THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 2, February 1977
Printed in U.S.A.

1A Processor:

Maintenance Software

By P. W. BOWMAN, M. R. DUBMAN, F. M. GOETZ,
R. F. KRANZMANN, E. H. STREDDE, and R. J. WATTERS

(Manuscript received July 16, 1976)

Comprehensive maintenance software is required to meet the system reliability objective of less than an average of 2 minutes per year of outage from all causes. The function and interrelationship of the four basic maintenance programs (fault recognition and recovery, diagnosis, trouble location, and error analysis) are detailed here. Results of extensive laboratory testing and early field experience indicate that the maintenance objectives will be achieved despite the size and complexity of the 1A Processor.

I. INTRODUCTION

Like the processors for the earlier electronic switching systems (e.g., No. 1 ESS, No. 2 ESS), the 1A Processor depends on integrating maintenance software with the hardware to (*i*) quickly recognize a fault condition, (*ii*) isolate and configure around the faulty subsystem, (*iii*) diagnose the faulty unit without interfering with normal processor functions, and (*iv*) assist the maintenance personnel in locating and correcting the fault. The system usually detects a fault and reconfigures itself within a few milliseconds without affecting calls being switched through the office. The 1A Processor writable program stores and high-throughput disk and tape subsystems, including direct memory access, have introduced both new maintenance problems and new avenues for their solution. Although these bulk storage systems have complicated the process of fault recognition and diagnosis, they have also provided the key to vastly improved trouble location and error analysis.

Each aspect of maintenance software is designed to minimize the likelihood of system outages. As discussed in Ref. 1, a reliability objective

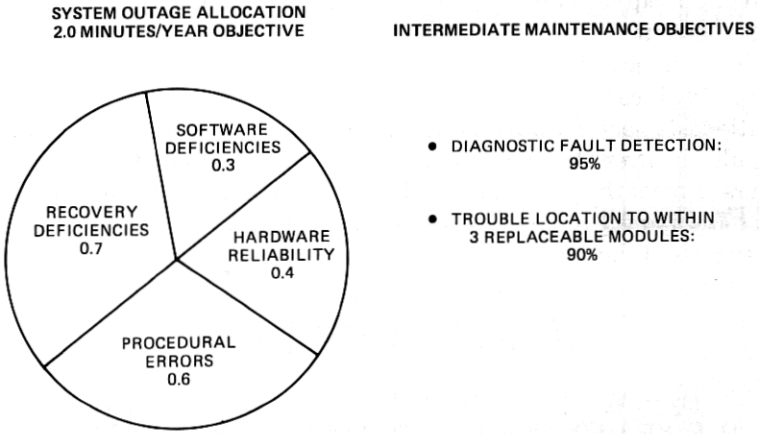

Fig. 1—System outage allocation and intermediate maintenance objectives.

is to keep the average accumulated downtime of 1A Processors at no more than 2.0 minutes per year. To achieve this objective, the probable causes of system outages are assigned to one of the four general categories shown in Fig. 1 and allocated a reasonable portion of the total system downtime. This allows the intermediate reliability objectives discussed in the following to be set for some components of maintenance software.

"Software deficiencies" can cause outages by improper system cycling. To minimize this source of downtime, overall program cycling is continually monitored, data integrity is checked using extensive auditing procedures, and thorough system integration tests are performed after program changes are introduced. Outages resulting from software deficiencies are not expected to average more than 0.3 minute of downtime per year.

When outages occur because a full complement of working units is not available to establish a system configuration, they are included in the "hardware reliability" category and are allocated an average 0.4 minute of downtime per year. Two maintenance software components, diagnosis and trouble location, bear strongly on hardware availability. As a result, intermediate maintenance goals have been set. Experience shows that, while it is costly in both hardware and software to develop diagnostic test programs capable of detecting every fault that can occur in a unit, it is generally feasible to develop sufficient maintenance access and diagnostic tests so that 95 percent of the faults (as measured using simulation results) can be detected. Therefore, the minimal level of fault

detection required for each individual diagnostic program has been set at 95 percent.

Fault-detection capability is by far the most important diagnostic property since the processor assumes a unit to be fault free if all diagnostic tests pass. However, it is not sufficient to detect the presence of a fault; the average repair time of units must generally be less than 2.0 hours to meet the previously stated hardware reliability objective. With time-consuming repairs, the behavior of the unit might be marginal or intermittent; however, most repairs should be completed in less than half the average repair time. Hence, an intermediate maintenance objective was established; at least 90 percent of the faults should be isolated to no more than three replaceable modules by the on-line trouble-locating procedure.

"Procedural errors" are expected to account for about 0.6 minute of downtime per year. Attempts to minimize this major cause of system outage include careful design of the human interface with emphasis on documentation clarity and uniformity, reduction in the number of manual operations and translations, and defensive programming implementations.

As depicted in Fig. 1, the largest source of system outage is expected to be "recovery deficiencies." Building on the strategy implemented in No. 1 ESS, the 1A Processor fault-recognition programs are generally interrupt driven and attempt to assemble a working configuration whenever a system error or fault is detected. Setting intermediate performance objectives for this software-maintenance component is difficult because of the large number of variables involved (the system can be in almost any state when a fault occurs) and because recovery is strongly related to all other components of maintenance. For example, recovery can be either facilitated or thwarted by manual procedures and can be easily misled by incomplete diagnosis. Clever use of failure symptoms collected by the error analysis program could obviate the need of some later system-recovery actions, but there is no guarantee that all impending troubles will be identified and isolated before they can jeopardize system operation.

In summary, all four maintenance software components are strongly interrelated. The fault-recognition-and-recovery program, in response to an interrupt, makes a tentative decision about the health of a unit; it automatically requests diagnosis upon suspicion of a fault, deferring the ultimate decision to the diagnostic program. Most diagnostic failure results are used to automatically pinpoint the fault to a few replaceable modules. Problems that elude detection or isolation by any of these maintenance components will have to be manually located. To assist maintenance personnel in these situations, the error-analysis software collects and files all trouble symptoms, and retrieves them on request.

II. FAULT RECOGNITION AND RECOVERY

2.1 Subsystem redundancy

The subsystem loss objective of only 0.4 minute of downtime per year is achieved in part through subsystem redundancy. The functions implemented in some units are so critical (for example, the central control) that they require at least full duplication of these units. Reliability calculations show that redundancy greater than full duplication is not required in any subsystem. In fact, unlike No. 1 ESS, subsystems containing many units do not require full duplication to meet the hardware reliability objectives. For memory units, redundancy is influenced by the ease with which data stored in a failing unit can be regenerated. Since a program store can be reloaded from the file-store system in less than a second, a roving spare redundancy plan is sufficient for the program-store community. The call stores, however, contain both data backed up on the file stores and call-related transient data. This transient data can be regenerated only through phases of memory reinitialization that interrupt system operation for many seconds and terminate calls that are not in the talking state. Therefore, the limited-spare concept used for the program store is expanded for the call stores to include sufficient spares to provide full duplication of transient data. The file stores contain the backup data for the call/program stores and transient data that is accumulated over long time periods. Neither type of data can be regenerated easily and, therefore, full duplication of the file stores is provided. Table I summarizes the redundancy plan for each processor subsystem.

2.2 Hardware fault detection

Hardware redundancy alone is not sufficient to meet the 1A Processor reliability requirements. Rapid fault detection and reconfiguration is also needed. Fault detection is accomplished through both hardware and software checks, with the emphasis on hardware due to its inherent

Table I — 1A Processor redundancy plan

Full Duplication	Limited Spares	Duplicated Bus Access
Bus systems	Call stores (CS)*	Call stores (CS)
Central control (CC)	Input/output unit channels (IOUC)	Central control (CC)
Data unit selectors (DUS)	Program stores (PS)	Input/output unit selectors (IOUS)
File stores (FS)	Tape units	Master control console (MCC)
Input/output unit selectors (IOUS)		Program stores (PS)

* Sufficient spares to duplicate transient data.

Table II — Hardware fault detection techniques

Unit	Techniques
Central control	Matching of bus transmissions and internal operations Parity Protected address range Timing
Call program stores	Parity Operation checks (access current, etc.) Timing Acknowledgments
File stores	Parity Cyclic redundancy code Internal matching Timing Operation checks (disk speed, etc.) Acknowledgments
Data unit selectors/ tape unit controllers	Parity Cyclic redundancy code Timing Operation checks (invalid mode/command, etc.) Acknowledgments Loop-around checks
Input/output units	Internal parity Timing Operation checks (invalid command, etc.) Loop-around checks Acknowledgments

speed.[2] Since a single fault-detection technique cannot solve the problems of all subsystems, several techniques are used. Table II summarizes the 1A Processor units and the techniques used for each. A detailed description can be found in the articles describing the 1A Processor units.[3,4] The 1A Processor design was improved over that of No. 1 ESS by extending the self-checking capability of all units. Each unit employs one or more of the techniques listed in Table II to verify its own operation. Self-checking speeds up fault detection by minimizing the reliance upon timing and software checks. It also aids the fault-recovery process by providing error indications that help to isolate the faulty unit.

2.3 Software error detection

While designed primarily to detect and correct data mutilation due to translation or program errors, software error detection provides a backup for the hardware fault-detection circuits. Undetected hardware faults may lead to data mutilation or loss of program sanity. The fault-detection circuits also provide a backup for software error detection. Invalid program actions, such as out-of-range memory references, generate hardware check failures. Because error-detection techniques are interrelated, the hardware- and software-recovery philosophies are also

Table III — Maintenance interrupt structure

Function	Level	Source
System configuration	A	Manual from MCC
	B	Processor configuration, CC activity switch, CC pulse-source failure
Fault detection	C	CC mismatch
	D	CS or AU failure
	E	PS failure
	F	PU failure
Test	G	Interval timer Utility match tests
Fault detection	Maintenance Interject	AU failure PU failure
	Base level Maintenance	AU failure PU failure

interrelated. Software recovery is initiated when the level of maintenance activity due to invalid program operations becomes high enough to affect service. Tests and reconfigurations of the 1A Processor are initiated when software recovery is unable to resolve error-check failures through transient memory initialization.

Similar to hardware fault detection, software error detection can take on many forms. These include timing checks, error codes, in-line defensive checks, data-structure checks, and reasonableness checks based upon redundancy in the data. A detailed discussion of these can be found in Refs. 5 and 6.

2.4 Maintenance interrupt structure

When a fault is detected by a check circuit, call processing is interrupted and fault-recovery actions are initiated. This interruption can fall into one of three priority categories determined by the severity of the fault: (*i*) immediate interrupt (maintenance interrupt) if the fault is severe enough to affect program execution, (*ii*) interrupt deferred until completion of the currently active task (maintenance interject) if the problem could affect several calls or tasks, and (*iii*) interrupt deferred until detected by routinely executed base-level jobs (base-level maintenance) if the problem affects only a single call or task.

Maintenance interrupts are assigned a priority based upon the subsystem in which the fault is detected. High-priority interrupts are allowed to occur during the processing of lower-priority interrupts, but not vice versa. The only exception to this rule is that B-level interrupts, which generally indicate a loss of system sanity, can occur while processing a manually initiated A-level interrupt. Table III summarizes the 1A Processor maintenance interrupt structure.

2.5 Fault-recovery strategy

The goal of all fault-recovery programs is to recover call-processing capabilities. This is accomplished in three steps: identification of the faulty unit, isolation of that unit, and reconfiguration and initialization of spare units. While specific actions performed by the fault-recovery programs are determined by each subsystem for which the program was designed, there are several features common to the design of all 1A Processor fault-recovery programs.

The major common feature is minimizing the effect of nondeferrable maintenance activity. This generally means minimizing the execution time. A balance between fault detection and execution speed is generally achieved through a first-look strategy in which fault-detection testing is directed towards a particular unit or part of a unit based upon the circumstances in which the fault was detected. For example, the central control fault-recovery program functionally partitions the central control based upon the instruction being executed when a mismatch occurred. A subset of all fault-recovery tests is then selected based upon the partitioning.

When fault recovery involves accessing the disk files, the duration of the interrupt is determined by file-store access time and not by the test-execution time. Therefore, when the first-look checks indicate a disk-file problem, the file-store fault-recovery program terminates interrupt processing and accesses the disk file as a deferred time-shared job. The call/program-store fault-recovery programs attempt to minimize the loading of call/program stores from file store on interrupt. This is accomplished by assigning the spares to duplicate units in which transient errors are occurring and loading these stores through a deferred time-shared job. When a load on interrupt cannot be avoided, the effect upon the system is minimized by interleaving critical call processing with the load.

In the auxiliary data system, minimizing the effect upon system operation means elimination of configuration changes which require manual tape changes. Therefore, the data unit fault-recovery program will leave in service units that have configuration-sensitive faults if a configuration can be established that passes access tests.

Another major common feature of 1A Processor fault-recovery programs is the use of short-term error analysis, which improves the tolerance of the programs to intermittent faults over that achieved with No. 1 ESS programs. Short term does not imply a common time interval. Instead it refers to those error records collected and analyzed by the fault-recovery programs as opposed to those collected and analyzed by the system error-analysis program. The records indicate units in which faults or transient errors have occurred and the response of the fault-

recovery programs to those faults or errors. If analysis of the records indicates that the system has not been restored to interrupt-free operation, the fault-recovery program modifies its response to the next fault or error. The next fault detected generally results in abandoning the first-look strategy and executing all fault-detection tests. The next transient error results in isolation of the unit experiencing a high error rate.

A third major common feature is the "bootstrap" strategy. Fault recovery normally consists of identifying the faulty unit and replacing it with an operational spare. If a spare is not available, the fault-recovery program executes what is referred to as a bootstrap. During a bootstrap, the previous status of all units in the subsystem upon which fault-recovery actions are being performed is ignored, the units are tested by the recovery program, and a working subsystem configuration is established using units that pass the tests. Repeated entries to a bootstrap routine in a predetermined time interval indicate a failure to configure a fault-free subsystem, perhaps due to inadequate subsystem tests. When this occurs, the fault-recovery programs combine short-term error analysis with test results to systematically isolate units on successive interrupts that result in bootstraps.

Another common feature of the 1A Processor fault-recovery programs is control of all deferrable configuration requests. All manual and diagnostic requests to modify a 1A Processor subsystem configuration are submitted to the appropriate fault-recovery program. The configuration is changed only after determination that there will not be an effect upon system operation. With one exception, this means that a unit must be isolated and replaced with a spare before it can be diagnosed. The exception occurs when data unit selectors and tape units must be diagnosed. Since the auxiliary data system fault-recovery program may leave in service units that have configuration-sensitive faults, but which are currently in an error-free configuration, it allows them to be diagnosed when not in use by the data-unit administration program.

2.6 Software audits

Each fault-recovery and administrative program includes program units designed to audit or initialize transient data. These audits are executed on a timed basis or by the application audit controller on a routine basis. The 1A Processor software package also includes two audits designed to detect and correct errors in the nontransient call/program-store and file-store data. The first of these is a routinely executed audit that verifies the data through the calculation of error codes or hash sums. The hash sums isolate errors to 1024-word blocks and identify which copy of data (call/program store, file-store copy 0, or file-store copy 1) is valid.

This copy is then used to correct those in error and to identify specific words in error.

The second audit is manually initiated when the first audit detects an error that it cannot correct. It identifies errors through a simple comparison of the data with a tape backup. The backup tapes are periodically generated in the office and may not reflect the most recent changes to the office-dependent data. Therefore, the tape audit checks all apparent errors against an internally stored list of approved changes before marking a word for correction.

2.7 Processor configuration recovery

A general 1A Processor bootstrap recovery is automatically executed when check circuits indicate loss of processing sanity or when a fault-recovery program fails to configure a working subsystem. This bootstrapping, referred to as processor configuration (PC) recovery, occurs on one of three levels corresponding to the three sets of states in the PC sequencer in the central control. In each level, a complete processor is configured by building upon the basic configuration (central control, program store, and program store bus) established by the PC sequencer. Test and configuration routines in each of the fault-recovery programs are executed as subroutines of the PC recovery program to configure each processor subsystem. The first level, corresponding to states 0 through 15 of the PC sequencer, attempts to minimize execution time by executing the call store and program store copies of the fault-recovery programs. In the second level, corresponding to states 16 through 48, all nontransient data are verified before being used during the recovery. This level begins with a hardware-initiated load of a small program from a file store. The small program initiates the verification of data through subroutines in the nontransient data audit and also initiates the execution of the fault-recovery test and configuration routines. The final level corresponds to PC sequencer states greater than 48 and is called the repeated PC. It is entered once the fault-recovery programs have unsuccessfully attempted to build a complete processor from each unique basic configuration. Recovery steps in this level are similar to those of the second level. They differ in the selection of fault recognition tests. Less stringent tests are executed in the third level in an attempt to configure a system capable of performing very basic call-processing functions.

2.8 Manual recovery

Failure of the processor-configuration recovery sequence to establish a viable processor configuration necessitates manual-recovery procedures. These are invoked through controls at the master control console.

The first manual-recovery step taken consists of establishing a basic configuration using the override-control keys and requesting the second or third level of PC recovery. The override-control keys have the advantage over the processor-configuration sequencer of being able to force a basic configuration which fault-recovery programs cannot change.

Failure to recover system sanity through the override controls may be due to mutilated nontransient data in both the call/program stores and the file stores. Therefore, the next step in manual recovery is to reload this data from tape. This type of recovery (called a system reinitialization) is begun by using manually activated sequencers to load a small bootstrap program from tape into the basic processor. The program initiates the load of the remaining data from tape and directs programs loaded with this data to configure a complete processor.

The final set of manual-recovery procedures has no counterpart in No. 1 ESS. It involves forcing the system into an emergency mode of operation in which only manually initiated tasks are executed. All other tasks including call processing are excluded. In the event of excessive call/program-store or file-store failures, this emergency mode can be entered with a minimal processor configuration that consists of a central control and only sufficient call/program stores to execute maintenance tasks. It may also be entered with a complete memory configuration in the event of peripheral faults or program problems that cause the loss of system sanity.

Manual procedures are also available to load new versions of generic programs or office data with minimal disruption to call processing. Most of this system-update procedure is time shared. It moves the data from tape to a single file-store copy. Once this copy has been fully updated, call processing is interrupted long enough to load the call/program stores from that file-store copy. The old data base remains on the mate file stores until it is overwritten manually from the updated copy. It is available for quick reload of the call/program stores if the system lacks sanity on the new data.

III. DIAGNOSIS

3.1 Overview

3.1.1 General description

The prime functions of the 1A Processor diagnostic programs are fault detection and generation of failure data used to locate faults. Diagnostics are developed using a high-level macro language and are table driven to facilitate multiple applications. The diagnostics are resident in the auxiliary unit (AU) community on disk and are paged into main memory when executed. They are specifically designed to run in a rela-

tively short elapsed time and to minimize the storage requirement. The table-driven/macro language design simplifies diagnostic design, test development, and debugging. It also simplifies modifications and fosters standardized documentation.

3.1.2 Diagnostic objectives

(*i*) Maximum fault detection—This objective concerns applying tests to a unit and having one or more of these tests fail if the unit is malfunctioning. However, economic constraints prevent detecting all faults.

(*ii*) Consistent test results—The diagnostics contain sufficient hardware initialization and test-output analysis to insure consistent test results for a given hard fault.

(*iii*) Protection of memory—The diagnostics are designed to minimize the possibility of destroying information stored in either main memory (call stores or program stores) or auxiliary memory (disk and tape).

(*iv*) System noninterference—The diagnostics are designed not to interfere with the normal operation of the system.

(*v*) Single replaceable module resolution—The objective is that test failures will allow resolution of a fault to one replaceable module (circuit pack). However, economic constraints prevent this resolution for all faults.

(*vi*) Program flexibility—The diagostics are designed so that various test options are available to maintenance personnel. The environment for testing a unit can be controlled by executing only part of the diagnostic, by removing or restoring other system resources, and by specifying other system units in the test configuration.

(*vii*) Efficient tests—Efforts were made to minimize the number of tests required to hold down the program size and execution time.

(*viii*) Program documentation—The diagnostics are designed with a high degree of standardized documentation to simplify program maintenance and to aid in the repair process.

3.2 Design approach

The table-driven/high-level-macro approach is used to design and develop the diagnostics. The diagnostic tests are a collection of macros that expand (when assembled) into a data table and drive (when interpreted) a control program that applies the tests to a particular unit. Section 3.3 explains the structure in more detail. The high-level-macro approach facilitates using the diagnostics as a common data base for several applications. By designing a macro-expansion package for a particular application, the data base is assembled to provide the driving

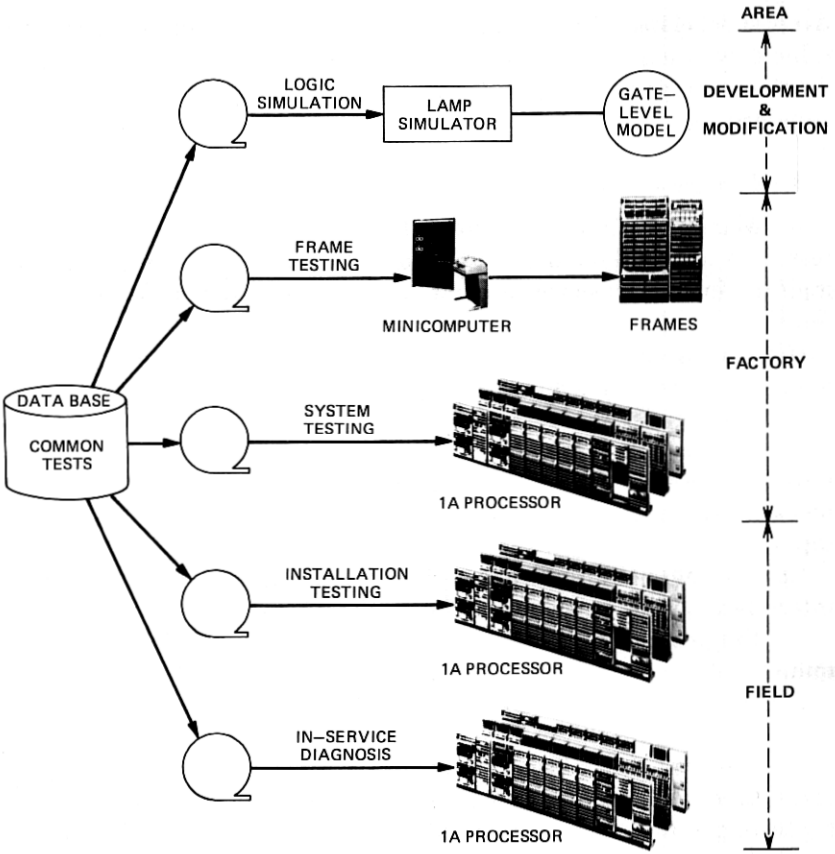

Fig. 2—Multiapplication of diagnostics.

inputs for the application. Figure 2 shows the current applications of the diagnostic programs. In the area of hardware and diagnostic development and modification, logic simulation plays an important role. The tests are assembled as inputs to a simulator called LAMP[7] and are applied by LAMP to a gate-level model of each unit. This application simulates the effects of the tests on the units and provides logic and diagnostic verification from the start of the design process to the completion of development. In the factory environment, the diagnostics are used for frame and factory system testing. For frame testing, the tests are assembled as inputs to a minicomputer which controls and drives a unit. This testing permits extensive circuit and diagnostic verification prior to interconnecting any of the units. For factory system testing, the processor units are interconnected and are driven by an installation test version of the diagnostics. In the field environment, the diagnostics are used for initial

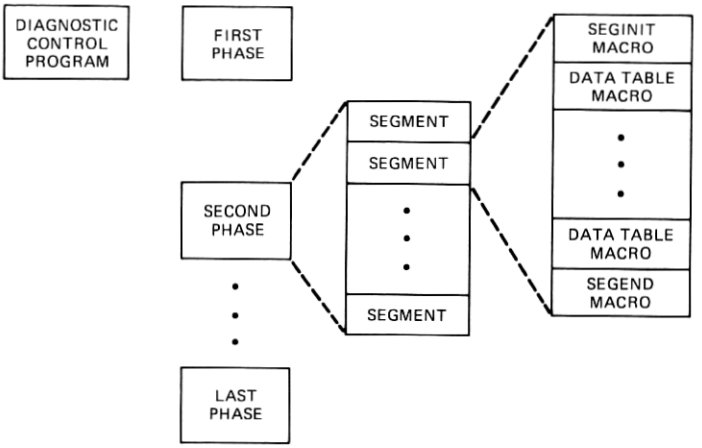

Fig. 3—General diagnostic structure.

installation testing and for the in-service diagnostic tool. The initial installation test consists of applying the same version of the diagnostics used for factory system testing to the completely interconnected processor in the field when it is first installed. The final application is the permanent in-service diagnostic, which is part of the generic maintenance software package.

Another important impact on the design process is that the diagnosticians and hardware designers work as a team, from the initial concept of the hardware design through the completed system. The hardware and diagnostic designs proceed in parallel, and each is used to verify the other. Diagnostic personnel have to agree to hardware changes made to a unit to insure the high level of fault detection required for overall system reliability.

3.3 Organization

3.3.1 General structure

The 1A Processor diagnostic is made up of individual unit diagnostics. Each unit diagnostic is a collection of many diagnostic phases and a control program. A diagnostic phase is a paged program module that is brought into main memory from auxiliary-unit memory when required for execution. Each phase is a collection of diagnostic segments and each of these segments is made up of data-table macros. Figure 3 illustrates this structure. The data-table macros expand when assembled into the data table that drives the diagnostic. These macros appear as a high-level language to initialize and test a portion of the unit being

diagnosed. Each diagnostic segment begins with a SEGINIT macro, which performs certain initializations required for the particular unit, and ends with a SEGEND macro, which performs a clean-up function and takes a real-time break. Each segment is designed to run in less than 2.5 ms when failing tests do not occur (less than 3.5 ms if all tests fail). In some special cases, additional real-time breaks must be taken inside a diagnostic segment to meet this design requirement.

3.3.2 Data structure

The macro language for the 1A Processor diagnostics is called DL/1 (Diagnostic Language/1). Diagnosticians specify sets of these DL/1 macros that perform (when implemented) basic read, write and associated test functions for the various units. A typical example of two DL/1 macros is:

CCWRITE WORD (address), DATA (data)
CCREAD WORD (address), EXPECT (data).

These two macros perform a simple test of the standby central control (CC) by writing a data pattern into an internal location and then reading the internal location and comparing the results of the read with the expected results.

Each DL/1 macro expands when assembled into an INDEX word and perhaps additional DATA words. The INDEX word contains the index field, which is a unique value associated with the particular macro type; the remainder of the word is used for data. A typical expansion for the two CC macros is:

Write address	CCWRITE index
Data to be written	
Read address	CCREAD index
Expected data	

INDEX WORDS

3.3.3 Control structure

Each unit diagnostic has a control program that is comprised of a small task dispenser and a set of task routines. The control program is table driven and uses the data structure described in the previous section. An interpreter for the data table is required to pass control to ESS assembly language routines that perform the required work. The interpreter is a program unit called a TASK DISPENSER. It has a pointer to the next INDEX word in the phase being executed. It fetches this word and transfers control to the TASK ROUTINE associated with the value of the index field in this word. The TASK ROUTINE is the ESS language

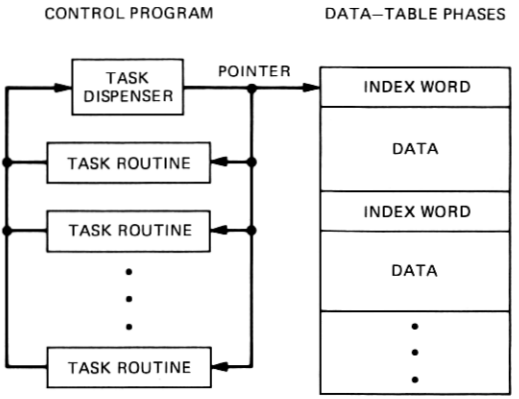

Fig. 4—Diagnostic table-driven structure.

routine that fetches any additional data words associated with the macro, performs the appropriate work, updates the pointer so it points to the next INDEX word, and returns control to the TASK DISPENSER. Figure 4 represents the entire table-driven diagnostic structure.

3.4 Test generation and execution

3.4.1 Test design

Test design is an iterative process guided by logic-simulation results and by experience gained from the various applications of the diagnostics, as described in Section 3.2. An objective of the diagnostic tests is to achieve complete coverage in the sense that the tests result in all logic nodes being exercised. However, a constraint is placed on this objective by economics.

Unit diagnostics are designed by diagnosticians who are familiar with the available DL/1 macros and with the hardware to be tested. Most tests are derived to exercise a part of the unit functionally. The remainder of the tests are designed using either automatic test-generation techniques or manually designed tests based on a gate-level logic diagram of the circuit. Great care is taken to test the input and output nodes of a unit before using them to test into the interior logic. Such a test-design sequence is feasible because of the maintenance involvement early in the hardware design, which is directed towards obtaining a sufficient degree of maintenance access for each of the units.

3.4.2 Test execution

Diagnostics are designed for straight-line execution. This design philosophy may be interpreted as running all tests and using a post-

processing scheme to determine the problem. To put it another way, no attempt is made to evaluate test results during diagnostic execution making branching decisions based on these results. However, the ability to perform simple forward branching is provided using data-table jump macros. The jump macro provides the capability to skip over tests that require other units that are not in service at the time they are needed or are not equipped in the system. Another reason for the jump macro is that a particular unit-type can exist in different systems at various hardware design levels. A diagnostic can handle a few of these design levels by running or skipping sets of tests sensitive to particular hardware design states. The final use of the jump macro is to terminate the diagnostic early when system integrity could be destroyed or when additional useful information cannot be gained by further execution. Care is taken when using the early-terminate function so that enough test data have been generated to solve the problem before terminating.

3.5 Interfaces

3.5.1 Diagnostic triggers

There are three ways to trigger a diagnostic. The first is automatic fault-recognition and recovery programs. During normal system execution, when a problem is detected, the fault-recognition and recovery programs request a diagnostic to be run on a unit to begin the repair process.

A second method is automatic routine exercise. Periodically, each unit is removed from service and diagnosed to detect latent faults.

Manual initiation is the third method. Manual diagnostics are initiated by either frame-control action or by teletypewriter (TTY) requests. Each unit has a frame-control switch. By changing the state of this switch, craft personnel can remove the unit from service, diagnose the unit, and restore the unit to service. This is especially convenient when repairing the unit. The TTY requests can perform the same functions. A unit is diagnosed when an input message to diagnose or restore a unit to service is received. An example is:

```
RMV:CC O! #remove CC O.
DGN:CC O! #diagnose CC O.
RST:CC O! #restore CC O.
```

If the diagnostic triggered by the RST input message executes without failing a test, the unit will be restored to service. When using the DGN input message, a special parameter, TLP (trouble location procedure) can be employed to trigger a post-processing system that evaluates the failing result of the diagnostic and generates an ordered list of suspect

replaceable modules to be changed one at a time. A detailed description of this capability can be found in Section IV.

3.5.2 Backup techniques

The first-line maintenance-repair procedure consists of executing the entire diagnostic for a unit and employing TLP to evaluate the resulting failure data and resolve the problem. However, if the fault in a unit is either not resolved or even not detected using this procedure, various backups can be employed. When a unit diagnostic is triggered by either of the first two methods discussed in Section 3.5.1, the automatic phases for the diagnostic are executed. Some units also have a set of phases called demand phases. These phases can only be executed by specifically requesting them via a manual TTY request. These tests are usually long-running exercise-type tests that are particularly helpful in locating intermittent faults and resolving access or memory faults in the primary or secondary storage devices.

When using a manual DGN TTY request, any subset of the automatic and demand phases can be selected to test the unit. Individual phases, groups of phases, or entire diagnostics can be run repetitively to establish consistency. The detailed failing results of these tests, referred to as raw diagnostic data, can be used to aid in the repair of the unit when the first-line maintenance-repair procedure fails to resolve the problem.

For even greater flexibility, an additional diagnostic tool called the exercise mode (EX) is available. EX is an input message verb like DGN, RMV, and RST that allows full and partial diagnostic runs along with repetitive execution of a phase or phases, and also permits probing inside a particular phase. Using this tool, one can step through a phase executing one or more segments at a time, advance to the end of a segment and stop, or can loop over one or more segments a specified number of times or indefinitely until manually terminated. When looping indefinitely over a set of tests, a SYNC pulse can be generated at a specified place allowing the circuit to be analyzed with an oscilloscope.

The DL/1 macros used have various self-documenting capabilities, such as producing a test number and specifying the address, data, and expected data. This effectively yields an in-line documentation that is valuable for resolving faults that elude TLP. Various documents are also available to assist maintenance personnel when evaluating raw diagnostic data.

IV. TROUBLE LOCATION

A standard trouble-location procedure (TLP) has been developed for the 1A Processor that encompasses an on-line TLP program and office-resident data bases. Programs were also developed to produce diagnostic

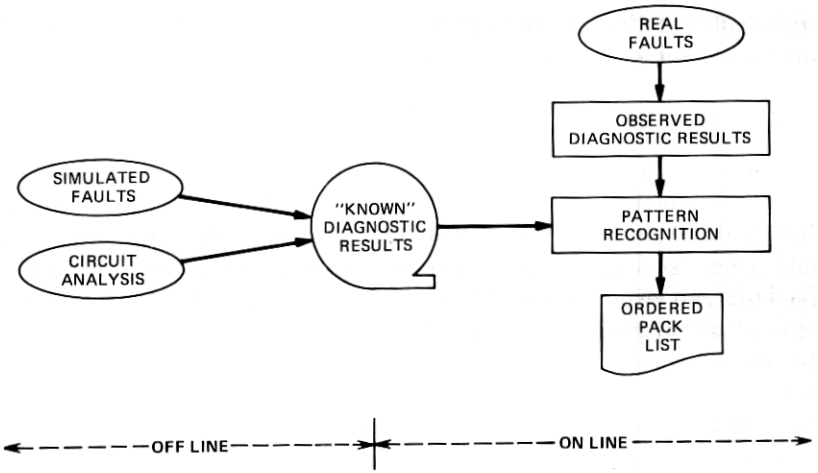

Fig. 5—Generation of pack list for trouble location.

failure results used by off-line programs to generate the resident TLP data bases. The primary output of the TLP program is an ordered list of suspected faulty equipment modules, which is referred to as the TLP "pack list."

When a fault is detected by the diagnostic program, the diagnostic results are passed to the on-line TLP program. The diagnostic results contain one data entry for each failing diagnostic test. Each entry contains the unique test number assigned to the diagnostic test and a 24-bit error word computed by exclusive ORing the expected test result with the observed test result.

Considerable processor resources would be needed to use the entire set of diagnostic results in an unprocessed form in subsequent TLP operations. Instead, each diagnostic result is summarized by extracting distinctive pattern features in the form of numerical quantities assembled into a signature. This signature is used to classify patterns obtained both off-line and on-line and to associate with each signature an intermediate circuit pack list. As shown in Fig. 5, off-line data is generated from circuit analysis or simulation of "classical" faults (i.e., idealized fault conditions such as logic nodes stuck at "0" or stuck at "1," or gate terminals opened or shorted to ground) using a physical or computer model.

The result observed on-line after diagnosing a real fault may not resemble simulated fault behavior; in fact, it may reflect the presence of an arbitrary failure condition including marginal circuit performance and multiple malfunctions. Primarily for this reason, a pattern recognition process is used to compare the signature of the observed result

with an office-resident data base of known signatures to obtain the closest matching patterns. The associated intermediate pack lists are then merged into one resulting list, which is ordered by the probability of implicating the fault. The remainder of this section will describe the three basic TLP components: feature extraction, on-line trouble location, and TLP data-base generation.

4.1 Feature extraction

Diagnostic failure patterns are summarized by extracting their significant features. Three methods of feature extraction are used depending upon the type of diagnostic analysis employed. These methods are: connectivity, first-test-failure (FTF), and pattern analysis (PA). With the connectivity approach, diagnostic failure patterns are analyzed by relating individual test failure results to an associated addressable point inside the circuit being tested. An intermediate pack list is derived by processing design file information and tracing all electrical connections emanating from these addressable points. Note that these intermediate pack lists are derived from the connectivity of the circuit; the behavior of the circuit is not used (and, hence, simulation is not required). Since the 1A Processor diagnostics use behavioral approaches (FTF and PA), the connectivity method will not be further discussed in this article. The connectivity approach is used extensively in the No. 4 ESS periphery and is described in detail in Ref. 5.

In the behavioral approaches, the selection of features is based to some extent on the way the tested circuitry is interconnected, but is principally based on the way in which the circuit behaves in the presence of faults. Choice of the behavioral approach is based on both the type of circuit to be tested and the ability to acquire the necessary data for pattern recognition. If it is feasible to simulate the behavior of a circuit under diagnosis in the presence of all classical faults, the FTF approach is used. If it is feasible to manually analyze entire failure patterns, the PA approach can be chosen. When neither behavioral approach is feasible, the connectivity approach can be employed.

4.1.1 First-test-failure approach

Most of the 1A Processor circuitry is embodied in sequential logic units. The fault behavior of these units can be analyzed using "complementary simulation" (Fig. 6). With this technique, faults can be simulated physically (in the system laboratory) and logically (on a general-purpose computer), and the results from both methods can be compared for reasonableness and accuracy. This dual method provides both a convenient method for validating the results and more extensive fault-simulation data than would normally be available if either process

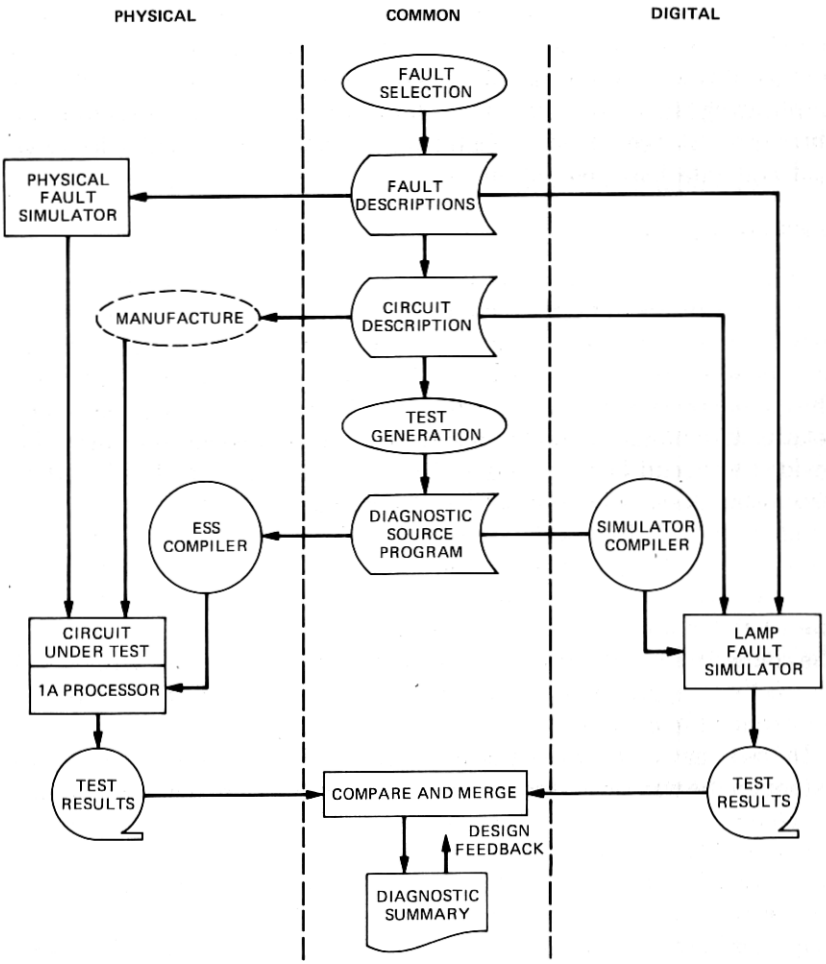

Fig. 6—Complementary fault-simulation system.

were used individually. Complementary fault simulation is discussed in Ref. 6.

ESS diagnostics are characteristically designed so that with each succeeding test there is a minimal dependence on circuitry not yet tested within the diagnostic. To capitalize on this characteristic, features extracted from the corresponding simulation results are heavily weighted by the earliest test failure results. The most significant feature is the first test failure; hence, the set of features derived from sequential logic simulation has been called the FTF signature. Additional features extracted from diagnostic failure patterns are: the first error word, the number of test failures in the first failing phase, the number of failing

phases, the total number of failing tests, and a scrambled number computed over the entire pattern.

Associated with each signature is an intermediate pack list, ordered according to the number of simulated faults on the pack that result in the associated signature. The bulk of the behavioral data base is composed of FTF signatures and associated pack lists.

4.1.2 Pattern analysis (PA) approach

Memory media and large regular circuit arrays require numerous diagnostic tests and, hence, simulation of all classical failure modes is frequently impractical. Fortunately, the location of a fault in such circuitry can be deduced from analysis of the overall pattern of failures rather than from early failures. This technique is termed pattern analysis (PA). To make the best use of the PA technique for call store, program store, and file store, the associated diagnostics employ exhaustive location-by-location test sequences at various readout thresholds and "worst-case" test patterns.

Frequently, memory testing will not produce exactly the same failure results from one run to the next for the same fault. However, there are invariant pattern features, and these are extracted for PA signatures. Some examples are: an unusually high error count for certain address ranges, an unusually high error count for specific portions of the data word, the confinement of all errors to a specific memory module or sector, or an unusually high error count associated with a particular memory cell transition state. It has been found that approximately twenty to thirty separate features, depending on the type of memory, can be used to classify such failure patterns using a PA signature. Since the required PA data base is relatively small (several hundred signatures) and the circuitry is quite regular, the intermediate pack list associated with each signature in this data base can be manually derived by the diagnostician. The PA data base is verified and augmented by sample fault simulation.

4.2 On-line trouble location

The on-line TLP program is a five-step process which (*i*) collects the diagnostic results, (*ii*) summarizes the diagnostic results into one or more signatures and places them on a disk holding queue called the TLPQUEUE, (*iii*) locates the desired data base on the TLP tape, (*iv*) determines the sequence of closest signatures (according to "weighted distance") and merges the associated intermediate pack lists, and (*v*) prints the final pack list. A weighted distance measure is used to take into account the relative significance (or weight) of each signature pa-

rameter. The TLP program requires a significant amount of processing time and storage, but due to the inherent reliability of the 1A Processor, the TLP program is infrequently executed. These considerations dictate that the TLP program be a disk-resident (paged) program that executes as a segmented maintenance client. To efficiently utilize the maintenance resources of the 1A Processor, the TLP program stores a summary of the diagnostic results in the TLPQUEUE and releases these resources while the TLP tape is being positioned to the desired section of the data base. Once the positioning is complete, the maintenance resources are again used by the TLP program to complete the task of producing the pack list.

The process of matching behavioral signatures with like entries in the FTF or PA data base is one of pattern recognition. The process finds an optimum match between the signature of the observed result and one or more entries in the appropriate data base by computing weighted distance functions. The weights account for the relative significance of each pattern feature quantized in the FTF or PA signature. In nearly all cases, several intermediate pack lists are generated even if there is an exact match. Near matches are considered if they are within a predetermined distance. A system of weighting functions is applied to each of the intermediate pack lists and a composite list is produced based on these weights. Weights are applied so that lists referencing the best matching signature will move to the top. Confidence factors (numbered 0 through 10) are computed to indicate to maintenance personnel the degree of match (10 represents an exact match).

Several 1A Processor diagnostics (e.g., call store, program store, and file store) occasionally require the simultaneous use of both feature extraction processes, FTF and PA. For these cases, the final pack list is produced by combining component pack lists, which are produced by the two signature types according to a merging algorithm.

Magnetic tape was chosen as the storage medium for the office resident data bases instead of the other available memory systems because of the large size of the data bases and the low frequency of access. The 1A Processor data bases alone contain over 2.5 million bytes. Data bases of comparable magnitude are also required for application maintenance. The frequency of access is expected to be less than three times a day in a stable office for both processor and application maintenance.

In addition to low cost, magnetic tape has other advantages: costly printed trouble-location manuals are eliminated, the process of updating data bases in the field can be controlled and is accurate, and updating does not have to be linked to reissue of the generic program.

However, the use of tape does exact a price. Increased access time is required to locate the desired data base on the tape. Additional TLP program complications occur in positioning the tape, and administration

of the TLPQUEUE is needed to hold the summary during repositioning.

Because the 1A Processor tape units move relatively slowly (800 bits/inch at 25 inches/second), several minutes may be needed to position the data-base tape; therefore, several entries may exist in the TLPQUEUE at one time. I/O messages are provided to allow maintenance personnel to examine and modify the contents of the TLPQUEUE and/or the activity of the TLP program. When the TLPQUEUE is full, diagnostic results from subsequent diagnostics cannot be immediately used by the TLP program. However, the diagnostic results are processed to the extent that a summary of the TLP data is printed for later use. Data are automatically removed from the TLPQUEUE when the associated pack list is printed.

When diagnostic results are not entered in the TLPQUEUE, maintenance personnel can simply rerun the diagnostic when there is space in the TLPQUEUE. However, if the fault is marginally detected (i.e., the diagnostic may not always fail the second or third time), special input messages are provided so that maintenance personnel can manually reconstruct the TLPQUEUE entry from the earlier terminal printout of the TLP summary data. This feature can also be used to remotely generate pack lists for other offices in emergency situations.

The on-line nature of the TLP program also has several advantages: it supplies a common interface for each TLP approach; it allows for more complex approaches, which result in better pack lists and therefore faster frame repairs; and it allows for future modification of and addition to existing approaches without the necessity of retraining maintenance personnel.

The TLP program structure permits additional trouble-location approaches to be easily added. Frame-dependent interface programs control the overall TLP process by using subroutines supplied by the TLP program. This flexible control is used to dynamically select the TLP approach to be applied based on individual frame requirements.

4.3 TLP data base generation

Figure 7 shows the major data flows required to generate the TLP data base used by the 1A Processor. There are three data base components: FTF, PA, and connectivity. Each component consists of a data base composed of signatures and associated intermediate pack lists. The connectivity pack lists are generated by automatically tracing the interconnections among circuit components by processing design-file information.[5]

Pattern analysis (PA) pack lists are usually derived by manual analysis of the behavior of the unit; however, sample fault-simulation results are used to verify and augment the PA data base.

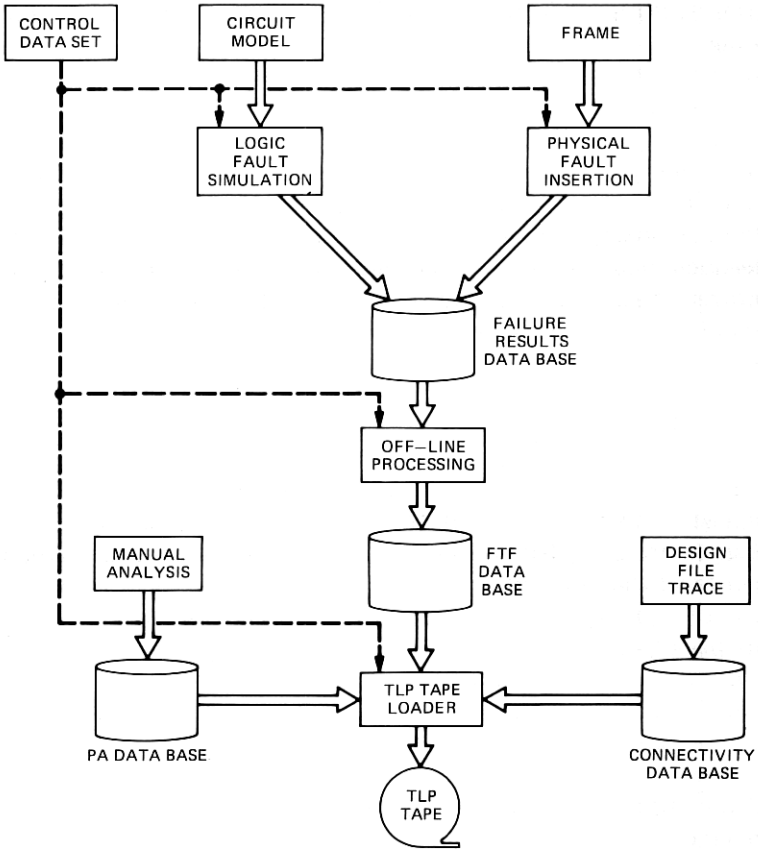

Fig. 7—TLP data-base generation.

As stated earlier, most of the TLP data base for 1A Processor units consists of FTF signatures and pack lists derived from complementary fault simulation. The physical simulation results are primarily obtained by inserting classical faults at circuit pack terminal-pin connections. Classical faults interior to the circuit pack and faults analogous to the above physical terminal-pin faults are simulated using LAMP.[7] In contrast to the universe of "real" faults, the simulated fault set does not include open power buses, open ground buses, marginal logic levels, marginal gate delays, crosses between signal leads, etc.

A basic assumption in the application of the FTF behavioral trouble-location technique is that the universe of all physically possible FTF signatures will be sufficiently covered by dealing only with classical faults. This assumption has been born out by the success, to date, of TLP performance (see Section VI).

Faults for both physical and LAMP fault simulation were directly derived from certain equipment files in the data management system (DMS). These files describe the interconnections within circuit packs and circuit-pack interconnections within an ESS unit. Fault collapsing (choosing one fault to represent a group of logically equivalent faults) is used extensively to minimize the number of faults handled.

The initial diagnostic results produced from physical and LAMP fault simulation reflected an ideal diagnostic environment in the ESS system. Quite often, however, diagnostics are run under less than ideal conditions; e.g., bus access to the unit under test might be limited, or other units associated with the unit under test might not be available to participate in the diagnosis. Under these conditions, diagnostic tests are sometimes skipped and the FTF behavioral trouble-location process suffers. Compensation for these effects is usually achieved by computing a "family" of signatures for each fault. One signature represents the ideal situation, and the others represent various off-normal equipment-availability conditions.

All stages of TLP off-line processing (Fig. 6) are controlled by software data sets, which serve not only to control the processing but also to document how the data bases are generated. The majority of the intermediate data bases are automatically generated from circuit-description files. This automation, coupled with other administrative facilities, permits the orderly updating of the TLP data bases and the TLP tape.

V. ERROR ANALYSIS

5.1 Objectives and uses

The fault-recognition, diagnosis, and trouble-location programs are the primary components of the maintenance software system, which is provided to the field to help isolate and identify faulty modules. However, even with high-quality maintenance-software design and thorough debugging, there is a need for backup procedures that can be applied to system troubles that elude the normal maintenance defenses. Trouble-isolation difficulties frequently occur as a result of transient, intermittent, or marginal malfunctions. These are usually classified as system "errors" rather than "faults." The resolution of these problems typically involves using past occurrences to extract patterns that may implicate a particular subsystem or component. Such analyses can be time consuming and often require special consultation with off-site maintenance experts.

The 1A Processor error analysis program was conceived as a backup problem-solving tool based on the use of a readily accessible system trouble history. This program, referred to as ERAP (for error analysis processor), has been implemented as an on-line storage and retrieval

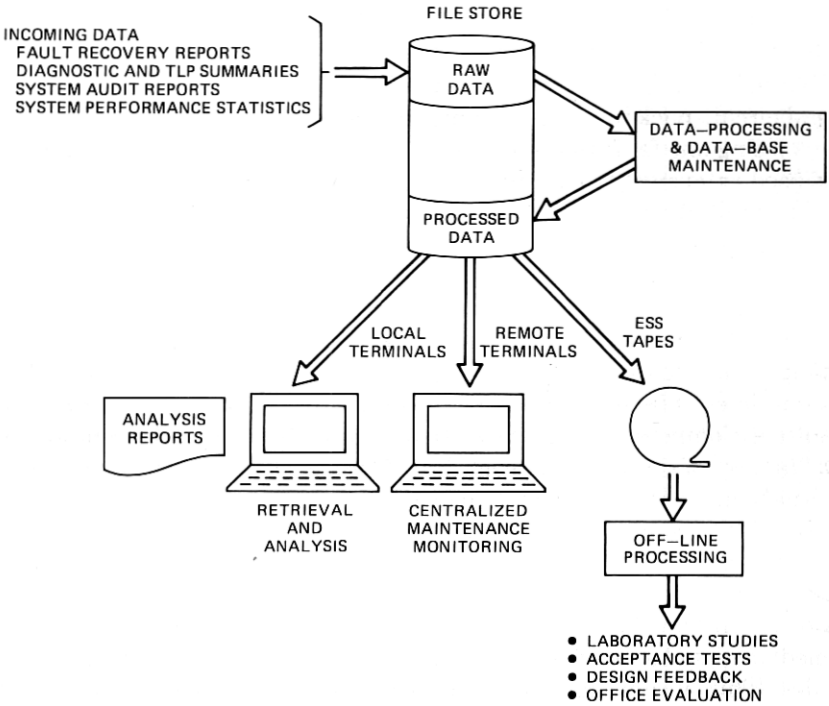

Fig. 8—Error-analysis system.

system. It automatically collects and formats appropriate trouble-history data, stores the data records in a dedicated area of file store, maintains the data base with its own audit program, and provides extensive retrieval capabilities via TTY input/output messages. In the design of the ERAP program, provisions were made to take advantage of the magnetic tape storage facilities of the 1A Processor. Thus, the entire ERAP data base may be transferred to tape at any time for subsequent off-line processing, and also data preserved on tape may be read back into file store and retrieved on-line.

An overview of what is effectively an error analysis system is depicted in Fig. 8. The incoming data accepted by the program consists of various types of records which fall into four main categories: fault recovery reports associated with system maintenance actions taken in response to detected malfunctions; diagnostic and TLP summaries; certain system audit reports; and overall system performance statistics. Incoming data records are initially buffered in main memory and then transferred to a file-store buffer area which holds the raw data until it can be processed. The data processing involves putting the records into standard formats suitable for subsequent retrieval, adding the formatted records to the

permanent file-store data area, and updating directory tables and link lists. The data-processing routines are executed as clients of the maintenance-control program, MACP, and are paged in from file store when needed.

The data base maintenance and retrieval routines also operate in a paged program mode. Data retrieval is carried out in response to input TTY messages with displays on office-maintenance channels or on dial-up terminals located off-site in a centralized maintenance facility. The retrieval capabilities include pattern searching and data-scanning features that serve as manual analysis aids. Automated analysis procedures are not provided in the ERAP program. Instead, companion programs are available to process on-line-generated magnetic tapes on a commercial computer. In addition, advantage can be taken of the 1A Processor library-program feature to perform on-site reporting and data-analysis functions. The library program approach offers flexibility in introducing various analysis procedures, which may be run relatively infrequently, without requiring generic-program changes and also without occupying dedicated program space on file store.

The off-line processing capabilities considerably extend the scope of the original error-analysis design concept. The processing of error-analysis tapes is used in special laboratory studies, such as fault-insertion projects to evaluate recovery software, and in acceptance tests of new generics. Data sent back from the field are used to provide hardware and software design-feedback information to help evaluate office performance (particularly near the time of cutover) and to aid in the development of analysis procedures.

5.2 Data base formation and maintenance

The remainder of this section is devoted to a description of the on-line ERAP program, beginning with specifics on the nature of the file-store data base and the methods used to maintain that data base.

The fault-recovery records collected by the program consist of reports of each maintenance interrupt, maintenance interject, and base-level maintenance action taken by the system. These reports, which are also printed on an office TTY when the action occurs, provide dumps of the contents of selected internal unit registers and also contain data passed on by the fault-recovery programs. Included in the category of fault-recovery reports are records of each system phase of memory reinitialization, whether automatically or manually generated, and also failure reports resulting from deferred fault-recognition tests.

The next category of data collected (see Fig. 8) consists of summaries of each diagnostic carried out indicating, for example, the source of the diagnostic request and the main diagnostic results. TLP summary records

provide further information on the diagnostic results, although such summary records are included in the data base principally for the purpose of off-line TLP evaluation studies. Diagnostic and TLP summary records, which arise as a result of fault-recovery actions, are associated with the recovery records by a link-listing scheme used to create error-analysis files. An error-analysis file consists of a collection of individual records that bear a common "maintenance file number." The error-analysis program maintains a file-number counter, which is passed on by a recovery program when it makes a diagnostic request and which is incremented each time a distinct recovery sequence is terminated. Records of manually-requested diagnostics and associated TLP summaries also result in error-analysis files that are separate from the recovery files.

Other types of data collected by the program, which are in the same category as TLP summary records, include optional detailed diagnostic results (raw diagnostic data), records of deletions from the TLP queue, and frame-repair records. A frame-repair record is manually input via the TTY to describe a TLP-based repair action involving the replacement of a particular pluggable module. These repair records are link-listed into the files containing the diagnostic and TLP summary records as well as any associated fault-recovery records.

The system-audit reports collected by ERAP summarize memory-mutilation errors found and corrective actions taken for all nontransient memory areas in main memory or file store. The resolution of memory-mutilation problems typically requires correlation of historical data and is a prime candidate for an error-analysis approach. The only other audit information collected by the program is that generated by ERAP itself when it corrects errors found in its data base.

The final category of data collected by ERAP is principally intended for off-line processing of error-analysis tapes for system-evaluation purposes. It includes overall performance statistics generated by traffic- and plant-measurement programs that are provided in the No. 1A and No. 4 ESS systems (see, as an example, Ref. 5). Also included are reports every half hour on units that are out of service for maintenance reasons.

The error-analysis program is designed to preserve its data base through severe system troubles, including phases of memory reinitialization and file-store memory mutilation. There is an extensive error-analysis audit program that is automatically requested after system phases and when ERAP program defensive-check failures are detected during normal execution. The audit program verifies the internal consistency of the reference tables (directories, file-number table, link lists, etc.) and also can compare the reference tables with the totality of records stored in the data base. Sufficient information is stored within each

record to rebuild the reference tables if the consistency checks fail. Furthermore, each record has stored within it a check sum (formed when the record was originally constructed), which reliably indicates any unintentional alteration of the record. Mutilated records are automatically discarded from the data base, which is subsequently repacked. The audit program will use both file-store copies of its data base to find a correctable copy and will update the other copy automatically, as required. If both file-store copies are uncorrectable, the data base will be reinitialized. As mentioned earlier, all modifications in the data base by the audit program (including automatic and manual reinitializations) are themselves recorded in the data base for subsequent investigation.

Intentional deletion of records from the data base is accomplished manually by means of TTY input messages. (Automatic data-clearing operations are planned for future issues.) If the file-store data base overflows, data collection is halted. Warning messages are issued every half hour prior to overflow when the data base has less than 1/8 of its total assigned space available. Also, the current data-base occupancy may be determined at any time by means of error analysis TTY input messages.

5.3 Retrieval capabilities

Data output from the program is provided in response to error analysis TTY input messages that specify, via arguments of keywords, the type of data desired and the file numbers of the data of interest. In one retrieval input message any combination of subrecords, records, or entire files may be requested mnemonically by listing the appropriate arguments. Another form of retrieval takes place where specified files are scanned and particular words or items are extracted from named data blocks within the files. On output, the items are listed in matrix form with columns for the individual items and rows for the individual data blocks and files. This type of output is especially useful as an aid to manual recognition of data patterns involving sequences of interrupts or diagnostics. The program also has the capability to accept mnemonic specification of items to be retrieved so that specific layouts of data blocks need not be consulted.

Sample ERAP output of the form just described is depicted in Fig. 9. The input message shown in the figure requests that items named INS, SCA, SDA, SPA, CES, and SBYCES be extracted from all DLEV data blocks (associated with D-level interrupts) that may happen to be present in the file number range 1 through 77. The output message given in response to this request indicates that five files were found to contain DLEV-type data, and the desired items are displayed (in octal) along

```
INPUT MESSAGE REQUEST FOR DATA OUTPUT

   OP: ERAPDATA  DLEV: ITEM (INS, SCA, SDA, SPA, CES, SBYCES). MFNUM 1–77! PF

OUTPUT MESSAGE RESPONSE

M 28 OP:ERAPDATA                COMPLETED
       NUM FILES = 5            MFNUM RANGE: 00000015 THROUGH 00000056
       DATA TYPES : DLEV
        MFNUM      INS          SCA        SDA        SPA        CES      SBYCES
       00000056  00002003  14652043  07753410  14652046  00504040  02001500
       00000027  00002100  14430015  00004000  14204050  01003100  02004040
       00000024  00002100  14430015  00004000  14204050  01003100  02004040
       00000022  00002100  14430015  00004000  14204050  01003100  02004040
       00000015  00002003  14654760  07752540  14654762  00500310  02004140
         04/15/76  10:28:17
```

Fig. 9—Sample ERAP output.

with the file numbers. The similarity of three of the D-levels is immediately apparent from this display.

One of the most powerful features of the error-analysis program is the capability of associative retrieval implemented by means of searches through the data base according to user-selected search conditions, which are input via TTY. A search condition imposes an arithmetic restriction on an individual item of a record or on the relationship between two different items from records within the same file. An example of the use of a search condition is the specification that a fault-recovery-requested diagnostic on a particular frame passed all tests, possibly indicating a transient error. Another example is that a memory-access failure occurred at a particular address of interest. When a search condition is entered via an input message, the translated condition is stored in the error-analysis data base where it remains for future reference until it is manually cleared by a special input message.

An actual search through the data base is requested by a TTY input message, which names the search conditions to be used. Two or more conditions appearing in the same input message will be logically ANDed. The result of a search is a memory block indicating the file numbers of those files which meet the input conditions. The search results are themselves stored in file store for subsequent reference by input messages used to output data or to perform additional searches. In this way, it is possible to output from files previously found in a search without having to enter the individual file numbers, and it is also possible to perform searches among existing search results with additional conditions imposed.

VI. EVALUATION

A definitive evaluation of 1A Processor maintenance objectives cannot be made until enough systems have accumulated sufficient operating experience in representative service environments, although several

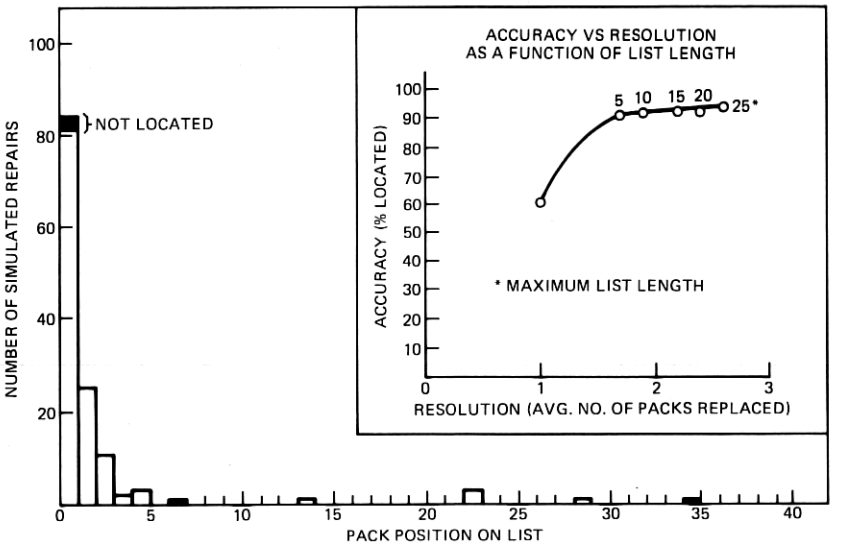

Fig. 10—TLP performance on 132 simulated repairs using field returns.

studies have been conducted to evaluate fault recognition and recovery, diagnostic detection, and trouble location to provide early estimates of 1A Processor dependability.

In one study, 2071 single faults were inserted by connecting randomly selected backplane points to ground using relays while the system laboratory 1A Processor was in a normal operating mode. Automatic system recovery occurred in 99.8 percent of the cases; manual assistance was required for only five of the faults. In another study using a random selection of 2400 classical faults, simulation results indicated that 95 percent of the faults were detectable by the diagnostic programs.

To evaluate the TLP process, circuit packs that had failed while in field or laboratory service or during installation were reinserted in one or more laboratory equipment locations. After each circuit pack insertion, the associated unit was diagnosed and the trouble-locating process initiated. In only five of the 132 simulated repair cases did the TLP fail to include the inserted pack in the resulting list. Figure 10 shows the distribution of simulated repairs as a function of pack position on the list; the maximum list length was used in the five cases where the inserted pack was not listed. Notice that only seven cases (5.3 percent) would involve more than five pack replacements to locate the failed pack or exhaust the list.

It is of interest to consider the effect of truncating the pack list at lengths of 5, 10, 15, 20, and 25 packs. Figure 10 depicts the resulting accuracy (percent of failures located) versus resolution (average number

of pack replacements). In this experiment, a reduction of maximum list length from 25 to 5 would degrade TLP accuracy from 96.2 to 92.4 percent while improving resolution by one pack replacement per repair (2.7 to 1.7).

VII. SUMMARY

To achieve the rather stringent system-dependability objectives of the 1A Processor, four maintenance-software subsystems were designed: fault recognition and recovery, diagnostics, trouble location, and error analysis. Fault recognition has been enhanced by more extensive use of self-checking circuitry than that used in the earlier systems, such as No. 1 ESS. System recovery, while more complex because of the writable program store and dependance upon the rather complex file store, has been improved by increased use of short-term error analysis, additional bootstrap implementations, and the consolidation of control of deferred configuration requests. Extensive use of physical fault insertion in system laboratories has verified that, with very high probability, the 1A Processor can recover from faults without manual assistance.

Diagnostics have received considerable emphasis during maintenance-software development. Extensive physical and computer simulation, which began in the early stages of subsystem development and continued throughout the development, resulted in unit tests capable of detecting at least 95 percent of the classical faults. Using a macro-level test-specification language, a high degree of standardization was achieved. Essentially the same set of diagnostic tests is used to exercise a unit at the frame level, verify it at the factory, and diagnose it during normal 1A Processor operation.

Trouble location has been automated to the point where failure results from a unit diagnostic are translated to an ordered pack list, with the particular combination of distinct failure-analysis procedures used transparent to the maintenance personnel. Early data from in-service failure indicate that the objective of at least 90 percent of 1A Processor repairs requiring three pack replacements or less is being met.

Problems that elude diagnostic detection or the normal trouble-locating process usually must be resolved by analyzing error symptoms. Extensive data-collecting-and-processing software has been provided in the 1A Processor to assist such error analysis. Much of this error-analysis capability is used in the course of normal system maintenance; substantial amounts of the data are being saved for off-line performance and design evaluation.

VIII. ACKNOWLEDGMENTS

The work described in this article could not have been accomplished without the combined efforts of the numerous system designers who have

been involved in maintenance software. The authors particularly wish to acknowledge the significant contribution of W. R. Hudgins and D. E. Lutz in the development of the TLP feature and in the preparation of this article.

REFERENCES

1. R. E. Staehler and R. J. Watters, "1A Processor—An Ultra-Dependable Common Control," International Switching Symposium, Institute of Electrical Communication Engineers of Japan, Kyota, Japan, 1976 Symposium Record, pp. 636–642.
2. G. F. Clement and R. D. Royer, "Recovery from Faults in the No. 1A Processor," Proceedings of the 4th Annual International Symposium on Fault Tolerant Computing, June 1972, pp. 5-2—5-7.
3. A. H. Budlong et al., "1A Processor: Control System," B.S.T.J., this issue, pp. 135–179.
4. C. F. Ault et al., "1A Processor: Memory Systems," B.S.T.J., this issue, pp. 181–205.
5. "No. 4 Electronic Switching System," B.S.T.J., (special issue), *56,* 1977.
6. F. M. Goetz "Complementary Fault Simulation," Proceedings of 3rd Annual Texas Conference on Computing Systems, University of Texas, Austin, IEEE Computer Society (November 1974), pp. 9.4.1–9.4.6.
7. "LAMP: Logic Analyzer for Maintenance Planning," B.S.T.J., *53,* No. 8 (October 1974), pp. 1431–1555.