DEVELOPING INTELLIGENT AUTOMATIC TEST EQUIPMENT

by

William R. Simpson
John W. Sheppard

ARINC Research Corporation
2551 Riva Road
Annapolis, MD 21401

ABSTRACT

The increasing complexity of electronic systems and the decline in the numbers of skilled technicians are leading to increased reliance on automation in maintenance. Automated maintenance is typified by built-in test (BIT), embedded diagnostic systems (EDS), and improved automatic test equipment (ATE). The BIT and EDS concern themselves with the first level of maintenance (organizational level) while ATEs are developed for the second and third levels of maintance (intermediate and depot). ATEs typically consist of several elements, including an instrument suite, an interface test adapter, a computer, a test executive, and a test program set (TPS). The TPS, which defines how a piece of equipment is diagnosed, usually has been based on an approach using static (i.e., predefined) fault trees. The static fault tree has, inherent in its formulation, limitations in flexibility and adaptability.

Diagnostic modeling techniques have evolved to the point where we can place a limited form of "intelligence" into the ATE process. An intelligent ATE should take into account any known information (e.g., BIT readings, pilot reports, logistic history), adapt to changing conditions during maintenance, and provide maximum flexibility to the maintenance technician. Further, the diagnostic process should be flexible enough to work around deficiencies in test equipment and other factors.

In this paper, we discuss internal research of the past 10 years which led to the evolution of several maintenance tools and an architecture for intellient ATE. Specifically, we review the objectives of the research program, some of its results, and its current applications. We also describe a demonstration intelligent ATE system consisting of a Racal-Dana VXI, instruments-on-a-board automatic test unit for an AV8B power supply. Finally, we discuss the capabilities of this system and its implications for more generic ATE architectures.

INTRODUCTION

Test equipment has been vastly improved since the first appearance of sequential state, paper-tape-driven "automatic" tests systems of 20 years ago. The advent of high-speed digital computers, microprocessors, sophisticated instrumentation, and bus structures has resulted in state-of-the-art automated test systems capable of isolating failures in equipment operating over large frequency ranges and at digital rates exceeding 150 MHz [Dill 1990]. Unfortunately, the advance of software used to harness and drive this powerful instrumentation has failed to keep pace. Repair facilities still struggle with test programs that require "full-up" instrumentation systems and can take little or no guidance from skilled technicians. These are the same problems that plagued the operators using the paper-tape-driven systems. Such problems suggest that we need to overhaul the current approach to controlling test equipment by incorporating intelligent approaches to fault isolation.

This paper addresses one aspect of the entire maintenance problem—fault diagnosis with automatic test equipment. Typical automatic test equipment (ATEs) include 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 unit under test (UUT) to the test station, a test program set (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 [Melendez and Hart 1990]. 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, then currently a fixed fault tree would fail to reach a conclusion.

AUTOMATIC TEST EQUIPMENT

With the lack of effective tools to develop efficient TPSs, current ATE systems are cumbersome and subject to errors in the way they perform diagnosis. Precomputed fault trees are used to guide the test program, and sometimes faulty conclusions are reached. Further, ATE systems are incapable of adapting to lost (i.e., failed) instruments or insights available from the technician.

Several initiatives are under way to standardize ATE architectures. However, the current focus of these systems is on ways to standardize and integrate analysis tools and instumentation interfaces with little attention being given to developing TPSs.

MATE—Air Force. The Modular Automatic Test Equipment (MATE) program began in 1980 as an attempt to address concerns of life-cycle cost in the maintenance of weapon systems. Integral to the MATE concept is an architecture for ATE systems. The program serves to provide a set of standards and specifications for Air Force ATE systems and attempts to implement a structure for controlling and standardizing acquisition, development, and maintenance of ATE systems [Cross 1987].

CASS—Navy. As with the Air Force, the Navy is concerned with reducing the life-cycle cost of systems and ensuring that ATE systems are standardized to help meet that need. The Navy instituted the Consolidated Automated Support System (CASS) program to provide an industry definition of operational constraints and maintenance policy for automatic test. It then went to industry to find a way to fulfill these goals. Further, the test systems were to be available to a weapon system designer so that issues of testability could be addressed early, and more time could be spent focusing on system performance [Najaran 1986].

IFTE—Army. The Army Intermediate Forward Test Equipment (IFTE) program was an attempt to meet the need of increasing tactical flexibility. The purpose of IFTE was to provide ATE equipment for line replaceable units (LRUs) close to their operational units and was to replace all existing test methodologies for all of the maintenance levels in the Army. The end result is a shorter logistic chain for system support [Espisito et al., 1986].

CAM—Automotive Industry. The Computerized Automatic Machines (CAM) system [GM 1989]—now referred to as the T-100 system—is a fully integrated diagnostic system developed by EDS for General Motors. Its purpose is to provide a means for automatically testing automobiles in the maintenance shop by connecting an umbilical line directly to the automobile. The system evaluates the car and produces a troubleshooting tree (fault tree) for the mechanic to follow.

SMART™—Airline Industry. The Standard Modular Avionics Repair and Test (SMART) system is a standard for avionics ATE for the airline community. The system was designed by Aeronautical Radio, Inc. (ARINC), and is being developed by ARINC, TYX, and Aerospatiale. SMART is a modular architecture including a set of standards for a generic test system, which allows a freedom of choice in selecting test instruments, TUAs, and TCCs. The adaptable standard ATE is designed to drive and monitor most avionic units regardless of the manufacturer [Melendez and Hart 1990].

THE CURRENT APPROACH TO TPS DEVELOPMENT

Current approaches to TPS development involve the development of static fault-isolation strategies as the control structure for the test program. Such strategies incorporate no knowledge about the changing state of the system as tests progress and, thus, become 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 with little or no optimization. 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. This is appropriate and will continue; however, nothing is done to determine if the set of tests available can be minimized. Some of the more prominent simulation tools used for test program development include 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 [1987] or Calandra and Leahy [1990].

Since 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 the problem with directed search, and they are now using analysis tools to assist in building optimized fault trees. The System Testability and Maintenance Program (STAMP®) [Simpson 1985] and the Weapons System Testability Analyzer (WSTA) [Franco 1988] are two tools 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.

Once most of the tests have been designed and specified through a set of test requirements documents (TRDs), the test programs can be written. Often additional tests are provided during the coding process, when most of the shortcomings are detected. This process leads to potential diagnostic errors in the final TPSs. Generally, test programs are written in ATLAS (A Test Language for All Systems); however, there is an emphasis today on writing the programs in Ada (for military applications) or C (for commercial applications). Another important part of the TPS development process is the design of the TUA. TUAs are usually unique to the UUT and must be included with each TPS.

LIMITATIONS OF CURRENT APPROACHES

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 and incidence of improper diagnosis.* Another concern with the current approach to TPS development is that resulting TPSs do not provide any means for a technician to capitalize on his or her experience in testing the system. Often, technicians learn to recognize failures from the reports provided on the system, thus enabling 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 end result is that technicians are treated as operators of the ATE with no capacity for problem-solving.

Incorporating symptomatic information is hard in the current context because multiple TPSs must be developed for each desired symptom. If combinations of symptoms are indicated, then the problem suffers from combinatorial explosion. Some people are examining the possibility of using expert systems to drive TPSs, but this approach makes the optimization process mentioned above even more difficult. Thus, either predefined fault trees are computed with an epsilon-optimal approach and symptoms are ignored, or symptoms are included in the analysis and much of the optimization capability is lost. As with the symptom problem, failed instruments may be considered in the definition of TPSs, but to include all possible combinations is impractical. Thus the need exists for the TPS to be able to adapt to the loss of equipment as it occurs.

INTELLIGENT AUTOMATIC TEST EQUIPMENT

In this section, we discuss an approach to intelligent 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 for overcoming these problems. The approach has worked for several other disciplines [Pople 1977; Shortliffe 1976]. So far, this approach has not worked well in the ATE problem for several reasons:

1. In new systems it is difficult to identify an expert on the maintenance of the system.

2. Expert systems do not optimize well because either there is too much information to process or the information is not well defined enough to enable optimization.

3. 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 applied the modeling approach developed with STAMP and the inference capability of POINTER™ to this problem. In developing the knowledge base for this system, because of the unique nature of the model, no commercial shell was used. Rather, the ARINC STAMP tool was used to generate the model, which was then directly manipulated by the POINTER engine. In the intelligent test environment described below, the test program will no longer assume that required resources are available and that tests act on evolving system states.

STAMP and POINTER

STAMP is a tool for developing information flow models of complex systems [Simpson 1985; Johnson and Unkle 1989]. These models are then analyzed by STAMP to evaluate the testability of a design and generate diagnostic strategies. A tremendous advantage to this approach is that STAMP will provide an analysis of test suite shortcomings that can be addressed before the TPSs are coded. This can improve efficiencies and remove potential sources of error. Because of the nature of the model, STAMP may be applied to systems varying in complexity and technology, and it is fully hierarchical and capable of handling problems that cross maintenance levels. The companion tool, POINTER—Portable Interactive Troubleshooter—provides an adaptive, interactive environment for diagnosis [Simpson, Sheppard, and Unkle 1989]. 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. STAMP and POINTER together provide a framework for

* 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 okay (RTOK) event where a unit replaced at one level is found to be functioning nominally at the next level. See for example, Simpson [1985] for an extended discussion.

an integrated diagnostic architecture by using a common modeling technique and processing system for all diagnostic problems.

The Information Flow 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). The developer, instead, constructs a model describing the information flow in the system. This model can then be used in an up-front analysis to improve system testability before coding TPSs. Next, the simulation process changes. The TPS writer can no longer assume that the tests are using evolving states. Instead, neutral points must be determined for each test (and test group) and the simulations run from the neutral points. This is called 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 the application. The model provides the logical relations 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 [Sheppard and Simpson 1990]. Other systems use similar approaches, such as IDSS [Franco 1988] and ICAT [Cantone, Chesoweth, and Cassaro 1990].

The Test Executive—POINTER

Given an information flow model as discussed above, we can apply an information-theory-based inference engine to the model in order 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 was directly applicable to other problems in maintenance and diagnosis, such as ATE and BIT, so we adapted POINTER to be able to run independent programs. The result was an intelligent diagnostic shell which became the test executive for our intelligent ATE [Sheppard 1990].

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" [Shannon 1948]. Second, the inference engine and the metarules employed to guide the diagnostic process are derived from STAMP. These metarules provide the model-based reasoning capability [Sheppard and Simpson 1990]. Third, a number of methods were incorporated for modifying the optimization process to meet the requirements inherent in real-world problems. These modifications are optimization process overrides 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 have been 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 [Sheppard 1989]. Last, we incorporated reasoning under uncertainty into the test executive, and we adapted the Dempster-Shafer approach [Dempster 1968; Shafer 1976] to evidential reasoning to overcome some limitations. We added elements of Fuzzy logic [Zadeh 1981] and neural networks [Sheppard 1990] to devise a complete uncertainty engine.

THE ARINC INTELLIGENT AUTOMATIC TESTER

In this section, we discuss the specific elements of our application. The ARINC Intelligent Automatic Tester (IAT) is a small test station constructed using the principles discussed above. The system uses off-the-shelf test equipment, an MS-DOS-compatible test control computer, the POINTER software as the test executive, and a library of the encapsulated test programs written by ARINC engineers [Dill 1990]. The IAT is shown in Figure 1.

Application—AV8B Power Supplies

The current application of the ARINC IAT is limited to two systems: two cards from a high-voltage power supply for the AV8B (Harrier) aircraft being used by the Navy. The IAT system is intended to be used for depot-level maintenance and is currently used by two technicians in Annapolis, Maryland. 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. The ARINC IAT is now a part of the complete ARINC line of maintenance and diagnostics products.

Architecture

There are seven major elements in the ARINC IAT. 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

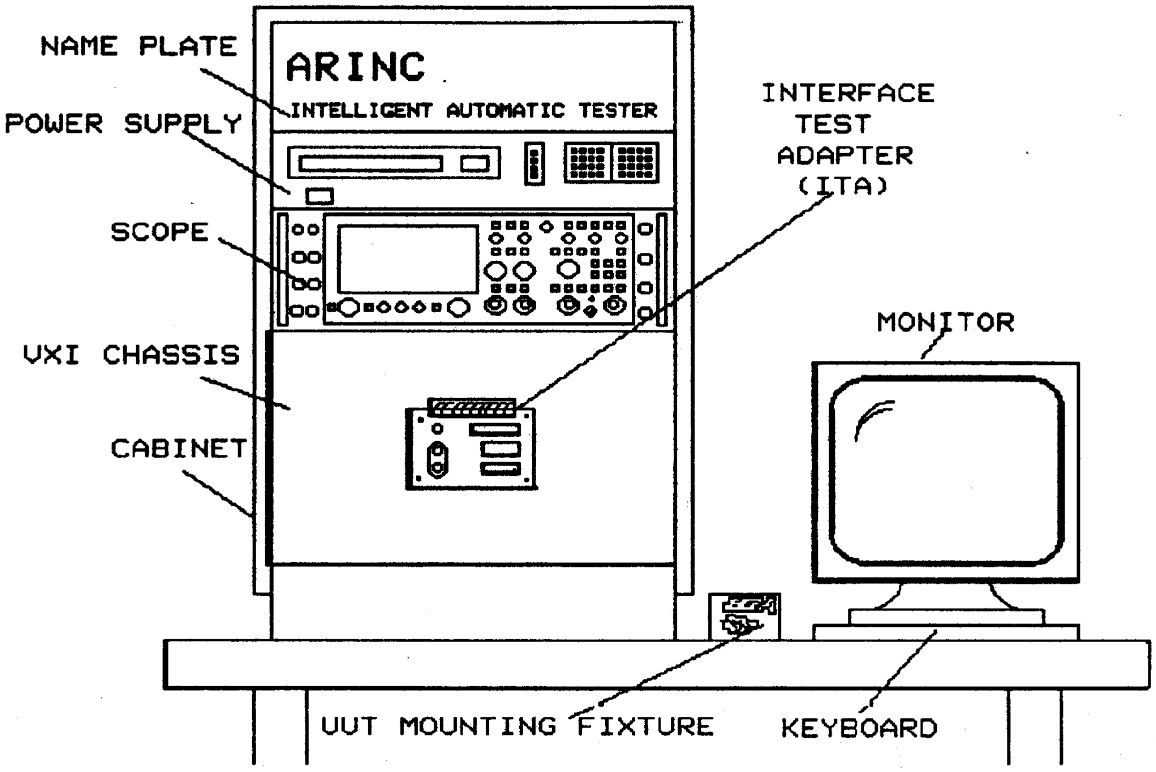

Figure 1. The Intelligent ATE

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. Figure 2 shows the basic system configuration [Dill 1990].

The IAT is hosted in a C-size Racal-Dana 1262-10 VXI chassis. This chassis houses a Racal-Dana Model 2251 counter/timer, a Datron Wavetek Model 1362 digital multimeter, a Racal-Dana Series 1260 universal switching system, and a Colorado Data Systems 73A-270 arbitrary pulse/pattern generator. In addition to the VXI instrumentation, this system contains two IEEE-488 bus-controlled instruments: a Tektronix Model 2430M oscilloscope and a Hewlett-Packard Model 6624A power supply. Figure 3 shows the IAT with the instrument boards. The interface test adapter is a Virginia Panel Series 90 Ten Module Hybrid Connector System and modified enclosure. This interface device is configured for testing the two UUTs for the AV8B and contains no active circuitry.

POINTER, as described above, can be thought of as the test executive for the IAT. POINTER operates on a generated model to select tests to perform, calls the appropriate test program 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. They 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 five test results to POINTER as described above: good, bad, untestable, manually interpret, or manually perform.

Three Test Scenarios

To demonstrate the capability of the ARINC IAT, we use three scenarios that may occur in testing a unit. Actually, the IAT is capable of handling much more; therefore, these scenarios serve as examples only.

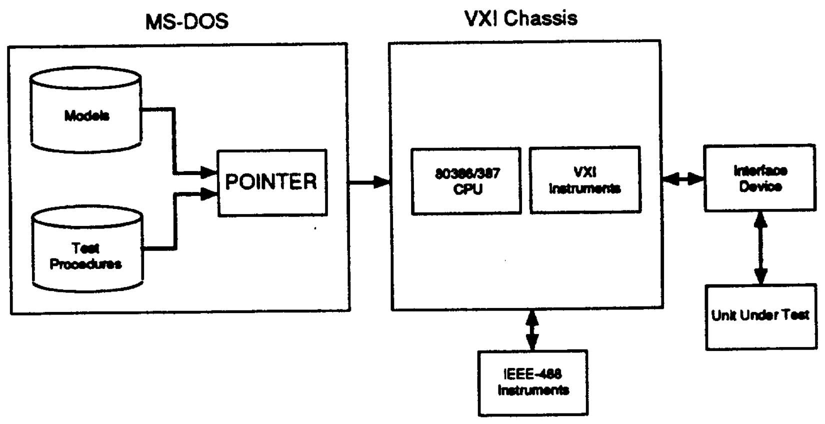

Figure 2. Architecture of the ARINC IAT

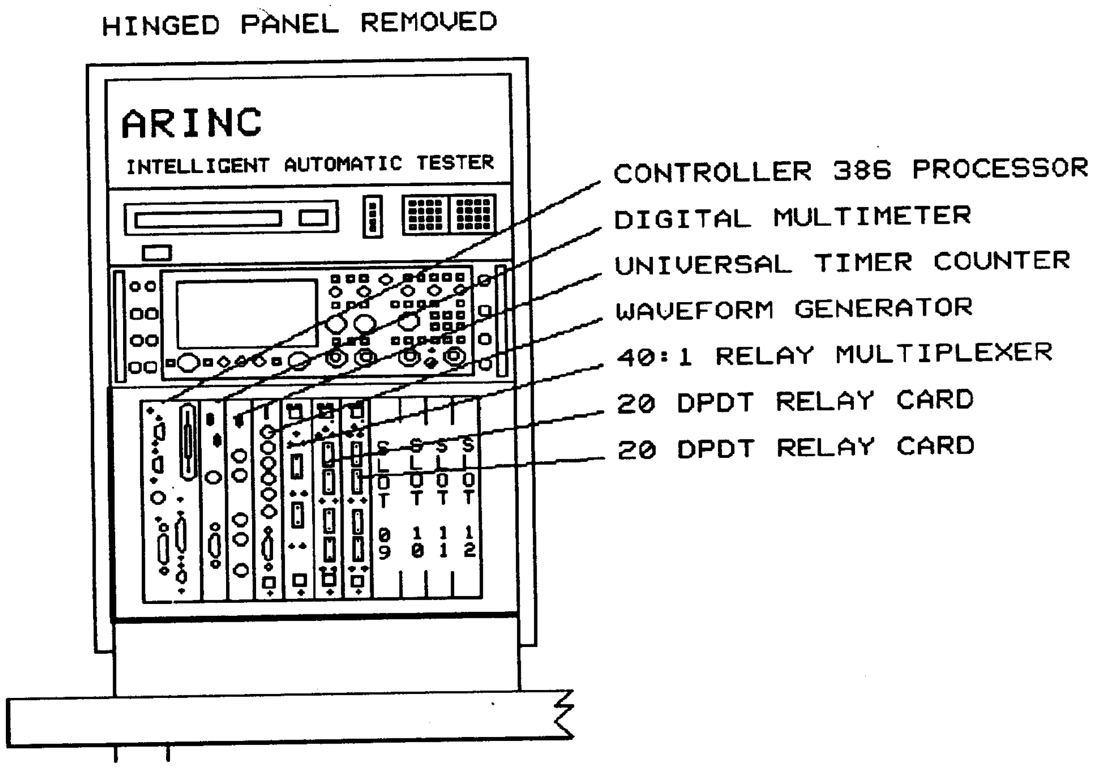

Figure 3. Instrument Boards Used in the ARINC IAT

Full Test Scenario

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

Test with Failed Instrument

In the second scenario, diagnosis proceeds as in the first. The difference lies in that we fail one of the test instruments during fault isolation. In a traditional ATE system, a failed instrument could cause fault isolation to terminate without an answer. For the example, we will turn off the oscilloscope during the fault-isolation run to simulate that the oscilloscope has failed. The IAT will detect the problem and continue to fault-isolate by choosing alternative tests.*

Test with Hypothesis

In the final scenario, we will assume that the technician is experienced with the UUT and, based on the reports, he or she has an idea of what the fault is. Following the safe-to-turn-on and signature tests, the technician may enter the expected fault as a hypothesis. The IAT then proceeds to choose tests to verify the hypothesis and, indeed, locates the problem in fewer steps. If the wrong hypothesis were entered, the IAT would collect that information, remove the hypothesis, and continue to fault-isolate.

CONCLUSION

Some interesting insights came from the development of this system. First, most real-world applications require multiple technologies in order to get them to work properly. When integrating artificial intelligence (AI) into an application, no single AI technique is likely to solve the problem. Thus integrating several techniques becomes necessary. Second, when integrating several technologies, each technique tends to be straightforward.

Thus, the difficulty does not come in implementing a particular technique. Rather, the difficulty comes in combining the techniques. Information provided by one approach may require conditioning or interpreting before it can be used by another approach. Further, different approaches are occasionally completely incompatible and cannot be integrated. These issues must be considered whenever combining technologies—not just AI technologies. Finally, it is clear that "academic" solutions to problems rarely work without modification in the real world.

For example, in the optimization problem, it is nice to have a provable optimal solution (or even an epsilon-optimal solution), but in diagnosis, most problems have so many constraints and time is such a critical factor, that optimizing becomes nearly impossible, and using the original optimization process on a single objective function may yield inappropriate or incorrect results.

This paper has described an application of several artificial intelligence techniques to the problem of performing intelligent, adaptive, and efficient automatic testing. The current approach to automatic testing involves writing test programs that follow a predefined diagnostic strategy to fault-isolate the system. Interaction from a technician is eliminated to the point that valuable experience and expertise are completely ignored. The application presented is the first real ATE system incorporating principles of inference, optimization, uncertainty, learning, and the ability to adapt to user requirements in one complete system. The techniques combined are necessarily innovative, and they provide an innovative, integrated approach to solving a difficult problem.

REFERENCES

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.

Cantone, R., D. L. Chesoweth, and M. J. Cassaro, "Self-Improving ATE," in *AUTOTESTCON 90 Conference Record*, 1990 IEEE Automatic Test Conference, San Antonio, Texas, September 1990, pp. 145-152

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

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

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

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

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

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

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

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

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

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

Najaran, M. T., "CASS Revisited—A Case for Supportability," in *AUTOTESTCON '88 Symposium Proceedings*, 1986 IEEE Automatic Test Conference, San Antonio, Texas, September 1986, pp. 323-327.

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

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

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

Sheppard, John W., *Learning Diagnostic Information Using a Matrix Based Approach to Knowledge Representation*, Master's Thesis, The Johns Hopkins University, 1989.

Sheppard, John 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.

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

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

Simpson, W. R., "The Application of the Testability Discipline to Full System Analysis," 1985 Automatic Test Program Generation Workshop, San Francisco, California, March 1985, pp. 12-18.

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

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