,000 lines.

The minimum LTF/MMM consists of one TAS that can accommodate up to 18 no-test trunks, circuit packs to activate two test ports, either two dial pulse or two multifrequency dialers, one Test Package (TP), and one Dial Tone Analyzer. The maximum LTF/MMM consists of four TASs to accommodate 98 no-test trunks, circuit packs for nine test ports, two dial pulse and two multifrequency dialers, four TPs, two Dial Tone Analyzers, and one Rotary Dial Analyzer.

VI. CONCLUSION

The MLT system had its field trial in Nashville, Tennessee, beginning in late April 1978. By the end of 1978, four Bell Operating Companies had implemented standard MLT systems produced by Western Electric and, by the end of 1979 thirteen companies were using MLT. Conversion to MLT systems during 1979 and 1980 was running at almost 2 million lines per month.

The economic and operational success of the MLT system is documented elsewhere in this issue.[2] At this writing, Southern Bell Telephone and Telegraph Company has installed more MLT systems (197 LTFs and 59 Controllers by the end of 1980) than any other Bell Operating Company. In Southern Bell, the operational availability of the LTFs and associated MMMs is running at 99.7 percent, while the overall availability (Controllers, data links, LTFs, etc.) of the system is currently 99.3 percent.

VII. ACKNOWLEDGMENT

The MLT design, development, and introduction was a large team effort involving the contributions of many highly skilled systems engineers, electrical engineers, computer scientists, physical designers, and experimental psychologists. To single out individuals would be unfair to those not mentioned, so the authors wish to acknowledge that the ideas and results stated in this article belong to those many individuals who gave unselfishly and tirelessly to the creation and implementation of the MLT system.

REFERENCES

1. P. S. Boggs et al., "Automated Repair Service Bureau: Evolution," B.S.T.J., this issue.
2. E. A. Overstreet, "Automated Repair Service Bureau: Economic Evaluation," B.S.T.J., this issue.
3. O. B. Dale, "The Evolution of the Automated Repair Service Bureau With Respect to Loop Testing," International Symposium on Subscriber Loops and Service—Conference Record, March 20–24, 1978, IEEE Cat. No 78CH1279-9 COM.

4. F. J. Uhrhane, "Automated Repair Service Bureau: Loop Testing Strategies and Techniques," B.S.T.J., this issue.
5. G. H. Leonard and J. E. Zielinski, "Automated Repair Service Bureau: Human Performance Design Techniques," B.S.T.J., this issue.
6. R. F. Gauthier and W. A. Harris, "Automated Repair Service Bureau: Two Examples of Human Performance Analysis and Design in Planning the ARSB," B.S.T.J., this issue.
7. R. L. Martin, "Automated Repair Service Bureau: The System Architecture," B.S.T.J., this issue.
8. S.G. Chappell, F. H. Henig, and D. S. Watson, "Automated Repair Service Bureau: The Front-End System," B.S.T.J., this issue.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Second-Generation Mechanized Loop Testing System—A Distributed Microprocessor Application

By H. RUBIN

(Manuscript received December 2, 1980)

The Mechanized Loop Testing (MLT) system is that part of the Automated Repair Service Bureau that provides automatic acquisition and analysis of electrical test data for customer telephone loops. The MLT-2 system is a second-generation MLT that has as its basic architectural features communication, loop access, and loop test distributed as closely as possible to the point of testing. Architectural components include a wire center-based Loop Testing System (LTS) and a centrally located Data Communication Network (DCN). Each LTS contains communication, access, and test capabilities, and is logically connected by the DCN to each controlling minicomputer (up to 12). The LTS and DCN are each composed of multiple microprocessor-based circuits. The architecture of MLT-2 is presented. Particular attention is given to the subjects of partitioning both hardware and software, to the development of change-tolerant software, and to intrasystem communication capabilities powerful enough to support a large number of distributed processors. In addition, special measurement techniques employed by MLT-2 that take advantage of analog and digital large-scale integration technology are discussed. Operational scenarios are included for an appreciation of how the MLT-2 system works.

I. INTRODUCTION

At Bell Telephone Laboratories, Mechanized Loop Testing (MLT) is a generic term used to describe that part of the Automated Repair

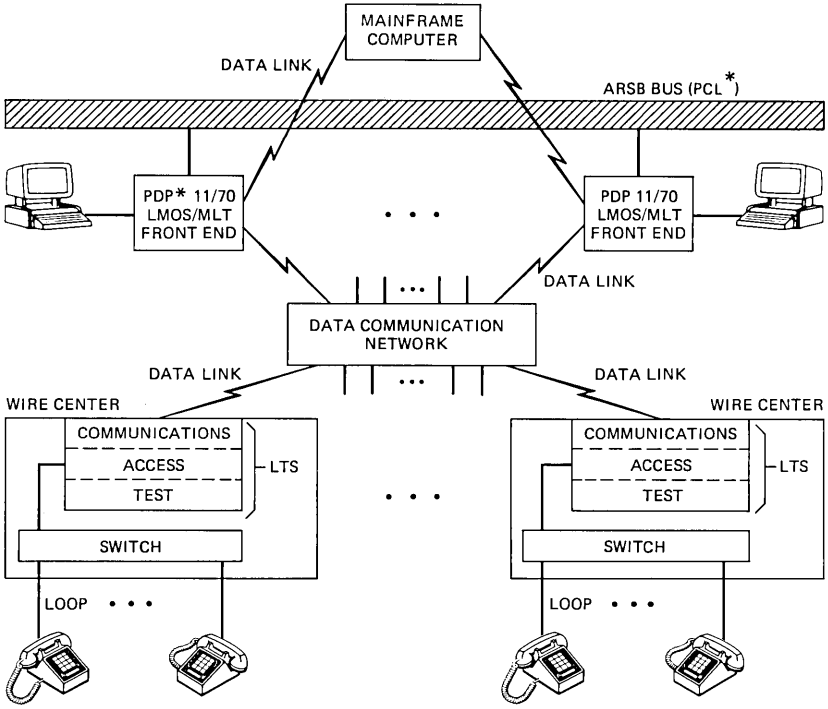
Service Bureau (ARSB) that provides automatic acquisition and analysis of electrical test data for customer telephone loops. As pointed out in Refs. 1 and 2, the acquired data are translated into an equivalent electrical circuit model, and matched against the Loop Maintenance Operations System (LMOS) data base information in an attempt to isolate loop faults or to determine that the loop is operating properly. The first such system, MLT-1, is described in Ref. 2. The MLT-2 system is an alternative solution to the loop testing problem which takes advantage of recent technological advances, and can be used either to augment an existing MLT-1 installation or to provide the total loop testing function in a given loop environment.

There are two major reasons for the development of an alternative MLT system. First, the Loop Testing Frame[2] (LTF) of MLT-1 does not represent a cost effective solution in very low population density areas.[3] Second, Bell Operating Companies (BOCs) have recently started consolidating their testing bureaus; this requires the relocation of the associated manual testboard positions (Local Test Desk No. 14 and No. 16). Relocation of the electromechanical testboards is an expensive operation because they were not designed to be moved from their initial installation site. For reasons explained in Ref. 2, MLT-1 does not eliminate completely the manual testboard system. Consequently, it has become important that the new automated testing system replace the manual system. The MLT-2 system is cost effective in a small wire center environment, sophisticated enough to eliminate the existing manual systems, and it makes loop testing independent of how the testing bureaus are organized within the BOC. Distributed processing techniques using microprocessor-based circuitry allow the realization of these system characteristics.

Automated loop testing that is coupled to LMOS is a rather natural application of distributed computing; subscriber loops are dispersed over a wide geographical area, whereas the LMOS data-base is centralized within the front-end (FE) processors. For economic reasons, the MLT-1 testing vehicle [the LTF[2]] is designed to take advantage of functional concentration to realize economies of scale. Hence, distributed processing in MLT-1 stops at the MLT-1 Controller.[2] The MLT-2 system relies on the use of microprocessors and new techniques for loop measurements to meet the challenges of cost and performance, and to take more complete advantage of the distributed nature of the testing problem.

II. AN OVERVIEW OF THE MLT-2 ARCHITECTURE

To test telephone loops (as opposed to building a complete testing system with its human interface),[2] three basic functions are required: access, test, and communications. These three basic functions are



* REGISTERED TRADEMARK OF DIGITAL EQUIPMENT CORPORATION

Fig. 1—Architecture of MLT-2.

present in all manual and automatic testing systems. The testing system has to gain control of the loop and have physical access to it in order to test it. In addition, the testing system has to have two-way communications with a central controller that determines the testing pattern and collects the test results for analysis. The reader can recognize these basic functions in the MLT-1 system described in Ref. 2. The main architectural technique used in the design of MLT-2 is to distribute the access, test, and communications functions as closely as possible to the point of testing, i.e., to the loop itself. The use of this technique tends to minimize the data flow in the system, and is consistent with design strategies for functional distribution that have been advocated elsewhere.[4,5]

Figure 1 shows a block diagram of the MLT-2 architecture. The ARSB high-performance Parallel Communication Link (PCL)[6] and connections to the mainframe computer are shown for completeness. (See Refs. 7 and 8 for a discussion of the PCL and the mainframe computer.) In Fig. 1, each wire center served by the system contains a microprocessor-based Loop Testing System (LTS) that consists of access, test,

and communications capabilities. The use of microprocessor technology makes it economically possible to distribute these functions and their control on a wire center basis, where the loops terminate on the central office equipment. The communications function that resides in each LTS is incomplete without the Data Communication Network (DCN), which is part of the architecture in Fig. 1. The DCN allows any one of the PDP* 11/70 LMOS/MLT computers (FEs) to communicate with the LTS in any wire center served by the system. The DCN is itself a microprocessor-based distributed processing machine that off-loads communications processing for all FEs (up to twelve) attached to the PCL. The MLT-1 architecture restricts each FE to serve a unique subset of customer loops, whereas the architecture of MLT-2 allows any FE to test any customer loop.

The MLT-2 system operates in the following manner. Repair Service Bureau personnel have a CRT interface to the LMOS/MLT system, and are able to input data (a telephone number to be tested) and receive output (test results) from the system.[9] Each CRT is connected by data link to one of the FEs in the ARSB. If the user requests MLT testing, the MLT software that resides on the FE interacts with LMOS functions on that machine to retrieve the data base information for the loop to be tested. This information is passed to an application process that initiates and guides the loop access and testing. The application process may contain the adaptive loop testing algorithm discussed in Ref. 1, or it may contain software to implement interactive test control and other functions that allow the elimination of the manual test board systems referred to earlier.

Because of the intelligence located in the LTS microprocessor circuits, only high-level commands need to be generated by the FE software. The first command requests the LTS to access a specified telephone number. The message header contains a parameter that identifies the LTS data link for the system to use. This message is routed by the DCN to the appropriate LTS data link (see Fig. 1). When access is completed, the response is routed through the DCN to the FE that requested the access. Subsequent message transactions that occur between the LTS and the FE involve high-level requests for tests to be performed, followed by detailed responses containing raw test data (the amount of current that was measured on the loop wires when a particular source was applied to the loop, etc.). See Ref. 2 for a description of the tests typically performed by MLT. The last request made by the FE is to have the LTS drop the access to the loop under test. The number of loops that may be accessed simultaneously at any LTS site and the

* Registered trademark of Digital Equipment Corporation.

number of simultaneous tests that may be in progress at a given LTS are discussed in the sequel.

The MLT-2 architecture has the following attributes:

(*i*) Since FEs can control testing on any loop, they provide active backup to each other, thereby increasing system reliability.

(*ii*) High-level commands are passed from the FE to the LTS, thereby minimizing the volume of communications required.

(*iii*) The FE MLT function controls the testing of a loop, but does not control the details of each test. Less processing overhead means that higher system throughput can be achieved.

(*iv*) Increased throughput means that many more wire centers can be handled by the MLT-2 system than by the MLT-1 system. In addition, MLT control software and LMOS software can share the same FE.

(*v*) Distribution of function to the point of testing eliminates long test trunk connections.

This overview shows the basic design philosophy of MLT-2, and illustrates the benefits that arise from adopting a distributed processing approach to loop testing. However, many problems need to be overcome in implementing the architecture. The distributed functions have to be made cost effective in small loop environments, and the communications and control mechanisms must be sufficient to support the interactions of many distributed intelligences. Hardware and software functions need to be partitioned so as to reduce module complexity in order to simplify module design and maintenance. The following sections describe the microprocessor-based LTS and DCN in detail from both a hardware and software perspective, and it is shown how the "divide and conquer" technique is used to render a practical design.

III. MLT-2 HARDWARE IMPLEMENTATION

3.1 Loop Testing System

The wire center-based LTS is a collection of loosely coupled* distributed microprocessors organized to perform the communications, loop access, and loop testing functions. The block diagram in Fig. 2 shows an LTS controller that is responsible for communications with the DCN, for control of circuits used to provide a talk function for ARSB personnel, and for local control of the other LTS processors. The port controller is responsible for the access function. It provides tip, ring, and sleeve lead control for connections to no-test trunk circuits that enable MLT to interface to the switching machine. The Precision Measurement Unit (PMU) is a general-purpose testing instrument that is used to

* The term "loosely coupled" is used here to denote an organization of processors that share no common memory but communicate by passing messages over a serial or parallel interface.

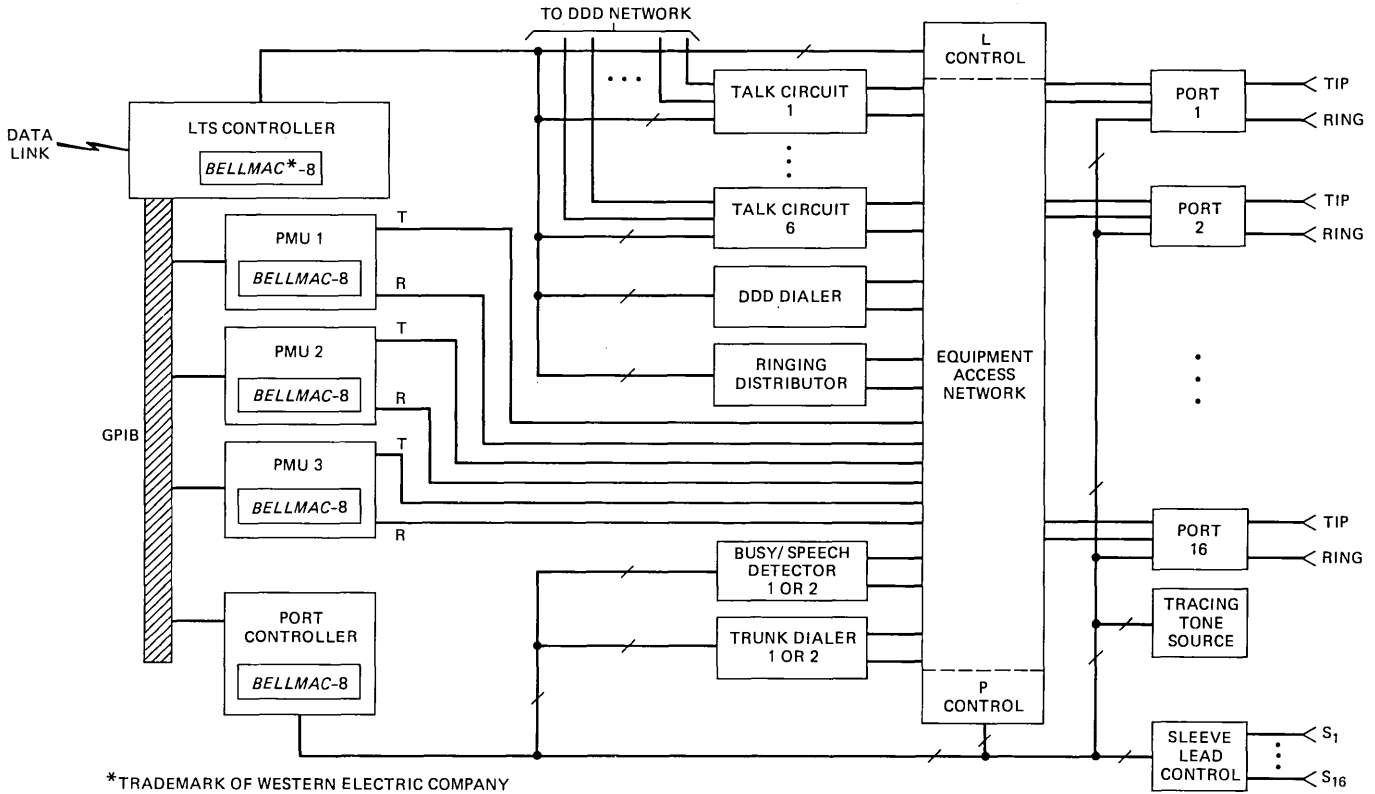


Fig. 2—Loop Testing System.

make loop measurements. Each LTS may contain from one to three PMUs. Each PMU is based on a *BELLMAC*†-8 microprocessor[10] module, as are the LTS and port controllers. All processors in the LTS are loosely coupled, and use an IEEE STD-488 General Purpose Interface Bus (GPIB)[11] as the interprocessor communication mechanism. The GPIB is controlled by the LTS controller. The data link connection to the DCN is a synchronous, full duplex 1200- or 2400-baud link utilizing the BX.25 level 2 communication protocol.[12]

The system is designed to be modular so that wire centers ranging from roughly 1000 to 100,000 lines can be served economically. As the wire center size increases, more PMUs can be added (up to three per LTS), up to sixteen port circuits (interfaces to no-test trunk circuits) can be accommodated, and the Equipment Access Network (EAN) internal to the LTS can be expanded to enable any of the PMUs, dialers, talk circuits, etc., to be connected to any port circuit. Hence, the largest size LTS can have up to sixteen loops simultaneously accessed for testing, and can time share the three identical PMUs to perform requested tests. The largest size LTS, therefore, contains five *BELLMAC*-8 microprocessor modules, as well as several single-chip microcomputers (four per PMU) to perform special functions for the PMU.

The separation of the testing function from the access and communications functions simplifies the LTS organizational structure in the case where multiple PMUs are required to handle a given testing traffic load. The assignment of access and communications to separate processors increases significantly the throughput of the LTS. With the arrangement shown in Fig. 2, communications, access, and testing can be going on simultaneously. It is also evident that distributing the local processing provides additional memory space for the LTS functions. A system such as MLT has great potential for functional growth, and additional program memory size is an asset.

3.1.1 Loop Testing System operation

The reader is referred to Fig. 2 for the following discussion. The LTS controller implements the BX.25 level 2 protocol function on the data link. Received messages are parsed and interpreted by the LTS controller. An access request causes the LTS controller to establish some data in RAM that is used to track and time the request. A message is then generated, and passed over the GPIB interface to the port controller. The port controller proceeds to access the loop specified in the message by attaching a trunk dialer to an appropriate port, dialing the telephone number, and attaching a busy/speech detector circuit to determine whether the loop is idle. When loop access is obtained, the

† Trademark of Western Electric Company.

port controller sends a response across the GPIB, and the LTS controller proceeds to satisfy any test requests that may have been contained in the original request message. If no test requests are present, a response is generated for the FE, and is transmitted over the BX.25 link. At this point, the loop is accessed, and the LTS controller awaits subsequent requests for tests.

Most test requests require the services of a PMU. However, some requests can be satisfied by either the LTS controller or the port controller and their associated circuitry. Test requests are coded so that the LTS controller can determine which LTS circuits can satisfy the request. The LTS controller, therefore, acts as a resource manager for the LTS.

Fig. 2 indicates that the port controller can perform sleeve lead manipulation, can apply a tone source to the customer loop, and (not shown) can monitor the loop in a coarse sense for a shorted or open condition on tip and ring leads. Sleeve lead control is used to signal the trunk circuit in order to pull in the customer line circuit for testing or to gain access to certain types of testing circuits associated with the central office. Tone is applied to the loop to help outside plant personnel locate a particular wire pair in a cable. Coarse detection of shorts and opens on a customer loop is also required as an aid in pair identification by outside plant personnel. Although the latter two features can be implemented with a PMU, the duration of the tone application or monitor function is in the order of minutes. Consequently, MLT-2 does not attempt to use the sophisticated PMU for these purposes. The port controller functions mentioned above allow MLT-2 to replace the manual testboard system.

Another feature required to allow replacement of the manual system is the ability to alternately talk to a customer (or craft person) and test the customer loop. This feature is provided by the LTS controller via the LTS talk circuits. If an interactive session is desired, the initial access request contains, in addition to the telephone number of the loop to be tested, the number of a telephone that is associated with the craft person at the ARSB work center. The LTS passes the customer telephone number over the GPIB to the port controller as before, but now proceeds to use a dialer to place a call over the DDD network to the work center. When both connections are made, the LTS can be commanded to ring the customer loop, and, subsequent to ring-trip detection, connect the test trunk and the DDD path through one of the LTS talk circuits. The craft person can enter requests through the CRT, and cause the LTS to break the talk path temporarily, perform the requested test, and reconnect the talk path.

Tests and features other than those mentioned above require the use of a PMU.

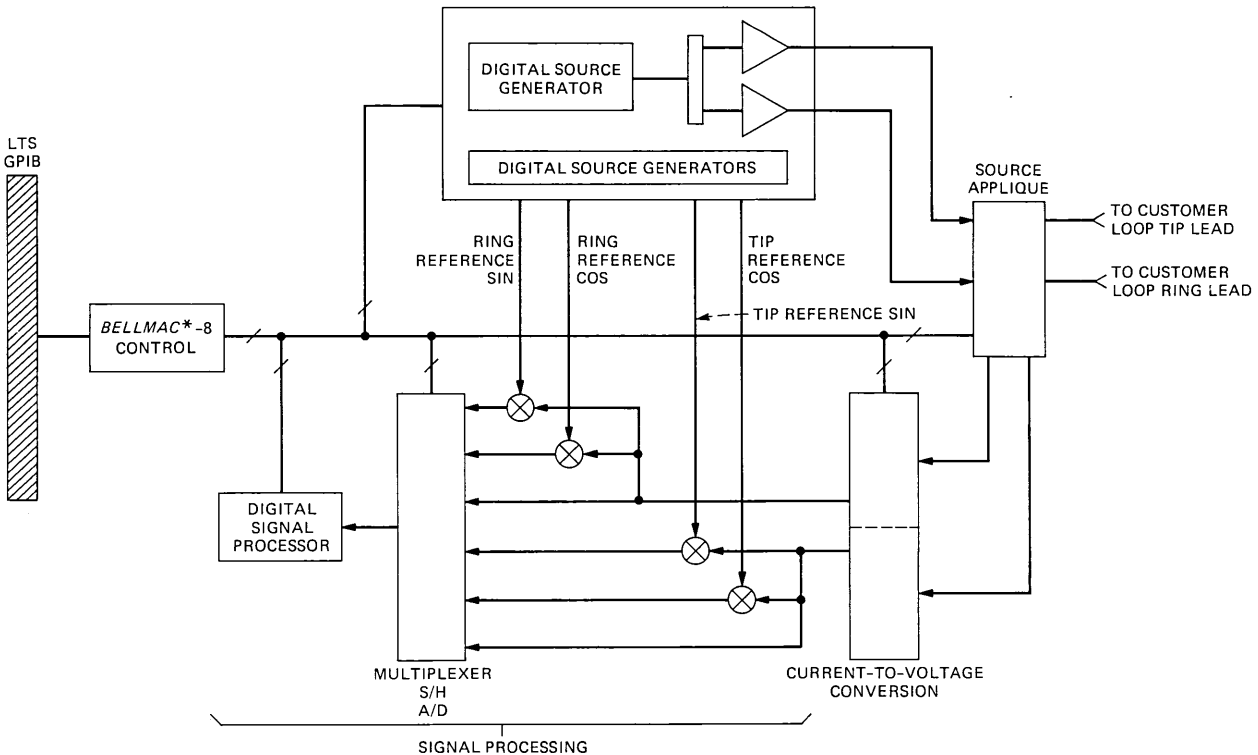
3.1.2 Precision Measurement Unit

No part of the PMU is dedicated to performing any particular test. Fig. 3 presents a block diagram of the PMU circuitry, and shows the *BELLMAC*-8 controller, a source generation section, a source applique (terminations), loop current detectors, and a signal processing section. The PMU is capable of performing all tests that MLT-1 can perform with its several test circuit types,[2] and can also perform many tests that MLT-1 cannot perform. The *BELLMAC*-8 processor receives test requests via the GPIB interface, sets up the test by interfacing to the components shown in Fig. 3, and transmits the results across the GPIB when the test is completed.

The source generation section of the PMU consists of a set of three single-chip microcomputers. Each microcomputer can generate digital samples of quadrature sinewaves at programmed frequencies ranging from 1 to 3200 Hz in 1-Hz steps. A table lookup algorithm is used for this purpose. The digital samples are converted to analog form by means of digital-to-analog converters, possibly combined with a dc level, and applied to a power amplifier. Signals up to a 135-volt peak and up to 125 mA, from dc to 3200 Hz, can be generated by this circuitry. The single-chip signal generators can produce several functions, including pulsed waveforms, swept waveforms, and dual frequency waveforms.

Mechanized loop testing systems perform mainly admittance measurements.[2] Hence, in order to test a loop, the signal voltage sources are applied to the tip and ring leads, and the resultant current flow is measured. In MLT-2 the current is measured by means of a magnetic technique that uses components that are substantially smaller and more efficient than those used in MLT-1 for the same purpose.[2] The outputs of the two magnetic sensing circuits are voltages proportional to the currents in the loop conductors. The two channels can be configured to produce the metallic and/or longitudinal loop currents.

The signal processing section of the PMU consists of signal conditioning circuitry for each current sensing channel, an analog multiplexer, an analog-to-digital converter, and a digital filter. Of the three microcomputers in the source generation section, only one is used to generate the voltage waveform that is applied to the loop. The other two are used to generate inputs to the two signal conditioning channels. Waveforms at precisely the frequency applied to the loop can be generated in quadrature and used to synchronously demodulate the voltage outputs from the current sensors. Analog multipliers are used for this purpose. The PMU circuit phase shifts can be accommodated simply by shifting the phases of the demodulator sources relative to the loop source. Quadrature waveforms are used to generate demodulated voltages that are proportional to the real and imaginary compo-



*TRADEMARK OF WESTERN ELECTRIC COMPANY

Fig. 3—Precision Measurement Unit.

nents of the loop current. Harmonics of the frequency applied to the loop can be generated and used to detect nonlinearities in the loop current. The use of two signal conditioning channels and two demodulating sources allows the PMU to make multiple measurements simultaneously. The reader should note that in this scheme, all results are dc values, either initially or after the demodulation and filtering process.

As the above paragraph suggests, the signal processing section's input circuitry can produce several outputs simultaneously, depending on the test being performed. An analog multiplexer is used to select the desired outputs, and feed them to a sample/hold and A/D converter circuit. Digital samples are fed to a Digital Signal Processor (DSP) chip,[13] where most of the filtering in the system is performed. The DSP contains several digital filter programs and a dynamic settling algorithm that can decide when a final value has been obtained from a measurement. Test results are passed from the DSP device to the PMU control processor, and are sent over the GPIB to the LTS controller.

The general-purpose design of the PMU is evident in the above discussion when one realizes that, within the voltage and frequency limits specified, the PMU can make measurements to characterize any three-terminal network. The PMU intelligence is also used to provide self-calibration functions and a sanity/diagnostic function.

The description of the LTS shows how the hardware is mapped onto the physical needs of the loop testing problem. The result is a loosely coupled distributed architecture in which each processor's operational details are hidden from the others. Getting something done in the system requires only that a message be passed between processors. The technique of mapping the hardware onto the problem is extended to the software as well, and represents a continuing theme in the MLT-2 design. The result is an easily understood and maintainable system.

3.2 Data Communication Network

Figure 1 indicates that the DCN has a very simple functional requirement, namely, to route messages between any FE and any LTS. The architecture to be described below allows from one to twelve FEs to exchange data with up to 768 LTSs, using the BX.25 level 2 protocol. (The limit of 768 is determined by physical design constraints and by expected needs of the application. The total capacity of the architecture to be presented is 1800 LTS data links.) Two other requirements include having the ability to operate when remoted from the ARSB FE complex, and having enough redundancy to withstand a single failure.

Figure 4 shows a three-tiered multiple processor architecture that realizes the DCN function. The microcomputers based on the *BELL-*

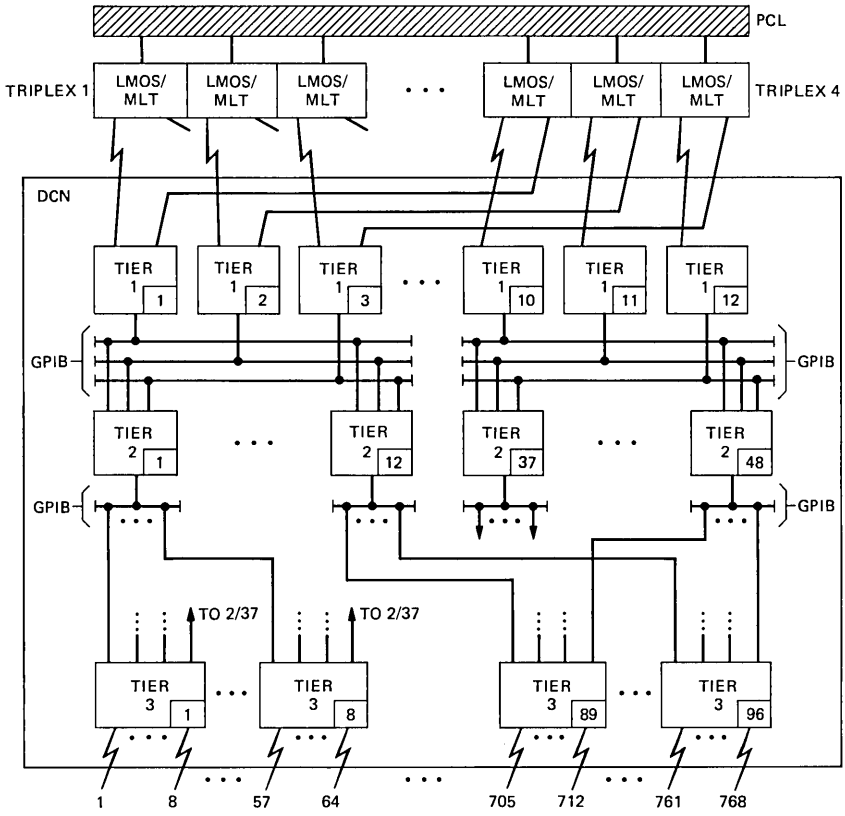


Fig. 4—Data Communication Network.

MAC-8 microprocessor are loosely coupled, and the design is modular in that eight LTS data links can be added at a time, and/or a single FE can be added to the system until its capacity is reached. Only five separate circuit pack codes are required to assemble the DCN.

Each LMOS/MLT FE is connected to two distinct DCN Tier 1 circuits via synchronous 9600-baud BX.25 data links. Dual active data links are used to meet reliability and throughput requirements. The use of data links for the FE interface allows the DCN to be remoted from the ARSB computer complex. Each third-tier circuit in the DCN provides eight 1200- or 2400-baud BX.25 data links that interface to the LTSs. The second-tier circuits serve to interface the first-tier and third-tier microcomputer circuits via IEEE STD-488 busses.

As can be seen from Fig. 4, the FEs are grouped in triplexes.* The

* The triplex configuration is derived from the LMOS application in which two FEs are active and one serves as a backup. All three share a common set of data links to user terminals.

second-tier DCN circuits are grouped in units of from one to twelve circuits and are dedicated to two distinct triplexes. Each second-tier function has a GPIB interface to each of three first-tier functions. The first-tier functions are backed up by virtue of the existence of two paths into the DCN for each FE. The second-tier functions are backed up only if more than one triplex is connected to the network, for in that case, an additional set of second-tier functions is used. In general, two unique paths exist in the DCN for message traffic between any FE and the eight LTSs served from a particular third-tier function. Message re-routing is used to deliver messages in the face of single circuit failures.

Figure 4 shows that the third-tier DCN circuits have GPIB interfaces to each set of second-tier functions present in the system (up to four GPIBs per third-tier function). Third-tier DCN functions are not backed up, and consequently, a single failure can result in the loss of communications up to eight LTSs. Sanity and diagnostic software functions are provided to identify these failures and help correct them within reasonable time limits.

Message routing in the DCN is accomplished by having two bytes in the message header contain an LTS data link identifier. This identifier is filled by the FE when it constructs the message. When a message passes into the DCN, the first-tier microcomputer software fills a third byte in the header that is reserved for the FE identifier. The response message from the LTS contains the same routing information as the original request message. Hence, the FE that generates the request receives the response.

The MLT-2 system can be seen to consist of a large number of microcomputer elements that are connected by communications facilities, including data links and local IEEE STD-488 busses. A typical MLT application may contain roughly three hundred *BELLMAC*-8 microprocessors; the largest installations may contain over a thousand *BELLMAC*-8 microprocessors. Functional decomposition is used to render the hardware design of such a system comprehensible. Similar techniques are applied to the software designs for these microprocessor subsystems.

IV. THE MLT-2 MICROCOMPUTER SOFTWARE

When the PMU receives a request to perform a test, it proceeds to do that function and that function only. Its operation is seen to be "single-threaded" in that it completes its activity without interruption. By contrast, all other MLT-2 microprocessor environments are "multi-threaded." In the DCN circuits, I/O operations can be occurring simultaneously over several different interfaces. In the LTS controller and port controller, activities for up to sixteen loop accesses may be in

different states of completion at the same instant. Furthermore, the operations required for interfacing to the central office equipment or for controlling the data link have many spans of time during which activity ceases before the *BELLMAC*-8 microprocessor needs to perform the next step. Hence, the MLT-2 microcomputer circuits operate for the most part in a multitasking environment. To cope with this environment, a small operating system (designated m8os) has been developed for MLT-2, and provides the multitasking facility, intertask communications via messages and semaphores, and buffer management. A user-defined interrupt structure is also supported. An attempt has been made to keep m8os as small and as fast as possible. The size of m8os is just 2200 bytes of text and data.

The provision of a multitasking environment facilitates the partitioning of the software into entities known as "tasks." Each user task is known to the operating system, and can be scheduled to execute when there is some operation to be performed by that task. Usually, a semaphore is set or a message is passed by some task or I/O that alerts m8os to the need to schedule a particular task. The main advantage to partitioning the software is to create functions of manageable size. Another important advantage is to create a structure that is change tolerant; when something in the environment forces a software change, not all of the software has to be changed. Software tasks are more or less isolated from one another, and have a very simple and precisely defined interface with one another via the operating system. One guideline used in MLT-2 is to partition the software in such a way that it maps onto the system hardware as much as possible. This mapping tends to make the software system change tolerant. Another guideline followed is to buffer the application tasks from the hardware drivers as much as possible. Again, a greater degree of change tolerance is achieved.

4.1 Input/Output structure

A unified I/O structure is used for all MLT-2 microcomputer circuits. The main idea is to buffer the user tasks from the hardware drivers (usually interrupt-level software functions), and to standardize the task/driver interface. The constraint imposed is that a task never communicates directly with a hardware driver. Instead, the task makes a function call, and passes some necessary parameters (the address of a message buffer, a byte count, and the address of a result flag). The function called is referred to as the interface function. A RAM control block for the I/O hardware is defined and known only to the interface function and to the driver software. The interface function does only two things. First, it fills the control block with the user parameters. Second, it performs one action to start the I/O. The I/O operation is

completed by the driver at interrupt level. The only connection between the driver and the interface function is the common control block. The only connection between the driver and the user task is a semaphore that is set by the driver when its activity is completed. The user task waits for the occurrence of the semaphore, after which it is scheduled to execute by m8os. Since the only interaction between the driver and the operating system is the setting of the user semaphore, an extremely fast and error-free interface is achieved between task, driver, and operating system.

4.2 Task structure

User tasks are defined so that the software maps onto the system hardware as much as possible in order to minimize the impact on the software of changes in the hardware or in the hardware drivers. Hence, each I/O facility of sufficient complexity has its driver and interface function, and also a unique task that controls that I/O facility. Any other task that needs I/O from (to) that device merely receives or sends a message buffer via m8os to the controlling task. The interface between any task and the I/O is now very simple. Since only one task controls the I/O, there is no confusion about when the device is ready for the next operation, and user tasks do not have access to global variables that describe the I/O function. Potential software errors are thereby avoided, and an application task can change and not complicate or even cause changes to the drivers.

Figure 5 shows the software architecture of the DCN third-tier function, and is representative of the architecture in the remaining DCN circuits. The large ovals indicate an application task, whereas the small ovals indicate an interface function. Because of the simplicity of the DCN operation, only three types of tasks are required. One controls the GPIB interface. Since a third-tier circuit contains four GPIBs (Fig. 4), four large ovals of this type are indicated. Although there are four distinct GPIB tasks, all execute the same program text. Similarly, there are eight BX.25 tasks shown in Fig. 5 to control the eight LTS data links. A message received over a GPIB is simply sent to the appropriate BX.25 task via the m8os message passing facility.

Each MLT-2 microcomputer contains an administrative task (ADMIN in Fig. 5). This module performs all nonoperational functions required in the local environment. For example, ADMIN controls sanity and diagnostic functions and the reporting of trouble count and usage statistics. The MLT-2 message header contains the message source and destination circuit types and task identifiers. Hence, ADMIN tasks in different microcomputer modules can communicate to synchronize actions across circuit-board boundaries. Lack of space prohibits a more detailed discussion of MLT-2 microcomputer sanity and diagnostic strategy and software.

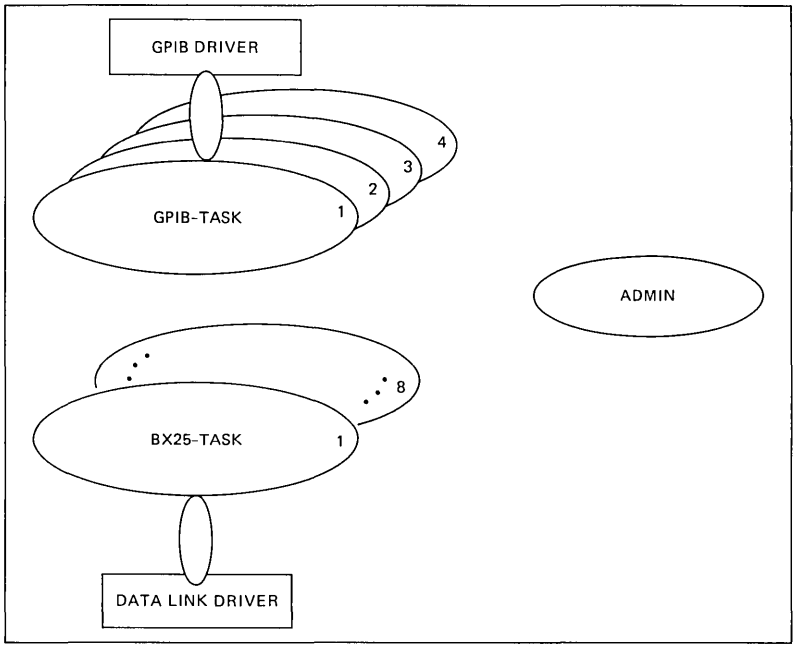


Fig. 5—Data Communication Network software structure.

Other MLT-2 microcomputer modules have software architectures that are similar to that shown in Fig. 5. For example, the port controller contains a GPIB task to control its communication interface. It also contains one PORT task for each port in the LTS (i.e., 16 tasks). Each PORT task can control all activities on a particular port in the system. If the GPIB task receives an access request, it passes the message to a PORT task, and access is controlled from this software function. All 16 PORT tasks execute the same program text.

The reader should note the similarity that exists in all MLT-2 microcomputer software environments. This similarity is of course intentional, and provides several benefits. Different MLT-2 microcomputer circuits have the same types of I/O interfaces. The structure allows us to include a single software module in all environments where it can be used. Similar structures make it easier to understand and cope with the large amount of software required to produce the MLT-2 functions.

V. CONCLUSIONS

The second-generation MLT is a microprocessor-based distributed processing system that performs the loop testing function in the ARSB. The main architectural technique used in the design of MLT-2 is to

distribute the access, test, and communications functions as closely as possible to the point of testing. New architectural components include the LTS and the DCN. The LTS can appear in each wire center served by the system, and is composed of a collection of loosely coupled microprocessors. One of these is a sophisticated general-purpose testing instrument, the PMU. The architecture and operation of the LTS are discussed in some detail. The DCN allows any LMOS/MLT FE to exchange data with any LTS. The DCN is itself a collection of loosely coupled microprocessors, and is constructed from five basic circuits. The modularity and fault-tolerance of the DCN are discussed.

The architecture of the MLT-2 microcomputer software is described. The multitasking environment is explained, and techniques for dealing with this environment are presented. Major software design goals are to make the system tolerant to change, easy to understand, and easy to maintain. These goals are achieved by standardizing I/O interfaces, partitioning software so that it maps onto the system hardware and/or the problem being solved, and making use of functional commonality across the different MLT-2 microcomputer software environments.

The use of a distributed testing architecture provides many advantages to the ARSB FE complex, including the ability to support testing in a very large number of wire centers. Installations of MLT-2 can have many hundreds of microprocessors involved in the testing function for the totality of loops served. The functional decomposition techniques used in the design of MLT-2 help deal with the complexity that accompanies this large distributed processing system.

VI. ACKNOWLEDGMENTS

Many people have made significant contributions to the MLT-2 development. I gratefully acknowledge the efforts of the members of the Maintenance Systems Engineering Department and the Mechanized Loop Testing Department.

REFERENCES

1. F. J. Uhrhane, "Automated Repair Service Bureau: Mechanized Loop Testing Strategies and Techniques," B.S.T.J., this issue.
2. O. B. Dale, T. W. Robinson, and E. J. Theriot, "Automated Repair Service Bureau: Mechanized Loop Testing Design," B.S.T.J., this issue.
3. E. A. Overstreet, "Automated Repair Service Bureau: Economic Evaluation," B.S.T.J., this issue.
4. C. Weitzman, *Distributed Micro/Minicomputer Systems,* Englewood Cliffs, New Jersey: Prentice-Hall, 1980.
5. S. Muftic and N. Husovic, "On Functionally Distributed Computing Systems," IEEE Symp. on Trends and Applications: Distributed Processing, 1978.
6. *Terminals and Communications Handbook,* Digital Equipment Corporation, pp. 272–5, 1979.
7. R. L. Martin, "Automated Repair Service Bureau: The System Architecture," B.S.T.J., this issue.
8. C. M. Franklin and J. F. Vogler, "Automated Repair Service Bureau: Data Base System," B.S.T.J., this issue.

9. G. H. Leonard and J. E. Zielinski, "Automated Repair Service Bureau: Human Performance Design Techniques," B.S.T.J., this issue.
10. T. Holub, "Bell Labs Announces Powerful New Microprocessor for Wide Range of Bell System Applications," Bell Labs News, February 17, 1977, Murray Hill, N.J.: Bell Telephone Laboratories.
11. *IEEE Standard Digital Interface for Programmable Instrumentation*, New York: Institute Electrical & Electronics Engineers, 1978.
12. *Operations Systems Network Protocol Specification: BX.25*, Issue 2, Bell Telephone Laboratories, March, 1980.
13. J. S. Thompson and J. R. Boddie, "An LSI Digital Signal Processor," Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, Denver, Colorado, April, 1980.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Cable Repair Administrative System

By P. S. BOGGS and J. R. MASHEY

(Manuscript received July 13, 1981)

The Cable Repair Administrative System (CRAS) is a recent addition to the Automated Repair Service Bureau (ARSB) that helps extend ARSB usage beyond Repair Service Bureaus into the Outside Plant Maintenance organization. The CRAS system helps this organization to schedule repair forces efficiently and to locate the trouble-prone outside plant facilities that are most in need of rehabilitation. This paper includes an analysis of the unusual features of CRAS, and comments on its development under rapidly changing circumstances.

I. INTRODUCTION

The CRAS system helps Outside Plant Maintenance (OSPM) managers to identify people and plant items that need attention on a priority basis. For example, CRAS helps managers schedule repair forces efficiently and locate organizational trouble spots in the complex process of OSP repair. The CRAS system also helps locate those cables or terminals that, over periods of time, cause an excessive number of customer troubles or especially high repair costs. Thus, repair money can be directed where it is most effective.

This article gives the background for CRAS and describes its goals, usage, and architecture. It also comments on the design approach used, unusual features included, and lessons learned from its development.

II. HISTORY AND STATUS

For several years, both AT&T and the Bell Operating Companies (BOCS) wanted a system that combined and improved on the cable trouble analysis features of the Computerized Cable Upkeep Analysis Program (CCUAP) and the manual Cable Repair Force Management

Plan (CRFMP). The CCUAP system was designed to help determine the nature and location of cable troubles, but lacked flexibility. Also, it could not accurately account for the hours worked on troubles because self-reporting was used, rather than payroll records. Trouble counts were also provided through self-reporting, rather than by obtaining the counts from LMOS. The manual CRFMP was used to forecast trouble loads and the repair force needed to handle those loads. It required a large manual effort to collect and analyze data, and had problems like those of CCUAP in providing accuracy. These major limitations are overcome in CRAS through the CRAS/LMOS/Mechanized Time Reporting interfaces described later. The goal of CRAS was to replace the "pieces" with a complete, integrated system. Thus, CRAS provides a complete cable trouble and expense data collection system that has enough detailed data to associate "hours spent" with specific type of trouble, specific part of plant, and specific work forces.

After initial functional requirements were issued in mid-1978, the New England Telephone and Telegraph Company was chosen to field test the system. Some work on CRAS architecture was done during the second half of 1978. In January 1979, software development started, with six software developers, a systems engineer, and a human performance engineer. Given strong pressure to build CRAS quickly, an ambitious development schedule was used to allow the field trial to begin July, 1979; writing and testing much of the software was scheduled for later phases of the trial. The trial was completed successfully in July, 1980, with users who were quite happy at that point. The system's economics appeared sound, and it was well documented and well packaged. The first standard version of CRAS was installed at Southwestern Bell Telephone Company in February, 1981.

This brief history shows that it was possible to build a usable system in a short time span, i.e., two years from starting to write code until a standard installation. Following a description of CRAS and its environment, later sections offer a retrospective on the development process, for it did not happen as smoothly as the speed of implementation might indicate.

III. BACKGROUND

3.1 Customer troubles and cable trouble tickets

For a Repair Service Bureau (RSB), the customer trouble report is the entity that is tracked (by LMOS) and later analyzed (by TREAT). For an OSP maintenance organization, the corresponding entity is the Cable Trouble Ticket (CTT), which is generally a more complex entity than a customer trouble. For example, suppose that a problem causes two customers' cable pairs to be crossed. The repair of this problem is considered to be one case of work, even though it involves several

customers, and may even involve work at several locations. In addition, more organizations are likely to be involved in the repair of an OSP problem. Not only must an RSB handle the problem in the first place, but sometimes station repair craft persons may be dispatched, and then determine that OSP repair craft must become involved. As an extreme, but real example, if a contractor cuts a cable, tens or hundreds of customers may be out of service, and many people may work on its repair. The repair work would still be described in one CTT.

The RSB is primarily organized to handle customer troubles, which exist because they affect customer service. The OSPM organization handles this type of work when cable is involved, tracking each piece of work as a Service Affecting (SA) CTT. Unlike the RSB, the OSPM organization handles an additional type of work, termed Nonservice Affecting (NSA) or routine. The NSA work should be done sometime, even though it does not immediately affect service. Sometimes a craft person may do enough work to restore service, create a temporary closure that will need later work, then go on to the next SA problem. In other cases, people notice problems in cables or terminals that have not been reported as customer troubles, but are likely to cause problems later. In any case, the OSPM organization maintains a list of such "programmable work" that can be used to fill slack periods. As shown in Fig. 1, a complete SA CTT needs the following data:

(*i*) Data from each associated customer trouble, such as the number of trouble reports, circuit number, cable, pair, etc., all of which CRAS obtains automatically from LMOS. Before CRAS, these data were gathered manually from LMOS reports.

(*ii*) Force data associated with the CTT, such as hours worked, account charged (such as Aerial Cable or Underground), craft persons who did the work, what day(s) they worked, etc. The CRAS system obtains these data automatically from the local Mechanized Time Reporting (MTR) system. Before CRAS, these data were gathered manually via discussions with the craft person.

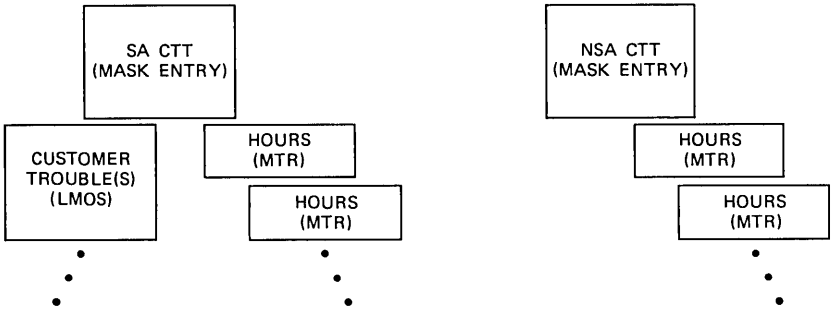


Fig. 1—Components of SA and NSA CTTs.

(*iii*) Data manually entered to complete the CTT, including trouble location, source of the trouble report, type of work, repair performed, etc. These data are obtained from the craft person.

An NSA CTT needs only items (*ii*) and (*iii*), since it has no customer troubles. The CRAS combines these pieces by a key that consists of the Cable Trouble Ticket Number (CTTN) and a Wire Center (WC) identification. Much of the CRAS design exists to assure that these different pieces of data are matched together, and that errors cause as few problems as possible.

3.2 Relationship to LCAMOS

The CRAS system is one module of three that will together comprise the Loop Cable Administration and Maintenance Operations System (LCAMOS). In the future, the LCAMOS tracker module will be used to (*i*) track individual troubles (CTTs) that are OSP-related, (*ii*) correlate multiple individual troubles into a related trouble and track this item (as a CTT), and (*iii*) transmit data on completed CTTs to CRAS for further analysis.

In addition, the predictor module of LCAMOS will be used to analyze switching machine messages and predict likely cable problems before they are reported, i.e., it will help identify NSA troubles before they turn into SA troubles.

Although CRAS is the "back end" of the LCAMOS system, it is being introduced first, for several reasons:

(*i*) It improves, integrates, and replaces several existing systems or manual plans, i.e., CCUAP and CRFMP.

(*ii*) Its financial benefits are high and quickly identifiable, because it eliminates a great deal of clerical effort. Benefits from improved management of OSP repair are also expected, although they are less easy to quantify.

(*iii*) The LCAMOS tracker will be built on LMOS front end (FE) computers, whose software has been evolving rapidly over the last few years [see Ref. 1, part VI]. The software is now flexible enough to support the effective construction of the tracker.

IV. WORK FLOW AND INPUT FOR CRAS

4.1 Workflow with LMOS and CRAS

In the LMOS environment, a trouble report is received at a central location by a Repair Service Attendant (RSA) who inputs the report to LMOS through a cathode ray tube (CRT) terminal. The trouble report becomes part of the Basic Output Report (BOR) that is transmitted to the RSB that is responsible for coordinating maintenance on the affected customer line. The BOR also contains information on past troubles on the affected line, commitment time, a number at which

the customer can be reached, and the results of an automatic verification test on the line. The trouble is screened at the RSB and routed to the work center that should take responsibility for correcting the problem. The OSP repair craft are managed from a center that may be co-located with an RSB or may be a separate center that serves several RSBs. This center is currently called a Maintenance Center (MC). If the trouble should be routed to the MC, the RSB sends a BOR to the MC, using the LMOS Request Basic Output Report (RBOR) transaction, and a Cable Trouble Ticket Number (CTTN) is assigned to it. After service has been restored by the MC, the trouble must be removed from LMOS and the CTT data supplied to CRAS.

In the LMOS environment, a customer trouble report is closed out by a Final Status Transaction (FST) at the CRT terminal. In the LMOS/CRAS environment, if an OSP trouble is being closed, the person who closes the trouble is automatically reminded that CTT information is required. A Service-Affecting (SCTT) mask may be requested and displayed on the CRT to receive information on the location and cause of the trouble.

The close-out procedure described has the advantage of encouraging individuals in work groups other than OSP maintenance to enter the necessary CTT information in the CRAS/LMOS database. This is important because not all OSP repair work is actually done by OSPM forces, and yet, OSPM management needs to know where such repair work is being done to permit effective analysis.

Information on routine or NSA problems is also supplied to CRAS. The data entry procedure is similar to that for SA troubles, except an NSA (NCTT) mask is requested and displayed in place of the SCTT mask.

4.2 Interaction of CRAS and MTR

Each BOC has built a computerized time reporting system that collects all data on time charged to various work activities, then supplies that data for payroll computation and for other systems. Mechanized Time Reporting (MTR) is the generic name for these time reporting systems, which differ from company to company. Mechanized Time Reporting supplies the hours expended on various OSP activities to CRAS, to ensure accurate records with minimal input.

All craft technicians, including members of installation, business, coin and residence repair, construction, and other groups charging time to OSP repair ("R") accounts, must have a CTTN for each trouble case. All such hours are accumulated from craft persons' time reports and supplied to the MTR system.

The information derived from MTR provides data on the number of trouble cases and on the number of hours expended per trouble case.

Several LMOS/MTR/CRAS edit procedures provide users with discrep-

ancy reports that identify most input errors. These reports enable the user to correct errors before they are propagated through the system and to prevent similar input errors from being repeated.

V. CRAS ARCHITECTURE AND DATA FLOWS

5.1 The LMOS front end and host computers

Figures 2 and 3 display the hardware architecture of CRAS and major data flows. Both MCs and RSBs use CRTs connected to an LMOS FE computer, which handles some transactions directly (such as the FST) and passes others through to the LMOS host. The CRAS SA and NSA CTT entry transactions are of the latter type. Each adds one record to the corresponding CRAS host database (SACTT and NSACTT). Host batch programs obtain customer trouble data from the LMOS Trouble History (TH) database,[2] and through several steps, attach their data to the corresponding SA CTT records. The MTR data are read and processed on the host also.

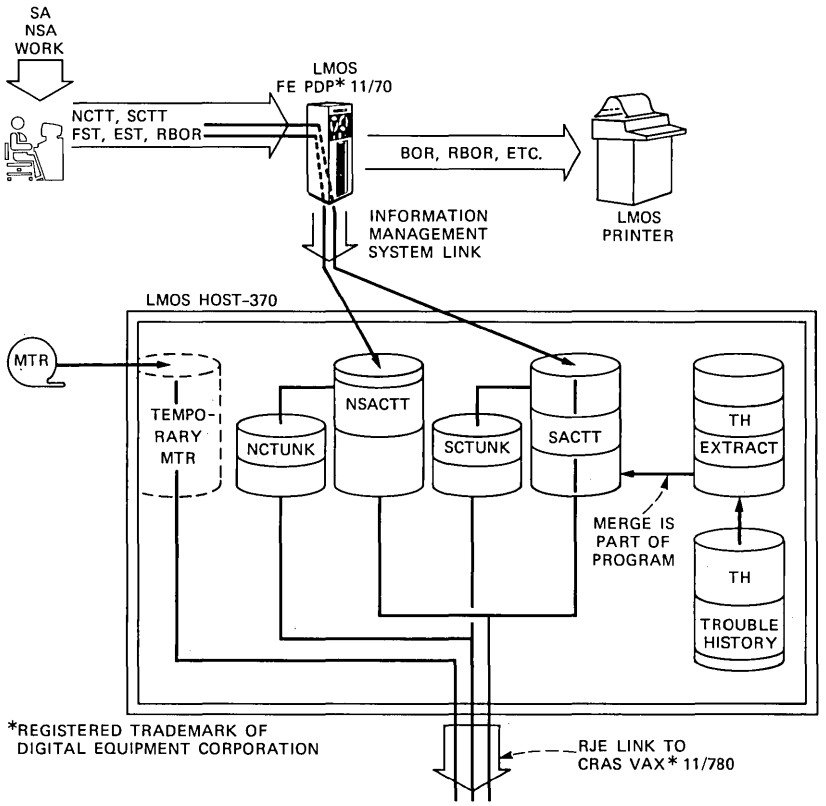


Fig. 2—The CRAS use of FE and host computers.

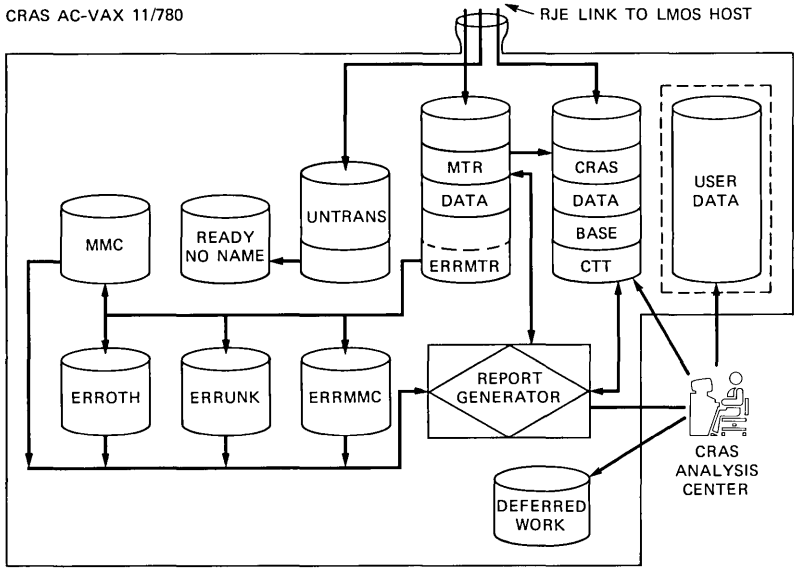


Fig. 3—The CRAS administrative computer.

5.2 The CRAS administrative computer

All the data from the previous stage is passed across a Remote Job Entry (RJE) link to another system, called the CRAS Administrative Computer (AC). See Fig. 3. The AC is a VAX-11/780 that runs under the *UNIX** operating system. Most of the CRAS capabilities are provided by the AC. These include matching of MTR data to CTT data, production of all reports, control of all CRAS host jobs, on-line documentation support, and all system administration. The AC is accessed by dial-up terminals, most frequently by MC analysts, but also by other managers and clerks.

Figure 3 shows the databases used on the AC. The CTT database has "good" CTTs, both SA and NSA, from which most trouble analysis reports are generated. Here, a "good" CTT at least possesses a legal wire center number.

The UNTRANS database includes CTTs that are "stuck" on the host, either because they contain illegal wire center numbers or because they are SA CTTs waiting to accumulate LMOS customer troubles. It also includes LMOS customer troubles not yet matched to CTTs. If many unmatched LMOS troubles are shown, CTTs are being entered incorrectly, very late, or not at all. The UNTRANS database contains images of two virtual databases (SCTUNK and NCTUNK) and a real one (TH

* Trademark of Bell Telephone Laboratories.

EXTRACT), all of whose primary copies reside on the host. The data are copied to the AC to aid the error reporting and correction process.

The MTR database contains hours data not yet matched into the CTT database. This data recirculates until it is matched correctly or ages enough that it is unlikely to ever match. The CRAS system aids matching by assuming that a CTTN is correct, but that someone specified a wire center found within the organizational boundary, but not the correct wire center. For some reason, this error turned out to be the most common one found, not the more expectable ones, such as CTTN digit transposition or omission.

The MMC database contains OSPM employees' payroll data that cannot be attached to CTTs. These include items like vacation, sick time, nonpaid absence, and any other data that are necessary to have a complete picture of OSPM employee's use of time.

Several databases (ERROTH, ERRUNK, and ERRMMC) hold MTR data known to be in error, categorized in ways to identify the sources of error. For example, CRAS maintains lists of organizational responsibility codes whose time records should be examined. If a time record appears that contains such a code, it is saved, even if it contains an unknown wire center or an improper CTTN. This assures the OSPM manager that even incorrect data entry by any relevant organization can be detected.

The CRAS system uses lists of employees, organizations, and wire center numbers to scan MTR data and save relevant records. Anything that does not match one of these lists is discarded immediately. Some spurious matches occur from data entered by parts of a BOC that do not yet have CRAS. These are kept long enough to make sure that the tables are correct, then they are discarded also.

5.3 Design commentary

The CRAS architecture is more complex than one might wish. Some of the complex matching procedures are only temporary, because they will be eliminated with the arrival of the LCAMOS tracker (Section 3.2). The AC database complexity arises from trying to make sense of data that arrives in no guaranteed order, and that may have errors that cannot be mechanically corrected. The CRAS system did not originally support the error databases. These databases seldom contain much data, so that people often do not bother using fix-up procedures to empty them. However, they must exist, if only to permit detection of consistent sources of errors, and to provide confidence that time records are being saved properly.

A host-only architecture was kept as an alternative until late in the design process. The separate AC design was chosen for several reasons:

(*i*) It permitted a faster schedule, since the *UNIX* system's tool

kit and environment fit the problem well. Database requirements fit the *UNIX* system's hierarchical file system well. The low ratio of updates-to-accesses permitted simple approaches to reliability that fit into a standard *UNIX* environment. For example, in any given day, only 2–4 percent of the database files are likely to be updated.

(*ii*) It offered the flexibility of the *UNIX* system's tools, which was necessary to survive expected changes in requirements.

(*iii*) Rough economic estimates did not clearly favor either approach over the other.

VI. OBJECTIVES AND OUTPUTS OF CRAS

6.1 Objectives

Many systems exist mainly to keep an accurate, up-to-date database from which any record can rapidly be retrieved. On the other hand, CRAS is a decision support system, whose value lies not in supplying records of data for immediate use, but in extracting relevant patterns from masses of data to support effective decisions, e.g., by ranking areas of plant by maintenance costs. It does a better job when it provides the least output to isolate problems. The objective of CRAS is to have managers use this information to improve service results, ensure good performance, and reduce cost.

6.2 The CRAS reports

The CRAS system provides about 40 reports, split into the five categories described below.

6.2.1 Outside plant trouble analysis reports

The CRAS OSP trouble analysis reports are directed at the following:

(*i*) Identification of locations with high customer report rates.

(*ii*) Identification of areas with high maintenance costs.

(*iii*) Identification of all SA and NSA OSP troubles by geographic location, type of trouble, type of work, and average clearance rate.

(*iv*) Identification, by work group, of average time-to-restore service.

6.2.2 Force management analysis reports

The CRAS force management reports are designed to provide measurements of such things as:

(*i*) Individual craft performance for the OSP maintenance organization in terms of hours expended per cable trouble case, percentage of trouble found, and types of work accomplished.

(*ii*) Performance results in terms of hours per trouble report by all groups—OSP maintenance, construction, business, coin and residence repair, installation, or others charging time to OSP "R" accounts.

(*iii*) Identification of hours, including overtime, and cases where two or more people worked on the same trouble assignment.

(*iv*) Identification of hours expended and resultant performance by repair category and report source.

6.2.3 Deferred work reports

Deferred work is a listing of NSA tasks that have been identified or started but have not yet been completed. These tasks can be the result of partially completed assignments, such as temporary closures, terminal replacements, etc., or potential problems that have been noticed by BOC employees. The CRAS deferred work reports are designed to provide the following:

(*i*) A temporary closure log that summarizes temporary closure activities for a given period of time, usually one month.

(*ii*) A listing of NSA maintenance tasks that can be completed on a scheduled basis.

6.2.4 Budget reports

This category of reports assists management in budgeting and scheduling tasks by providing (*i*) a budget report that provides an analysis of the hours expended, by type of work, and (*ii*) a report that allows managers to determine work schedules by analyzing work loads in terms of days of the week and times of day.

6.2.5 Administrative reports

The CRAS administrative reports are used to monitor and correct databases. They display the following types of information:

(*i*) Database completion summary, which shows the level of database completeness, i.e., what fraction of the CTTs have acquired MTR hours and LMOS trouble records.

(*ii*) Missing geographic identifiers (low-level identifiers, which must be added to CTTs by map lookups, and sometimes must be added much later than original CTT entry time).

(*iii*) Incomplete CTTs and unmatched LMOS troubles, used for database fixup.

(*iv*) Invalid tickets, i.e., those whose data are correct according to LMOS or MTR, but that contain errors noticeable to CRAS, which has tighter error checking.

(*v*) Miscellaneous administrative databases, such as employee lists, organizational hierarchies, etc.

(*vi*) The LMOS host run summary, which shows data flows between LMOS host and the AC.

6.2.6 Miscellaneous features

The reports are all implemented as shell procedures, which use both existing tools (especially *sort*, *awk* for report computation, and *nroff* for formating) and those newly written for CRAS. A powerful report compiler exists to select and display CTTs according to complex criteria.

Many reports include statistics on the completeness of their own input data. This is a necessity in an environment where it is too expensive to get perfect data, but where people want to know just how complete the data are. Thus, CRAS is able to provide good decision support from partial data.

VII. FEATURES OF CRAS AND ITS DEVELOPMENT

7.1 Team approach with computer assistance

The CRAS development has always attempted to include an integrated team of systems engineers, human performance engineers, software developers, managers, and end users. With rare exceptions, everyone (including many who had not used the *UNIX* system before) has used the machine-enhanced communication facilities provided by the *UNIX* system. Strong efforts have been made to use the system to provide leverage for human effort, not only in generation and control of code and documentation, but in multisite communication, exploratory requirements analysis, and product distribution.

The structure of the software was strongly influenced by the nature of the personnel involved. No single person had all the knowledge necessary to do the project. The software development supervisor was both newly-promoted and new to the ARSB. Only one member of the original development group had any *UNIX* system experience; several members had neither host nor *UNIX* system experience; and a few had no software implementation experience at all. Much of CRAS was built by managers who were on sabbaticals to learn software development. Some new person always had to be brought on board. Thus, the software structure had to consist of small, easily understood, relatively independent modules. In support of the educational function, people were taught to read each other's code, to steal it when possible, and to trade it back and forth as appropriate. Thus, while there was always individual responsibility for code, there was seldom individual ownership in any restrictive sense. *UNIX* system tools were used to keep track of the activity and automate drudgery so that people could get on with the job.

7.2 Tools

When attempting to build a project quickly, under conditions of

change and uncertainty, use of tools and other existing code is a necessity, if only because any prudent project manager avoids unnecessary risk. The nature of the CRAS AC permitted the use of the standard version of the *UNIX* system, so that expensive operating system changes could be avoided. Existing *UNIX* system tools were used heavily, to the point that most of the system's visible function is provided by the *UNIX* system's command language programs (shell procedures). Code sizes in the first issue of CRAS were as follows:

Lines	Type
16K	PL/I
10K	C
15K	Shell
6K	Miscellaneous
33K	Documentation

Some of these figures are misleading, since heavy use is made of program generators that operate on the source code stored above to produce the compilable source code. For example, CRAS includes a package of generators for PL/I access routines that are used to simplify the use of the host's Information Management System database system. Some product code and most development procedures were adapted from previous projects.

Tool benefits included cheapness of construction, speed of implementation, ability to demonstrate function quickly, and the chance to cope with surprises.

7.3 Data

From the beginning, CRAS development used a simple data dictionary system, which consisted of a text file of data item descriptions, plus several shell procedures for its manipulation. The dictionary was used to generate PL/I, C, and deliverable documentation. Use of a complex data dictionary system appeared unnecessary; use of some data dictionary from the beginning has proved invaluable.

The CRAS AC stores data in "packet" format, which has also been used in the new LMOS FE software. A packet is an abstract data type, which represents a collection of self-identifying data items. For CRAS, the benefits of packet use were as follows:

(*i*) It was easy to add fields, because programs that do not need the new fields need not be recompiled.

(*ii*) Storage was saved when fields varied greatly in length, or were missing entirely; CRAS has numerous such fields.

(*iii*) It was easy to write general tools to process packets.

When stored on disk, each CRAS packet is represented as a line of text ended by a new line. All but the largest CRAS packets can be

manipulated directly by many *UNIX* system tools (such as *sort*, for example). This was especially helpful in the early stages of development, since occasionally used functions could rely on the existing tool kit, rather than requiring expensive development. For example, the *UNIX* text editor was often used as a database patching utility.

7.4 Documentation and planning

The CRAS system provides on-line documentation that includes an unusual degree of automation from development through distribution to end usage. It also provides an advance planning system whereby BOC users may dial a *UNIX* system, look at preliminary documentation, try out report retrieval on a test database, and use hardware sizing programs. This work is reported in detail elsewhere in this issue.[3]

VIII. DESIGN ISSUES AND PHILOSOPHY

The following discussion highlights some of the design issues faced during the implementation of CRAS.

8.1 Relationships with other systems

The CRAS system was required to interface with both the LMOS host and MTR. Previous systems have used self-reporting for both customer trouble counts and payroll hours. For the sake of accuracy, CRAS was required to obtain these data items from their true sources. In addition to the new CRAS software, CRAS implementation required a few changes to the LMOS FE, new input edits, and other software in MTR. The CRAS system also provides data to the Loop Activity Tracking Information System (LATIS). It is well known that any system interface must be examined for potential problems.

As a system designed to improve, integrate, and replace existing systems, CRAS was subject to some constraints regarding compatibility with existing systems. Because of the sensitivity of some of the CRAS data, upon which people's performance ratings may depend, people must retain confidence in the outputs of CRAS. They assure themselves of its accuracy by comparing its outputs with those of other systems. Thus, CRAS was subject to constraints such as the following:

(*i*) When counting customer troubles, it should be consistent with TREAT.

(*ii*) When counting cable troubles, it should be consistent with CCUAP and should be able to produce all existing CCUAP reports.

(*iii*) When counting hours worked, it should be consistent with MTR internal reports and with the methods of the CRFMP.

(*iv*) All its own reports should be consistent.

These constraints represented great potential for surprise, especially where the counting rules of different systems were imprecisely speci-

fied. Even worse, some of the rules were inherently inconsistent. The CRAS system was the first to try to put these particular pieces of data together, so the task of discovering the problems fell on it.

8.2 Data asynchronism

As noted earlier, a CTT contains up to three kinds of data (CTT mask, customer troubles from LMOS, and hours from MTR). Data validation would be helped if these pieces of data arrived in a consistent order. Unfortunately, every possible arrival sequence can happen in practice. The CTTs are usually entered during the day, and customer troubles (closed by the LMOS FE's FST transaction) are transmitted to the host at night, and then can be attached to the CTT. However, FST is not supposed to be used until the customer has been notified, which implies that some customer trouble records straggle in over a period of days. When there is an overload of customer troubles, CTTs are often held and entered later, so that trouble records arrive before CTTs. Depending on local policies, MTR records arrive before or after either of the other pieces.

Even if a fixed sequence were provided by normal operations, computer failures would certainly disrupt such a sequence. Thus, CRAS must tolerate all arrival sequences, and it must be immune to machine failures. It also must handle partial CTTs, as when a CTT never acquires MTR hours because of coding errors. These issues were thoroughly addressed and handled by the design.

Also addressed was the need for an unusual type of multisource error detection, which depends on examination of groups of related data to find patterns, rather than simple errors in individual items. For example, it is difficult to know whether two identical CTTNs exist because one is an extra (and can be deleted), or because one contains a typing error (and should be edited to renumber it), without looking at both together.

8.3 Change and uncertainty

During the period when CRAS was being implemented, changes were occurring and were expected to continue in the recommended organization of Bell System repair activities.[4] Some of the change and uncertainty arose from anticipated regulatory changes; other parts of it came from attempts to improve the repair process; some arose in the process of ARSB evolution.[1]

The CRAS system would often be required to deal with organizational combinations that differed from BOC to BOC and changed over time. When CRAS was started, independent MCs were only starting to come into existence, and did not exist at the field trial company.

All the above argued strongly for an emphasis on flexibility.

8.4 Implementation speed and maintainability

For various reasons, CRAS needed to be built and deployed quickly. Thus, there was pressure to get something working quickly. However, the existence of change implied that rigid, unmaintainable software would not survive.

8.5 Philosophy

Given the complex nature of the environment into which CRAS had to fit, and the need to survive continual change, CRAS design strategy assumed that perfect requirements would never be available. The resulting philosophy included the following:

(*i*) Think small, for the complex environment will stretch even small software into growth. Figures 2 and 3 offer a good example, since several of the databases came into existence only during the field trial.

(*ii*) Build a system quickly, thus lessening personnel turnover and requirements changes caused by change in the external environment.

(*iii*) Build a system that can be changed quickly.

(*iv*) Build a prototype, get it into the field, make it evolve, and make requirements converge on reality. For CRAS, it would not be feasible to discard the prototype, so it had to evolve.

IX. LESSONS

9.1 Team approach and philosophy

The approach of using an integrated team, with strong machine assistance, has worked well. In fact, the most troublesome areas have been those where we could not integrate people or functions into the environment shared by everyone else. For example, although documentation was an integral part of the system from the beginning, it was much more difficult to merge training into the machine-supported product.

The "small is beautiful" development philosophy worked well, in that it helped produce a usable product quickly. It helped CRAS survive the unusual amount of turmoil caused by major rearrangements of every organization involved in CRAS development. The idea of getting into the field quickly worked well, although the CRAS field trial probably started about three months too early. We assumed that some functions were needed to start the trial, and that others could be developed as we continued. All of the former were ready at the start, but some of the latter were actually needed earlier.

9.2 Systems engineering and human performance engineering

In a tools-based project that is moving quickly, it is important that Systems Engineering does not "throw the requirements over the wall"

and then go do something else. The CRAS software developers suffered somewhat from not having enough systems engineers to try ideas with, when building software prototypes. Eventually, more systems engineering support was assigned to report to the development supervisor, who was then acting as project manager. The resulting team synergism yielded rapid solutions to problems, and a generally satisfying working relationship.

When the software developers are using good tools, and when the product structure is a tool-oriented one, the result implies that more of the staff resources must be placed in Systems Engineering and Human Performance Engineering. When it is easier to create software, a higher percentage of effort must go into knowing what it should do, not how to build it.

9.3 Other systems as data sources

If data are received from another system, and if the correctness of that data is of only marginal importance to the other system, it should be expected to contain many errors.

9.4 Complexity of environment

Organizations that have complex operations tend to evolve complexes of computer systems to help them accomplish their work. Few organizations have been able to foresee all contingencies and provide totally integrated, comprehensive systems on a timely, cost-effective basis. If something new is implemented as a stand-alone system, it is irritating to its users because it almost invariably needs data from another system, or needs to give data to another. Any new system faces an increasing number of possible interfaces, compatibility constraints, and schedule interactions.

A particular "catch-22" exists in the area of schedules. Suppose newly-planned system A needs data from already-deployed system B, and the two are controlled by different organizations. If the developers of A request changes in system B long in advance of possible deployment, they may face indifference, if only because the developers of B probably have a long list of pending change requests anyway, and cannot be sure that system A will actually ever work, or if it does work, will arrive on schedule. On the other hand, if system A is far enough along to obtain high priority from B, it may be too late to get enough real data to use for testing A, especially where the crucial need is to know the types and frequency of errors.

Another schedule difficulty arises when long lead times are required to obtain changes in other systems or procedures. Sometimes a prototype must be built quickly to discover problem areas in other systems early enough that they can be changed in time to support a later

production system. A major problem faced by CRAS was the continual discovery of inconsistencies among other systems.

Complexity problems also exist in the number of organizations who own databases or systems and must be involved in planning of new systems. It is well known that multiplicity of organizations contributes to the difficulty of getting and keeping good requirements. They have needs and perspectives that contain legitimate differences, and whose relative priorities can be difficult to compare.

Such problems are hardly the property of any given organization, but are ones to be faced by any new system. It is ironic that the earlier and faster an organization computerizes, the more difficult will be the planning and implementation of later systems.

X. CONCLUSION

We have described the design and development of a successful operations system, whose history illustrates the problems to be overcome when building software in a complex and rapidly changing environment.

XI. ACKNOWLEDGMENTS

Many people contributed in the development of CRAS. We particularly thank J. A. Bayer, T. R. Schiller, J. Burtoft, E. A. Overstreet, and D. Kurshan (Systems Engineering), R. J. Glushko, G. T. Vesonder, K. A. Wright, and J. E. Zielinski (Human Performance Engineering), and M. Baade, W. F. Bauer, M. H. Bianchi, B. J. Beare, B. L. Cruse, C. M. Franklin, S. D. Rhodes. J. H. Shoemake, and R. O. Sinclair (Software Development). We are also grateful for the assistance of AT&T (especially T. Ballen, D. Webster and W. Bassham), Western Electric Company, and New England Telephone & Telegraph Company.

REFERENCES

1. S. G. Chappell et al., "Automated Repair Service Bureau: The Front-End System," B.S.T.J., this issue.
2. C. M. Franklin and J. F. Vogler, "Automated Repair Service Bureau: Database System," B.S.T.J., this issue.
3. R. J. Glushko and M. H. Bianchi, "Automated Repair Service Bureau: On-line Documentation: Mechanizing Development, Delivery, and Use," B.S.T.J., this issue.
4. L. S. Dickert and S. P. Rhodes, "Automated Repair Service Bureau: The Trouble Report Evaluation and Analysis Tool," B.S.T.J., this issue.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Human Performance Design Techniques

By G. H. LEONARD and J. E. ZIELINSKI

(Manuscript received July 6, 1981

Successful implementation of the Automated Repair Service Bureau (ARSB) Systems in the Bell Operating Companies is the result of the early integration of a variety of disciplines in the development process. This paper provides an overview of one of the basic human performance design techniques and an example of its application in the design of the Mechanized Loop Testing System. The expanding role of human performance design in the ARSB systems is also reviewed.

I. INTRODUCTION

In this issue, individuals from the disciplines of software, hardware, and systems engineering relate the design and development of ARSB systems from their perspectives. In this article, we discuss one of the functions that psychologists perform in the design and development of ARSB systems. This particular function is labeled personnel subsystem development (PSD), which means the integrated design of those factors that affect human behavior in the system, including the design of manual procedures, human-machine interfaces, training, performance aids, and documentation as part of the total system.[1] The PSD process also includes the systematic testing of the personnel subsystem prior to the initial field installation. As an illustration of PSD, we will first give a synopsis of the process and then relate the process as it was applied to the Mechanized Loop Testing System (MLT).[2] Our intentions are not to relate a "how-to-do-it" guide but to illustrate the approach and emphasize that system development is not just software development but the integrated design of hardware, software, and human information processing components.

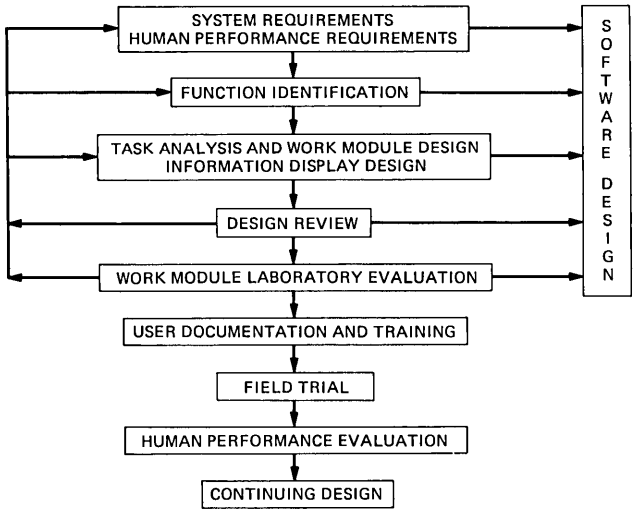


Fig. 1—Automated repair service bureau human performance design process.

II. OVERVIEW

Consideration of human performance in the MLT system began with the functional requirements for MLT and continued through system design, development, installation, and enhancements. An overview of this process is shown in Fig. 1. As system requirements were defined, the functions to be performed by people were defined, and aspects of the system which would affect human performance requirements were identified. In the MLT system, these manual functions included such activities as trouble analysis and system maintenance. Specific tasks were then identified to describe all the manual activities within each function. Related tasks were combined into work modules with specified inputs, outputs, and performance objectives. These work modules could later be combined to form complete jobs in the telephone company environment. At this point, we also provided initial specifications for the design of the information displays (CRT displays, printouts, forms, etc.). With preliminary versions of system operational features and human/machine interfaces, we conducted a formal design review with systems engineers and hardware and software designers. This design review provided the first opportunity for everyone involved in the design process to review the proposed system operation from the user's perspective.

When the initial design was complete, we conducted laboratory evaluations of work module design and preliminary documentation and training under conditions simulating actual system operation. For these evaluations, telephone company personnel who met the specified

minimum skill and knowledge requirements received training on the new activities. During the simulation, the participants processed typical inputs as they would on the job. Performance data were collected during the simulation, and the participants were interviewed to determine their subjective reactions to the system design, training, and documentation. If data from the evaluations indicated that redesign was required, the procedures and human/machine interfaces were redesigned prior to the development of final user documentation.

This design process was highly iterative, with changes at each phase affecting design decisions made earlier, as shown in Fig. 1. But the most significant effect of this approach was the impact on the design of the software. At each phase, as various design decisions were reviewed or tested, many recommendations for software changes were made and implemented to improve the total system. This process illustrates a view of the system as an integrated unit of people and machines processing information to achieve stated objectives. This view of the system requires that human performance considerations be integrated with software design from the earliest phases of system development.

The final stages of the human performance design process for MLT involved the preparation of user documentation and training for the field trial. We then trained the Repair Service Bureau personnel at the trial site and monitored their on-the-job performance during the field trial. Follow-up activities included field evaluations of human performance within the system and recommendations for design changes to improve overall system operation.

III. EXAMPLE FROM MLT SYSTEM DESIGN

To illustrate the impact of this design process on the MLT system, we will describe the initial design and subsequent redesign of a specific manual function. As a result of the initial examination of system functions, the trouble analysis function was defined to include all the manual activities that must be performed when a trouble report is first printed in the work center. These activities consisted of the examination and integration of the trouble description, line record information, and automated test results to determine what the next stage of processing should be.[3] As a result of the initial task analysis, one work module was designed to accomplish all the activities in the trouble analysis function.

The primary input to this work module was the Basic Output Report (BOR) containing line record information, the trouble description, and detailed test results. The output of this module was a decision to route the report for either manual testing, additional automated testing, dispatch, or close out. Some design work had previously been done on

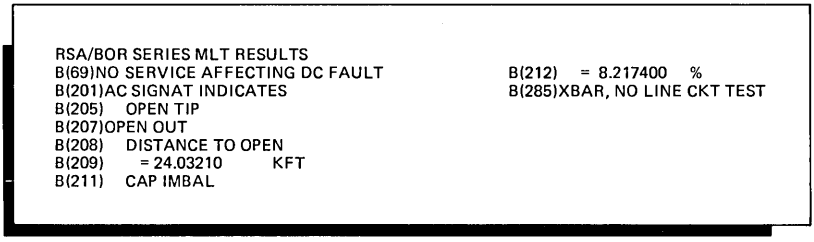


Fig. 2—Original test results format.

both the content and the format of the test results displayed on the BOR (Fig. 2), and this was accepted as the initial human/machine interface design.

This initial design of the test results output and trouble analysis procedures was based on two major assumptions. First, we assumed that in-depth interpretation of *all* the detailed test results would be required for all trouble reports. Therefore, very little interpretation of the raw test results was provided by the software. All results were displayed, along with software codes, in the order in which they were obtained during the mechanized testing process. We also assumed that a person's prior trouble analysis experience in the manual Repair Service Bureau, plus some limited training in the new test results, would enable him or her to accurately analyze and process trouble reports containing MLT results. Therefore, the original training focused on providing "translations" of the new test results into familiar terms, with little procedural instruction in the integration of the test results with other information on the BOR to make trouble report routing decisions.

To gather performance data to validate these assumptions, we conducted a laboratory evaluation of this work module. Four operating telephone company craft employees who met the specified minimum requirements were given the work module training. Then they processed trouble reports (BORs) selected to provide a valid sample of the trouble descriptions, line records, and test results found in typical Repair Service Bureau operations. Analysis of the performance data revealed that the participants incorrectly routed 16 percent of the trouble reports, and most of these errors involved routing troubles for manual testing when the MLT test results provided sufficient information to directly dispatch the trouble report. Since a major goal of MLT was to reduce the number of troubles that required manual testing, this error rate would have a significant adverse effect on overall system operations and economics.

Based on the performance data and comments from the evaluation participants, we reexamined the initial assumptions and redesigned

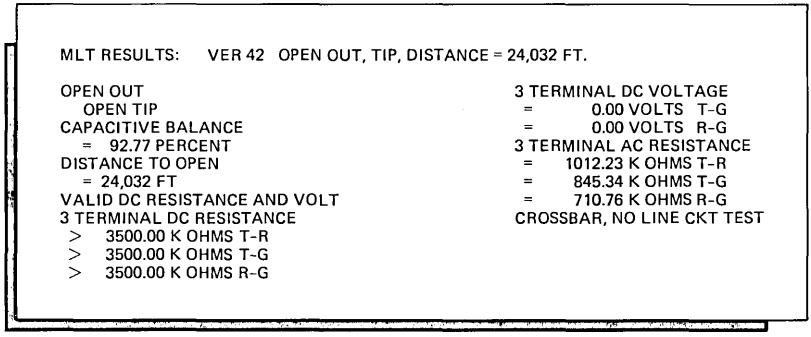


Fig. 3—Test results format—field trial.

both the human/machine interface and the analysis procedures. It was evident that some interpretation of the detailed test results could be accomplished by a software algorithm and that a brief summary statement could be provided at the beginning of the full test results (Fig. 3). Therefore, many routing decisions could be based solely on the summary without detailed examination of all the test results. In addition, extraneous information not needed for trouble analysis (e.g., software codes, nonsignificant digits) was eliminated. The detailed test results describing the fault condition were presented at the beginning of the results to facilitate trouble diagnosis. Training and documentation of the analysis procedures were expanded to include decision guidelines in a performance aid for work center personnel.

We expected that the redesigned procedures and output format would enable a clerical level person to process many trouble reports. Also, fewer trouble reports would be routed incorrectly, thus reducing the need for manual testing. Data from a follow-up field evaluation confirmed these expectations concerning the effectiveness of the redesigned procedures and test results. In some work centers, clerical people were processing all of the trouble reports, with fewer instances of incorrect routing than were found in the laboratory evaluation.

Continuing engineering of the MLT system has resulted in refinements and improvements in the testing hardware and the software algorithms that control testing. Similarly, follow-up studies of field installations have provided more detailed information on the relationship between various line fault conditions and specific MLT test results. This information on the operational use of the system has been used to enhance the software interpretation of the detailed test results to provide over 50 additional test-result summaries. Based on the results of field evaluations of several MLT installations and a laboratory study comparing human performance on alternative test-results formats,[4] the display format has also been redesigned, as shown in Fig. 4. This

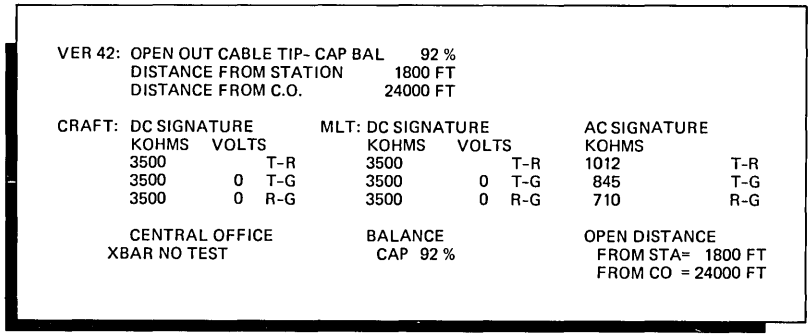


Fig. 4—Proposed test results format.

new format provides most of the test results information required for rapid trouble analysis in the summary. In addition, the summary will include the information to be relayed to outside repair technicians on dispatch.

IV. COMMENTS

The process of human performance design begins with and not after the development of system requirements. Human performance considerations begin not with a review of the requirements document but with an understanding of what the system is supposed to accomplish and the examination of alternative approaches.

The role of human performance designers in the design and development of the ARSB systems is continuing to grow along with the evolution of the systems. The initial systems were designed to mechanize the more routine, repetitive tasks performed by clerical or craft personnel (e.g., trouble report tracking, initial line testing). With the mechanization of many of these routine activities, more recent ARSB systems are addressing more complex craft and managerial activities. For example, the next generation of the MLT system will provide computer-assisted loop testing and analysis currently performed by experienced craft persons using old, but functional, work positions. The complexity of the information processing activities requires that the design of all system components be closely integrated.[5] Similarly, systems such as CRAS and Predictor are used by managers to support their decision making in such areas as plant analysis, force management, productivity, and budgeting.[6] In the design of these new decision support systems, the psychologist as designer and developer will play a major role, since these systems now address the issues of problem solving, direct managerial use of the system, design of the system for group problem solving, the incorporation of models and artificial intelligence to support decision making, the evaluation of systems in

supporting decision making, and a host of other complex human-computer issues.

As the role of human performance designers increases, so does the variety of tools and techniques applied during the design process. The basic systematic design process illustrated here is employed in the design of ARSB systems. In addition, laboratory experiments and field studies have been conducted to address specific design issues.[6] Areas such as job quality are being examined, both in existing ARSB systems[5] and in current system design efforts.

For the development of ARSB systems, it has been effective to dedicate and organize psychologists on a group level to a particular system development effort. In their system design and development work, these people draw on numerous branches of Psychology and Engineering, including Experimental, Social, Cognitive, and Organizational Psychology, Industrial Engineering, and Human Factors. This dedication and continuity of effort ensures that systematic human performance design takes place as opposed to having the psychologist intermittently critique the development efforts with the result of a retrofitted system at best.

REFERENCES

1. H. O. Holt and F. L. Stevenson, "Human Performance Considerations in Complex Systems," Science, *195* (1977), pp. 1205–9.
2. O. B. Dale, T. W. Robinson, and E. J. Theriot, "Automated Repair Service Bureau: Mechanized Loop Testing Design," B.S.T.J., this issue.
3. P. S. Boggs et al., "Automated Repair Service Bureau: Evolution," B.S.T.J., this issue.
4. T. S. Tullis, "An Evaluation of Alphanumeric, Graphic, and Color Information Displays," Human Factors, *23* (1981), pp. 541–50.
5. R. F. Gauthier and W. A. Harris, "Automated Repair Service Bureau: Two Examples of Human Performance Analysis and Design in Planning the ARSB," B.S.T.J., this issue.
6. P. S. Boggs and J. R. Mashey, "Automated Repair Service Bureau: Cable Repair Administrative System," B.S.T.J., this issue.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Two Examples of Human Performance Analysis and Design in Planning the ARSB

By R. F. GAUTHIER and W. A. HARRIS

(Manuscript received June 19, 1981)

Two case studies are described in which human performance methodologies were applied during the development of the Automated Repair Service Bureau (ARSB). The first case study illustrates some of the human performance analysis and design conducted during the specification of the computerized work station that would replace the local test desk in Repair Service Bureaus. The second case study shows how job quality considerations led to improvements in the job of the Repair Service Attendant who receives initial telephone trouble reports from customers. At the same time, a net savings in the time to process the troubles was achieved.

I. INTRODUCTION

This paper discusses some of the procedures and results of human performance design and analysis work carried out during the development of the Automated Repair Service Bureau (ARSB). It illustrates several ways that human performance analysis has benefited the design and operation of the ARSB. Two case studies are presented:

(*i*) Analysis and design for the computerized work station that replaces the local test desk for testing telephone lines in telephone repair bureaus.

(*ii*) Application of job enrichment techniques to the Repair Service Attendant's (RSA's) job in the ARSB.

II. PROCEDURES FOR HUMAN PERFORMANCE ANALYSIS AND DESIGN IN THE ARSB

2.1 Background: Use of the local test desk in the pre-ARSB environment

The local test desk is an electromechanical console characterized by

analog meter scales, toggle switches, electrical cords, and flashing buttons. Its basic testing modes of operation have been in use, virtually unchanged, for over forty years. It includes both testing and communication capabilities, and permits several telephone lines to be accessed simultaneously for testing.

In the pre-ARSB environment, skilled technicians called testers, working at the local test desks, perform tests on telephone lines to establish probable causes of trouble on a line. They work from paper trouble reports passed to them by repair clerks who receive customer complaints. Testers also assist craft persons in the field who call in, requesting tests on lines they are working on. The tester is the one who normally makes dispatch decisions and finally ensures that particular troubles are repaired. Telephone trouble reports requiring dispatching are routed from testers to dispatchers, who assign jobs to craft persons calling in upon completion of previous jobs.

Analytical skills, as well as interpersonal skills, are required for a person to be a successful tester. In addition, the tester has to be a good coordinator of the repair process. The tester is normally the one who is responsible for ensuring that a trouble is dealt with to the customer's satisfaction.

In the first phase of the introduction of Mechanized Loop Testing (MLT), the number of testers required at a repair bureau was reduced. The content of the tester's job tended to shift towards those tasks which MLT was not designed to do conveniently, such as testing interactively with craft persons in the field. The tester, when provided with a Video Display Terminal (VDT) giving MLT tests, often continued to use the local test desk as before.

The first phase of MLT was not designed to replace the local test desk entirely. This is principally because of the highly interactive nature of the tester's job and the power and flexibility of the local test desk in providing a variety of specific tests suitable for troubleshooting more complex telephone problems.

The systems engineering requirement to replace the local test desk with a VDT interface and appropriate communications capabilities came during the second stage of the implementation of the ARSB. This required that all testing be carried out under computer control.

2.2 Definition of the problem

When replacing one testing system with another, the general requirement is to ensure that the new system permits the functional capabilities appropriate to the new work environment. The human performance design goal is to help specify the user interface to the new system and to ensure that the tasks associated with the new system are suited to the abilities and needs of the users of the new system.

The MLT system specification required that we design a VDT interface that, together with a proper communications set, would be the functional equivalent of the local test desk. That is, the new work position would provide the testing and communications capabilities so that users could perform the functions that were previously performed at the local test desk. Of course, the absence of the local test desk, and the presence of a set of tests at the VDT that provides the equivalent or better testing capabilities, will permit operating companies to structure the processing of trouble reports in a more flexible way than is currently possible with the presence of the local test desk.

2.3 Rationale for analyzing the previous system

While the testing and communications features of the local test desk are known, we needed to know the way these features were applied in an operational environment, and which features were considered important or crucial by users. This knowledge would facilitate the transfer of testing capabilities to a VDT testing mode, and permit us to retain, where possible, the operational features considered to be the most advantageous.

We needed to understand fully what testers perceived to be the main advantages of the test desk. By this approach, we could attempt to incorporate as many as possible of these advantages into a VDT testing design. At the same time, we wished to make full use of the advantages of the VDT medium, as applied to testing areas.

2.4 Analysis of testing requirements

We decided to observe testers doing their jobs in a normal operational environment. This observational study approach was chosen to assess what types of tests were actually done at a local test desk, and the conditions under which testing was done.

Over a period of several months, we closely observed testers' activities in three different operating companies by sitting with and talking to testers in their normal course of work activities. One of our early observations concerned the highly interactive nature of the tester's job. Much of a tester's time is spent communicating and coordinating repair activity among a variety of telephone company personnel. Actual testing of phone lines is only one of a large variety of tasks that testers carry out.

It became clear that the design of the local test desk replacement would have to facilitate both the testing and the communications portions of the tester's job. Neither portion could be neglected. The interactive nature of the tester's job often requires the tester to work on several problems in parallel. A tester can be working on a problem with one craft person when a second craft person calls in requesting

assistance. If the second request can be dealt with quickly, the tester may take care of it, then return to assisting the first craft person. Otherwise, the second craft person, as well as subsequent callers, will be temporarily placed on hold until the tester has finished working on the earlier problems.

We observed a high degree of interaction between the tasks of testing and communicating. Testers often would be testing a line at the same time they were talking with the person who needed the test. We also observed some of the work pressures confronting testers on their jobs. Pressures could be caused by work load, ambiguities in the analysis of telephone troubles and deciding what actions to take, and delays caused by waiting to talk to other telephone company personnel. Also, working simultaneously on several troubles or having several other craft persons waiting on hold can further add to work pressures. We came to appreciate why one tester characterized his job as "fighting the test desk," although this characterization was mainly because of the pressures of interpersonal interactions, decision making, and time limitations rather than because of problems with the test desk itself.

2.5 Results and implications of observational study— guidelines for design

The preceding analysis of the tester's job yielded a set of design guidelines that will be used in creating a new work position and a new job category. The new job title will be Maintenance Administrator or MA.

The MA should be able to use the system flexibly for requesting interactive testing with craft persons and customers. Interactive testing requires that a craft person or customer participate in some way while a line is being tested. Voice contact between the MA and the craft person or customer is present during the interaction, usually both before and after testing the line. The system should as much as possible facilitate this interactive testing process so that it proceeds smoothly and efficiently. We decided that the required system for interactive testing would have the following characteristics:

(*i*) Required tests can be requested easily by the MA.

(*ii*) Appropriate communications and conference capabilities will be available.

(*iii*) The system will permit the request for interactive tests by MAs with craft persons or customers in as close to a "real time" mode as possible, i.e., interactive testing will not be held up unduly while waiting for test results from the system.

(*iv*) Talking and testing over the same line will be possible in a way that facilitates interactive testing.

(*v*) Several lines can remain accessed simultaneously and be sta-

tused appropriately so that the MA can conveniently work with several craft persons requiring intermittent assistance.

(*vi*) It will be possible to conveniently check detailed records for a line that is being tested.

(*vii*) The system will supply certain relevant line record information when test results on a line are returned.

2.6 The design of the human interface for testing and communications

(*i*) *General features*—The preceding testing and communications requirements led, after many design and redesign efforts, to a human-computer interface tailored specifically to the needs of the MA for requesting the tests formerly done at the local test desk. The requests available to an MA consist of test and information requests that permit testing to occur in a timely fashion in the normal working environment. The VDT display, or mask (called the TV mask, for Trouble Verification), developed for this purpose permits the MA to request these tests in a convenient manner. The TV mask provides a sophisticated visual display terminal or VDT interface for testing, as well as initiating certain communications with craft persons and subscribers.

A status region of the mask indicates all lines that are currently accessed by the system. For maximum flexibility, tests can be requested on any of the lines accessed. Up to five lines can be accessed at the same time so that, for example, tone may be placed on several lines, while the MA is working on another line. If the MA switches to another mask containing records, the original TV mask with all status information is stored and returned to the MA on request.

The communications console associated with the VDT will provide the MA with the necessary features for communications needs. Such features include conferencing capability, speed dialing of commonly used numbers, and appropriate sizing and arrangement of incoming and outgoing lines.

The VDT and communications console can be used in an integrated manner to allow the MA to talk and then request tests over the same telephone line, without having to drop and then reaccess the line. When the MA enters a telephone number on the TV mask and requests a talk line, the test system accesses the designated line for testing, and calls back the MA at the MA's work position over the communications console. When the MA answers the phone, the test system connects the MA's line to the line to be tested. The MA can then talk over the line, for example, to a customer or a craft person in the field, and then run tests on the line from the TV mask. When each test ends, the talk path is automatically restored so conversation can continue. This talking and testing path greatly facilitates the MA's job, which requires

a high degree of verbal interaction integrated with testing of phone lines.

(*ii*) *Specific examples of VDT test capabilities*—Here are three examples where an orientation towards the MA's needs and preferences resulted in appropriate testing capabilities being specified and implemented. All of these tests take into account the need for rapid system responses required for interactive testing with outside craft persons.

(a) *Tone*—Often a tester in a telephone repair bureau puts a tone on a pair to enable a craft person to find the pair by using a variety of tone detecting apparatus. When the craft person finds the pair, often the craft person wishes to signal the tester for assistance with testing the line. Here is how this is accomplished with the new system.

A craft person calls the MA requesting a tone on a certain telephone number to locate the pair of wires associated with that telephone number. The MA will type in the telephone number on the TV mask displayed on the VDT screen. The MA then types in TONE in a request field. The MA can then go on to testing other telephone lines, while tone is supplied to the requested telephone number.

When the craft person finds the telephone line with the tone on it and shorts that pair of wires, the MA's screen will signal this by a beep and brighten the status line on the screen corresponding to the appropriate telephone number. The MA can then request T (for Talk) on that telephone number. The tone will then be dropped automatically and one of the MA's phones will ring, providing the MA with a talking path over that telephone line to the craft person in the field. Further testing and talking over that telephone line can then proceed by the MA's entering appropriate test and communication requests from the TV mask.

(b) *Pair identification*—Often a craft person will identify a pair of telephone wires in the field by placing an electrical short or ground on the pair and asking a tester, who is accessed to a specific pair, if the tester "sees" the short or ground by an immediate deflection of the needle on the tester's electrical meter. The craft person may try to locate a particular telephone pair by shorting or grounding a number of neighboring pairs often in rapid succession. To permit this pair identification capability from a VDT, the ARSB system was designed to measure and report a shorted or grounded telephone pair in a "real time" mode, i.e., without the several-second delay required for a result to be reported from the remote test equipment to the VDT screen.

To get into this "look for short" mode, the MA enters a telephone number and requests the command S (for short) at the TV mask. This sets the test equipment in a continuous measuring mode for detecting a short or ground placed on a line by a craft person. At the same time, the system calls one of the MA's telephones, providing a direct connection between the remote test equipment and the MA.

As soon as the craft person places a short or ground on the pair corresponding to the entered telephone number, the remote test equipment detects it and sends a tone back to the MA over the telephone line. The MA, who is in voice contact with the craft person over another line, can then immediately tell the craft person that the correct pair has been contacted. In fact, the MA may direct that the system call back the craft person directly to provide tone information that indicates when the craft person has shorted the pair. The tone remains on until the craft person removes the short or ground that was applied, and reappears if the short or ground is reapplied. By design, the tone is not sent back to the MA for a preexisting short or ground on the line. Only a change of conductivity corresponding to the application of a short or ground by a craft person is signaled.

This "look for short" mode will be very useful for interactive testing with outside craft persons, where rapid response times are highly desirable. Without human performance design input at an early stage in the project, this useful capability would certainly not have been implemented in its present form. The capability required that specific changes be incorporated into the system design. These changes would have been more costly to introduce into the design at a later stage.

(c) *Quick single-ended test series*—Another way was found to meet the MA's need for rapid test results when interacting with outside craft people. A stripped-down version of the previously implemented full test sequence was designed. The full sequence is carried out on a line to characterize the full extent of problems on the line, and to compare observed results with stored records. We determined that much of this detailed information is not always necessary or relevant during interactive work with outside craft people. The new QUICK test that resulted is expected to reduce the time to get useful test results by a factor of four or so, to about 5 seconds as compared to about 20 seconds for the complete test sequence.

2.7 Summary

In the full ARSB environment, a new computerized work station will replace the local test desk. A variety of tests and information requests will be carried out by an MA from a VDT, with a communications terminal being used as appropriate to facilitate the testing and coordination process. The MA will also provide assistance to craft persons who call in with requests for tests or information on lines they are working on in the field.

2.8 Conclusion

In conclusion, the application of human performance approaches

during the ARSB system design stage yielded significant benefits. It provided the opportunity for human performance designers to contribute to an ultimate system design that would be efficient and effective from the final user's point of view.

Section III illustrates another area where human performance design considerations had an important effect on job design and job quality in the ARSB.

III. JOB ENRICHMENT IN THE CENTRALIZED REPAIR SERVICE ANSWERING BUREAU

With the implementation of Loop Maintenance Operations System (LMOS) in ARSBs, the job of the RSA was removed from individual Repair Service Bureaus (RSBs) and placed in Centralized Repair Service Answering Bureaus (CRSABs). Each CRSAB serves several RSBs. Centralizing the RSAs takes advantage of the efficiencies possible from combining several smaller work forces that perform the same function into a single larger unit.

The RSA is the initial telephone company contact when a customer calls to report a telephone service problem. The RSA works at a VDT and operates a keyboard, which is connected to a computer. A customer calling to report a telephone problem is automatically connected to an RSA. The RSA obtains the customer's phone number, and enters it into a "Trouble Entry" mask on the VDT. The computer responds by displaying on the terminal a Trouble Report mask. This mask provides the customer's name, the address for that particular phone number, and the date of the last trouble cleared.

The RSA verifies the name and address, asks the customer to describe the trouble, enters the trouble description into the Trouble Report mask, and determines whether the trouble requires a repair visit to the customer's premises. If a repair visit is needed, the RSA makes an appointment with the customer and gives a time of day commitment for repair of the trouble. When the customer hangs up, the RSA transmits the trouble report to the computer.

The computer forwards the trouble report to the RSB, where a printout is made available. The computer updates its history files and presents a new Trouble Entry mask so that the RSA is ready to answer the next customer's call.

3.1 The problem

One consequence of centralizing the RSAs that was not fully anticipated was the effect on the RSA's job. In the decentralized environment, the RSAs frequently had the necessity and opportunity to discuss trouble reports with other employees. They also had the opportunity to do a variety of tasks (fill work) when the number of trouble calls

was low. In the CRSAB environment, the RSA's job does not normally require them to converse with other employees and the work is more repetitive with less variety. For example, a busy day may require an RSA to handle up to 270 calls—one every 100 seconds.

A survey of RSA attitudes regarding their jobs in the centralized environment indicated dissatisfaction with the repetition and lack of variety.[1] Another key area of concern was their lack of decision-making authority. Further study of the RSA job in the centralized environment led to the hypothesis that the quality of the job could be improved.

3.2 Approach to a solution

The purpose of the present study was to determine if the quality of the RSA job could be improved, while preserving the efficiencies provided by centralization. The approach taken was to apply one of the principles of job enrichment and try it out in a CRSAB. These principles indicate that a job designed to maximize interest, challenge, and satisfaction for an employee must do three things:

(*i*) Allow the worker to do a *complete* piece of work for an *identifiable* client.

(*ii*) Give the worker a high degree of *control* and *decision-making authority* over how that complete piece of work is carried out.

(*iii*) Provide *direct feedback*—from the worker's clients and through the work itself—on how well the worker is doing the job.[2]

The trial discussed here focused on giving the RSAs more decision-making authority. This is not to imply that the other principles were well fulfilled in the RSA job, but only that they would be dealt with in later trials.

Increased decision-making authority in the RSA job was made possible by the development of the MLT, which automatically tests the customer's telephone line while the customer and the RSA are talking. The results of the test are then displayed on the RSA's display terminal.

Based on the information made available by the MLT—along with customer information—the RSA can make three important decisions during the customer contact:

(*i*) whether to obtain access to the customer's premises,

(*ii*) whether to attempt front end close-out of the trouble, that is, to get agreement from the customer that there is no longer a problem with the line, and

(*iii*) whether to suggest that the customer take the phone set to the Phone Center Store for replacement.

Each of these decisions is discussed below.

If the MLT indicates that the trouble is in the central office or in the cable, no access to the customer's premises is needed. The time that the RSA must spend talking with the customer can be shortened.

If the MLT results indicate the customer's line is OK and certain other conditions exist, there may be no need for the telephone company to take any action on the trouble report and the RSA can attempt a front end close-out. For example, a customer may try to use his phone before leaving for work in the morning and discover he has no dial tone. By the time he calls in from his place of work to report the trouble, the problem may have been corrected at the Central Office. When the service attendant uses the MLT, it will indicate TEST OK. The attendant can then tell the customer that the line checks OK, and indicate that no further action is required. If the customer agrees, the RSA can complete a front end close-out transaction that updates the computer but sends no trouble report to the RSB. This saves the RSB from handling and processing a trouble that is nonexistent.

The third type of decision also requires information from the customer. If, in describing the service problem, the customer indicates the trouble is with only one telephone set among several in the home, and if MLT shows that no other problem exists, then the RSA can ask the customer to take the suspected set to the Phone Center Store. If the customer accepts the suggestion, the attendant can use the front end close-out transaction to update the computer, but send no trouble report to the RSB. Again, the Bureau is spared the handling of a trouble report that actually requires no action on the Bureau's part.

3.3 Design of the trial

The trial included a sample from one CRSAB of six RSAs who used MLT results during their contact handling. The RSAs selected for the trial were considered to be "typical" attendants—not the best nor the worst. Three males and three females participated. Their ages ranged from 18 to 27 years and their experience ranged from 4 to 18 months. Following initial training, the six RSAs used the new procedures during a two-month period.[3]

3.4 Results of the trial

As the six trial RSAs used MLT results during contact handling, the impact on RSA performance and job attitude, impact on the RSB and reactions of customers were assessed.

(*i*) *RSA job performance and attitudes*—One prime RSA performance measure considered was the impact on holding time of using the MLT results during the contact. Holding time is defined as the interval starting when the RSA first receives the customer's call until the RSA transmits the Trouble Report mask for that trouble. Table I provides a summary of results.

The data indicate that, on the average, front end close-outs took about 35 seconds longer to handle than ordinary customer contacts.

Table I—Holding time for customer contacts

Type of Trouble	Number of Calls Monitored	Mean Holding Time (seconds)	Range of Holding Time (seconds)
Front end close-out troubles	41	145	62 to 274
Cable and Central Office troubles	61	77	38 to 238
Ordinary troubles	654	110	33 to 559
Total	756	109	33 to 559

The close-out transaction and the need for RSAs to call back some customers to obtain agreement to the close-out account for the extra time.

On the other hand, for contacts where the trouble was found to be in either the Central Office or in the cable, the average holding time was about 33 seconds less than would be required on an ordinary customer contact. This shorter holding time was because the RSA did not have to arrange access to the customer's premises.

The attitudes of the trial RSAs toward the use of MLT results in their contact handling were very favorable. When interviewed, the RSAs said they particularly liked the increased information that they could share with customers, the increased challenge of making more decisions, and the increased importance of their job in that they could save the RSB paperwork on front end close-outs.

(*ii*) *Impact on the RSB*—One impact that the new MLT procedures have on the RSB is measured in terms of the amount of trouble report processing saved. Every trouble that is closed out by an RSA means one less trouble to be processed by the Bureau. Data from the trial show that if a trouble closed out by the RSAs had been transmitted to the Bureau, it would have required an average of 129 seconds of handling by Bureau personnel. As noted earlier, the increase in average RSA holding time because of front end close-out was 35 seconds. Therefore, it seems that for an additional 35 seconds of RSA time, 129 seconds of RSB time can be saved. The data in Table II show that nearly 8 percent of the calls taken were candidates for front end close-out.

Table II also shows that more than half the customers who were

Table II—Frequency of use of MLT results by RSAs

Type of Contact	Number	Percentage
Front end close-out candidates:		
Accepted by customer	125	4.6
Refused by customer	88	3.2
Cable and Central Office troubles	272	10.1
Ordinary contacts	2212	82.1
Total	2697	100.0

offered a front end close-out accepted. Those who refused usually did so because they had checked their telephone recently—within the last hour—and felt there still was a legitimate problem, even though the RSA's test equipment indicated that the line was working.

Another impact the new MLT procedures might have had on the RSB would be to change the number of repeat calls made on troubles originally closed out by the RSAs. However, the data showed that for troubles closed out by RSAs during the trial, the repeat rate was the same as that for troubles of this type closed out by the RSB during the same time period.

(*iii*) *Reactions of customers*—To learn how customers reacted to the use of MLT results in their contact with an RSA, several hundred contacts were monitored. Customers always reacted favorably when told their telephone trouble was in the cable or Central Office and that access to their premises was not necessary. Similarly—but to a lesser degree—customers appeared to appreciate the RSA's informing them that the test equipment had identified a definite fault, even though that meant the customer had to arrange access to the premises.

To find out what customers thought of their telephone line being automatically tested during their contact with the RSA, a sample of 60 customers was surveyed by telephone. When asked about their reaction to the automatic testing, half gave positive reactions and half were neutral. None gave a negative response.

3.5 Conclusion from the trial

In conclusion, having RSAs use MLT results during customer contacts has far more benefits than costs. The benefits are savings in time for the RSB, improved customer relations, and a more satisfying job for the RSAs. The cost is the time required to train RSAs to use the new procedures.

REFERENCES

1. W. A. Harris, private communication.
2. L. York, *A Radical Approach To Job Enrichment*, New York: American Management Association, 1976.
3. W. A. Harris, private communication.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July–August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

On-line Documentation: Mechanizing Development, Delivery, and Use

By R. J. GLUSHKO and M. H. BIANCHI

(Manuscript received June 17, 1981)

We describe the design and development of on-line documentation for a minicomputer-based management information system. We outline the design choices, compare on-line documents with paper ones, and review human engineering and "software psychology" issues. On-line documents are accessed from any dial-up terminal. Document retrieval shares a common user interface with other information activities like report generation, trouble reporting, and interuser communication. Documents are "modular" with properties that make them easier to create, use, and maintain.

I. INTRODUCTION

Surveys confirm that documentation does not always meet the information needs of computer system users. Documents often arrive later than the system they are supposed to support. Related information may be scattered through several documents, inadequately cross-referenced, or inconsistent. The users may need to rewrite documents to reflect local policy, tariffs, or company organization, adding more costs and delays to the expensive and time-consuming documentation process.

These problems with document timeliness and organization have grown increasingly severe in the last decade as computer systems have proliferated. The traditional procedures for developing and delivering paper documents have not kept pace with the increased information needed to support complex and volatile software products. Instead, with computer systems for text editing and storage becoming readily available, a growing number of Bell System applications are hoping to

resolve the mismatch between software systems and paper documentation by developing and delivering documents along with software in computer-based form.

Mechanizing the documentation process by using computer text editing systems and by treating documents the same as software seems like an obvious solution to the documentation problems for software systems (see Refs. 1 and 2.) Unfortunately, this solution can be as simplistic as it is obvious. Documentation problems rarely reduce to separate problems of development, delivery, or use. When we set out to create an on-line documentation system for the Cable Repair Administrative System (CRAS), we struggled to cope with the radical changes in the traditional ways of developing, delivering, and using documentation that came about when we transformed paper documents into computer-based ones. We found it challenging to create the tools and techniques we needed to coordinate the development of software and documentation. We emphasized eliminating the duplicated and disjointed efforts that carried over from methods for delivering software and documents in different media through separate channels. We worked hard to build a system that was easy to use for the telephone company personnel whose needs for information underlay our entire effort. We share some of our insights and hindsights in this paper.

The CRAS system generates management reports on the performance of telephone cable maintenance forces and the condition of the outside plant. The system runs under the *UNIX** operating system on a Digital Equipment Corporation VAX† 11/780 minicomputer. We will not discuss the role of CRAS in day-to-day telephone company operations (see Ref. 3)

II. DEFINITIONS AND GOALS FOR COMPUTER-BASED DOCUMENTATION SYSTEMS

Computer-based documentation is a broad goal that can be realized in many different ways. Different applications may face different documentation problems, user populations, or computer hardware that shape the design and goals for a mechanized documentation system. Before describing the specific system that was developed for CRAS, it will be helpful to define some of the varied forms that computer-based documentation may take to provide context for the specific design choices.

The documentation cycle begins with document definition, determining which documents need to be created. Next, these are written

* Trademark of Bell Laboratories.
† Trademark of Digital Equipment Corporation.

and edited. The finished documents are delivered to the users. Finally, the documents are used and maintained by the telephone companies, whose needs may lead to revised or new documents, new software or even to new systems. We distinguish three basic forms of computer-based documentation that differ primarily in how much of the entire documentation cycle they mechanize: the *electronic factory*, the *electronic library*, and *on-line documentation*.

The *electronic factory* is where computer-based documents are developed. Documents are captured at their source as they are written using computer text-editing systems, thus saving at least one costly and time-consuming translation from paper to computer-based form. The document development computer then serves as an electronic factory from which paper documents are manufactured as required. The same text source produces either typewritten or photocomposed output, and changes can be made with considerably less effort than for documents that exist only on paper. Note, however, that the intended user of the documents is not allowed direct access to the documentation in computer-based form.

The *electronic library* extends the electronic factory one step. It offers information on-line in a form organized for the intended users. The electronic library is typically a computer that serves as a central repository of documents, training materials, or planning information that users can access on-line from dial-up terminals. This is especially useful when the specific end users are unknown in advance and are dispersed in many different locations. However, the electronic library is logically, and usually physically, distinct from the computer system whose documentation it contains.

On-line documentation spans the usage stage of the documentation cycle by making information available to users in their normal work environment as they use a computer system at a terminal. The goal is to minimize the user's need to maintain paper documents, since the most reliable and current copies are always available on the computer. The *man* command that prints pages from the *UNIX* User's Manual is a rudimentary example of on-line documentation. To our knowledge, CRAS is the first telephone company operations support system with computer-based documentation that is on-line in development, delivery, and use.

III. COMPUTERS VERSUS PAPER

The potential for computer-based documentation systems to reduce the documentation problems of the telephone companies seems so obvious that it is almost impossible to imagine a Bell System application that would not benefit substantially from even the basic electronic

factory idea. Nevertheless, we resisted the temptation to "damn the paper, computerize at full speed" when we realized that transforming documents from paper to computer-based form had its costs, as well as its benefits. We will discuss some of the trade-offs that directed and constrained the CRAS approach.

Paper documents are "frozen" when they are written. Since they are not easily changed, they must be detailed and complete, hence the problem of bulky paper documents that contain more information than any one person ever seems to want. In contrast, the fundamental property of computer-based documents is that they can be easily changed. When coupled with selective display of information and flexible organization, computer-based documents can be thorough, yet easy to use and maintain.

On the other hand, it is important not to overlook the desirable properties of paper documents that are not possible to recreate on-line. Few computer terminals can reproduce pictures or graphics with the quality available in print. While low-cost graphics terminals someday may be commonplace, and computer systems far more universal and reliable, people will always be more familiar with paper documents. Only paper documents can be personalized with margin notes or "dog ears" to make them unique and useful information sources.

The complementary features of paper and computer-based documents suggest to us that even the most technologically advanced on-line documentation system will be surrounded by paper documents of its own creation. Every prophecy of a "paperless office" or "paperless society"[4,5] seems countered by more conservative forecasts in which paper remains in an important, albeit changed, role.[6,7,8] People will not quickly abandon all paper documents for electronic images that glare at them from television screens. Instead, users of on-line documentation systems will create some mix of computer and paper documents that takes advantage of the benefits of each medium. New users often bring with them inappropriate ideas of what computers can and cannot do, and these expectations may lead them to reluctant, inefficient, and, from a designer's standpoint, unpredictable uses of the computer. Many on-line documents will be printed and carried to the computer terminal in loose-leaf notebooks and dutifully replaced when new electronic versions appear.

We might respond to such uses of our on-line documentation system as weaknesses in its design to be overcome with some clever program or technique. Instead, we remind ourselves that we set out to meet the information needs of people, and note that technological issues seem to be secondary to the human engineering and "software psychology"[9] issues in the design and use of on-line documentation systems.[2,10]

IV. THE CRAS DOCUMENTATION SYSTEM

The general theme is that we seek to make documentation part of CRAS instead of just support for it. Using the computer to mechanize a system's documentation is a first step, but the challenge is to unify documentation and software, simplifying the user's interaction with the computer. We present four different perspectives on the CRAS documentation plan that define this objective:

- CRAS is an integrated information system. All of the CRAS information activities—report generation, document retrieval, trouble reporting, and interuser communication—share a common user interface.
- CRAS has a broadened view of documentation. Its on-line documentation contains the "standard" information that would be delivered with any operations system, but also contains information that might normally be called "performance aids" and "training." Locally-generated documents can be installed in the on-line system.
- CRAS documents tend to be "modular" with properties that make them more manageable and more easily maintained on the computer or in a user's notebook.
- CRAS documents are "computerized," as well as "computer-based"—the computer uses information implicit in the CRAS file structures and software to rewrite key documents and to provide a degree of "intelligence" to the document retrieval process.

4.1 The Cable Repair Administrative System as an integrated information system

The primary user of CRAS is an analyst who uses CRAS reports to understand and improve the condition of the telephone cables and the performance of cable repair forces. Let's observe a new analyst at the terminal using the various information subsystems of CRAS to see how the activities fit together and complement each other.

4.1.1 On-line performance aids

Our analyst wants to generate the CRAS Geographic Area Summary Report, Report 02. The analyst needs to be reminded by some documentation or performance aid how to specify the report request. The CRAS system provides both kinds of information when the analyst types *rpt02* at the terminal:

```
$ rpt02
Usage: rpt02 lev org period [-c geoid geoidvalue] [-t ncode4 number]

For more detail type: prtdoc cmd.rpt02
```

The "Usage" reminder is modeled after those provided by many

UNIX system commands. We expand its function by giving the user explicit instructions (the specific CRAS command) for retrieving the on-line document called *cmd.rpt02*. The "Usage" line is a "performance aid," a memory jog for experienced users. The other line shows less expert users the exact step for obtaining information about retrieving Report 02.

4.1.2 The "prtdoc" document retrieval command

The *prtdoc* command prints on-line documents at the analyst's computer terminal. The syntax of *prtdoc* is not the result of a "shotgun wedding" between document names and a retrieval procedure; we carefully organized and named the on-line documents to make them easy to retrieve. Overview and one-of-a-kind documents are retrieved by one-word names; for example, a table of contents that lists all on-line documents is retrieved by typing *prtdoc contents*. All other documents are arranged in categories with names made up of the category name and the document name, as in *cmd.rpt02*. All documents in a category are retrieved by typing just the category name (e.g., *prtdoc cmd*).

Prtdoc usually begins presenting the requested document to the user's terminal in five seconds or less, because documents are preformatted in files. Storing "finished" documents rather than source files takes up slightly more space, but saves the user the one-minute or more wait if the text formatting were done at the time of document retrieval. The storage space that CRAS documents occupy is a minuscule requirement on the computer sized for storing the large data bases needed to generate CRAS reports. Documents take up at most a few percent of the disk space in the standard CRAS configuration.

The *prtdoc* command has an option that prints any documents that have changed since a specified date. We provided this feature to help the many CRAS analysts who prefer to keep personalized paper copies of the most relevant documents in loose-leaf notebooks that they bring with them to the computer terminal.

Another useful option in *prtdoc* allows the user to find documents whose names, titles, or major section headings match key words. This helps users find information when they know what they are looking for but cannot remember the names of the relevant documents.

4.1.3 Other on-line information facilities: "crasprob" and "crasmail"

Suppose the analyst successfully generates the desired CRAS report, but suspects an error. A problem reporting command called *crasprob* allows the analyst to report problems or suggestions to the CRAS administrator and to support groups at Western Electric and Bell Laboratories. *Crasprob* maintains accurate records of all problem

reports, their status, and their solutions. *Crasprob* is efficient and natural because it is part of the analyst's work environment at the terminal, where problems are usually encountered.

Some time after the analyst submits the problem, the CRAS administrator or someone in the support organizations determines that the analyst's report is not in error, but contains some curious rules for counting data items. The CRAS administrator then invokes the *crasmail* command to notify all CRAS analysts throughout the company of the problem and its resolution. *Crasmail* also archives all interuser communication for future reference. When the analyst next types *mail* the answer to the problem will appear at the terminal.

If CRAS problems require changes to software and documentation, the CRAS support organization transmits the new information to CRAS installations over dial-up lines. The local CRAS administrator simultaneously installs the software and documentation with a single command, ensuring consistency. This coordination is possible because files containing documents and files containing software look identical to the computer.

Crasprob and *crasmail* are both built around the *UNIX* system's electronic *mail* and evolved from problem-reporting and interuser communication systems devised for CRAS development.

4.2 Broadened view of documentation

The CRAS system blurs the usual distinctions between information that might be called documentation and that normally classified as performance aids or training. The CRAS on-line documentation scheme contains information of all types that has the common purpose of enhancing the user's performance. We have already described the "standard" documents retrieved using the *prtdoc* command and the on-line usage reminders that tie software to documentation. Other kinds of information aids are as follows:

- *Report prompting*—Normally, when generating a report the analyst types the command name followed by any required or optional items. However, in report prompting the user types *prompt* followed by just the command name and the computer asks for each item in turn and continues until the analyst supplies enough information to generate the report correctly. Prompting takes more time, but the computer can determine whether the entries are valid as the analyst supplies them, so it is a valuable training aid for inexperienced users or a performance aid for entering complicated report commands.
- *Training data base*—The reference data base that is used to create sample reports for on-line documents is available for on-line training in report generation.
- *Local documents on-line*—Documents created by the telephone

company to reflect local procedures or policies, or "nonstandard" documentation, such as lists of names or telephone numbers, can be installed in the "standard" *prtdoc* on-line data base by the CRAS administrator.

- *Usage records and analysis*—All commands that analysts type at their computer terminals are recorded in a command log used by the CRAS administrator and support and development groups to improve CRAS performance. A statistical summary of the log is provided to the CRAS system administrator. This information about how CRAS is used has been the source of many of its most useful features. For example, we noticed that certain analysts periodically retrieved the table of contents of the on-line documents and then painstakingly studied it line-by-line to determine if any documents had been changed. We automated this task and made it an option in the *prtdoc* command.

4.3 Document modularity

The CRAS system contains about 250 on-line documents that average three pages in length. One popular definition of a software module is that it is a piece of code that "hides one secret." The analogous definition of a document module is that it is a unit of information that "tells one secret." For example, when a programmer creates a new piece of software, it is easy and natural to write a new document that describes it.

Not all CRAS documents are modular—nor should they be. One-of-a-kind documents like indexes and tables of contents are not broken up, and some overview documents that "glue" modules together are necessary. But in general, the modular organization of CRAS documents as short, self-contained units of information creates useful properties:

- *Terminal compatibility*—The CRAS documents are easy to use at a computer terminal. Even at the slowest display speeds, most documents are so short that they print in only a few minutes.
- *Work-unit organization*—Most documents contain the information needed for a single task or unit of work—retrieving a report, sending a message, or doing some other single activity. The CRAS changes over time as reports or commands are created, changed or deleted, but the modular organization of the documentation makes these changes easy to manage. Adding a command only requires adding a single short document, which is easily installed at the same time as the new software. There is no need to accumulate a large number of changes to justify reissuing a complete user manual.
- *Flexible job definition*—A job or position is made up of a collection of tasks or work units. For example, the analyst retrieves certain reports, analyzes them, and executes various communication and out-

put control commands at the terminal. The work-unit organization of CRAS documentation allows the analyst to have the information needed for the job, without any irrelevant information, even when companies define the analyst job differently. For example, Telephone Company A may make data base maintenance reports part of the analyst's job, while Telephone Company B may assign them to a clerk. Since each report is described in a separate document, analyst A and analyst B can have customized documentation manuals with just the right information, even though A and B perform different jobs.

- *Integrating local documentation*—Modular organization makes it easy to integrate local information with standard information. Local information that would not fit into a bulky paper manual naturally fits into collections of short pieces of information in an on-line system. In the past, telephone companies either maintained their local documents separate from standard documents or reissued standard documents at considerable cost.

4.4 Computerized, not just computer-based documentation

The CRAS system uses the computer for more than simply storing documents. Whenever possible, CRAS uses the computer to make documents more current, more reliable, and more useful. In particular, "dynamic" documents are generated by computer programs and use information implicit in the file structure or software to update themselves. Some of the most important of these are:

- *Table of contents of CRAS documents*—This is the master list of all documents that can be retrieved using the *prtdoc* command. The table lists them by the short name used to retrieve them, along with the complete title of the document, the number of pages it contains, and the date that it was last changed. The table is regenerated when any document is changed or added to the on-line documentation system. A related program automatically regenerates a permuted index to all the titles.
- *Dictionary of data fields*—This lists the elements of CRAS data base records that are used to generate the reports. If more types of information are added to the data base to allow for more specialized reports, this document is automatically updated from the data dictionary.
- *Run folder documents for host jobs*—These documents are the "Run Folders" or "Run Books" for several data base extraction jobs run for CRAS on a remote IBM computer. CRAS run folders are created from the Job Control Language for each job and are automatically customized to reflect the specific run-time options, sizes, and file names selected by the CRAS administrator.

V. OTHER LESSONS FROM CRAS ON-LINE DOCUMENTATION

5.1 Constraints on document development

Delivering documents along with software in CRAS required that programmers become part-time technical writers and increased our direct development costs. In most projects, documents are written by technical writers from other organizations, which may make them of higher quality, but which usually makes them late and less precise. The CRAS developers used the computer aids for writing and editing—the "Writer's Workbench"—that became available during the system's development.[11] We think that any trade-offs we made in document quality are far outweighed by gains in document timeliness and better coordination with software. Nevertheless, management should allocate greater resources to the development team and regard documentation as more than a nuisance activity beneath the efforts of programmers.

5.2 Easy compliance with document standards

On-line modular documentation easily lent itself to the new standards recently adopted by AT&T for Operations Systems Deliverable Documentation (OSDD). The OSDD recommends that documents be organized to meet the information needs of particular jobs or users rather than letting a single document contain "everything for everyone." Even though CRAS development was nearly complete when the OSDD standards were adopted, in a few weeks we devised customized collections of our document modules and created an on-line command that automatically printed the documents in OSDD form. In general, maintaining documents in computer-based form makes compliance with document standards much less painful than it has been in the past.

5.3 An adjunct dial-up advance information system

When documents are developed on-line in an "electronic factory," the intended users can have a preview that helps them plan while letting them provide valuable input to the development process. We provided a log-in on our development computer to telephone company personnel so that they could examine documents under development and experiment with the on-line retrieval system. They were better able to plan for CRAS, and by monitoring the dial-up use of the *prtdoc* command, we significantly improved the organization and usability of the documentation. This system became available almost without effort because we were developing the documents on-line anyway.

VI. PROSPECTS FOR ON-LINE DOCUMENTATION

The CRAS system is part of a larger network of many different kinds

of computer systems, terminals, and personnel subsystems. Some systems run on large computers like an IBM 370, others run on minicomputers in both stand-alone and networked configurations, and still others run on microprocessors. Dial-up asynchronous terminals used by analysts and managers sit next to dedicated synchronous terminals used by data entry clerks.

One form of computer-based documentation cannot work for every application—we know because we are facing the challenge of developing a plan for mechanizing the documentation of the entire family of systems of which CRAS is a part. Nevertheless, we have "deCRASsified" many of the programs and techniques used in CRAS, and they are being adopted by other projects as we work together toward that goal.

VII. ACKNOWLEDGMENTS

The CRAS documentation system was shaped by the collective efforts of the CRAS development group, who merged documents and software just as a motley collection of systems engineers, psychologists, and software developers merged and became the CRAS team in 1979–1980. In particular, J. R. Mashey, G. T. Vesonder, and K. A. Wright made substantial contributions to the work and ideas presented in this paper.

REFERENCES

1. K. L. Heninger, "Specifying software requirements for complex systems: New techniques and their application," IEEE Trans. Software Eng. (1980), *SE-6*, pp. 2–13.
2. A. I. Wasserman, "Information System Design Methodology," J. Amer. Soc. Inform. Sci., *31* (1980), pp. 5–24.
3. P. S. Boggs and J. R. Mashey, "Automated Repair Service Bureau: Cable Repair Administrative System," B.S.T.J., this issue.
4. E. Hanson, "Preparing our Thinking for the Year 2000," Word Processing World, *5* (1978), p. 40.
5. A. Toffler, *The Third Wave*, New York: William Morrow, 1980.
6. W. Benedon, "The Paperless Society: Fact or Fiction?" Inform. and Records Management, *13* (1979), pp. 101–2.
7. V. A. Vyssotsky, "Computer Systems: More Evolution Than Revolution," Journal of Systems Management, *31* (1980), pp. 21–7.
8. D. Wallace and P. Yates-Mercer, "A Paperless Society: Reality or Myth? Management Services, *24* (1980), pp. 12–7.
9. B. Schneiderman, *Software Psychology: Human Factors in Computer and Information Systems*, Cambridge, Mass.: Winthrop, 1980.
10. S. Hiltz and M. Turoff, *The Network Nation: Human Communication Via Computer*, Reading, Mass.: Addison-Wesley, 1978.
11. L. T. Frase, "Computer Aids for Writing and Text Design," Symposium presented at the Annual Meeting of the American Educational Research Association, Boston, Massachusetts, April, 1980.

THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 61, No. 6, July-August 1982
Printed in U.S.A.

Automated Repair Service Bureau:

Economic Evaluation

By E. A. OVERSTREET

(Manuscript received March 3, 1981)

Throughout the conception, development, and deployment of the Automated Repair Service Bureau (ARSB) systems, economic studies have played a continuing and important role. Prior to the development of ARSB, economic studies quantified the potential costs and benefits, leading to the funding of the ARSB project. Throughout the ARSB development phase, the economic studies were refined and used to influence system features. Several field trial evaluations were conducted in Bell Operating Companies (BOCs) to verify the economic models and to refine operational procedures. Prior to system availability, economic planning guides for Loop Maintenance Operations System (LMOS) and Mechanized Loop Testing (MLT) were generated to assist the BOCs in planning for the implementation of ARSB. This article describes the purposes, methods, and results of each of these economic activities. Currently, approximately 75 percent of the Bell System lines are served by LMOS and 50 percent by MLT. Significant reductions in repair expenses, as well as flexibility in the organization of maintenance centers, have been achieved as a result of ARSB development.

I. INTRODUCTION

Throughout the conception, development, and deployment of the Automated Repair Service Bureau (ARSB) systems, economic studies have played a continuing and important role. As shown in the economic "flow chart" in Fig. 1, economic studies are used to

(*i*) help identify the need for a new system,
(*ii*) lead to a decision to develop a new system,
(*iii*) influence system requirements,

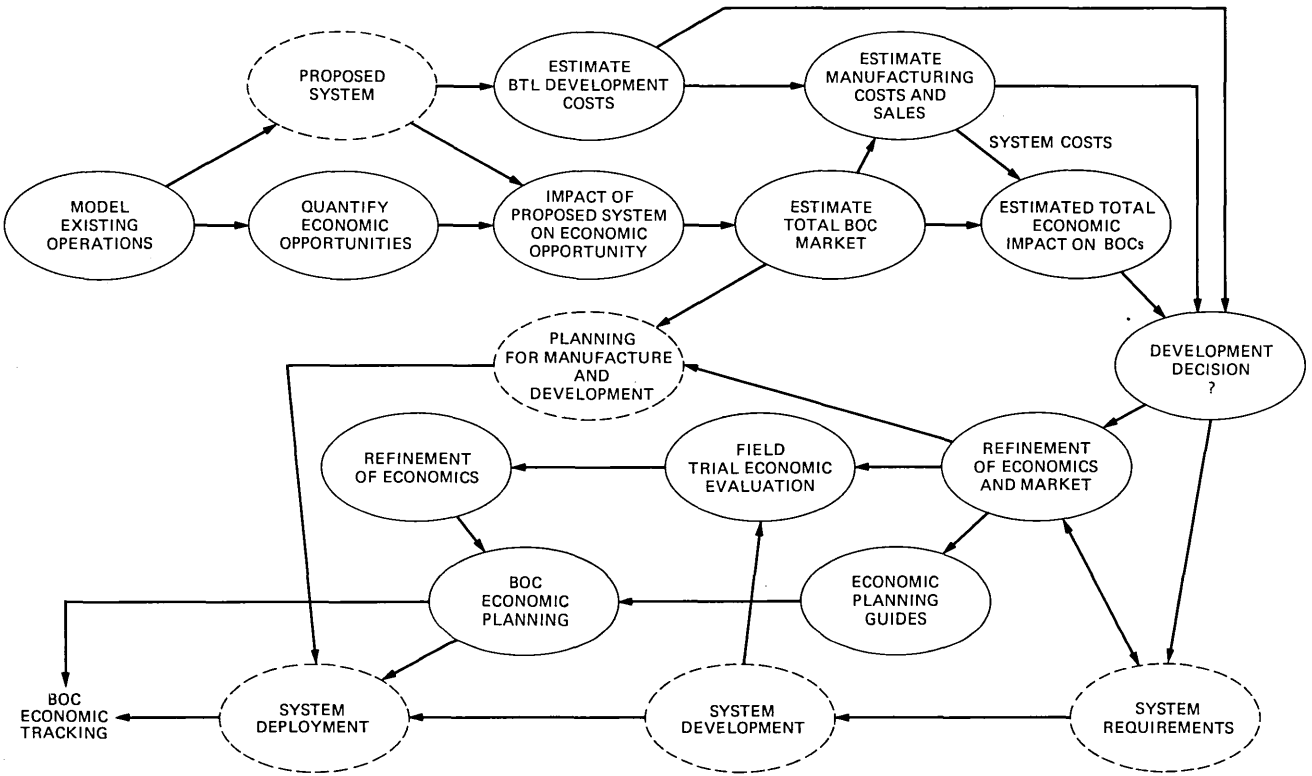


Fig. 1—Role of economic analysis in the life of a system.

(*iv*) support the supplier in pricing and planning for manufacture and/or implementation, and

(*v*) support BOC planning for system implementation.

During the planning and development of Loop Maintenance Operations System (LMOS) and Mechanized Loop Testing (MLT) system, three major economic milestones occurred. These were (*i*) the economic summary support to the Western Electric Case Authorization which led to the funding of ARSB, (*ii*) the publication of LMOS and MLT Economic Planning Guides, and (*iii*) economic results from field-trial evaluations. In addition, there were refinements of the economic analysis throughout the ARSB program.

Section II of this paper describes in more detail the purposes, methods, tools, and results used in each step of the ARSB economic evolution. Sections III and IV provide detailed models of costs and benefits for LMOS and MLT.

Additional ARSB modules include the next generation of both LMOS and MLT, plus the LCAMOS systems. The economic analysis of these systems are not discussed in this paper. However, the economic analysis methods discussed for LMOS and MLT apply to the newer systems.

The reader is encouraged to read the introductory article of this issue of *The Bell System Technical Journal* for background on the loop maintanenace process and an overview of the ARSB systems.

II. EVOLUTION OF ARSB ECONOMIC STUDIES

In this section, the role of economic studies in each step of the decision-making processes involved in defining, developing, manufacturing, and deploying LMOS and MLT is discussed. The evolution of ARSB economic studies discussed in this section covers a time span of approximately ten years.

2.1 Early estimates—support development decision

In 1968, a Repair Service Bureau (RSB) Task Force warned that unless mechanization is introduced to the repair process, the cost of repair service operations would increase at a rapid rate, because of Bell System growth and increasing labor rates, and that service to the customer would degrade. As a result of their studies, they recommended that loop records be computerized, improved testing equipment be developed, and repair service record keeping and analysis be computerized.

The next step was to model the costs required to perform the loop maintenance function in sufficient detail to allow estimates of opportunities for economic benefits. By gathering and analyzing data on the costs required to test trouble reports, to maintain the manual line record files, and to dispatch repair craft resulting in no trouble being

found, it was estimated that significant benefits might result from mechanization. The economic benefit model was then refined concurrently with the formulation of a conceptual system which appeared both technically and economically feasible. A conceptual system was first proposed in a Prospectus for New Methods and Equipment for Exchange Special Services and Local POTS Testing Systems in 1971. The economic benefit model was refined by visiting a number of Bell System RSBs and modeling their existing costs. The estimated economic impact of the proposed new system was incorporated into the economic model and both the potential benefits (i.e., reduced maintenance expenses) and the cost of the system for a typical BOC "installation" were derived.

The next step was to extend the "typical installation" economics into a Bell System view which assessed the economic impact of the proposed system on all of the BOCs. Market data were gathered from a survey which included statistics on the number and sizes of all BOC RSBs and distribution of wire center sizes. Also required were estimates from Western Electric on the costs to manufacture, sell, install, and support the system, as well as estimates of Bell Laboratories development costs. The total Bell System economic impact was then analyzed and a development decision was made based upon net economic savings to the BOCs, projected Western Electric sales and required Bell Laboratories resources. The initial request for development funding was submitted in 1973.

2.2 Market studies

Inherent in the estimate of the total economic impact of ARSB on the BOCs is an estimate of the potential penetration of ARSB in the BOCs. However, the market studies have value beyond that of simply supporting the development decision. In particular, market studies are used for the following purposes:

(*i*) To serve as a basis for the software supplier to develop a pricing strategy to ensure that early customers are charged properly and that the supplier recovers its development costs.

(*ii*) To serve as a basis for the hardware manufacturer to estimate the manufacturing capacity that must be planned and implemented.

(*iii*) To serve as input to the system design requirements for those features that are sensitive to the market characteristics.

The market estimates for LMOS were rather straightforward. It was assumed, and is still expected, that all BOCs would install LMOS. In addition, it was also assumed that each BOC would serve all of their lines with LMOS. This later assumption was based on

(*i*) the high initial cost of LMOS (e.g., for host computer, front-end processers, conversion process) and the low incremental cost to add lines, and

(*ii*) the desire for consistency within a BOC in receiving trouble reports, tracking reports, and post-trouble analysis—all of which are significantly impacted by LMOS.

Using estimates from the LMOS developers of the capacity of the LMOS computers, the number of host and front-end computers was derived from the total number of lines in each BOC. Allowances were made for growth and for special data processing administrative boundaries within some BOCs. These estimates were made originally by Bell Laboratories and Western Electric and then updated as the BOCs started formulating their own LMOS deployment plans.

Estimating the potential market for MLT was more complicated, primarily because there are no large initial centralized costs for MLT and each wire center in a BOC must be examined individually to determine whether it can economically justify MLT. Also, unlike LMOS, MLT does not impact the procedures for processing trouble reports significantly enough to create a desire for universal deployment. In other words, it is not economical for MLT to cover all sizes of wire centers in the Bell System and the individual BOCs deployed it selectively. Furthermore, since a single MLT system can test several wire centers if they are all located within MLT testing range (approximately eight miles), wire center data alone are not sufficient to estimate the number of MLT systems required. The geographical clustering and trunk routes between wire centers must also be known. The first step in the MLT market estimate was to model the costs and potential benefits of MLT to determine the number of lines required in a wire center cluster for MLT to be economically justified. This threshold was used as a guideline but may vary significantly from BOC to BOC depending on trouble report rates, dispatch strategies, travel times for repair and RSB consolidation plans. The next step was to examine several BOCs in detail to determine the impact of wire center clustering on the ratio of MLT systems to wire centers. These ratios were then applied to a distribution of wire center sizes for the Bell System, and an estimate of total MLT market was derived.

As of the end of 1980, 18 BOCs had implemented LMOS with approximately 50M lines being served, and 14 BOCs had initiated their MLT implementation serving approximately 25M lines. It is estimated that by the end of 1983, all lines in the Bell System will be served by LMOS. The deployment of MLT, expected to eventually serve approximately 85 percent of the Bell System lines, is expected to be completed in the mid-1980s.

2.3 Economic planning guides

The ARSB system requires a significant investment by the BOCs and impacts their maintenance procedures. Therefore, to be ready to deploy ARSB when it became available and to realize the system

benefits, the BOCs began their ARSB planning at least two years before actual system deployment. Economic planning guides for LMOS and MLT, produced by Bell Laboratories, were essential to this planning process. The purposes of the two planning guides were to

(*i*) provide guidelines for each BOC to size the LMOS and MLT systems to meet their application requirements,

(*ii*) allow each BOC to estimate LMOS/MLT costs and benefits for their particular operating environment,

(*iii*) support the BOC planning and budget process to ensure that the required ARSB investment is included in the corporate budget, and

(*iv*) allow BOCs to estimate changes in their personnel (both numbers and types) in advance of system deployment.

The format of the two planning guides followed the general outline given below:

(*i*) Systems architecture showing the relationship of system components, the modular flexibility in sizing these components and the restrictions on the capacity of each component. (For example, Fig. 2 shows the ARSB computer hierarchy, Fig. 3 illustrates the modularity of the MLT testing hardware, and Table I gives the restrictions on the MLT hardware and controller.)

(*ii*) Definition of BOC data required as input to the planning guide, e.g., RSB and wire center parameters, such as lines served and trouble report rates.

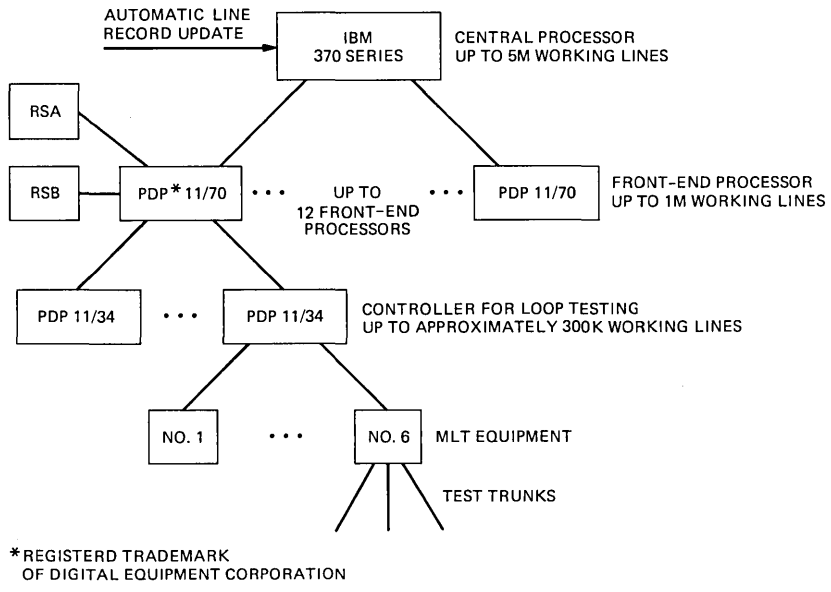


Fig. 2—Automated Repair Service Bureau architecture.

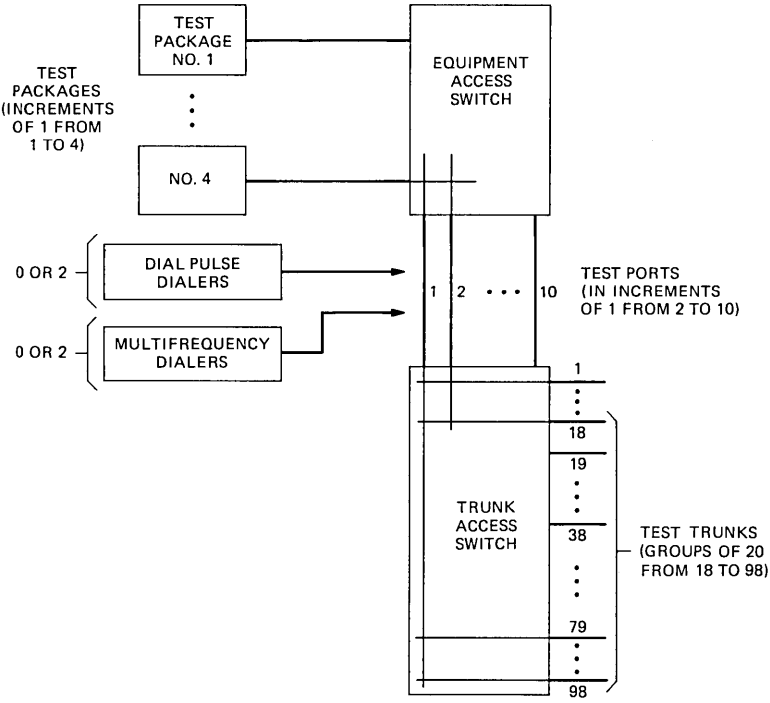


Fig. 3—Modularity of MLT hardware.

Table I—Restrictions on MLT application

MLT hardware	
Maximum number of test trunks	98
Tests per busy hour	600 maximum
Distance limitation (test trunk and loop)	3000 ohms or 100 kft
MLT controller	
Maximum number of MLT frames	6
Maximum number of test trunks	120
Maximum number of exchange codes	40
Maximum number of tests in busy hour	600

(*iii*) Model of the system costs and benefits with procedures for applying the BOC data to these models.

(*iv*) Sizing guidelines for determining the amount of modular equipment (e.g., front-end computers, terminals, MLT controller, etc.) required for each specific application of the system.

(*v*) Worksheets for use by the BOCs to size the LMOS and MLT equipment and to estimate costs and benefits using their input data.

One of the more critical parts of the planning guide discussed above is the generation of equipment sizing guidelines. These guidelines address two opposing objectives; to minimize the amount of equipment (and hence the cost) that the BOC must purchase while providing sufficient equipment to respond quickly to test requests.

The equipment sizing guidelines were particularly sensitive to two parameters: holding times for each piece of equipment and the number of MLT test requests generated by a given population of lines served. The holding times were originally estimated by the developers and used in the planning guides. During the MLT field trial, equipment holding times were monitored using computer log tapes and the sizing guidelines were updated to reflect the actual holding times. Originally, equipment-sizing requirements were provided to the BOC as a function of number of lines served by the equipment. This assumed a relationship between lines served, trouble reports, and MLT tests per trouble report. As the BOCs implemented and started using MLT, it was found that MLT usage varied among the BOCs depending on operational procedures. As a result, the equipment sizing tables were revised to provide equipment sizing requirements as a function of lines served, trouble reports, and expected MLT tests per hour, as shown in Table II.

For LMOS, the planning procedure was basically a straightforward pyramid approach. The approach was to do the following:

(*i*) For each RSB, Centralized Repair Service Attendant Bureau (CRSAB), and training center, determine the number of I/O terminals (CRTs and printers), number of lines served, I/O ports required on the front-end processor, and the transaction rate, based on trouble report and service order activity.

(*ii*) Determine number of front-end processors required, taking into consideration processor limits on I/O ports, lines served, and transaction rates.

(*iii*) Allocate RSBs, CRSABs, and training centers to specific front-

Table II—MLT Equipment sizing guidelines

	MLT Tests Per Busy Hour	Trouble Reports Per Busy Hour	Lines (1000)
Test trunks			
1	<6	<3	<3.5
2	6–43	3–22	3.5–30
3	43–110	22–60	30–80
4	110–185	60–100	80–130
Test ports			
2	<20	<10	<15
3	20–65	10–35	15–45
4	65–145	35–75	45–100
5	145–225	75–115	100–150
6	225–300	115–155	150–210
7	300–390	155–200	210–270
8	390–480	200–250	270–330
9	480–570	250–300	330–400
Test packages			
1	<20	<10	<15
2	20–145	10–75	15–100
3	145–390	75–200	100–270
4	390–580	200–300	270–400

end processors so that the limits of no front-end processor are exceeded.

(*iv*) Determine the number of host computers required and allocate front-end processors to hosts.

For MLT, the process flow was similar but could require several iterations to minimize overall MLT costs. This was due to the flexibility that the BOCs had in clustering wire centers on MLT hardware and in assigning MLT hardware to MLT controllers to achieve a balance on controller load in terms of LTFs, NNXs, test trunks, and testing load (i.e., see Table I).

The initial planning guidelines provided a fundamental model for sizing the LMOS and MLT systems and for estimating costs and benefits. As changes in configurations, costs or sizing guidelines occurred, the BOCs were notified either directly by Western Electric (prices) or via an ARSB Equipment Design Requirement which has been periodically updated and reissued by Bell Laboratories. However, the planning procedures used in the Economic Planning Guides are still applicable.

2.4 Refinements of economic estimates

As LMOS and MLT were being developed and deployed, the projections of system costs and benefits were periodically reevaluated as new data became available and the system configuration and features were becoming firmer. Included in these economic evaluations were (*i*) revised MLT benefits as a result of experimenting with an early model of MLT in a real operating environment in 1974, (*ii*) revision of LMOS and MLT benefits based upon pretrial characterization data gathered in the MLT trial site in 1975, and (*iii*) revision of LMOS/MLT benefits based upon data gathered from 15 BOCs in 1977. The results of the latter two studies will be discussed in more detail in Sections III and IV.

To gain experience with MLT in an environment of real trouble reports, an experiment was arranged to use an early model of MLT in several RSBs. Objectives of this experiment were to show that MLT could be useful in a functioning RSB, to gather data to refine MLT testing strategies, and to refine the projected MLT economic benefits. The experiments demonstrated that more than 80 percent of the troubles entering the RSB could be processed using only MLT test results; i.e., less than 20 percent would require local test desk testing. Direct benefits observed for MLT were less manual testing, fewer and more accurate dispatches, and the possible immediate close-out of some trouble reports as test-okays at the time they are received.

Throughout many of the ARSB economic studies, computer programs were used to mechanize much of the manipulation of economic data. These programs provided the user with standard and consistent treat-

ment of such variables as taxes, depreciation, salaries, cost of money, inflation rates, and calculations of conventional economic results, such as discounted cash flows, net present value of savings, rate-of-return on capital, and discounted payback periods.

2.5 Field trial results

Before deploying LMOS and MLT in the total Bell System, field trials or evaluations were conducted to ensure that the systems were viable in an operating environment and to refine the economic benefits and cost estimates for the systems. Three field trials or evaluations were conducted on the ARSB system:

(*i*) An evaluation by Bell Laboratories of the first installation of the Mechanized Line Record (MLR) system, a predecessor to LMOS which contained many of the features of LMOS, but with a different computer architecture, in 1973.

(*ii*) An operational and economic review by AT&T, with assistance from Bell Laboratories, of the first LMOS installation in 1975, and

(*iii*) A field trial of the MLT system in 1978.

2.5.1 Mechanized Line Record system evaluation

The general operational conclusions of the evaluation were as follows:

(*i*) The MLR system handled the work load in the trial RSB in a fashion acceptable to the users.

(*ii*) Service to the RSB customers appeared to improve with MLR.

(*iii*) A reduction in work force was possible with MLR.

Economic benefits for MLR were estimated from the evaluation as follows:

(*i*) Reduction in clerks previously required to update and maintain the manual line record file.

(*ii*) Savings in trouble tickets and associated computer costs to perform trouble analysis.

Thus, economic benefits were observed, with a possible additional benefit if the clerical labor required to manually update the MLR mechanized records could be eliminated. This potential benefit led to the development of an automatic interface between the LMOS system and the service order network.

2.5.2 Loop Maintenance Operation System evaluation

A performance evaluation of the first LMOS system was conducted by AT&T Customer Services and Data Systems organizations during 1975. The evaluation encompassed eighteen RSBs, the first of which was converted to LMOS in 1975, and one Centralized Repair Service Attendant Bureau (CRSAB) serving 1.3 million lines. Evaluation methods involved the use of questionnaires, interviews, and the examination

of pertinent data and reports. Among the specific areas reviewed were system performance, service and cost benefits, data conversion efforts, documentation, and employee acceptance of the system.

In the area of cost and benefits, the evaluation report indicated that the economic benefits attributed to LMOS in the LMOS Economic Planning Guide were substantiated. Specifically, a large reduction in the clerical force had been realized from the employment of the CRSAB concept and the elimination of clerical activity associated with line card maintenance. Additionally, substantial net capital savings was realized in recovered plant as the line records and assignment data were purified for entry into LMOS. However, these savings were not universally claimed for LMOS since it was recognized that other areas may not obtain comparable savings since these savings are a function of growth rate and present condition of plant records.

In summary, the evaluation team concluded that "LMOS is facilitating improved RSB administration in addition to providing substantial economics—we recommend that the BOCs proceed with their implementation plans."

2.5.3 Mechanized Loop Testing field trial

The objectives of the MLT field trial were (*i*) to verify and refine the projected economics for MLT, (*ii*) to refine the operational use of MLT, and (*iii*) to provide feedback to the developers on system changes and proposed enhancements. More of the verified benefits came from RSB savings (i.e., tester reduction) and less from a reduction in outside repair forces than was predicted in the model in Section 4.1. This difference between the predicted and verified sources of MLT benefits was due, in part, to the introduction of a successful Repair Force Management plan in the RSB prior to the MLT trial but after the initial MLT benefit estimate. The Repair Force Management program yielded a reduction in repair dispatches at the expense of additional testing in the RSB; thus, increasing the opportunities for MLT benefits in the RSB, while decreasing the possible MLT benefits in the repair force area.

The major impact of MLT was to eliminate two of the seven testers in the 70,000-line RSB. With the introduction of MLT, the percent of trouble reports requiring testing at a local test desk was reduced from 70 percent to less than 15 percent. Of the remaining five testers, only one was required for trouble report testing; the other four performed cable testing and dispatch, construction testing (e.g., cable throws) and interactive testing support of repair craft in the field. After the official MLT trial ended, it was possible to further deload the test desk to two testers by having screening clerks and dispatchers, using MLT, test cable throws and provide much of the testing support to the repair craft in the field.

In the area of reduced dispatches, the primary contribution of MLT was that it allowed dispatchers, with no testing experience, to perform a quick MLT test on a trouble immediately before it was dispatched. Using this "predispatch" testing, troubles that had "come-clear" between the time of the initial test and the time of dispatch could be closed out without dispatching, thus, avoiding a dispatch which would have resulted in no trouble being found. In addition, MLT was able to accurately detect Receiver-Off-Hook (ROH) conditions and avoid dispatches on these nontrouble conditions.

An additional benefit was projected based on MLT's ability to identify some cable problems (and, thus, avoid an initial dispatch of a station repair craft) and to provide distances to an open cable fault. This benefit would be realized in an environment where station repair forces are restricted from working in the cable plant.

In summary, the more important results from the MLT field trial evaluation were as follows:

(*i*) The projected benefits of MLT (Section 4.1) were substantiated.

(*ii*) The MLT system provides accurate measurements of the electrical characteristics of both good and faulted loops, as measured from the central office.

(*iii*) The RSB personnel readily adapted to the MLT system after receiving the prescribed training and gaining confidence in the accuracy of the system.

(*iv*) The MLT system and its operational use were improved because of changes and refinements throughout the trial period.

(*v*) Since MLT tests could be initiated from any LMOS console and most (>85 percent) of the MLT test results could be interpreted without testing experience, RSB testing functions could be distributed throughout the RSB rather than confining them solely to the local test desk.

III. LOOP MAINTENANCE OPERATION SYSTEM BENEFIT/COST MODEL

3.1 Loop Maintenance Operation System benefits

The methodology used in the estimate of LMOS benefits was as follows:

(*i*) The major clerical functions performed in an RSB were identified by interviewing RSB managers and observing operations in several RSBs,

(*ii*) The time (expense hours) required to perform each of the identified clerical functions was modeled based on detail data gathered over intervals ranging from two weeks to three months in 15 selected RSBs throughout the Bell System,

(*iii*) The impact (percent reduction in required hours) of LMOS on each of the clerical functions was estimated by comparing the features of LMOS with the clerical functions,

(*iv*) Resultant reductions of whole people were estimated based upon the reduction in required hours because of LMOS and the regrouping of remaining clerical functions into reasonable job positions.

Each of these steps is discussed with quantified results in the following sections:

3.1.1 Clerical functions

The major clerical functions performed in a manual (i.e., pre-LMOS) RSB are described below. The relationship of five of the seven functions to the overall trouble processing flow in the RSB is illustrated in Fig. 4.

(*i*) *Receiving trouble reports*—Includes customer contact and generation of trouble ticket.

(*ii*) *Line card activity on trouble reports*—Pulling line card, attaching to trouble report, recording trouble data on line card, and filing of line card when trouble is closed out.

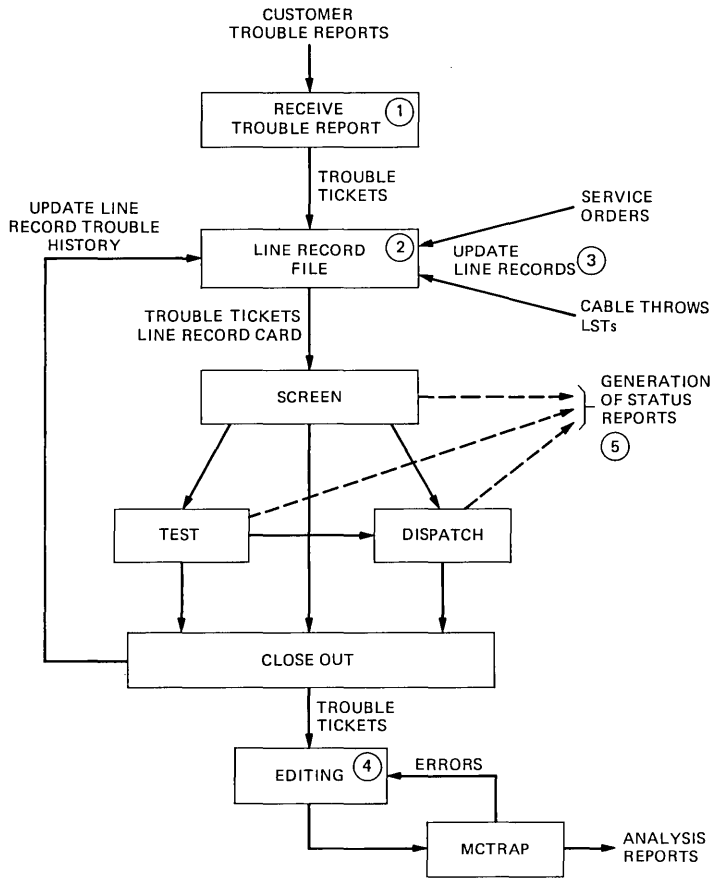


Fig. 4—Repair Service Bureau trouble report processing.

(*iii*) *Service order processing*—Updating of line record information in response to service orders, line and station transfers, cable throws, and service denial/restorals.

(*iv*) *Preparation of trouble tickets for analysis by the Mechanized* Trouble Report Analysis Program (MCTRAP)—editing of tickets for proper coding and correction of tickets rejected by MCTRAP.

(*v*) *Generation of reports*—Includes the generation of RSB work load status reports, reports of service times, and numerous other reports.

(*vi*) *Handling information requests*—Checking line cards or trouble tickets to answer requests for assignment information and trouble status.

(*vii*) *Miscellaneous*—Includes all minor functions not identified in other categories.

3.1.2 Time required to perform functions

To quantify the time spent on each of the above functions, repair clerks in 15 RSBs were asked to charge their time to one of the above functions for a period of from two weeks (in most RSBs) to three months in a few RSBs. Table III shows the percent of total clerical hours spent on each of the major clerical functions. The average RSB in the Bell System had approximately one clerk per 9K lines served, so a typical 70K line manual RSB would require eight repair clerks. In Table III, the percent of clerical hours can be translated into equivalent people, with partial people per function being completely reasonable, since a single clerk may perform several functions throughout the day.

3.1.3 Estimated impact of LMOS

The next step was to estimate the impact of LMOS on each of the clerical functions which could be affected in varying degrees by LMOS. The impact of specific LMOS features upon each of the clerical functions is discussed below:

Table III—Distribution of RSB clerical functions and estimated LMOS benefits

Function	Percent Clerical Hours	Percent LMOS Impact	Percent LMOS Reduction
Receiving trouble report	34	25	−9
Line card activity	23	100	−23
Service order processing	10	50	−5
MCTRAP preparation	7	100	−7
Report generation	8	50	−4
Information requests	9	50	−4
Miscellaneous	9	25	−2
Total reduction			−54

Assume reduction of 50 percent manual clerical hours.

(*i*) *Receiving trouble reports*—LMOS provides customer line record information to the Repair Service Attendant (RSA) during the customer contact and provides for entering trouble report information directly into LMOS, thus, eliminating the manual trouble ticket. In addition to improved customer service, the absence of manual line records and trouble tickets allows the RSAs to be physically separated from the RSB responsible for clearing the customer's trouble. This allows for centralization of RSAs, thus, allowing improved balancing of personnel and work load, particularly for night and weekend coverage of trouble calls. Once the trouble report is taken at the centralized location (typically serving one million or more customers), the LMOS system automatically prints out the trouble report and associated customer record information in the responsible RSB.

(*ii*) *Line card activity on trouble reports*—The line record data base in the LMOS computer completely eliminates this manual function.

(*iii*) *Service order processing*—A large percentage of service orders (i.e., the simpler orders) are processed by the Automatic Line Record Update (ALRU) module of LMOS and automatically updates the line record data base in LMOS. Some manual update effort is required with LMOS to input via a CRT transaction the more complex (e.g., large business) service orders and to update the line record data base for cable throws and line-and-station transfers.

(*iv*) *Preparation of trouble tickets for analysis*—The Trouble Report Evaluation and Analysis Tool (TREAT) module of LMOS provides a comprehensive trouble analysis program using the trouble report data collected within LMOS. Thus, LMOS completely eliminates the paper trouble tickets and the previous analysis tools.

(*v*) *Report generation*—Each person (e.g., screener, tester, dispatcher, etc.) that handles a trouble report enters into LMOS the current and pending status of the trouble. The LMOS can then generate status and jeopardy reports which provide a snapshot of the work load at any position in the RSB and for the outside repair craft. This significantly reduces the clerical effort involved with status reports.

(*vi*) *Information requests*—Because LMOS contains the latest customer line record information, the status of open trouble reports, and abbreviated trouble history for each customer; information requests can be handled much quicker by the RSB clerical forces.

(*vii*) *Miscellaneous*—The more efficient RSB operation provided by LMOS will have some benefit in the miscellaneous functions.

3.1.4 Quantifying LMOS benefits

The estimated reduction in RSB clerks because of LMOS is summarized in Table III. Of the clerks required in the manual RSB, approximately 50 percent could be eliminated by LMOS. Of the remaining

clerks, one-half would be transferred to a CRSAB for receiving trouble calls and the other remaining one-half would either remain in the RSB or in a data base management group to process LMOS line record updates not handled by ALRU and for miscellaneous RSB clerical functions.

In addition, there will be a reduction in first-line managers and the elimination of expenses associated with the paper trouble ticket and the computer processing previously required to perform trouble report analysis.

3.2 Loop Maintenance Operations System cost model

The calculation of costs and benefits expressed in terms of measures like the cash flow on a yearly basis and including such effects as inflation, tax treatment, accelerated depreciation rules, etc., requires the use of an economic analysis software tool.

The LMOS cost components can be divided into three major categories:

(*i*) Initial capital costs, e.g., purchase of PDP* 11/70 minicomputers,

(*ii*) Initial expenses, e.g., initial data conversion and loading expenses, and

(*iii*) Annual expenses, e.g., data processing labor.

Each of the above categories can be converted into equivalent annual charges by applying appropriate conversion factors based upon cost of money, depreciation rates, etc.

IV. MECHANIZED LOOP TESTING BENEFIT/COST MODEL

4.1 Estimated MLT benefits

The purpose of this section is to illustrate the type of data and methodology used in the earlier estimate of MLT benefits. As discussed previously in Section 2.5.3, the MLT field trial generally confirmed the total predicted MLT benefits, although the sources of the benefits differed between the predicted model and the trial results. The model of the projected MLT benefits presented in this section is based upon the pretrial data characterization of the MLT trial site, supplemented by lesser amounts of data gathered from 15 RSBs throughout the Bell System.

The expected areas of MLT benefits are listed below:

(*i*) Tester reduction

(*ii*) Reduced repeat reports

(*iii*) Reduced dispatches resulting in no trouble found

(*iv*) Reduced number of station dispatches on cable troubles

(*v*) Reduced trouble locating time for open cable troubles.

---

* Registered trademark of Digital Equipment Corporation.

Additional potential areas for MLT benefits, such as the elimination of local test desks and the use of MLT for testing cable throws, may be realizable in some situations but are not included in the benefit model discussed below.

4.1.1 Tester reduction

In the Bell System, there is approximately one tester for each 10,000 lines served. Therefore, in the typical 70K RSB depicted in Table IV, there would be 7 testers with their functions distributed as shown between regular trouble report testing and other testing functions. The distribution of testing functions is based on a sample of 15 RSBs, where the testers recorded the time spent on each of these functions for several weeks. As shown, approximately one-half of the total testing time was spent on trouble report testing prior to dispatch, with the remaining time spent interacting with station repair craft, dispatching, and testing with the cable repair forces and other tasks, such as ALIT testing, testing defective pairs, special service testing, etc. Based on observations of the regular testing function and the testing capabilities of MLT, it was estimated that 15 percent of the troubles will require testing at the local test desk in an MLT environment, compared to 60 percent in a manual testing environment. In an ARSB, one regular tester could handle a busy-day load by testing one trouble every eight minutes. The remaining troubles (85 percent) can be screened and routed by one full-time and one part-time MLT screener, if they maintain an average of screening a trouble every two minutes during the busy hour. Both the testing and screening rates appear to be readily achievable based on observations of testing operations.

4.1.2 Reduced repeat reports

A "repeat" report is defined as a customer trouble report received within 30 days of a previously cleared customer trouble report on the same line. The Bell System repeat report rate is 14 percent of all customer trouble reports. Data gathered on repeat reports in several RSBs revealed that 35 percent of the repeat reports occurred within two days after the supposed clearing of a previous trouble. Thus, it is

Table IV—Tester reduction (in a typical 70K line RSB)

Testing Function	Tester Required	
	Manual	MLT
MLT screener/tester	—	1.5
Regular trouble report testing	3.5	1.0
Interacting with repair	1.0	1.0
Cable testing and dispatch	2.0	2.0
Other	0.5	0.5
Total	7.0	6.0

strongly suspected that the original trouble was not cleared, even though the repair craft may have repaired or changed some equipment. It is estimated that with post-dispatch testing by MLT on the original dispatch, at least half of the repeat reports occurring within two days could be avoided.

4.1.3 Reduced dispatches resulting in no trouble found

The Bell System FOK-OUT rate is approximately 8 percent of all troubles received. In addition, a review of actual repair results from trouble reports indicate that on an additional 2 percent of the troubles no trouble was found on a dispatch, but the trouble was closed to dispositions other than FOK-OUT. Thus, on 10 percent of the trouble reports (equivalent to 20 percent of the dispatches), no trouble was found or repaired by the repair craft.

A distribution of the initial test results of these troubles is shown in Fig. 5. Some of the dispatches on the FOK test results possibly could have been avoided or could have resulted in found troubles because of the MLTS' more sensitive and comprehensive tests. Most of the 100-volt shorts could have been identified by MLT as a receiver-off-hook, thus, avoiding a dispatch. The remaining 23 percent of the troubles which indicated an initial fault would have, in many instances, either been detected as a time-varying fault by MLT or found to have cleared when retested prior to dispatch. Therefore, it is estimated that at least 20 percent of the no-trouble found dispatches could have been influenced by MLT, either by avoiding a dispatch or by helping the repair craft to clear the trouble and avoid a possible repeat report.

4.1.4 Reduced number of station dispatches on cable troubles

Data collected in areas where station repair craft attempted to locate and clear cable faults that were easily assessable revealed that 5 percent of all trouble reports resulted in a dispatch to both a station repair craft and to a cable repair craft before they were cleared. An examination of the test results on these double-dispatch troubles revealed that 40 percent were either open or initially tested okay but

TEST RESULT	PERCENT OF NTFs	
NO TEST	30	
GOOD LINE	40	
100-VOLT SHORT (ROH?)	7	
OTHER SHORTS	5	
GROUND	9	30
OPEN	4	
FEMF	2	
NOISE	3	

Fig. 5—Initial test results on dispatches which resulted in no trouble found.

retested open or FEMF once they were dispatched to the station repair craft. With MLT's open fault sectionalization and with retesting prior to dispatch, the initial station repair dispatch on the 40 percent could have been avoided.

4.1.5 Reduced trouble locating time for open cable troubles

In addition to avoiding the initial dispatch of a station repair craft on open cable faults, the MLT open fault sectionalization capability can be used to initially direct the cable repair craft to the area of the fault, thus, reducing the time required to locate the fault. Approximately 16 percent of all trouble reports result in an Outside Plant disposition. However, because some cable faults are cleared by station repair craft and multiple trouble reports can be associated with a single cable fault, the actual ratio of cable repair dispatches to trouble reports is 8 percent, of which 30 percent have been shown to be open. Experience from riding exercises in conjunction with the use of MLT indicates that approximately 15 minutes (out of an average dispatch time of two hours) can be eliminated from the fault locating time on open troubles.

4.2 Mechanized Loop Testing cost model

As was mentioned in Section 2.3, the cost of an MLT installation is sensitive to the number of lines that it serves.

As was described for the LMOS cost model in Section 3.2, the MLT costs can be divided into initial capital costs, initial expenses, and annual expenses. Each of these cost components can then be translated into equivalent annual costs to derive costs on a per-line-per-year basis.

V. SUMMARY

This article has illustrated the evolution of economic studies associated with the ARSB system from the early identification of a need through the development and successful field trial of the system. Based upon this experience covering more than ten years, the following recommendations are presented as being applicable not only to ARSB, but also to other systems either in development or being defined.

The economics of a system must be examined and refined at several stages throughout the definition, development, and deployment of the system. It is not sufficient to perform an economic analysis prior to the development of a system and then assume that these benefits and costs will remain constant. In particular, the continuing economic analysis must be alert to the following:

(*i*) Markets that decrease significantly because of changing BOC environments, technology changes, or alternative product availability.

(*ii*) Benefits that vanish because of changes in functions or the development of other systems that impact benefits.

(*iii*) Increasing system costs.

The economic analysis and market studies must be based upon realistic operational models. Field data characterizing the BOC operations to be impacted by a system should be gathered on-site from several different operational environments. The economic model must address the feasibility and methods for achieving reductions in people. For example, to maximize the number of people reduced by ARSB, some maintenance procedures had to be changed and some work functions redistributed among work positions in the maintenance center.

For systems, such as the ARSB, that require significant capital investments by the BOCs, it is critical that economic planning information be made available to the BOCs as soon as the system features, availability, and approximate cost can be estimated with a reasonable degree of confidence. The budgeting cycle in the BOCs is such that this planning information is typically required at least two years prior to the implementation of the system.

ACRONYMS AND ABBREVIATIONS

AC	administrative computer
ac	alternating current
ACS	active current sensor
A/D	analog/digital
ALIT	automatic line insulation testing
ALRU	automatic line record update
ALRUM	automatic line record update messages
AN	associated number
APL	application code
ARM	ALRU recovery monitoring
ARSB	automated repair service bureau
ASC	access switch controller
ATH	abbreviated trouble history
BOC	Bell operating company
BOR	Basic output report
BOS	Bell operating system
CA	cable
CAT	cumulative abbreviated trouble
CCC	communication control circuit
CCM	communication control manager
CDT	communications display terminal
CCUAP	computerized cable upkeep analysis program
CF	cable fail
CFS	C filing system
CMLR	change miniline record
COE	central office equipment
CPU	central processing unit
CRAS	cable repair administrative system
CRFMP	cable repair force management plan
CRSAB	centralized repair service answering bureau
CRT	cathode ray tube
CSO	completed service order
CTT	cable trouble ticket
CTTN	cable trouble ticket number
DBMS	data base management system
dc	direct current
DDD	direct distance dialing
DLL	dial long line
DCN	data communication network
DP	dial pulse
EAS	equipment access switch

ECC	enter cable change
EDP	electronic data processing
ESS	electronic switching system
EST	enter status transaction
FCM	frame communication manager
FE	front end
FID	field identifier
FOK-OUT	found OK outside of central office
FST	final status transaction
GPIB	general purpose interface bus
IMS	information management system
I/O	input/output
LATIS	loop activity tracking information system
LCAMOS	loop cable maintenance operations system
LMOS	loop maintenance operations system
LSI	large-scale integration
LSV	line status verifier
LTD	local test desk
LTF	loop testing frame
LTS	loop testing system
MA	maintenance administrator
MC	maintenance center
MCTRAP	mechanized customer trouble report analysis program
MDF	main distributing frame
MLC	miniline card
MLR	mechanized line record system
MLT	mechanized loop testing
MMC	maintenance management center
MMM	MLT measurement module
MOR	mini-output report
MF	multifrequency signaling
MTR	mechanized time reporting
MVS	multiple virtual system
NCTT	nonservice affecting CTT
NNX	exchange code
NPA	numbering plan area
NSA	nonservice affecting
OSDD	operations systems deliverable documentation
OSP	outside plant
OSPM	outside plant maintenance
PBX	private branch exchange
PCL	parallel communication link
PMU	precision measurement unit
POTS	plain old telephone service

PSO	pending service order
PTERM	physical terminals
RAM	random access memory
RBOR	request basic output report
RFM	repair force management
RJE	remote job entry
ROH	receiver-off-hook
RSA	repair service attendant
RSB	repair service bureau
SA	service affecting
SCC	supervisory and control circuit
SCTT	service-affecting CTT
SER	series control
SOH	service order history
SR	sanity reference
SS	special services
SUP	supervisor
TAS	telephone answering service (LMOS usage)
TAS	trunk access switch (MLT usage)
TE	trouble entry
TH	trouble history
TP	test package
TPA	test package auxiliary
TR	trouble report
TREAT	trouble report evaluation and analysis tool
TSO	time sharing option
TSL	transaction specification language
TSP	test supervisor
TV	trouble verification
USO	universal service order
USOC	universal service order code
VDT	video display terminal
WATS	Wide Area Telecommunications Services
WC	wire center
XFE	cross front end

CONTRIBUTORS TO THIS ISSUE

Salvatore J. Barbera, B.S. (Physics), 1960, Long Island University; American Telephone and Telegraph Company, 1947—. Mr. Barbera began his career as a switchman with New York Bell Telephone Company. He became a Maintenance Engineer in 1953, and held assignments as Equipment Engineer, Planning Engineer, and Budget Engineer before attaining the level of Division Engineering Supervisor-Construction Plans in 1960. Seven years later, following a number of Plant traffic and commercial assignments, he was promoted to General Plant Manager-Manhattan; and in 1971, he was named Assistant Vice President-Network Operations. He joined American Telephone and Telegraph Company as Engineering Director-Switching in May, 1973. From July 1, 1977 to December 31, 1979, Mr. Barbera was General Manager, Corporate Engineering and Product Planning in Western Electric Company's Corporate Engineering Division. In January, 1980, he was appointed Assistant Vice President and Director of Major Projects in American Telephone and Telegraph Company's Business Marketing Department. Mr. Barbera has published several technical papers in the International Conferences, 1976 and 1977, on Communications. In addition, he has published and presented technical papers at the International Switching Symposium, October 25, 1976—Assuring the Integrity of Switching Equipment in the Bell System—and at the International Switching Symposium, May 7, 1979—Assuring the Integrity of Electronic Switching Systems.

Robert F. Bergeron, Jr., Sc.B. (Applied Mathematics), 1964, Brown University; Ph.D. (Applied Mathematics), 1968, Massachusetts Institute of Technology; Bell Laboratories, 1968—. Mr. Bergeron's first project for Bell Laboratories was in the area of analytical fluid mechanics. In 1972, he became supervisor of a group doing exploratory work on the Automatic Main Distributing Frame, a system to automate connections to the cable plant in telephone central offices. After a year at M.I.T.'s Laboratory for Computer Science, he joined the Loop Maintenance Operations System project in early 1978 as supervisor of the Data Base Systems Group. He currently supervises a group responsible for LMOS front end transactions and data bases. Member, Sigma Xi, Association of Computing Machinery.

Michael H. Bianchi, B.S.E.E., 1970, Drexel University; Western Electric Co., 1970-1976; Bell Laboratories, 1976—. At Western Electric

Company, Mr. Bianchi was involved in developing systems for the SAFEGUARD Sprint Missile, circuit design, equipment use analysis, and business office support. At Bell Laboratories, he has worked on the design and implementation of the Cable Repair Administrative System and has been involved with the use of *UNIX** software since 1974. Presently, he is a member of the Information Processes and Architecture Group, and his responsibilities include building software tools to automate portions of the development process. Member, Association of Computing Machinery.

Phillip S. Boggs, B.S. (Applied Mathematics), 1968, Georgia Institute of Technology; M.S., 1970, North Carolina State University; Bell Laboratories, 1968—. Early in his career at Bell Laboratories, Mr. Boggs held various assignments associated with military systems. Later he was responsible for defining requirements for an operations support system for Bell System Repair Service Bureaus. Presently, Mr. Boggs is supervisor of the LCAMOS Predictor and Tracker Design Group. This group is responsible for the development of computer systems that perform pattern analysis of random inputs from multiple sources and for operations systems for tracking maintenance activities for various departments within Bell operating companies. Member, IEEE, Tau Beta Pi.

Miles W. Bowker, B.S. (Electrical Engineering), 1940, Swarthmore College; M.S. (Electrical Engineering), 1950, Stevens Institute of Technology; Bell Laboratories, 1940—. At Bell Laboratories, Mr. Bowker was initially involved with development of methods for the maintenance of coaxial and exchange cable in the Outside Plant Department. During World War II, he made contributions in the field of underwater systems. He has held various management responsibilities in the development of ocean cables, circular waveguide for long distance transmission circuits, underwater transmission, and maintenance planning for cables in the Bell System. He has also headed a group engaged in the design and development of new strategies and testing systems to ensure high quality transmission reliability for customers telephone facilities. Currently, Mr. Bowker is head of the Maintenance Systems Engineering Department, which is responsible for systems requirements and exploratory development associated with maintenance systems. Member, IEEE, Sigma Si, Sigma Tau.

* Trademark of Bell Laboratories.

Stephen G. Chappell, B.S. (Electrical Engineering), 1969, Georgia Institute of Technology; M.S. (Electrical Engineering) 1971, Northwestern University; Ph.D. (Computer Science), 1973, Northwestern University; Bell Laboratories 1969—.After joining Bell Laboratories, Mr. Chappell worked on LAMP (Logic Analyzer for Maintenance Planning), a logic circuit simulation and automatic test generation facility for the No. 1A and No. 4 ESS. He was promoted to supervisor in 1973 and was responsible for development of functional-level logic circuit simulators in LAMP. Later, he was responsible for the language and compiler for EPLX (ESS Programming Language—Extended), a high-level programming language for the No. 4 ESS. He was promoted to Department Head in 1978, responsible for the trouble processing part of the Loop Maintenance Operations System. Member IEEE, Tau Beta Pi, Eta Kappa Nu, Sigma Xi.

O. Bruce Dale, B.S.M.E., 1964, M.S.M.E., 1967, University of Maryland; Ph.D, 1970, Purdue University, Bell Laboratories, 1964—. Since joining Bell Laboratories, Mr. Dale has worked on the design of loop apparatus, the stress and vibrational analysis of ocean cables, the development of the COSMIC frame system and the COSMOS mechanized assignment system, and, presently is the head of the Mechanized Loop Testing Department.

Les S. Dickert, B.S.E.E., 1965, University of South Carolina; M.S.E.E., 1969, New York University; Bell Laboratories, 1969–1978. Mr. Dickert worked on development of Safeguard System support software. He supervised groups responsible for development of TREAT and LMOS software. He is currently Department Chief, Software Patent Licensing, Western Electric Co.

Carol M. Franklin, B.S. (Chemistry and Mathematics), 1973, Salem College; M.A. (Applied Mathematics), 1976, University of Carolina, Greensboro; Bell Laboratories, 1973—. Ms. Franklin has been involved in the development of the Loop Maintenance Operations System and the Cable Repair Administrative System. She is now supervisor of the LMOS Design Group.

Richard F. Gauthier, B.S. (Physics), 1967, M.I.T.; M.S. (Physics), 1971, University of Illinois; Ph.D. (Psychology), 1977, Stanford University; Bell Laboratories, 1978–1981. Mr. Gauthier was on the faculty at San Jose State University. At Bell Laboratories, he worked in the area of human performance analysis and design on the Mechanized

Loop Testing (MLT) project, as well as in the area of management information systems development. Member, American Psychological Association, Human Factors Society.

Robert J. Glushko, B.A. (Experimental Psychology), 1974, Stanford University; Ph.D. (Cognitive Psychology), 1979, University of California, San Diego; Bell Laboratories, 1979—. Mr. Glushko's first assignment after joining Bell Laboratories was the design and development of on-line documentation and other information subsystems for the Cable Repair Administrative System. This project evolved into a broader effort to build a "system for on-line information development." The goal is to improve the productivity of Bell Laboratories developers by "human factoring" the development environment and by borrowing and generalizing software development tools to mechanize and improve the traditional methods for developing and delivering documentation. Currently, he is a "computer psychologist" in the Information Processes and Architecture Group. Member, IEEE Computer Society, American Society for Information Science.

Ward A. Harris, Ph.D., (Behavioral and Management Science), 1971, University of Oregon; Bell Laboratories, 1978—. Early in his career at Bell Laboratories, Mr. Harris was engaged in human factors work on the user interface of the Loop Maintenance Operations System (LMOS) and the Mechanized Loop Testing (MLT) system. Currently, Mr. Harris supervises a group responsible for design of the user interface of the Job Management Operations System (JMOS). Member, American Psychological Association, Human Factors Society.

Frances H. Henig, A.B. (Mathematics), 1964, Wheaton College, Mass., M.S. (Mathematics/Computer Science,), 1967, Stevens Institute of Technology; Bell Laboratories, 1964—. Ms. Henig is currently Head of the Loop Engineering Operations Department, managing the development of computer systems to assist the operating companies in planning and implementing their Loop Plant facilities. In her last assignment, Mrs. Henig supervised a group developing the applications portion of the LMOS-2 front end software. She previously had responsibility for systems programming and computer center planning groups. Member, IEEE.

James P. Holtman, B.A. (Electrical Engineering), 1968, New Mexico State University; M.S. (Electrical Engineering/Computer Science), 1969, University of California, Berkeley; Bell Laboratories—February

9, 1968. Mr. Holtman began working at Bell Laboratories in the development of operating systems for military systems. He supervised the operating system development group on the Loop Maintenance Operations System (LMOS). His current department is responsible for the planning and development of a large transaction system to mechanize the service order processing in the Bell System.

Grace H. Leonard, B.A. (Psychology), 1965, Gettysburg College; M.A. (Experimental Psychology), 1969, University of Delaware; CIBA Pharmaceutical Company, 1965–1967; Bell Laboratories, 1969—. Ms. Leonard began her career in Bell Laboratories in the Human Performance Technology Center, where she provided human performance design and evaluation support for various microfilm and computer-based systems. In 1977, she joined the Mechanized Loop Testing Department and was responsible for human performance design for the Mechanized Loop Testing Systems. She is currently supervisor of the Human Performance Engineering Group in the Customer Services Advanced Development Department. Member, Human Factors Society, Psi Chi, Phi Beta Kappa.

Robert L. Martin, Sc.B. (Electrical Engineering) 1964, Brown University; M.S. (Electrical Engineering), 1965, M.I.T.; Ph.D. (Computer Science), 1967, M.I.T.; Bell Laboratories, 1967—. When Mr. Martin joined Bell Laboratories, he worked on the SAFEGUARD Antimissile system project for five years. He was assigned to loop maintenance work in 1972 and worked on the predecessor to LMOS, the Mechanized Line Record (MLR) system. He was named Director, Loop Maintenance Systems Laboratory in 1978 and Director, Assignment Systems Design Laboratory in 1979. In May, 1981, Mr. Martin was appointed Executive Director, Customer Network Operations Division. Member, Tau Beta Pi, Sigma Xi.

John R. Mashey, B.S. (Mathematics), 1968; M.S. 1969, Ph.D. (Computer Science), 1974, Pennsylvania State University; Bell Laboratories, 1973—. Mr. Mashey's first assignment at Bell Laboratories was with the Programmer's Workbench Project. There, and later elsewhere, he worked on text-processing tools, command-language development, and *UNIX** operating system usage in computer centers. Since 1978 he has worked in the Loop Maintenance and Engineering Operations Laboratory, where he managed the development of the

* Trademark of Bell Laboratories.

Cable Repair Administrative System (CRAS), and now supervises the Information Processes and Architecture group. His interests include programming methodology and the interactions of software with people and their organizations. Mr. Mashey is an ACM National Lecturer. Member, Association of Computing Machinery.

Edmond A. Overstreet, B.S.E.E., 1963, Virginia Polytechnic Institute; M.S.E.E., 1965, Duke University; Bell Laboratories, 1963—. When Mr. Overstreet joined Bell Laboratories, he worked on several military projects, including the SAFEGUARD system. In 1970, he moved to the Loop Maintenance area, where he was involved in requirements development, economic analysis, and field-trial evaluations of new loop maintenance support systems; in particular, the Loop Maintenance Operations System (LMOS) and the Mechanized Loop Testing (MLT) system. He was promoted to supervisor in 1977, and assumed his present position as supervisor of the Loop Maintenance Operations Planning Group in 1980. This group is responsible for generating requirements for computer systems which aid the Bell operating companies in the processing and repair of troubles in the loop plant. Member, IEEE, Phi Kappa Pi, Eta Kappa Nu.

Steven P. Rhodes, B.A. (magna cum laude, Mathematics), 1969, North Texas State University; M.S. (Computer Science), 1971, Ph.D., 1973, University of California at Berkeley; Bell Laboratories, 1973—. Mr. Rhodes began his career working in the area of host processor system support and performance tuning. With the increasing emphasis on minicomputers, he shifted to developing on-line software subsystems for front-end processors. Later, he was responsible for the software development of several systems to test telephone subscribers' lines. Presently, he is supervisor of the Data Base Design Group. This group is responsible for analyzing and designing software for large data base systems. These systems are for the host processor of the Loop Maintenance Operations System. The group is also involved in performance tuning and capacity planning activities.

Thomas W. Robinson, B.S.E.E., 1960, University of South Carolina; M.S.E.E., 1962, New York University; Bell Laboratories, 1960–1978; Southern Bell Telephone and Telegraph Company, 1978—. Mr. Robinson's initial assignments were associated with the SAFEGUARD project and involved electronic circuit design of missile-borne assemblies for the Sprint and Spartan missiles. Later assignments at Bell Laboratories included responsibility for the design of a detection of

unauthorized equipment system and portable test sets used by installation and cable repair technicians in the Bell Operating Companies. Mr. Robinson is currently a District Staff Manager in Southern Bell and in the Corporate Planning department.

Marc J. Rochkind, B.S.M.E., 1970, University of Maryland; M.S.M.E., 1972, Rutgers University; M.S.C.S., 1976, Rutgers University; Bell Laboratories, 1970—. From 1972 to 1975, Mr. Rochkind helped develop the Programmer's Workbench, which was a system of support tools that did for programmers what a carpenter's workbench does for carpenters. His primary contribution was the Source Code Control System, which manages changes to text files. From 1975 to 1977, he worked on DIRECT II, which was a system to generate telephone directories. He joined the Loop Maintenance Operations System project as Supervisor of the Transaction Languages Group in 1978. Mr. Rochkind is currently Supervisor of the Advanced Processing Group at the Denver Laboratories, where he directs exploratory work in office automation.

Harvey Rubin, B.S.E.E., 1965, M.S.E.E., 1966, Eng.Sc.D., 1970, Columbia University; Bell Laboratories, 1970—. Mr. Rubin has worked on switching system software design, transmission system design, and hardware and software design for mechanized loop testing systems. He is currently supervisor of the Loop Testing Communications and Applied Research Group. Member, Sigma Xi, Tau Beta Pi, Eta Kappa Nu, IEEE.

Leo Schenker, B.S. (Civil Engineering), 1942, University of London; M.S., 1950, University of Toronto; Ph.D., 1954, University of Michigan. Bell Laboratories, 1954—. After joining Bell Laboratories, Mr. Schenker was involved in the development of telephone station equipment, including *TOUCH-TONE*® dialing and the *TRIMLINE*® phone. In 1968, he was promoted to Director of the Military Electronic Technology Laboratory in North Carolina and, in 1971, became Director of the Loop Maintenance Systems Laboratory, which was closely involved with the development of new software and hardware systems aimed at reducing the expense of maintaining the customer's telephone service. In 1978, he became Director of the Loop Systems Engineering and Methods Laboratory and, in 1979, he was made Director of the Customer Network Operations Systems Engineering Center. Mr. Schenker is Executive Director of the Central Office Operations Division. This division provides operation systems to the telephone com-

panies and does a wide variety of related engineering and support activities. Mr. Schenker has been awarded seven patents in connection with *TOUCH-TONE* dailing, *PICTUREPHONE*® meeting service, and *TOUCH-A-MATIC*® repertory dialer. Fellow, IEEE; member, Sigma Xi, Phi Kappa Phi.

Eugene J. Theriot, B.S.E.E., 1956, Louisiana Sate University; M.S.E.E., 1965, North Carolina State University; Bell System, 1955—. Bell Laboratories, 1960—. Mr. Theriot's background includes employment by Southern Bell Telephone and Telegraph Co. and Western Electric Company. At Bell Laboratories his work included design of microwave circuitry for SAFEGUARD missile systems and development of a cable pressure monitoring system for telephone exchange cable use. He is currently head of the No 5 ESS Peripheral Circuits Department.

F. Joseph Uhrhane, B.S. (Physics), 1960, Ph.D. (Physics), 1971, Stanford University; Bell Laboratories, 1971—. At Bell Laboratories, Mr. Uhrhane first worked on radar methods for detecting underground obstacles in the outside plant. He then did system design and development work in interconnection for advanced distributing frame systems. More recently, he has been responsible for systems engineering for the Mechanized Loop Testing (MLT) test strategies and algorithms and for the integration of MLT with service order and line record systems of the Loop Maintenance Operations System (LMOS). Member, Sigma Xi.

Robert W. Vetter, B.E.E., 1968, Pratt Institute; M.S. (Electrophysics), 1969, Polytechnic Institute of Brooklyn; Bell Laboratories 1968—. After joining Bell Laboratories, Mr. Vetter was involved with the design of strip line circuits for the CAMAR and TACMAR RF Processor and RF Microwave Converter of SPRINT and SPARTAN missiles in the SAFEGUARD Antiballistic Missile system. Later he worked on the logic design of hardware for the Mechanized Line Record (MLR) system which evolved into the Loop Maintenance Operations System (LMOS). Currently, Mr. Vetter is supervisor of the Testing Requirements Group, which is responsible for specifying the functional requirements for the Mechanized Loop Testing System and for negotiating design requirements, as they pertain to loop testing, for pair gain system and loop electronics. Before his present assignment, he worked for Southern Bell for a two-year operating telephone company assignment and returned to Bell Laboratories to work on

systems engineering for Automated Repair Service Bureaus. Member, Tau Beta Pi, Eta Kappa Nu, Pi Mu Epsilon, IEEE.

John F. Vogler, B.S.M.E., 1958, North Carolina State University; Bell Laboratories, 1961—. Mr. Vogler is currently supervisor in the Maintenance Systems Engineering Department, with responsibilities in the area of data base management and mechanization of loop maintenance operations.

David S. Watson, B.S. (Mathematics), 1962, and M.S. (Mathematics), 1964, University of South Carolina; Bell Laboratories, 1964—. Mr. Watson's early work at Bell Laboratories included operating system development for military real-time systems and applied research in computer graphics software. He supervised groups which developed various software components of the Loop Maintenance Operations Systems (LMOS). In 1977, he was appointed department head responsible for design of software to support network operations and system test for the Bell Data Network Project. He assumed his current position as head of the FACS Operating System and Languages Department at Piscataway in 1980.

John E. Zielinski, M.E., 1964, M.S., 1968, Stevens Institute of Technology; M.B.A., 1981, Rutgers University; Bell Laboratories 1969–1972, 1976—. Mr. Zielinski is currently Supervisor, Information and Decision Sciences Group, with personnel subsystem development responsibilities for Loop Maintenance Operations System and Loop Cable Administrative and Maintenance Operations System. Additional responsibilities include applied research in Artificial Intelligence (Expert Systems), speech recognition/synthesis, and decision support systems.

THE BELL SYSTEM TECHNICAL JOURNAL is abstracted or indexed by *Abstract Journal in Earthquake Engineering, Applied Mechanics Review, Applied Science & Technology Index, Chemical Abstracts, Computer Abstracts, Current Contents/ Engineering, Technology & Applied Sciences, Current Index to Statistics, Current Papers in Electrical & Electronic Engineering, Current Papers on Computers & Control, Electronics & Communications Abstracts Journal, The Engineering Index, International Aerospace Abstracts, Journal of Current Laser Abstracts, Language and Language Behavior Abstracts, Mathematical Reviews, Science Abstracts* (*Series A, Physics Abstracts; Series B, Electrical and Electronic Abstracts; and Series C, Computer & Control Abstracts*), *Science Citation Index, Sociological Abstracts, Social Welfare, Social Planning and Social Development*, and *Solid State Abstracts Journal*. Reproductions of the Journal by years are available in microform from University Microfilms, 300 N. Zeeb Road, Ann Arbor, Michigan 48106.


Bell System