AN INTELLIGENT APPROACH TO AUTOMATIC TEST EQUIPMENT

William R. Simpson
John W. Sheppard
ARINC Research Corporation
2551 Riva Road
Annapolis, MD 21401

ABSTRACT

In diagnosing a failed system, a smart technician would choose tests to be performed based on the context of the situation. Currently, test program sets do not fault-isolate within the context of a situation. Instead, testing follows a rigid, predetermined, fault-isolation sequence that is based on an embedded fault tree. Current test programs do not tolerate instrument failure and cannot redirect testing by incorporating new information. However, there is a new approach to automatic testing that emulates the best features of a trained technician yet, unlike the development of rule-based expert systems, does not require a trained technician to build the knowledge base. This new approach is model-based and has evolved over the last 10 years. This evolution has led to the development of several maintenance tools and an architecture for intelligent automatic test equipment (ATE). The architecture has been implemented for testing two cards from an AV-8B power supply.

INTRODUCTION

Test program sets as currently configured for ATE have few characteristics of a "smart" technician. A smart technician, approaching a unit that has an indication of malfunction and using a collection of test equipment, would go about the task of localizing the failure in a structured manner. First, the observed anomaly would be cataloged and detailed, then a series of tests would be chosen to categorize the failure. In choosing these tests, the technician might hypothesize an answer and seek to verify it. Recognizing that some tests are hard to perform, take a long time, or are inaccurate, the technician could select alternative tests to minimize cost or uncertainty. Expecting some elements to fail more frequently than others, the technician could focus testing first on those high-failure-rate items or might let the symptoms of the failure direct the testing. If a test instrument failed, the technician might be able to select alternative tests to bypass the problem. In short, a smart technician would choose the tests to be performed based on the context of the situation. In the automatic testing approach to fault diagnosis, we would like to be able to use a similar reasoning process.

Automatic test equipment has evolved at an exponential rate over the last 20 years. We can now measure an order of magnitude more precisely, more often, and over far greater ranges than we could before [1], and our capability in this area continues to grow at an astonishing pace. However, we are not meeting actual test requirements. Repair facilities still struggle with test program sets (TPSs) that find no fault with a known bad unit under test (UUT) and incorrectly identify components containing no faults. These problems arise because UUTs are growing in capability and complexity at the same pace as the ATE. The technology that improves our ability to test makes testing more complex. These types of problems suggest a need to overhaul the current approach so that test equipment is controlled by applying intelligent approaches to fault isolation.

Typical ATE includes a suite of test instruments in a common test station, a test control computer (TCC) to control the instruments and interpret the results of tests, a test unit adapter (TUA) that connects the UUT to the test station, a TPS that instructs the computer on how to test the unit, and a test executive that determines the order in which tests are run and instructs the computer on how to drive the instruments [2]. Until now, TPSs and test executives have been developed to test a unit by following a predetermined fault-isolation sequence, called a fault tree. TPS developers attempted to minimize the mean time to fault-isolate by constructing more efficient fault trees. This approach has worked to a point, but the problem with writing TPSs around any fixed fault tree is the lack of flexibility in the resulting system. Separate fault trees and TPSs are often required for different symptoms, different optimization criteria, and different instrument suites. Further, should an instrument fail, a fixed fault tree would fail to reach a conclusion.

Several initiatives are currently under way to standardize ATE architecture in both the military and commercial sectors. These initiatives include:

- MATE (Modular Automatic Test Equipment)—Developed to provide a set of standards and specifications for U.S. Air Force ATE systems [3].
- CASS (Consolidated Automated Support System)—Developed to provide an industry definition of operational constraints and maintenance policy for automatic test in the U.S. Navy [4].
- IFTE (Intermediate Forward Test Equipment)—Developed to provide ATE for U.S. Army equipment close to their operational units [5].
- CAM (Computerized Automatic Machines)—Now referred to as the T-100 system, developed as a fully integrated diagnostic system by Electronic Data Systsms (EDS) for General Motors Corporation to provide automatic testing of automobiles in the maintenance shop [6].
- SMART™ (Standard Modular Avionic Repair and Test)—Designed by Aeronautical Radio, Inc., to provide the airlines TPSs that will port between a number of ATE instrument suites [2].

TPS DEVELOPMENT

Current approaches to TPS development entail the development of static fault-isolation strategies as the control structure for the test program. Strategies are not based on the changing state of the UUT, the instruments, or the ATE system as tests progress; thus, they are permanently fixed. In addition, the strategies assume that the required resources will always be available; thus, they cannot tolerate "soft failures" of the test equipment.

The most common form of search strategy is directed. Directed search consists of testing a system at its outputs and proceeding backward toward the inputs until the problem is isolated. It is equivalent to a sequential search through the set of possible failure modes. Circuit simulation is used to determine the effects of failures on the tests in the system. The use of simulation is appropriate and will continue; however, nothing is done to determine if the set of tests available can be minimized. A number of simulation tools are available for test program development, including HITS, LASAR, ZYCAD, and PSPICE. The first three provide models for common digital faults, and PSPICE is used for both analog and digital simulation. See, for example, Forster and Colburn [7] or Calandra and Leahy [8].

Because directed search and simulation are the primary means by which diagnostic strategies are constructed, no facility is available for an adaptive strategy. Directed search is fixed, and simulation models usually incorporate history in the test specification. In recent years, TPS developers have realized that significant inefficiencies result from directed search, and they are now using analysis tools to assist in building optimized fault trees. The System Testability and Maintenance Program (STAMP®) [9] and the Weapon System Testability Analyzer (WSTA) [10] are two tools that are capable of providing efficient fault-isolation strategies. These fault trees, however, are still fixed and do not provide the level of flexibility required to address common problems in automatic testing, as described previously.

Another concern with the current approach to TPS development is that resulting TPSs do not provide any means for technicians to capitalize on their experience in testing the system. Often, technicians learn to recognize failures from the reports provided on the system, which thus enables much of the diagnostic process to be short-circuited. But current TPSs force the technician to follow the same procedure every time the system is tested. The result is that technicians are treated as operators of the ATE, having no capacity for problem solving. Currently, little or no "intelligence" goes into developing efficient ATE systems. There is a need to improve on flexibility, adaptability, and technician control while decreasing time to fault-isolate an incidence of improper diagnosis.*

Incorporating symptomatic information is difficult under the current approach because multiple TPSs must be developed for each desired symptom. If combinations of symptoms are indicated, then the problem suffers from

*An improper diagnosis results in one of two field maintenance events. One is the cannot duplicate (CND) event where a fault indication is not repeatable. The other is the retest OK (RTOK) event where a unit replaced at one level is found to be functioning nominally at the next level. See, for example, Simpson et al. [11] for an extended discussion.

a combinatorial explosion. Similarly, failed instruments may be considered in the definition of TPSs, but including all possible combinations is impractical. Thus the need exists for the TPS to be able to adapt to the loss of equipment as it happens.

MODEL-BASED AUTOMATIC TEST EQUIPMENT

There is an approach to model-based automatic testing that addresses each of the concerns raised in the previous sections. In the past, some investigators have considered embedding expert systems into ATE as a means of addressing those concerns. The approach has worked for several other disciplines, particularly in medical diagnosis [12, 13]. So far, this approach has not worked well with the ATE problem for several reasons:

- In new systems it is difficult to identify an expert on the maintenance of the system.
- Expert systems do not optimize well, either because there is too much information to process or the information is not defined well enough to enable optimization.
- It is difficult to modify reasoning and search strategies in the middle of a session. Thus, technician interaction is limited.

In response to these difficulties, we developed a modeling approach and a model-based inference capability. In the intelligent test environment described below, the test program no longer assumes that required resources are available, and tests act on evolving system states.

Tools for Diagnostics

STAMP is a tool for developing information flow models of complex systems [9, 14]. STAMP then analyzes these models to evaluate the testability of a design and generate diagnostic strategies. A tremendous advantage of this approach is that STAMP provides an analysis of test suite deficiencies that can be addressed before the TPSs are coded. This can improve efficiency and remove potential sources of error. Because of the nature of the model, STAMP may be applied to systems of varying complexity and technology, and it is fully hierarchical and capable of handling problems that cross maintenance levels.

The STAMP system has evolved through 10 years of active application to numerous systems. It has provided solid and sometimes spectacular improvements to system testability [15]. The companion tool, POINTER™ (Portable Interactive Troubleshooter) provides an adaptive, interactive environment for diagnosis [16]. The POINTER engine can be embedded to provide intelligent troubleshooting in built-in test (BIT) or ATE, or it can be used as a manual troubleshooting aid. The POINTER system uses the model generated by STAMP and provides a dynamic, context-sensitive environment for manipulating this model. STAMP and POINTER together also provide a framework for an integrated diagnostic architecture by using a common modeling technique and processing system for all diagnostic problems.

Developing a Model—STAMP

The process by which intelligent ATE systems are constructed is completely different from the standard approach to TPS development. The developer no longer assumes the definition of a fault-isolation strategy (through either directed search or an optimization process). Instead, the developer constructs a model describing the information flow in the system. This model can then be used in an analysis to improve system testability before TPSs are coded. Next, the simulation process changes. The TPS developer can no longer assume that the tests are using evolving states. Instead, a neutral point (or points) must be determined for each test (or test group), and the simulations are run from the neutral points. This is the test encapsulation process. Finally, the individual tests are coded and stored in a library to be accessed by the test executive as required. There is no longer any need to develop a complete test program to include the diagnostic strategy.

The information flow model forms the knowledge base for an application. The model provides the logical relationships between tests and conclusions in the system and further enhances and expands on this information by providing descriptions of test inference types, weighting factors, test and conclusion groupings, and forced and recommended sequencing [17].

Manipulating the Model—POINTER

Given an information flow model that has been compiled as discussed above, we can apply an information-theory-based inference engine to the model to optimize the fault-isolation sequence and draw conclusions from the tests performed. The system that serves as the inference engine is POINTER. POINTER is an intelligent, interactive maintenance system that was originally designed to guide manual troubleshooting. We found that the process is directly applicable to other problems in maintenance and diagnosis, such as ATE and BIT, so we adapted POINTER to have it run independent programs. The result is an intelligent diagnostic shell that became the test executive for an intelligent ATE [17].

There are five major elements of the POINTER test executive. The first major element is the process of optimization. This process is based on Shannon's Theory of Information and is called entropy-directed search [18]. Second, the inference engine and the metarules used to guide the diagnostic process are derived from STAMP. These metarules provide the model-based reasoning capability [17]. Third, a number of methods are incorporated for modifying the optimization process to meet the requirements inherent in real-world problems. These modifications are overrides to the optimization process that allow the solution to be reasonable and still be verifiable. They include sequencing requirements, groupings, and other factors that make the test process realistic and achievable. Fourth, two levels of learning are incorporated in the POINTER system. These learning elements include the ability to adapt the optimization parameters to historical data and to identify errors in the model and correct them. Finally, reasoning under uncertainty is incorporated into the test executive. We adapted the Dempster-Shafer approach to evidential reasoning [19, 20] to overcome some of its limitations and added elements of fuzzy logic [21] and neural networks [22] to devise a complete uncertainty-based inference engine [23].

THE INTELLIGENT AUTOMATIC TESTER

The ARINC Intelligent Automatic Tester (IAT) is a small test station that was constructed using the principles discussed above. The system uses off-the-shelf test equipment, an MS-DOS-compatible test control computer, and the POINTER software as the test executive. All of the test programs are written using the test encapsulation concept described earlier. The IAT is shown in Figure 1 [1].

The ARINC IAT was developed for two systems to be used for depot-level maintenance. The two systems are high-voltage power supplies for the AV-8B (Harrier) aircraft being used by the Navy. A prototype manual diagnostic system for the power supplies incorporating the same models is being used in Cherry Point, North Carolina, for depot-level maintenance of the power supplies.

Architecture

There are seven major elements in the ARINC IAT [1]. First, a library of test procedures contains the executable programs for each test in the diagnostic model. Second, an information flow model describes the system to be tested and serves as the knowledge base for the IAT. Third, the POINTER software serves as an intelligent test executive for the IAT. Fourth, MS-DOS serves as the operating system for the environment. Fifth, an 80386-based microcomputer with an 80387 math coprocessor functions as the test control computer. Sixth, VXI and IEEE-488 instrumentation make up the test instrument suite. And seventh, a test unit adapter—TUA (or interface test adapter—ITA) provides a communications interface between the UUT and the IAT. Specific hardware elements have been described by Dill [1]. This interface device is configured for testing the two UUTs for the AV-8B and contains no active circuitry.

In its role as the test executive for the IAT, POINTER operates on a STAMP-generated model to select tests to perform, calls the appropriate test from the library, invokes the execution of the test, reads the results of the test, draws appropriate inferences from the test outcome, and either chooses the next test or reports the results of the fault-isolation process. The test selection process may be modified or controlled by the technician.

The test programs for the two power supplies are individual, independent test procedures that can be called in any order. They have been written to meet the definition of an encapsulated test as described. The programs are written in the C programming language and control both the VXI bus and the IEEE-488 bus instrumentation. Each test procedure can return one of several test results to POINTER, and the result indicates the appropriate action to be taken by POINTER to continue fault isolation. In addition to pass/fail indications, tests can return values that indicate that the test was not able to be performed, and either these results are interpreted by the technician or the test is performed manually.

Three Test Scenarios

Three example scenarios of what may occur in testing a unit serve to describe the capability of the ARINC IAT.

The first scenario is one in which no information is known about the system. A fault has occurred, and the IAT is going to attempt to isolate that fault. The first

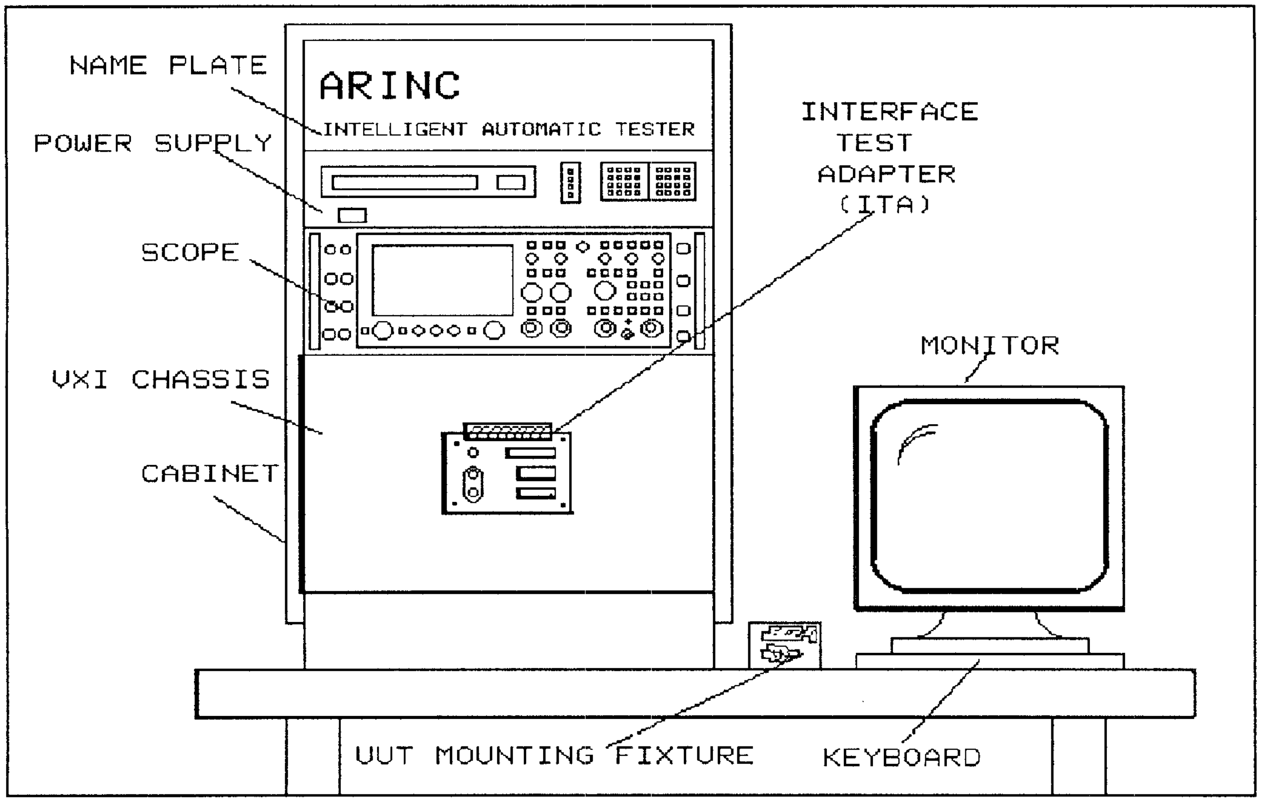

Figure 1. The Intelligent ATE

step in the process is to connect the UUT, via the TUA, to the IAT. Then the appropriate model is selected. Next, safe-to-turn-on and signature tests are performed to verify that fault isolation is ready to proceed. Finally, diagnosis begins, tests are chosen to optimally isolate the most likely failure, and the fault is isolated. Test times are recorded, and failure rates are updated so that the ATE will continue to improve its efficiency.

In the second scenario, diagnosis proceeds as in the first scenario. The difference lies in that one of the test instruments fails during fault isolation. In a traditional ATE system, a failed instrument could cause fault isolation to terminate without an answer. However, in the IAT, if, for example, the oscilloscope fails during one of the tests, the IAT can detect the problem and continue to fault-isolate by choosing alternative tests.*

In the final scenario, assume that the technician is experienced with the UUT and, based on the reports, has an idea what the fault is. Following the safe-to-turn-on and signature tests, the technician enters the expected fault as a hypothesis. The IAT then proceeds to choose tests to verify the hypothesis and, if the hypothesis is the actual fault, locates the problem in fewer steps. If the wrong hypothesis is entered, the IAT collects that information, removes the hypothesis, and continues to fault-isolate.

CONCLUSION

The existence of the IAT demonstrates that it is possible to develop an intelligent ATE system. Tools for model-based reasoning systems have evolved to the point where the use of such systems is practical. The model-based approach offers a large measure of flexibility that is not available from conventional approaches. When combined with the tools for integrated diagnostics, the model development approach will provide significant savings. Models developed during design can be used for testability evaluation and maintenance architecture

*The IAT will not try to derive replacement tests. Instead, it will choose from among available tests to provide the best possible diagnosis.

development. This same modeling process can be used for BIT evaluation and improvement. When fielded, these models can be used to develop maintenance manuals and intelligent ATE test executives. The model-based reasoning approach offers the proven flexibility for an integrated diagnostics program.

REFERENCES

1. Dill, H. "Test Program Sets—A New Approach," *AUTOTESTCON 90 Conference Record*, 1990 IEEE Automatic Test Conference, San Antonio, Texas, September 1990, pp. 63-69.

2. Melendez, E. M., and D. C. Hart. "Airlines Get SMART™ for Avionics Testing," *AUTOTESTCON 90 Conference Record*, 1990 IEEE Automatic Test Conference, San Antonio, Texas, September 1990, pp. 505-508.

3. Cross, G. "Third Generation MATE—Today's Solution," *AUTOTESTCON '87 Symposium Proceedings*, 1987 IEEE Automatic Test Conference, San Francisco, California, November 1987, pp. 289-292.

4. Najaran, M. T. "CASS Revisited—A Case for Supportability," *AUTOTESTCON '86 Symposium Proceedings*, 1986 IEEE Automatic Test Conference, San Antonio, Texas, September 1986, pp. 323-327.

5. Espisito, C. M., et al. "U.S. Army/IFTE Technical and Management Overview," *AUTOTESTCON '86 Symposium Proceedings*, 1986 IEEE Automatic Test Conference, San Antonio, Texas, September 1986, pp. 319-322.

6. General Motors Corporation. "GM Service Information and Diagnostic Technology," *General Motors Techline*, 1989.

7. Forster, P., and C. Colburn. "HITS: A Current Status Report," *AUTOTESTCON '87 Symposium Proceedings*, 1987 IEEE Automatic Test Conference, San Francisco, California, November 1987, pp. 145-150.

8. Calandra, V. P., and P. Leahy. "Applying Advanced System Simulation Techniques to INFOSEC System Development," National Aerospace Electronics Conference, Dayton, Ohio, May 1990, pp. 1066-1070.

9. Simpson, W. R. "The Application of the Testability Discipline to Full System Analysis," 1983 Automatic Test Program Generation Workshop, San Francisco, California, March 1983, pp. 12-18.

10. Franco, J. R. "Experiences Gained Using the Navy's IDSS Weapon System Testability Analyzer," *AUTOTESTCON '88 Symposium Proceedings*, 1988 IEEE Automatic Test Conference, Minneapolis, Minnesota, September 1988, pp. 129-132.

11. Simpson, W. R., J. H. Bailey, K. B. Barto, and E. Esker. *Organizational-Level Testability Prediction*, Report 1511-01-1-3623, ARINC Research Corporation, Annapolis, Maryland, 1985.

12. Pople, H. E. "The Formulation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning," *Proceedings of the Fifth Annual International Conference on Artificial Intelligence*, Cambridge, Massachussetts, 1977, pp. 1030-1037.

13. Shortliffe, E. H. *Computer-Based Medical Consultations: MYCIN*, American Elsevier, New York, 1976, pp. 1-77.

14. Johnson, F., and R. Unkle. "The System Testability and Maintenance Program (STAMP): A Testability Tool for Aerospace Systems," *Proceedings of the AIAA/NASA Symposium on Maintainability of Aerospace Systems*, Anaheim, California, July 1989.

15. Esker, E. A. "Testability Analysis: Applications in the Real World," 1985 IEEE Integrated Diagnostics Symposium, Dayton, Ohio, December 1985.

16. Simpson, W. R., J. W. Sheppard, and C. R. Unkle. "POINTER—An Intelligent Portable Maintenance Aid," *AUTOTESTCON '89 Conference Record*, 1989 IEEE Automatic Test Conference, Philadelphia, Pennsylvania, September 1989, pp. 26-32.

17. Sheppard, J. W., and William R. Simpson. "Incorporating Model-Based Reasoning in Interactive Maintenance Aids," 1990 National Aerospace Electronics Conference, Dayton, Ohio, May 1990, pp. 1238-1242.

18. Shannon, C. E. "A Mathematical Theory of Communications," *Bell System Technical Journal*, vol. 27, 1948, pp. 379-423.

19. Dempster, A. P. "A Generalization of Bayesian Inference," *Journal of the Royal Statistical Society*, Series B, 1968, pp. 205-247.

20. Shafer, G. *A Mathematical Theory of Evidence*, Princeton University Press, New Jersey, 1976, pp. 35-73.

21. Zadeh, L. A. "Possibility Theory and Soft Data Analysis," *Mathematical Frontiers of the Social and Policy Sciences*, L. Cobb and R. M. Thrall (eds.), Westview Press, Boulder, Colorado, 1981, pp. 69-129.

22. Sheppard, J. W. "Terminating Evidential Fault Isolation: A Neural Network Approach," STAMP Technical Note 353, ARINC Research Corporation, Annapolis, Maryland, May 1990.

23. Sheppard, J. W. "Uncertainty Computations in Model-Based Diagnostics," to be presented at the 1991 IEEE Automatic Test Conference, Anaheim, California, September 1991.