AT&T Technical Journal
Vol 64, No. 1, January 1985
Printed in U.S.A.

System 75:

Maintenance Architecture

By K. S. LU,* J. D. PRICE,† and T. L. SMITH*

(Manuscript received July 11, 1984)

Reliable service has been a cornerstone of customer premises communications systems for years. System 75 office communications system hardware and software have been designed to continue that high degree of reliability and availability. The hardware has been designed to detect and correct errors as they occur, to minimize the number of components that cause system outage, and to simplify fault isolation to a replaceable component. The software has been designed to recover from intermittent failures and to continue providing service with a minimum of disruption. These features have been implemented in the software as a group of processes running under a real-time operating system, which simplifies building and testing the software and makes it easy to extend its functions.

I. INTRODUCTION

Reliable service is one of the most important features of a customer communication system. Aspects include reliable hardware design, automatic system reconfiguration in the event of a hardware fault, defensive programming to minimize the impact of intermittent hardware failures or obscure program bugs, and a repair strategy that quickly corrects any hardware problems in the system. System 75 is

* AT&T Information Systems Laboratories, an entity of AT&T Information Systems, Inc. † AT&T Consumer Products.

Copyright © 1985 AT&T. Photo reproduction for noncommercial use is permitted without payment of royalty provided that each reproduction is done without alteration and that the Journal reference and copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free by computer-based and other information-service systems without further permission. Permission to reproduce or republish any other portion of this paper must be obtained from the Editor.

based on previously proved designs[1-3] to ensure reliable service. It includes some new features as well. Major enhancements include a CRT-terminal-based, human-engineered interface to simplify repair activities; a maintenance system that provides systematic organization and flexibility to system maintenance activities; and features to improve the reliability of the communications network by maintaining connected equipment such as stations, terminals, and trunks.

This paper describes various aspects of system maintenance including objectives, the plans for hardware and software maintenance, the maintenance software architecture, a discussion of how the system handles critical failures, the human interface to the maintenance features, and the remote maintenance capabilities.

II. DESIGN OBJECTIVES

A clear set of objectives was required to build a consistent and integrated set of maintenance features in a timely way. Some of these were external objectives, observable by people who use or repair the system:

1. The system must be highly reliable. Any component failure should affect the smallest possible piece of the system, and the time between system outages caused by a single component failure should be several years. Automatic recovery from hardware and software problems should be achieved as well.

2. The system, not the users, should detect failures. The system should be fully self-testing with tests run frequently enough to detect failures before the user. False alarms should be avoided; alarm only those faults requiring repair.

3. The system must be simple and quick to repair. Trouble analysis, testing, and direction of the repair activity should be provided by the system, not the repair person. The system should have an interface that can be used with little training, permitting the customer to participate in maintenance.

4. The scope of the maintenance should extend to the entire customer communication system. The system should help maintain stations, terminals, trunks, and other connected equipment.

In addition to the external objectives, there are several objectives which, though not directly visible, impact maintenance performance:

1. The system should have a common set of tests for each piece of the system. By using the same test for system fault diagnosis, periodic testing, and technician testing, test results are reproducible and the possibility for confusion is reduced.

2. The maintenance system should have the flexibility to deal with a wide diversity of objects. By using a maintenance operating system

approach, maintenance of each piece of the system can be written independently while using common structure and functions.

3. The system should be easy to extend. As the features and functions of the system increase, the maintenance must also expand with minimal revisions to existing features.

4. The system should have minimal maintenance interactions between parts of the system. This makes it easier to identify the part of the system that is failing and permits inherently simpler maintenance strategies to be developed.

III. MAINTENANCE OBJECTS

Maintenance programs for System 75 run under the Oryx/Pecos operating system.[4] (See Fig. 1.) The collection of processes that perform the maintenance activities is called the maintenance subsystem and is written as a maintenance operating system. Each part of the system to be maintained is called a maintenance object and is handled independently by the maintenance operating system. Maintenance objects are selected to be independent from each other, reducing the number of interactions inside the maintenance system and simplifying the maintenance strategies. Each object in the system has its own maintenance strategy. This strategy can include tests for detecting

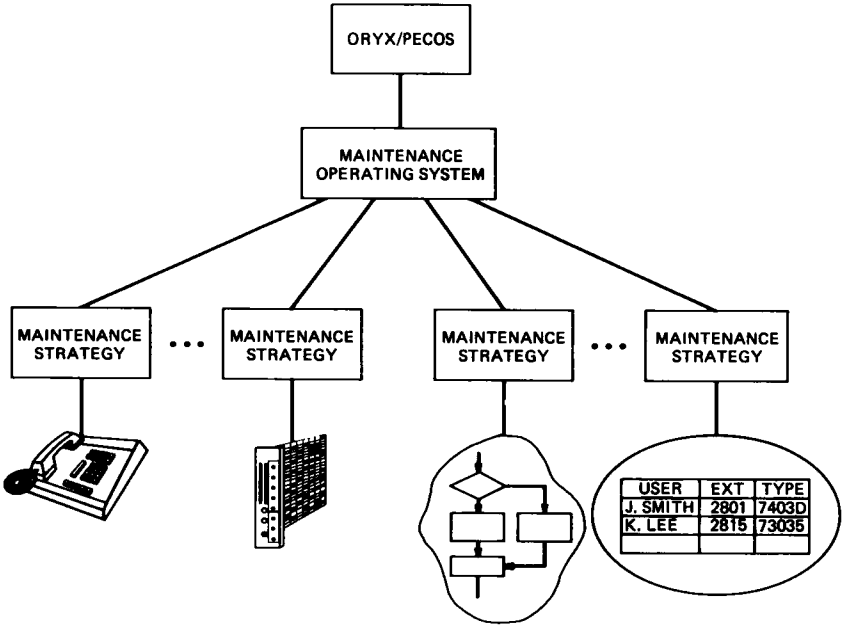

Fig. 1—Maintenance object organization.

and diagnosing problems, and recovery, reconfiguration, and repair activities. The maintenance operating system provides common functions such as scheduling, alarming and terminal interfaces.

There are three general categories of maintenance objects. The first category is hardware, including circuit packs, stations, and trunks. In general, each circuit pack or each separately replaceable unit is a separate maintenance object, but there are exceptions where several circuit packs belong to one maintenance object or where one circuit pack contains several maintenance objects. Hardware maintenance objects can be tested, alarmed, and removed from service. When a problem has been isolated to a hardware maintenance object, the object is replaced.

The second category of maintenance objects is software processes. Each process running under the Oryx/Pecos operating system is a separate maintenance object. If a process encounters trouble, it can be recovered or restarted.

The third category of maintenance objects is data relationships. Each group of data that has some internal structure or redundancy can be maintained. Data relationships can be audited and corrected.

IV. HARDWARE MAINTENANCE

4.1 Principles

4.1.1 Scope

Hardware maintenance covers all the equipment inside the cabinet including all the circuit packs (the processor complex, trunks, lines, and service circuits), the tape drive, the power converters, and cabinet-level functions such as power supplies, batteries, and environment (see Section 4.4.4). In addition the system maintains attached lines and trunks.

4.1.2 Error detection

Whenever possible, the system will detect and report errors automatically. Error detection is done through a variety of methods, with the particular method determined by weighing the severity of the problem against the cost of the method. The following methods of error detection are used:

1. Error detection hardware is added to the system, monitoring the operation and detecting errors immediately. This is cost-effective in only the most critical applications. An example is the error detection and correction circuitry on the processor memory. When a faulty memory location is accessed, the failure is detected immediately.

2. In-line errors are detected using software or firmware. Each time a circuit performs its function, additional software checks are run to ensure that the circuit is functioning correctly. This is useful in

important applications where quick detection of errors is worth the cost of processor real-time consumption. The checksum on control channel messages is an example.

3. Periodic tests are the most widespread testing strategy. Tests that run quickly or detect important troubles are run frequently, typically once an hour; a comprehensive set of tests is run once every 24 hours.

4. A few error-detection tests are potentially so service disruptive that they are not run on a periodic basis. A comprehensive memory test falls into this category. As long as memory is working correctly, no comprehensive test is run. However, if the system has experienced several crashes, additional time will be taken on the reboot to run a complete memory test.

4.1.3 System alarms

If a maintenance object consistently fails the error-detection tests, the system will automatically generate an alarm. This is a call for technician action to repair the system and restore it to a normal condition. There are three levels of alarms.

1. Warning alarms for failures that cause no noticeable degradation in the customer service. Also included are failures whose cause might be external to System 75 but need no immediate action.

2. Minor alarms for failures that cause marginal degradation of customer service while not rendering a crucial portion of the system inoperable. This condition requires action, but its consequences are not of a global or immediate nature. Problems might be impairing service to a few trunks or stations or causing problems to only one feature across the entire system.

3. Major alarms for failures that cause critical degradation of customer service and require immediate action. Processor faults and failure of an entire trunk group fall in this category.

Alarms are made visible in several different ways.

1. If the failing component is a circuit pack, a red lamp is lit on that circuit pack. This guides the technician in replacing the faulty circuit pack.

2. If the system detects any major or minor alarm conditions, they will be reported to the Operations Support Center computer.[5]* The system will automatically place a call and report the problem (see Section IX).

3. The system alarm status is displayed in lamps on the maintenance circuit pack alarm panel. A second lamp indicates the status of

---

* A typical centralized Operations Support Center computer system is RMATS (Remote Maintenance Administration and Traffic System).

attempts to notify the Operations Support Center computer of the problem.

4. If the system has any major or minor alarms, an alarm lamp on the attendant consoles will be lit.

5. A list of all alarms currently in the system is kept in an internal alarm log. This can be displayed on command from a system administration terminal (see Section VIII) or can be requested through the remote computer interface.

In addition to automatically raising alarms, the system will also automatically retire alarms. Sometimes a problem that is causing an alarm will disappear without human intervention. Since testing continues on an alarmed maintenance object, tests will begin to pass after the problem disappears, causing the alarm to be retired. This strategy eliminates needless repair activity to retire alarms for temporary faults. In addition, no technician activity is required to retire an alarm after repair.

The maintenance strategy is to eliminate intermittent alarms and to alarm only consistent, reproducible faults. This is done by requiring an error to be seen several times before it generates an alarm. Similarly, once a maintenance object is alarmed, it must pass its tests several times before the alarm is retired.

To generate alarms in a reasonably short interval of time, the system uses intensive testing. Once an error is detected on a maintenance object, that object goes into a testing mode where tests are run much more frequently. Either these tests will fail several times in a row and quickly generate an alarm on the maintenance object, or they will pass and remove the maintenance object from further suspicion. Intensive testing continues as long as any errors are being detected on a maintenance object, whether it is alarmed or not. This testing is done at low priority to avoid interfering with normal system operation.

All alarms lead directly to a specific repair activity such as replacing a circuit pack, changing a power supply, or repairing a station set. Such repair activities have a good chance of fixing the problem and retiring the alarm automatically.

4.1.4 System recovery and reconfiguration

The major recovery and reconfiguration strategy is to take selected pieces of equipment out of service (maintenance busy) to preserve good service for the remaining equipment. Faulty trunks and service circuits are removed from service, but care is taken to avoid disabling all the trunks in a trunk group or all the service circuits. Stations are not removed from service except when they have totally failed and cannot provide any service or when the failure mode may disrupt other parts of the system. If any failure prevents the system from processing

calls, emergency transfer stations are connected directly to trunks. Loss of ac power, continuous rebooting, and complete loss of the time division bus are examples of conditions that invoke emergency transfer.

All systems are provided with battery backup for the entire system. If the power outage is short (up to 10 seconds) the batteries will support the entire system. If the power outage is longer than 10 seconds, the system is put into a standby mode where battery power is supplied only to the control complex. All calls are dropped and emergency transfer is invoked. The batteries can power the memory for an additional ten minutes. This shortens the time required to bring the system back to an operational state since it takes several minutes to reload the memory from the backup tape.

4.1.5 System repair

The particular type of repair activity used depends upon which type of maintenance object has failed. Failures in circuit packs connected to the time division bus, including station, trunk, and service circuit packs, are easiest to repair. They can be replaced with minimum impact on the system since it is not necessary to turn off power. The faulty board, indicated by a red LED, is just unplugged and replaced; only those calls using the board are affected. Two additional LEDs are used to guide this repair activity. A yellow LED shows the pack is in use and a green LED shows the pack is under test.

Repair of control circuit packs requires system power down, board replacement, and system reboot. Station repair activities include locating the trouble, wiring repair, and station replacement. Trunk repair activities include verifying the trouble, fixing trunk wiring, and reporting troubles on connecting switching systems.

4.2 Processor complex

The critical boards required to provide system service are the processor, memory, and interface to the switching network.[6]

4.2.1 Processor board

A sanity or watchdog timer, which must be reset periodically, monitors major failures of the processor. If the software fails to attend to this function, the processor is reset and the maintenance circuit pack, an independent processor, is notified. Major system outages are covered in Section 6.2.

Several functions of the processor adjunct circuits are periodically tested, including software interrupts (such as divide by zero), memory management operations, privileged instruction detection, bus time-out exceptions, and initialization and bootstrap ROM checksums.

4.2.2 Memory

The memory has full multibit error detection and single-bit error correction circuitry. It can be reconfigured to run even if every memory access detects a single-bit error. The advantage of this feature is that single failures in the memory do not result in any increase in system outage.

Full, destructive memory tests are run on system initialization (before reboot), including a horizontal and vertical partition test and stuck-at-zero/stuck-at-one test. The following tests are run during system operation:

1. Read all memory test. Read every word in memory to correct soft errors and to force an error report.
2. RAM checksum test. Check that the text segments of all processes have not been modified.
3. Memory error correction test. Check that the error correction and detection circuitry is working and generating the correct failure reports.

If several uncorrectable errors are detected, the system initiates a reboot, memory is fully tested, and the corruption is corrected.

4.2.3 Network control

The network control circuit pack includes both an interface to supervise the port boards and four interfaces for data communication with peripheral devices. These are referred to as the control channel and data channels, respectively.

The control channel is monitored using several in-line error-detection mechanisms, since it is a critical link in switch operations, with all messages to and from the port boards flowing over it. Error-detection hardware immediately reports failures of the Time Division Multiplexed (TDM)* bus clocks. The control channel protocol provides error detection using sequence numbers, checksums and acknowledgments, and error correction using retransmissions. The control channel driver checks the consistency of the interface and maintains a sanity handshake. The control channel interface processor also checks its own internal RAM and ROM for faults during operation.

Several periodic tests check operation of the control channel. The control channel test verifies communications with several port boards that are installed in all systems. The control-channel processor loop-around test verifies operation between the main processor and the control channel processor. Finally, for severe problems, the reset test is used to reset the control channel processor and trigger local initialization tests without affecting the port boards.

* Acronyms and abbreviations used in the text are listed at the back of the *Journal.*

The data channels are similar to other port boards, except that the data side of the port talks to the processor instead of a terminal. Tests include a loopback through one channel, a data loopback between channels, and a dual-port RAM test of the processor interface. When problems escalate, a data channel processor reset is forced.

4.3 Port boards

Some maintenance features apply to all port boards. Except the tone/clock board, all can be replaced with the system powered and running, and only those calls using the replaced board will be disrupted. The system automatically detects the removal and insertion of port boards. Inserted boards are put into service and tests are run. Removed boards are taken out of service and alarmed.

The first priority for port board tests is to protect service. All port tests, including trunks, stations and service circuits, will not run if the port is busy. In addition, all tests will be aborted if an attempt is made to seize the port externally (off-hook on a station or seizure on a trunk) or if a call attempts to terminate on a port (someone calls that trunk or station). If tests must be run on a busy port, the port can be busied out using the system access terminal to force it idle.

4.3.1 Common port maintenance

The port board processor, bus buffers, and control channel interface are common to all port boards. When a port board processor is initializing, it runs a test of its RAM, ROM, and I/O devices and stops if there is any failure. During normal operation many of these tests continue and report in-line errors. The main processor tests the control channel to each board and audits the network connectivity.

4.3.2 Trunks

The tests listed below are run on most trunk types.

1. Seizure test. On trunks that are outgoing, the trunk is seized to verify the trunk signaling and provide good assurance of a working trunk. This tests dial tone reception on those trunks that provide dial tone.

2. Signaling diagnostic tests. Usually the trunk port board processor can exercise signaling mechanisms and detect correct operation of such items as ground detectors, ring detectors and battery feed. In many cases, special circuitry is used to simulate incoming seizures.

3. Tone loop-arounds. This tests that the trunk is able to transmit voice information successfully in both directions. The trunk is put into a loop-around mode, tones are sent, and the returned signal is tested for level and noise.

4.3.3 Stations

The tests listed below are run on stations.

1. Signaling channel test. Digital stations and multibutton stations have a data channel that transmits control information to and from the station. This channel is tested for correct and reliable data.
2. Station present and overcurrent tests. These tests determine if a station is present and test for shorted or open wiring by measuring battery current flow.
3. Tone loop-arounds. These tests are made in the same way for stations as for trunks (see Section 4.3.2).

4.3.4 Service circuits

Service circuits are tested by hooking two circuits together and testing them against each other. For example, tone generators are connected to tone detectors and both are tested at once. Another example is connecting two pooled modems back to back so that the data are converted from digital to analog and back to digital. The faulty service circuit is identified by testing against known good service circuits.

4.4 Miscellaneous components

4.4.1 Maintenance circuit pack

The maintenance circuit pack is periodically tested using a sanity handshake test and tests of the alarm-origination circuitry. If the handshake test fails, a dual-port RAM test is run to test the interface, and the maintenance circuit pack is reset. During reset, the maintenance circuit pack executes a complete set of initialization tests.

4.4.2 Tape

Most error detection for the tape subsystem occurs during normal use. Periodically the processor interrogates the tape subsystem for status and runs data loopbacks through the interface boards. Every 24 hours the tape subsystem self-diagnostics are started and read/write tests on unused tape areas are run. If errors are encountered, a test reads the entire tape and checks the consistency of data stored on the tape. Finally, warning alarms are raised if the tape is not installed or is write-protected.

4.4.3 Power supplies

The distributed power supplies in each carrier are monitored via the maintenance circuit pack, and all but the control carrier power supply can be recycled or shut down. A carrier supply is recycled if it fails; if this does not restore operation, alarms are raised. The battery charger monitors itself and the batteries for problems. Problems such as a

nonworking charger or an open cell in the batteries can be detected. In addition, worn out batteries that no longer hold a charge are detected if the charger does not shift from a high-charge rate to a low-charge rate within the time normally required to charge the batteries.

4.4.4 Environment

Temperature sensors in the cabinet monitor high temperature and lack of airflow. Alarms are raised when these conditions are detected. Airflow monitoring directly detects dirty air filters, eliminating the need for periodic inspection.

V. SOFTWARE MAINTENANCE

In a computer-based system, most features are provided using software. Software maintenance becomes as important as hardware maintenance, since the system can lose functionality when the software malfunctions. This section describes the principles and techniques used in maintaining the software.

5.1 Software maintenance principles

Software failures are fundamentally different from hardware failures. First, reliability of a software component does not change with use; it cannot break. Bugs are discovered and removed over time; but before the bug is found and fixed, the system must be able to recover and operate with the existing software package. Secondly, a bug in a software package does not necessarily prevent the software from performing its task. Only when the bug is exercised by a specific combination of events can the bug cause failure.

Software may also fail if the hardware executing the software has an intermittent error. Intermittent hardware errors may occur if a particular device has low noise margins or if the hardware deteriorates through aging effects. These intermittent errors may have symptoms similar to software bugs.

Although restarting software execution often permits the system to recover from a software failure, this should be done infrequently and with the least possible service disruption. The system has several levels of recovery to deal with software errors, including single-process restart, system-warm restart, system-cold restart and system reboot. More drastic recovery actions are more likely to be effective but are also more service disrupting. This leads to the following strategy: When the maintenance subsystem detects a system failure without knowing the specific reason, it tries a recovery action that has less impact on the system services. Only if that does not cure the problem is more drastic and service-disruptive recovery action invoked. If the cause of a system failure is known, the right recovery level is invoked

immediately, avoiding the risk of the escalation scheme slowing down system recovery.

5.2 Data audit

Auditing is a technique to ensure that system resources have internally consistent states and to check data consistency periodically. Errors detected by data audits are used to start automatic recovery and provide clues to the system developers for debugging purposes. System resource inventories (e.g., message buffers, path descriptors, etc.) are also audited to detect lost resources. Hardwares status (e.g., circuit port status) is audited against the data kept by software to detect any state inconsistency.

5.2.1 Audit philosophy

The fundamental audit philosophy is to find lost resources before the system performance is adversely affected. When an error is detected, the data are restored to a safe system state, one that has the highest probability of not denying service. For example, a trunk or line found in an erroneous state is restored to the idle state. Another basic principle is that if one error is found, all actions taken to restore that resource do not use any other status data associated with that resource because they may be incorrect or inconsistent.

5.2.2 Audit types

5.2.2.1 Data relation audit. Many audits have been developed to check the consistency of internal system status. For example, an audit checks the user station list stored in the user manager process. If a station is shown as on a call, then the service dispatcher process is checked to see if that user is still active on the call. If not, then the user station is marked idle by the audit. Another audit checks the call record list stored in the service dispatcher process. If there is no user on that call, the call record is released to terminate the stranded call record.

Another example is the audit of the touch-tone receiver status which is kept in the connection manager. The audit will check a call to see if an allocated receiver is still active. If not, the audit will release the receiver.

5.2.2.2 Defensive programming. The data used by a process may be obtained from other processes. To prevent erroneous data from propagating through the system, processes typically check input data before using it. Sometimes a process will check for abnormal local data. However, defensive programming must be used wisely so that it does not introduce significant real-time overhead.

5.2.2.3 Process sanity audit. The hardware sanity timer monitors the sanity of the operating system, which monitors the sanity of the

maintenance control process, which, in turn, monitors all other application real-time processes.

5.2.3 Audit control

Data audits are programmed as a set of functions, and each one can be invoked independently of other system operations. When an application process is restarted, it will invoke the path index audit in addition to other process-specific data audits. A central maintenance control process regularly invokes the data audit functions on a time-available basis.

VI. MAINTENANCE CIRCUIT PACK

One circuit pack in the system is devoted strictly to maintenance functions. As long as the processor is running correctly, the maintenance circuit pack serves as a peripheral device for the processor (see Fig. 2a). In this mode, the maintenance circuit pack is used to provide maintenance access to the system and auxiliary maintenance functions. It is not responsible for testing any pieces of the system on an ongoing basis. However, if a fault makes the processor complex unable to run programs effectively, the maintenance circuit pack will assume overall control and intervene to attempt recovery (see Fig. 2b).

6.1 Normal operating mode

Normally the main processor has responsibility for overall maintenance. The maintenance circuit pack provides the interface to several maintenance functions, including monitoring the status of the ac power, power supplies, battery, and charger. It passes the status of this equipment to the main processor and controls this equipment under the direction of the main processor. It also provides an interface to cabinet temperature and airflow information, external alarms, and system control panel lamps.

6.2 Operation with a critical system fault

The maintenance circuit pack monitors the operation of the main processor, taking control when the main processor becomes completely inoperative (unable to load or run Oryx/Pecos code). To prevent this control of the system from occurring prematurely, the maintenance circuit pack must see several failures over a period of time before it will assume control.

Once it has assumed control, there are some basic changes and simplifications to the maintenance strategy. The maintenance circuit pack will periodically attempt to restart the main processor. It will automatically place a telephone call and alert the Operations Support Center computer that a major problem exists. It will put the system

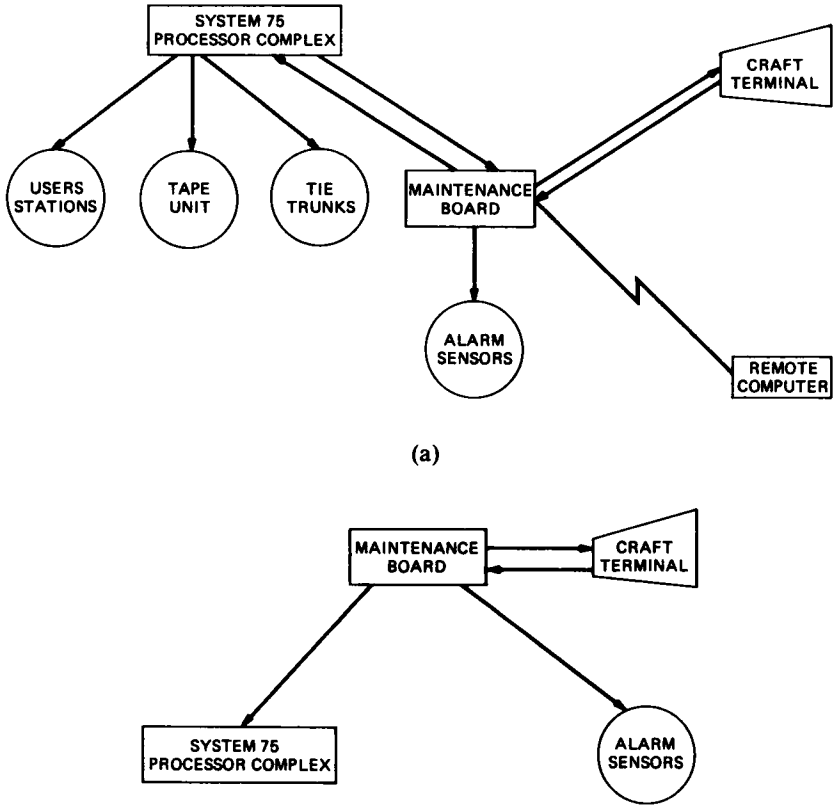

Fig. 2—Maintenance board connection (a) in normal mode and (b) in stand-alone mode.

into an emergency transfer mode where some telephones are connected directly to the central office trunks to provide basic telephone service. And the maintenance circuit pack will continue to directly monitor and control and most critical parts of the power system.

The maintenance circuit pack provides a terminal interface to help in isolating and fixing system problems. This is particularly important since symptoms of total failure are often the most difficult to diagnose. This interface is a proper subset of the commands used when the system is fully operational, eliminating the need to learn another command language. The commands include displaying the cause of alarms and testing critical circuit packs.

Once the cause of the failure is eliminated, through manual repair of a faulty circuit pack or because some temporarily disrupting con-

dition has stopped, control returns to the main processor and normal operations resume.

VII. MAINTENANCE SOFTWARE ARCHITECTURE

7.1 Overview

The maintenance subsystem is composed of three sections. The first section provides system testing and recovery activities, the second stores and retrieves error and alarm logs, and the last section provides an interface for the technician to request actions and query the status of the maintenance subsystem. The relationship of the maintenance processes is shown in Fig. 3.

A central control process called High-Level Maintenance Manager (HMM) provides a mechanism for the whole system to use in reporting errors. The HMM begins maintenance activities based on in-line error reports, technician commands, or time (periodic maintenance). The time-consuming testing and recovery functions are performed in several different instances of Maintenance Action Processes (MAP). The Maintenance Data Manager (MDM) controls the error and alarm log, condenses error and alarm records, and services queries of the log. The technican can issue maintenance requests through the Maintenance Command Process (MCP).

7.2 Process description

7.2.1 High-level maintenance manager

The HMM is responsible for the strategy of maintenance object operations. The HMM will send reported errors to the MDM process

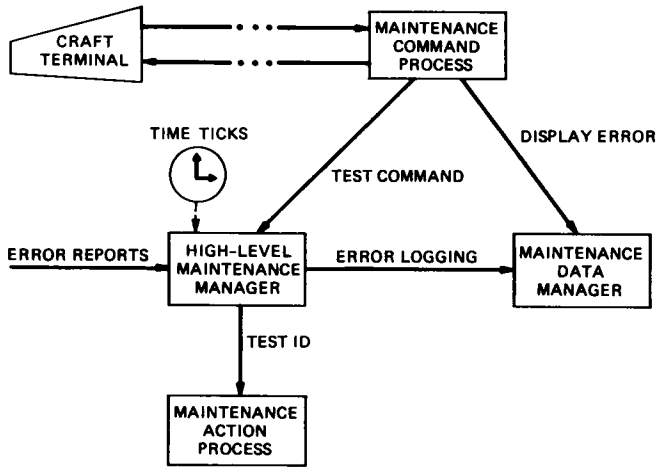

Fig. 3—Maintenance software architecture.

for logging and invoke proper recovery actions. Maintenance can be active and at different states of completion on many maintenance objects simultaneously. For this reason, the HMM multitasks the execution of maintenance strategies.

Errors of all types and severities, some demanding immediate attention, are reported to the HMM. The particular maintenance object strategy determines the priority of the resultant action by dispatching the testing and recovery tasks to a MAP of the appropriate priority. The HMM manages the various MAPs and allocates a MAP when requested by the maintenance object's strategy. Since the MAPs are valuable maintenance resources, the HMM has to control the maintenance load created by background testing to ensure the availability of a MAP should quick testing or recovery response be necessary.

7.2.2 Maintenance action process

There is a set of maintenance test and recovery routines designed for every hardware or software maintenance object type. These are grouped into several different kinds of MAPs, some with multiple instances to provide the required level of concurrency.

1. Initialization MAP (INITMAP). Runs all the critical test and recovery routines, including system initialization and recovery, system power failure handling, and software process recovery.
2. Hardware MAP (HWMAP). Tests for most of the hardware maintenance objects, including processor, port circuit packs, trunks, stations and the tape drive, are grouped into the hardware MAP. Instances of this MAP are created at different priorities to provide the required responsiveness.
3. Data Audit MAP (AUDIT). All the data relation audits are grouped into one MAP.

7.2.3 Maintenance data manager

The MDM is responsible for the error and alarm log. It receives error reports and requests to raise or resolve alarms, and maintains them in an internal database. It supports queries of this database for the maintenance command interface. When alarms are raised or resolved, the MDM controls the proper alarm indications.

7.2.4 Maintenance command process

The MCP converts external maintenance command requests into internal test requests and queries. For example, a request to test a trunk group is expanded into a sequence of tests on the members of the group.

7.3 Message flow examples

The following examples help explain the operation of the maintenance subsystem.

7.3.1 Maintenance command

This example covers a technician test request on a station. (Refer to Fig. 4.)

The technician's command originates in the administration subsystem, which is responsible for the forms-based human interface.[7] The command arguments are passed to the MCP, representing the request to run a test sequence on a specific extension. The MCP converts the station extension to a maintenance object identifier and initiates the tests through the HMM. The maintenance object strategy in the HMM requests a MAP to execute the tests. The MAP executes the test and sends the results back to the HMM, where they are forwarded back to the MCP. The MCP formats the output for display by the administration subsystem.

7.3.2 Digital station failure

This example covers the failure of a digital station leading to an alarm. (Refer to Fig. 5.)

Typically a port failure will appear as an in-line error detected and reported by the port board firmware. In this case a data link error is reported by the digital station board. This error message is routed to the HMM, where a threshold counter is incremented for this mainte-

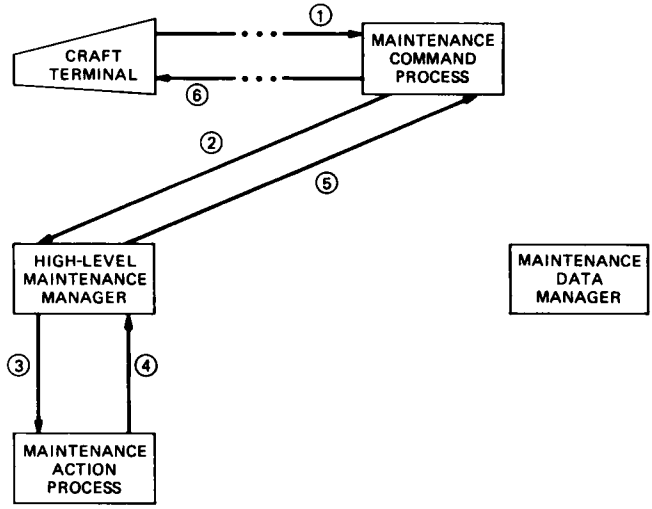

Fig. 4—Craft command message flow.

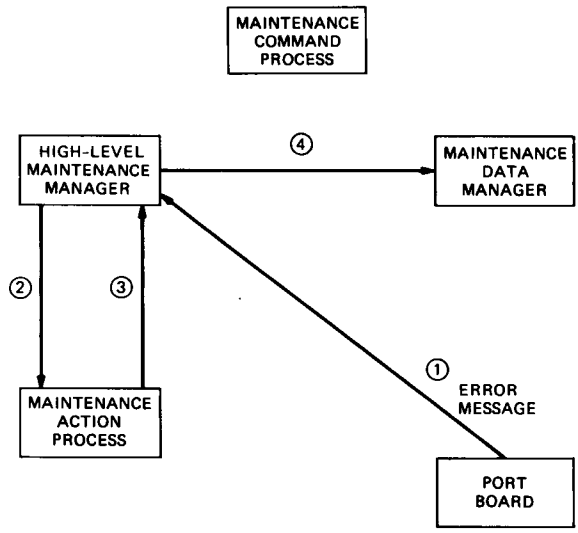

Fig. 5—Station link failure.

nance object. The counter being active causes the maintenance object's strategy to dispatch a test to verify the error.

One test to be run, based on the active threshold counter, is a data link loop-around. This test fails since the data link is faulty and causes the counter to be incremented again. The object strategy schedules more frequent testing while troubles are active on that port. With the test failing, the counter soon exceeds its threshold and starts the MAP alarming action to raise the alarm with the MDM.

VIII. PERSON/MACHINE MAINTENANCE INTERFACE

A CRT terminal was selected to be the system access terminal since this provides the best interface for repair personnel. The terminal provides a powerful and flexible human interface, with a simple command language that is easy to learn and remember. The same terminal and command interface is used for maintenance and administration.[7] Some examples best serve to illustrate the ease with which maintenance activities can be requested.

The simple English-like command language can be used to request tests of specific parts of the system. A circuit pack can be tested by entering:

```
test board A13
```

where `A13` is the location of the circuit pack in the cabinet. Station extension 235 is tested by the command:

```
test station 235.
```

The need to remember and call for specific tests has been eliminated. Each hardware maintenance object has a short test and a long test, with the short test as the default. The short tests are not service disruptive and complete quickly. The long test causes all tests to be run on a maintenance object, it takes longer to complete, and is more likely to disrupt service. This complete test is specified by adding the word *long* to the test command. For example:

```
test tape long
```

The interface also makes use of the terminal forms capability. The system has many options that allow the selective display of errors, restricting it to errors on specific pieces of equipment, to errors occurring over a specific interval of time, or to certain types of errors. Rather than complicating the command that displays errors and making it difficult to remember, the simple command:

```
display errors
```

is used, which brings up a form on the CRT screen. Then any display restrictions are made using form entries.

IX. REMOTE MAINTENANCE CAPABILITIES

System 75 has a complete set of remote maintenance capabilities. These capabilities are provided by a dial-up connection to AT&T Information Systems Operations Support Center centralized maintenance computer.[5] Communication between the system and the support center computer can be established in either direction using the X.25 protocol. This protocol provides error detection and retransmission and ensures that error-free information is exchanged between the two computers.

All the components required for this interface are part of the maintenance circuit pack, including a 212-compatible data set operating at 1200 baud, a line interface, and an automatic dialer. This integrated design speeds installation since it is only necessary to connect two wires from the central office.

From a maintenance point of view, one of the most important capabilities of this remote access is automatic alarming. Whenever an alarm is raised, the system will dial up the Operations Support Center computer and report the details.

The Operations Support Center computer allows remote execution of all maintenance commands that are supported by the system administration terminal. Commands can be issued from the support center regardless of which end made the call, allowing trouble analysis to begin immediately after a trouble is reported.

Other capabilities of the Operations Support Center computer include up-loading and down-loading of translations and remote program update. Any changes to the load module are transmitted and added to a separate patch area on the tape. A remote command then reboots the system, updating the program without a visit to the site.

These remote maintenance functions provide an advantage over strictly on-site maintenance since the remote site can respond more quickly and visits to the site can be more carefully planned. This results in better, faster service at lower cost.

X. SUMMARY

The major objectives in the maintenance design were to make the system reliable, self-testing, and easy to repair. Feedback from the field has been very positive. Maintenance operations, including the terminal interface, alarming methods, error logging, and automatic testing have been successful in providing low-cost support. Most changes based on field experience have been fine tuning maintenance strategies to cover unexpected failure modes or adjusting the sensitivity of alarming thresholds. Software maintenance capabilities have allowed us to weather early software faults and will continue to provide insurance against undiscovered problems.

XI. ACKNOWLEDGMENTS

The design described in this paper is the culmination of the ideas of many people at AT&T Information Systems Laboratories. We would like to acknowledge the efforts of all these people, with particular acknowledgment to M. C. Wei for his contributions to the maintenance software design.

REFERENCES

1. E. J. Braun, "Maintaining the DIMENSION® 400 PBX," Bell Lab. Rec., October 1976, pp. 244–8.
2. P. W. Bowman et al., "1A Processor: Maintenance Software," B.S.T.J., *56*, No. 2 (February 1977), pp. 255–87.
3. H. J. Beuscher, "No. 5 ESS Maintenance Software," IEEE Trans. Commun., *COM-30*, No. 6 (June 1982), pp. 1386–92.
4. K. T. Fong, G. R., Sager, and J. A. Melber, "System 75: ORYX/PECOS Operating System," AT&T Tech. J., this issue.
5. J. E. Smathers and N. T. Tsao-Wu, "RMATS—Remote Maintenance System for DIMENSION® PBXs," Conference Record—Int. Conf. on Communications, June 1979.
6. L. A. Baxter et al., "System 75: Communications and Control Architecture," AT&T Tech J., this issue.
7. H. K. Woodland, G. A. Reisner, and A. S. Melamed, "System 75: System Management," AT&T Tech. J., this issue.

AUTHORS

Kang-sen Lu, B.S. (Physics), 1974, National Tsing-Hua University; M.S. (Computer Science), 1976, National Chiao-Tung University; Ph.D. (Computer Science), 1981, University of Pennsylvania; AT&T Bell Laboratories, 1981–1982; AT&T Information Systems Laboratories, 1983—. Mr. Lu developed the maintenance control software and system recovery strategy used in the System 75 release 1. His current responsibility is on the maintenance software architecture of the System 75 release 2.

John D. Price, B.S. (Electrical Engineering) 1977, Cornell University; M.S. (Computer Engineering), 1978, Carnegie-Mellon University; Bell Laboratories, 1977–1982; AT&T Information Systems Laboratories, 1983–1984; AT&T Consumer Products, 1984—. Recently Mr. Price has worked on maintenance software and product delivery for System 75. He is currently a Supervisor of a group designing residential terminals. Member ACM, IEEE.

Thomas L. Smith, B.S. (Electrical Engineering), 1967 and M.S. (Electrical Engineering), 1969, The Massachusetts Institute of Technology; Bell Laboratories, 1969–1982; AT&T Information Systems Laboratories, 1983—. Mr. Smith has been involved in the design and systems engineering of maintenance features for several systems including 3 *ESS,*™ No. *5ESS*™ and minicomputer maintenance support systems. He is currently the Supervisor of System 75 maintenance and firmware group. Member Eta Kappa Nu, Tau Beta Pi.