Diagnostic engineering requirements

by JOHN J. DENT
International Business Machines Corporation
Kingston, New York

INTRODUCTION

There is a maze of diagnostic techniques in use today: Diagnostic Programs, Micro Programs, Test Panels, On-Line Diagnostics, Error Recording Techniques, Automatic Recovery Procedures; and the list goes on. The purpose of this paper is to take a look at the basic concepts applicable to Error Detection and diagnosis, in order to put into perspective the value of the various techniques for satisfying future requirements.

The terms malfunction, failure, and error are used rather loosely to refer to any deviation from the expected operation of the system, caused by a design error, fabrication error, equipment malfunction or program error. The most common method of automated diagnostic testing uses computer programs written in machine language. However, the same concepts apply whether these are implemented by software, hardware, or firmware.

Uses of diagnostic tests

The design of a diagnostic test is influenced by its intended use or test environments. Perhaps one of the most unique test environments is *Engineering Test*; that is, testing the very first model of a newly-designed system. Engineering Test is characterized by multiple design and fabrication errors, multiple component malfunctions, and multiple errors (bugs) in the diagnostic test itself. The purpose of Engineering Test is to verify the design. "Functional Tests" are written to determine whether the system operates in precisely the same manner described in its functional description or specifications. A test generated automatically from design automation tapes would be of no use in verifying the design itself. Functional tests are generally not sensitive to engineering changes. This is of particular value in the Engineering Test phase when there are a number of engineering changes. In addition to functional tests, random tests and worst case patterns must be developed to "shake down" the design.

The *Manufacturing Test* requirements are similar to the Engineering Test requirements. These tests must be designed to cope with multiple errors resulting from faulty components and fabrication errors. *Field Maintenance*, on the other hand, is slightly different. The assumption is that the system is in good operating condition and that failures are repaired as they occur. The diagnostic designed for this use can take advantage of the single error assumption.

There are several situations which require a *quick and thorough checkout* of the system: Ship Test, Installation Test, Acceptance, Early Morning Checkout, etc. In all such cases, the assumption is that the system is working. The requirement is for a rapid test to verify the fact. This is a GO/NO-GO type of decision requiring a fast and thorough test without regard for isolation.

Basic approaches to testing and diagnosing

The *Start-Small* approach is a building block approach where the first test starts with the smallest amount of circuitry possible. Each additional test adds a small increment to the circuitry tested. When a given test fails, the assumption is that the failing circuit is within the group of circuits added by that test. Figure 1 illustrates the general flow of the Start-Small approach. Notice that the flow is sequential. The sophistication of this approach is found in the design and sequencing of the individual tests. The assumption is that the first failure found in the test sequence is repaired before proceeding past that point. This approach is effective for multiple errors.

The *Multiple Clue* approach bases its diagnosis on the analysis of a series of individual test results. Figure 2 illustrates how the test flow is similar to the Start-Small approach. The difference is that a failure does not terminate the test. All tests are run, failure information is stored, and the diagnosis is determined by analysis of this failure data. Assuming a single error, this analysis can be quite sophisticated. Assuming more than one error, the analysis is extremely complex, if not impossible.

The *Start-Big* test approach has a more complex flow, as illustrated by Figure 3. Testing starts with a

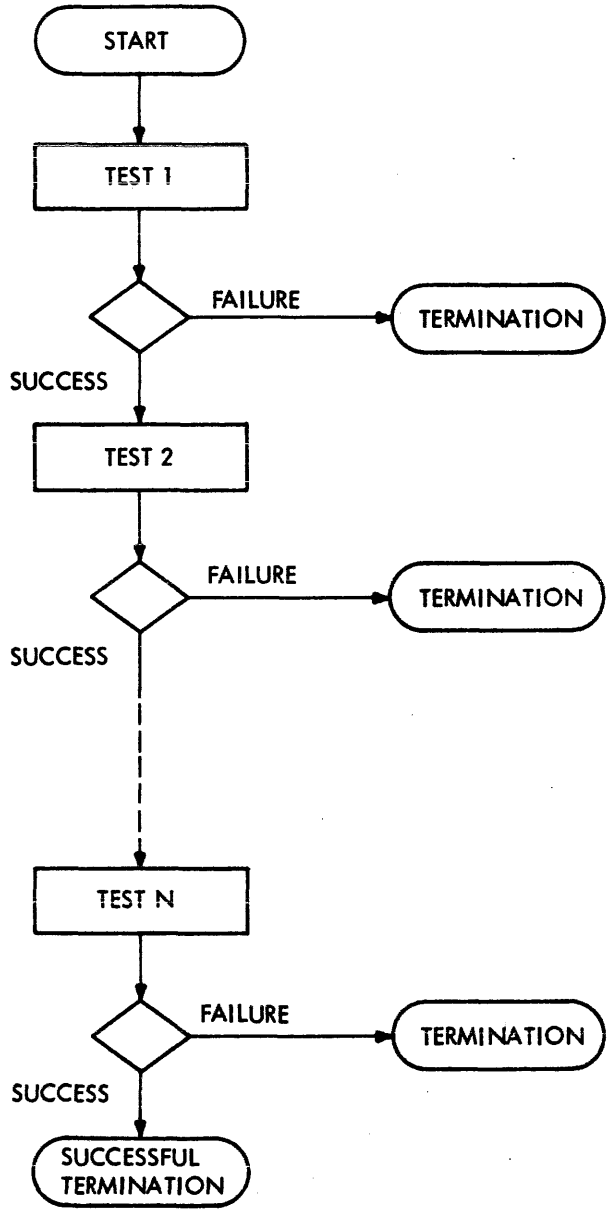

Figure 1—General flow diagram of start-small (building block) test approach

large portion of logic and, if successful, proceeds to another large portion of logic. A failure changes the test sequence by branching to a test designed to further pinpoint the trouble. The test results of each test determine which test is executed next. This approach appears to be the optimum strategy; under successful conditions, it provides a rapid checkout; under a single-failure condition, it provides rapid isolation. However, it is extremely complex in design and presents some hazards. One incorrect branch can lead the maintenance engineer astray. Furthermore, this approach is not applicable for multiple errors.

Figure 4 shows how the three test approaches are rated against the three test environments. The Start-Small test is extremely useful for Manufacturing

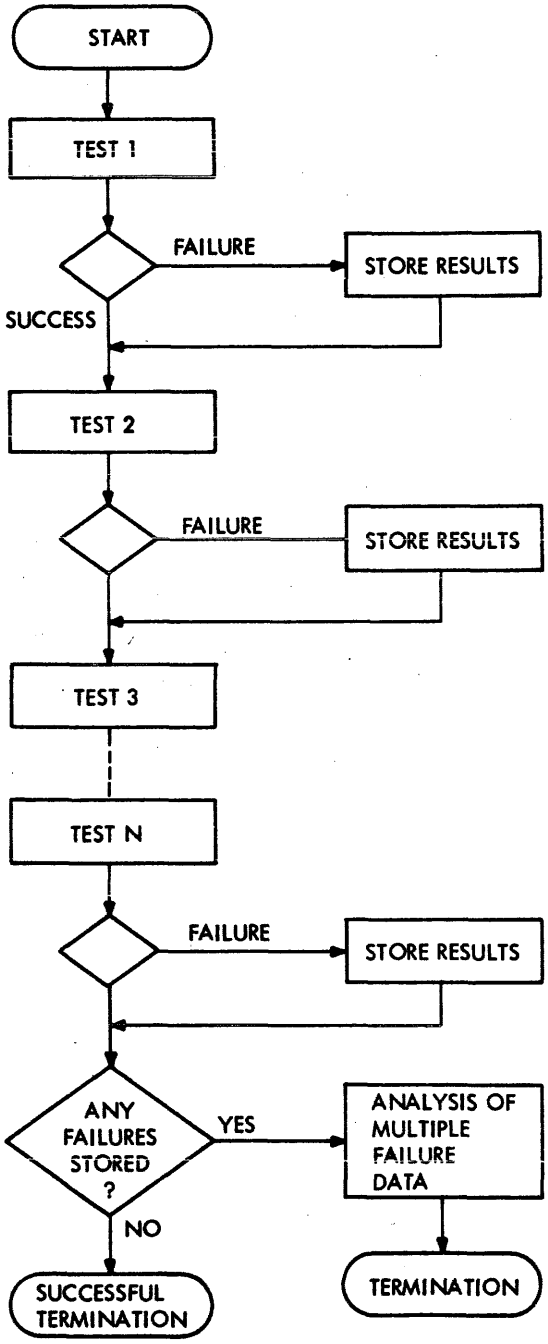

Figure 2—General flow diagram of multiple-clue test approach

Test, since the initial tests are simple and particularly suited for multiple failures. The Start-Small approach is also useful for field maintenance, along with the multiple clue approach to give better diagnosis in certain areas. The Start-Big approach satisfies the need for a quick checkout.

A mixed strategy can be developed to satisfy the different test environments with one series of tests. The Start-Small approach satisfies two of the three test conditions and can form the foundation of the mixed strategy. The multiple clue approach can be

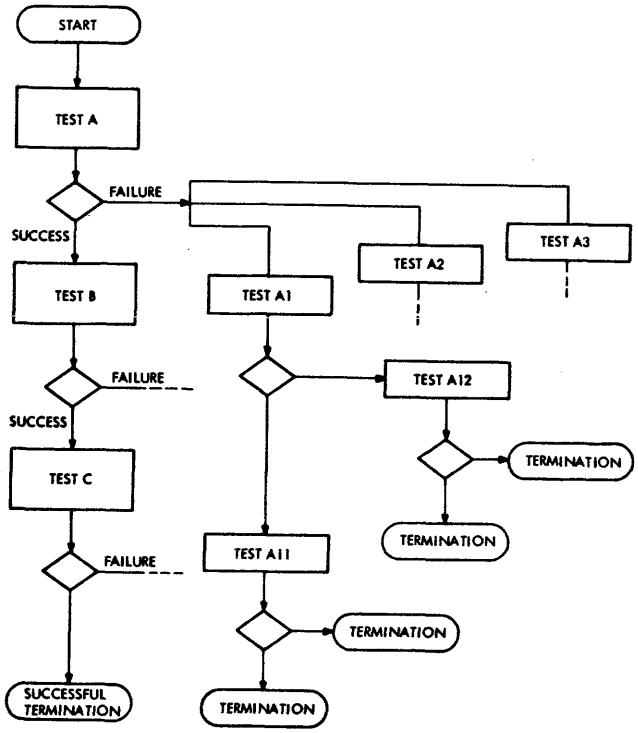

Figure 3 – General flow diagram of start-big test approach (process of elimination)

used where further isolation is required. The Start-Big approach can be used, sparingly and cautiously, to speed up the test process.

All of the techniques mentioned so far depend on the results of more than one test for proper diagnosis. They are susceptible to incorrect diagnosis of the intermittent error, and the diagnosis of a failure depends on the ability to recreate the failure under test conditions. The test environment is an artificial one and, while operating conditions may be simulated, they cannot be duplicated. Therefore, there is no guarantee that a failure which occurs during normal operation can be detected and isolated under test conditions. Error-detection circuitry, along with error-environment recording techniqes, has been used to detect errors during normal operation (on-line) for analysis at a later time (off-line). The ultimate diagnostic seems to be error-check circuitry with enough diagnostic resolution to isolate to the replaceable parts. This creates another diagnostic problem. Verifying the design and functioning of all the error-check circuitry is, in itself, a formidable task.

Field engineering

If Diagnostic Engineering is looked at from the Field Engineer's point of view, a very interesting perspective is obtained. Here is a man who is sent to company schools for extensive training in circuits, logic, theory of operation, programming, operating

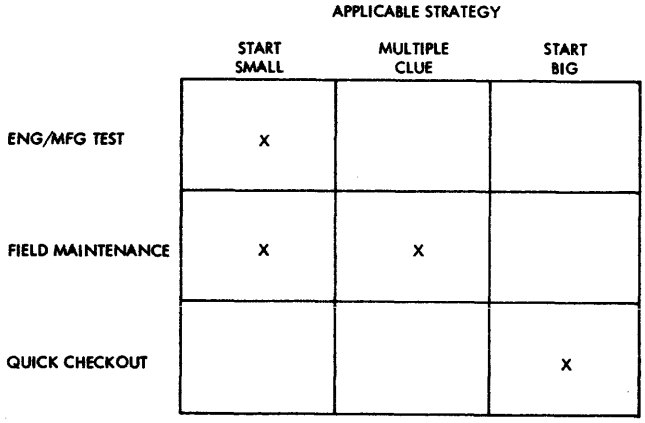

Figure 4 – Test use vs. strategy

systems, diagnostics, etc. He is furnished with a complete set of manuals describing in detail the operation of the system and its subassemblies: hundreds of pages of logic diagrams; documentation for the operating programs; diagnostic program writeups, flow charts and listings. He is furnished with many maintenance aides such as test panels, failure indicators, oscilloscope, tool kit and spare parts.

The problem the Field Engineer faces is that, when a failure occurs, it is not always obvious where to begin looking. The problem could be a program error or a hardware malfunction. Should he spend more time investigating the customer's error symptoms or should he try to recreate the error under test conditions? Should he go to the test panel or run diagnostic programs? Should he run all the available test programs in sequence, or should he select the test program for the suspected malfunctioning area? Can the user still operate a portion of the system while he isolates and repairs the malfunction?

There are no simple answers to these kinds of questions. The field engineer is exposed to a wide variety of problems and operational environments. He has at his disposal a wide variety of testing tools and techniques. He applies his judgment to each unique set of circumstances as he proceeds in the isolation and repair process. All the diagnostic tools, with their options and variations, should be ready and available as he needs them.

This interplay between Field Engineer and diagnostic tools is, in itself, a Human Factors problem. The Field Engineer must control or manage the maintenance system. He must continually select a procedure, execute the procedure, and interpret test results. While he has been furnished with sophisticated systems for automatic testing and diagnosis, the toughest problems have been left for his own ingenu-

ity. The tendency is to compensate for this by providing him with flexibility in the maintenance system. The Field Engineer must communicate with the maintenance system to select the option he wants. The diagnostic programs must communicate test results in meaningful terms.

The Diagnostic Control Program has evolved as a method of standardizing the interface between the Field Engineer and the numerous diagnostic programs. It provides a framework within which all of the individual diagnostics are developed. It provides a standard communication medium between the test engineer and the individual diagnostics. The control program automatically initiates and terminates tests as instructed by the test engineer.

Maintenance systems continue to get more complex with the increased complexity of the systems being installed. Techniques must be developed in the future to simplify maintenance from a Human Factors point of view.

Other factors affecting diagnostic techniques

Up to this point, the basic techniques for testing, detecting and isolating equipment malfunctions have been described. The continuous trend toward more sophisticated data processing systems has placed restrictions on the use of some of these techniques and has created requirements for new techniques.

The technological advance from the tube to the transistor, and now integrated circuits, has given us a significant increase in reliability. The failure rate per logical decision has decreased significantly. The same technological advance is providing us with higher logical density and more logical function per dollar.

In parallel with advances in equipment development is the development of new and more sophisticated applications for this equipment. The result is a continuous trend toward larger and more complex system configurations. As a result of this increased use of equipment per system complex, the typical large complex of today is susceptible to a higher failure rate, in spite of reliability improvements at the logic level.

System error management

Systems must be designed which are fault tolerant; that is, capable of continuing with their primary function in spite of individual equipment malfunctions. This implies the implementation of a total error management concept throughout the system. This concept involves program design, as well as system design. Basically, it includes the following:

1. The ability to detect system malfunctions while the system is operating.
2. The ability to recover by redoing the operation, or going back to some checkpoint in the system.
3. The ability to recognize and isolate a solid malfunction to a unit.
4. The ability to adjust the system (reconfigure) to continue operation without the failing unit.
5. The ability to repair the failing unit without impairing system operation.
6. The ability to return the failing unit to the system without impairing system operation.

All of the above have been implemented to some degree or another for various applications. In some cases, the solution to the problem is applications dependent. For example, in some tracking applications occasional pieces of tracking input can be lost without significant effect on the operation. However, the loss of one business order or stock transaction can be very serious. On the other hand, the tracking application must respond in real time—including the error-recovery procedures—while many business applications can tolerate an occasional lag in response.

System Error Management has been implemented successfully for specific applications. Many defense systems serve as good examples. In many cases the additional programming costs run high. More generalized solutions to the error management problem are required in the future.

SUMMARY

In conclusion, a few general observations can be made. Error detection and diagnosis through the use of automated testing and programmed analysis has several drawbacks:

1. The use of testing for diagnosis assumes the ability to recreate the error under a test environment.
2. Diagnosis to replaceable parts requires sophisticated programs.
3. The effectiveness of diagnosis for intermittent errors is questionable.
4. Automated testing for error detection provides no error detection during normal system operation.
5. Implementation of this technique on-line, concurrent with the operating program, adds more complexity to the maintenance system.

In spite of these drawbacks, the automatic testing approach will still be required in the foreseeable future for design verification and initial system shakedown.

After installation, the best test of the system is the actual operating programs. It is possible to design detection logic which approaches 100% detection of equipment failures during system operation. The same

error detection logic can be designed to provide diagnosis or isolation to replaceable parts for intermittent as well as solid failures. Error detection logic plays a vital role in error management or automatic recovery techniques. As the complexity and cost of programming increases, the economic tradeoff is swinging toward more error detection logic for the detection and isolation of errors.