ATE ENABLING TECHNOLOGIES

Larry V. Kirkland, OO-ALC/TISA
7278 4th Street
Hill AFB, Utah 84056-5205
Phone (801)-777-2142

Jeffrey S. Dean, SA-ALC/LDAE
404 Greig Street
Kelly AFB, Texas 78241
Phone (210)-925-4401

ABSTRACT

A discussion of the current and emerging core technologies and philosophies that will enable Air Force personnel to quickly, accurately and intuitively diagnose faults in increasingly complex systems.

INTRODUCTION

Until recently, ATE has been limited in capability by the hardware and software technology available. Rather than work with our strengths, ATS user interfaces have historically played on our weaknesses. The ATS user, through long training and experience has had to adapt to the machine, rather than the machine adapting to the user. After the user was trained, he often moved on to another job or eventually retired, taking his expertise with him. This would require another long learning period as a new user adapts to the ATS.

With the advent of fast microprocessors and graphical, intuitive software interfaces our computers, and ATE, are beginning to adapt to us. Graphical, object oriented user interfaces, machine-based intelligence and perception, virtual realism, speech recognition and other emerging capabilities are making or will make ATE easier to understand and use. In addition, the push for open hardware and software standards, object oriented programming and software libraries will allow faster ATS and TPS design and generation, providing the flexibility needed to adapt to changing requirements and technologies.

HUMAN WEAKNESSES AND THE COMPUTER

The ability to read, perform mathematical analysis, and to logically attack and solve a problem are, unfortunately, not natural functions for the human species. Instead we are trained over much of our childhood (hopefully) to learn to use these skills. Most of us can on the other hand recognize a friend in a crowd, make a guess based on incomplete information, or create solutions to problems by generalizing from similar problems solved in the past.

Computers, and ATE, have required us to operate in the realm of our weaknesses: details are demanded while we tend to generalize facts, keyboard/textual interaction (often in arcane codes or commands) is required of beings who process data visually (spatial interrelationships) and orally (speech). With time and training, we can acquire the skills needed to work with these machines. Our learned ability to operate in a manner that is not natural and requires more effort is usually referred to as a skill. The more skilled we are considered to be, the less natural our behavior and sensory processing have become. Eventually, after the person obtains the training, skills and experience needed to alter ones behavior to adapt to the computer, the person is moved somewhere else, starting a new cycle of frustration and
inefficiency.

We can, as adaptable humans, learn to operate and think in these skill-based, text dependent modes, but there are costs. Training time is required to use any ATE, errors are made that must be corrected, efficiency is lost by having to perform in a non-visual, skill intensive environment. As the Air Force downsizes (or rightsizes), it can afford less and less to pay these costs. We are hemorrhaging our skills base as our senior technicians take early retirement, while training funds are harder to come by for those who remain. Errors in repair become more important as the spares supplies dwindle, and efficiency becomes critical as our workforce gets smaller while the job of the Air Force only gets bigger.

**DRAWING FROM/ COMPLEMENTING HUMAN STRENGTHS**

Under what conditions can we expect a complex dynamics of information to emerge spontaneously and come to dominate human behavior[4]? Human beings are highly visually oriented. We look for patterns, we can "spot" mistakes, and when we understand something it is often acknowledged with "I see!" Someone with exciting new ideas is often described as being a visionary. Our visual orientation is reflected in our brain; a significant chunk of our cerebral cortex is devoted towards extracting, identifying and interrelating objects within our visual space, creating a world view of our surroundings within our mind. Within our internal visual world we can extend ourselves to incorporate phenomena, concepts or objects that are not physically existent, such as Einstein's experiments in the imagination that he used to understand relativity. Essentially we have within us a very flexible, massively parallel visual processor that has been largely unused in the past when trying to diagnose faults within electronic systems via ATE.

We can improve ATE effectiveness by tapping into this visual processing capability in two basic ways. First, by using graphical objects on the ATS monitor we can display the components on a circuit card as a visual, object oriented representation; by selecting chips, capacitors, resistors, etc. with a mouse we can call up applicable tests, voltage levels, and failure data associated with that object.

We can navigate through windows, icons, buttons, menus and synthetic test devices to drive the ATS to perform tests, initiate diagnostic algorithms, and display results. In other words we can create a graphical world, a "virtual reality" in which we manage our ATE and manipulate data in an intuitive manner. The appeal of operating in the graphical environment is obvious. Anyone who doubts the benefits and appeal of the graphical interface can try telling the attendees at any software convention how they love DOS, because it is easier to work with than Windows. Possible consequences of this action include being laughed at or being anesthetized and put in a rubber room.

The second way in which we can assist ourselves visually in diagnosing a faulty circuit card is to expand our ability to "see" the card. X-ray technology can allow us to look for broken circuit paths, spot bad or marginal solder joints, or derive interconnections for reverse engineering of

![Figure 1. Test Points.](image-url)
a card no longer in production. Infra-red imaging can spotlight chips operating hotter or cooler than normal, indicating the improper functioning of that chip or the components connected to it. The imaging of the electrical fields around the circuit card can show where the current is flowing and where it is not, and whether it is AC or DC (Fig. 2).

**ELECTROMAGNETIC SPECTRUM**

![Electromagnetic Spectrum](image)

**Figure 2.** Electro-magnetic.

Because of our ability to extract and compare visual patterns quickly, these methods give us a fast means of determining the location and type of failure involved.

Another information rich sense that humans use naturally is that of hearing, in particular the ability to speak and to understand speech. Being able to talk interactively with a computer has been a dream for decades and, until recently, has been considered science fiction. With the advances in speaker independent speech recognition, this capability may soon become science fact. Such a capability would enable an ATS user to operate the ATS with his hands free to manipulate connections to the UUT for various tests. With voice responses the system could provide test results verbally, freeing the user from being trapped at a monitor as tests are run. This freedom to leave the keyboard adds a dimension to the user's capabilities, allowing him to perform other operations in parallel with controlling the ATS.

**KEEPING HUMAN EXPERTISE IN THE MACHINE**

While the above technologies can make computers (and ATS) easier to learn and to operate, the user must still direct the machine to act in accordance with his experience and knowledge. An experienced ATS user knows the failure modes of the circuit cards he tests; he knows which tests fail most often. In possessing this knowledge, the experienced user can accurately diagnose the bad component(s) on a faulty card much faster than the novice, often just by looking at the card (heat discoloration, cold solder joints). Every test organization has its share of wizards and gurus, who understand the idiosyncrasies of a particular ATS or UUT. When such a person leaves, the organization loses capability and money, as it trains someone to replace him.

The answer to this dilemma is to retain expertise within the ATS itself. This can be achieved by incorporating various forms of Artificial Intelligence in the ATS, with the ability to extract failure information from each UUT tested and to learn from them. At first, any such machine would require some baseline knowledge about the UUT and its failure modes, drawn from the UUT design and specifications. This is necessary for the initial testing of the UUT in order to have some hope of rendering a correct diagnosis. After the testing is completed and the failed component identified and confirmed, the ATS can begin associating specific test failures, X-ray, infra-red or electromagnetic field image patterns, or signal strength levels with different failure modes. Each time that type UUT is tested, the ATS becomes smarter in identifying the failed components.

**MACHINE-BASED INTELLIGENCE**

Machine-based Intelligence can be viewed as Adaptive Information Processing Systems designed to alter its response to specific criteria or events.
which results in optimizing task quality and performance. Advanced emergent concepts including Neural Networks, Genetic Algorithms, Cellular Automata, Lindenmayer Systems, etc., can be used to effectively generate, maintain, and optimize operational processes. Machine-based Intelligence can be used on ATE to control the TPS testing sequence, to provide alternative, high-speed accurate fault-diagnostics and to provide advice as to the optimal testing technology.

RETARDUS ILLUMINOUS GIGANTUS
(Slow Blinking Monsters)

ATE has been technology limited. Depending on the application, to integrate all of the stimulus and measurement devices for a specific application could have required a test system 50 feet long or more. The computing power of the system was also technology limited and most definitely a closed system. ATE is evolving as technology evolves. Our task is to stay in line with the technology and work together as a united technology team, both industry and government, in an open architecture environment. An open-architecture hardware environment is a platform in which stimulus and measurement modules can be intermixed using equipment from different manufacturers to make up a system. An open-architecture software environment is a platform in which different software packages can be integrated into an interoperable system.

OPEN ARCHITECTURE

Off-the-shelf operating systems (i.e., DOS, UNIX, WINDOWS, etc.) provide a high performance platform for machine intelligent implementation on ATE. Personal computers provide versatility with their high speed, memory manageable, open systems. The IBM 486 AT architecture has more processing power than many of the large, multi-cabinet ATS packages in use today, and with it's universal availability, upgradability and price, it is surprising that everyone has not jumped on the bandwagon of hosting new or upgraded TPSs on this platform. In the near future, small businesses may very well be undercutting major ATS suppliers by offering the same functionality at less than a tenth of the cost, simply by taking advantage of the power of this architecture.

Open architectures in an integrated technology environment permit the combination of multiple information sources for the diagnostic evaluation of a UUT. Open systems provide interactivity, interoperability, multiple languages, graphics, multiple technology and a pick and choose testing platform. The Air Force currently has far too much proprietary ATS purchased to support specific weapon systems, which over time have become unsupportable and expensive, with no clear upgrade path.

VXI testing platforms generally are open, modular, compact and easier to integrate than any technology in existence today (1). VXI has inspired new approaches to building modular instrument systems, and it has also allowed advanced approaches to integrating these components with software to build system level open solutions. It is crucial that VXI remain an open standard, rather than have various proprietary spin offs that are incompatible, and cannot communicate or talk to each other. The Air Force is looking at long term, life cycle costs for new
ATS acquisitions, and closed systems are more expensive in the long run to manage and upgrade. Figure 4 shows a typical VXI layout.

Figure 4. VXI Tester.

INTEROPERABILITY STANDARDS

No standard exists at this time for the expression of information and knowledge content at the domain level for specific applications of Knowledge-based learning systems. Critical issues are therefore:

- How do we talk back and forth (how do programs communicate)
- How do we structure our sentences so we can understand one another (what are we conveying)
- What should be the specific domain level words or expressions.

An activity to develop a standard ontology of common terms to represent sensor and sensor-based signals in Knowledge-based learning Systems is in work (2). Ontology provides a lexicon of common terms, providing the users with a common dictionary of standard labels and units which have a precise, well-defined meaning, independent of the internal or individual processing system.

The Sensor-based Ontology is an ATE-specific ontology which provides for interoperability between different software packages, testers, sensors, and Instruments on A Card (IAC). Interoperability is talking the same language, using the same terms, and inferring the same meaning; so no matter what internal or individual system processing is used at least the conveyance of sensor-based information will be understood. Essentially, interoperability is the ability to communicate between software packages and to understand what is said. A simulation of potential information exchange is shown in Figure 5.

Figure 5. Interoperability.

A Broad Based Environment for Test (ABBET)

The DoD, like many large corporations, is energetically working on all reasonable methods to cut the cost of doing business. The cost of providing testing has been significant, especially for an organization that must be able to repair state-of-the-art electronic systems along with those designed and manufactured decades ago. Automatic test systems are acquired, TPSs are written, and over the years the ATS becomes obsolete and unsupportable. As the old TPSs were written for a specific tester the DoD must often create new TPSs, often with only limited technical information on the UUT, to maintain test capability. The cycle continues until the weapon system is retired, often 40 or more years after the system was first deployed.

The cost of having to recreate the data and test code for a UUT, added to the inability to reuse test code for testing similar UUTs, has created strong
interest in promoting industry software and data standards that reduce the cost of test. An IEEE standardization effort known as ABBET is actively working toward lowering the barriers that prevent the simple passing, manipulating and reusing of all forms of test related information.

ABBET specifies the interrelationships between different forms of test related information (design, test strategy, test requirements, test specifications, maintenance and diagnostic information, etc.) and by defining the interfaces to these forms of information, it should be possible to move information from one form to another (design->test) more easily. By creating UUT test specifications (TPSs) that are tester independent, coupled with tester resource descriptions that the test exec can match against TPS requirements, TPSs can be moved between testers with less integration (cost) required. By creating standard software interfaces to test instruments to make similar instruments accessed, controlled and used in the same way, ATS integrators are free to change instruments as needed without rewriting the TPS. Essentially, ABBET will reduce the cost of test by reducing the complexities of test. These complexities were caused by multiple proprietary information formats, test languages, instrument drivers and incompatible test approaches.

It should be noted that ABBET is not an all or nothing proposition. The standards under ABBET (interfaces, services, and information formats) can be taken together or separately, wherever they can be used most effectively. ABBET is also language independent, so that the interfaces and services can be implemented in almost any language. The intent is to enable, through the use of open standards, rather than to control industrial ingenuity in alternate solutions.

CONCLUSIONS

One of the major limitations of testing electronic components is that it requires a logical, knowledge intensive approach. This is a limitation because people think in an intuitive, vertical manner. Rather than playing to our strengths, computers (and ATE) have forced us to operate in the realm of our weaknesses; details are demanded while we tend to generalize facts, keyboard/text interaction is required of beings who process data visually (spatial interrelationships) and orally (speech). Eventually, after the person obtains the training and experience needed to alter their behavior to adapt to the computer, they are moved somewhere else, starting a new cycle of frustration and inefficiency.

People have a tendency to learn and operate in a multi-modal sensory world. Computers have traditionally forced us to rely on textual interchanges, a "limited bandwidth" form of communication. Graphical object oriented interfaces, combined with speech generation for perception, would allow users to interact with computers (and ATE) on multiple levels.

Not only should the ATE adapt to our ability to fuse various sensory inputs (i.e., visual and auditory) to create a better interpretation of the available data, but it should also be able to perceive and integrate information beyond human sensory abilities. Thermal and X-Ray imaging, electromagnetic field detection of the unit under test provide valuable clues to the ATE as to the cause of failure. The fusion and interpretation of extended sensory information into the ATE augments system accuracy, and usability, particularly for users with less experience. An extension of this graphical interface to a "virtual reality" could complete the transition from a totally logical, "left-brain" approach of problem solving to one of a balanced, "whole-brain" approach, blending logic with our ability to derive solutions. Interactive autonomous machines using the most efficient processes to communicate information will improve productivity in the future.

Adaptive Machine-based Intelligence (Neural Networks, algorithms) systems coupled with automated learning can help the user track and
utilize all available data needed for decision making, and can emulate a human's ability to extrapolate from the partial data available to reach conclusions. Because the expertise in diagnosing faults in a system resides largely within the test equipment, turnover of trained personnel does not significantly impact mission effectiveness.

Finally, we the Air Force depend on thousands of special purpose, unique pieces of test equipment, in which we cultivate "experts" to use and maintain. When the experts leave, the system fails and our mission is impacted. To counter this, we must move away from those factors which create specialized ATE systems; proprietary hardware, software and data structures that impede easy maintenance and modifications, and steepen the user learning curve. Standardized and open hardware and software interfaces, plus common data formats including inter-operability make it possible to mix and match equipment sets to meet testing requirements. In particular, the current power of the PC (486) will make it possible to replace large, expensive, proprietary ATS with equivalent or greater functionality in the near future.

The Air Force is looking at the long range costs of new ATS, including training requirements, effectiveness, and upgradability. These costs can be lowered by the acquisition of systems that are intuitive to use, intelligent, with open software and hardware for easy maintenance and upgrading. With the do more with less philosophy now in place, we can not afford to do otherwise.

REFERENCES:
2. R. Glenn Wright, ARPA/GMA Industries - Knowledge-Based system interoperability for Sensor-Based Processing Applications.
TESTER INDEPENDENT TISSS TOOLS (T2)

There are five tools represented in Figure 4, two of which are completely tester independent, one that is partially tester independent, and two that are tester specific. The tester independent tools are the VHDL translator tool (noted as T2-1 and partially independent), the TRF/TDL translator tool (noted as T2-3), and the IN-STEP tool (noted as T2-4). The part of the T2-1 tool that is tester independent reads the VHDL structural file, and automatically creates a simulation test bench for the model. The test bench is also compatible with the WAVES dataset. The generated test bench has been designed to capture stimulus and response information for all external nodes as well as internal nodes on the board as output. Further, if the incoming WAVES vector information is cyclized, the cycle time information is retained in the captured test bench output. Details of the test bench are provided in Reference [4] and therefore will not be discussed here. The VHDL model that is read and parsed by the T2-1 tool currently must adhere to the following specification:

The tool expects each model to contain, at most, a single instance of the following blocks of code:

```
BLOCK1: entity
  ...
  ...
  end

BLOCK2: architecture
  ...
  begin
  ...
  end

BLOCK3: configuration
  ...
  ...
  end
```

Any legal order of the above blocks is allowed.

Only signals of type 'std_logic' or 'std_logic_vector' will be allowed in BLOCK1 and BLOCK2 above. These signal types depend on using IEEE Std 1164, a nine state logic system developed for use with VHDL.

It is also assumed that the incoming VHDL model has been verified as a "working" model. The other output of the T2-1 tool that is tester independent is the file containing the signal input/output list and directional information.

This information is used later in the process flow to determine which signals in the test bench output file are external and the direction of those signals.

The other tester independent tools are the TRF-to-TDL tool (T2-3 in Figure 4) and the IN-STEP tool (T2-4 in Figure 4). The T2-3 tool takes a completed TRF and creates a test description file and a pins file in the TDL format. The test description file contains information on board voltage and current levels, timing information (if relevant to test), and basic test requirements information such as opens and shorts testing. The pins file contains information on the board connector pins and the board-level signals each pin is mapped to. These two files are among the major inputs to the IN-STEP tool, T2-4. T2-4 contains a parser program for TDL. Once parsed, T2-4 can be programmed to extract relevant information from the pins.
file and TDL to populate tester specific template files that are user created. The other input to the T2-4 tool, but not shown in Figure 4, is a file containing board-level test philosophy. In this case, test philosophy describes the different kinds of testing, including qualification testing, that could be performed, if desired. The test philosophy files are currently standardized and do not need to be re-created for each TPS. An example of how T2-4 works is shown in Figure 2. This example shows how a pins definition file is created for a Teradyne J953 component tester.

TESTER SPECIFIC TEST AUTOMATION TOOLS

The tools that have been developed to provide GenRad specific test files are T2-1 (partially tester specific), T2-2, and T2-6.

The T2-1 tool has routines built in that use the information contained in the various inputs to this tool to create the GenRad .ADS, and .CCT files. GenRad's .ADS (for Assembly Data Set) format is essentially a netlist file that describes each pin on each device on the board and identifies the board signals the pins are connected to. Other information in the .ADS file includes definition of the power signals (voltage source, current source, and ground), and identification of the external board signals and their directions. One of the inputs used to create the .ADS file is the .TBL file, which is a look-up table that contains a listing, for each component on the board, of generic component name, pin number, and direction. The current version of the T2-1 tool creates most of this look-up table from information contained in the VHDL file. The only remaining information to be added by the user is the pin numbers.

The table was necessary because the component pin number associated with each VHDL signal name is required for the .ADS file. This information is typically not contained in the VHDL model.

The T2-2 tool currently reads the output of the test bench and creates the three GenRad files designated as .TGO, .CAP, and .CTX in Figure 4. The .TGO file contains the information needed by the ATS to perform an "edge connector" test of the board, including cycle times and pin groups for drive pins and sense pins. This information is automatically determined from the "generic" output of the test bench. The tool determines which signals can be grouped into drive and sense sets. Further, if the vector information was cyclized in the WAVES formats prior to simulation, the tool also retains this information and passes it to the .TGO file. The .CTX and .CAP files are used by the GenRad Genesis tools to build a probing database for diagnostics. For further information on these files and the rules for their construction, see Reference [7].

The final tool that is GenRad specific is T2-6. This tool is noted as the TGO-TO-WAVES translator. This tool was developed to take the vector information contained in the .TGO file and convert it into an external file in the WAVES format. This tool is needed to handle situations where the incoming vectors (created via simulation) violated any of the rules for .TGO construction, and therefore, can not be run on the tester. One such rule is that the GenRad 2751 permits only eight drive and eight sense sets to be used (i.e., only eight groupings of pins that are either driven or sensed at the same time within each cycle are permissible).
There may also be timing problems that interfere with optimal diagnostics. In such situations, the vector set must be manually manipulated by the test engineer to achieve a desired result, or to make the vector set compatible with the tester. Once such manipulation is accomplished, the model must be resimulated using the modified vector set. The T2-6 tool converts the vectors in the GenRad .TGO format into a format that is WAVES compatible. A new WAVES dataset has also been developed to match the new WAVES external file, and the process shown in Figure 4 is repeated.

PROGRESS

A digital test program set (or test module) for a relay driver card has been successfully integrated on the GenRad 2751 using the automation process defined in Figure 4. The relay driver card represents a re-host example, as a test program for this board did exist for the MATE-390. Therefore, an existing vector file was available. However, it served as an excellent test case for verification of the total process (i.e., capture and simulation of the design in VHDL and subsequent porting of this information to the tester).

Some adjustments were made to both of the tester independent processes, primarily modifications to the VHDL test bench, and some have been made in the tester specific tools (primarily T2-2). One of the problems with the testbench related to the need to capture the event when a signal changed from a drive value to a sense value of the same logic level (or vice versa) within the same vector cycle. There was also a problem with the need to maintain signal order when going from the testbench vector file to the GenRad specific .TGO file. However, all of these problems have been solved. In addition to these changes, the .CCT files created by the T2-1 tool needed additional modifications before they could be used for testing. Further, not all information available from the HILO simulator can currently be recreated with this process. First, it was not possible to grade the test vectors for fault coverage, as fault simulation is not yet available in the VHDL environment.

Second, the .CAP file, used to create the probe database for diagnostics, is not 100% complete. For bus nodes that can have multiple drivers, it is advantageous for guided probe diagnostics to know which devices are driving the bus node at any given time. Such knowledge allows the guided probe to track down problems without "blindly" having to check all inputs to each device that could be driving the bus node. This information is usually part of the .CAP file, however, it cannot currently be captured using the VHDL testbench developed under this project. This is because within the current VHDL test bench, whenever multiple pins are driving a node at the same time the value for the node is resolved to a single value based on the IEEE Std_1164 resolution function. It is not currently possible to capture the pre-resolved information within this resolution function. Further research into this problem is planned. However, this does not mean that a complete TPS cannot be developed; it indicates only that the ability to perform diagnostics is somewhat diminished.

In addition to the relay driver card, an Input/Output Control (IOC) board will be tested on both the GenRad and MATE-390 ATS, using test programs developed via the test automation process described.
LESSONS LEARNED

There were a number of different problems that were solved or investigated during this project. A list of problems addressed is provided below:

- How should the TDL format be modified for board-level test?
- How can VHDL be implemented for test development?
  - How can internal node data be captured from a board level structural VHDL model?
  - How can WAVES be implemented to enhance the design-to-test implementation in TISSS?
  - What information needs to be added to the VHDL design description for test purposes?
- What is the minimum number of GenRad specific files needed to create a working TPS?
- What is the minimum number of MATE-390 specific files needed to create a working TPS?
- Can creation of the VHDL test bench be automated?
- How can "print on change" vector data be cyclized?

While there were other problem areas that were investigated and remain to be solved, the above list were the primary ones. The first problem arose because TDL was originally designed for component-level testing, which has more detailed requirements, in general, than does a board. For instance, timing tests, voltage and current tests, and logic integrity tests are the three main elements to digital components testing. Further, TDL was also designed to capture component test qualification information, such as testing at more than one temperature. To modify TDL for board level test, the basic testing that is accomplished at the board level needed to be defined. Basically this amounted to opens and shorts tests in addition to functional and diagnostic testing. Although timing tests and voltage and current testing is sometimes done at the board-level, the TDL formats to be used most often do not include them, and are therefore simpler for a board. The second problem, "How can VHDL be implemented for test development?" was probably the most critical. The main points of investigation into this question are those listed above. The sub-problem listed initially arose because the board-level VHDL model was essentially like a single component, with only inputs and outputs available from the test bench. To be able to create a test program that enables the development of a diagnostic database, internal node data need to be captured. If the board-level VHDL model is used, then the internal nodes cannot be made visible to the testbench, which prevents capture of this information. Only external I/O can be captured. However, if the model is built from the components within the testbench entity, then all signal paths within the board-level model are visible to the testbench, and can be captured. Therefore, the initial VHDL models developed were modified to enable internal node data capture. The next sub-problem, having to do with WAVES, required first learning what the WAVES formats meant, and how they could be integrated with VHDL. WAVES was designed to enable a standard documentation format for simulation and test vectors to be created. Information such as timing and tolerance values are included as part of WAVES. In addition to the documentation aspects, WAVES is a subset of VHDL, and can therefore be integrated with VHDL simulation. To
describe the WAVES/VHDL integration process would be too lengthy for this paper. However, it can be said that making WAVES part of the overall process was felt to be a plus, as this added one more open standard format to the process. More detailed information on WAVES can be found in [8], and in IEEE Std 1029.1-1991. The final sub-problem listed above deals with an apparent disconnect between the design simulation world and the test simulation world. Many VHDL models do not require that power signals, or external board connector information be present for design development and verification. These VHDL models are designed only to emulate the functionality of the logic. Information needed for physical device emulation, such as power supplies, are needed for test, but not for design. The test world requires some physical representation as well as functional, or behavioral. Physical information such as what pins are connected to the external signals and determining which signals each pin on each board component is connected to are other examples required for test, but not necessarily for design simulation. Most of these problems were solved either by adding the information directly to the VHDL model, or by providing it as an external file input to the appropriate TISSS tool.

The next problem arose when the decision was made to bypass the tester specific simulator and use VHDL. An existing TPS, used as a baseline, contained approximately 1,000 files. SA-ALC test engineers knew that not all the files were necessary to create a "working" TPS, and that the complete TPS could be recreated from a subset of these files. Therefore, an investigation was performed to determine the minimum set of files required to create a valid GenRad test program. This resulted in identifying only about 25 files! Many of these files are very simple and never change from TPS to TPS. The primary files that required additional knowledge of their content and format are those listed in Figure 4. This exercise proved to be very useful in learning more about the specific details of each tester and how the software environments worked. The same process was accomplished for the MATE-390. The next problem, automation of the VHDL test bench creation, seemed very feasible primarily because all of the VHDL code developed for the structural description of the board is used in the test bench. Therefore, it was more or less an exercise in determining what additional information needed to be created for a working test bench that produced information needed for test.

A great deal of effort went into making sure that the test bench remained generic, and did not contain any processes that were inserted to target the information for either of the two testers. Details on the test bench and the modules within the T2-1 tool that create it can be found in Reference [4].

The final problem listed above, "How can "print on change" vector data be cyclized?" is discussed in the next section.

FUTURE WORK

At the time this paper was being written, the automation path for the MATE-390 was being developed. Figure 5 presents the flow diagram for the MATE-390 tester. Many of the tools remain the same for the tester independent information. The only new tool developments are noted by the shaded boxes. At the input to the VHDL simulation, the vector information that is
captured in WAVES will be converted to a fixed cycle time of 1000 ns. Unlike the GenRad 2751, which can apply a rich set of waveforms within variable-length cycles, the MATE-390 can apply only fixed values within fixed-length cycles. Therefore, it was necessary to "flatten" WAVES data sets to accommodate this restrictive timing model. The problem was solved by using a VHDL test bench to simulate the incoming vectors captured in the WAVES format. This test bench, shown as the "WAVES FLATTENER" in Figure 5, is designed to ensure that no minimum relative timing constraints are violated. It is important to note that the vectors are flattened before board model simulation, to guarantee the integrity of the probe database. Other tool development for the MATE-390 include adding a module to the T2-1 tool to produce the .CKT file, and building a translation tool for the vector files noted in Figure 5 (i.e., .STIM, .RES., .MSK, etc.). The TDL files and tester
specific templates for the MATE-390 have been completed, but not fully tested at the time of this writing. For the GenRad 2751, work is continuing on enhancing the $T^2$-2 tool. In particular, the module that creates the GenRad .TGO file will have some capability to cyclize vector information produced by the VHDL test bench for situations where no previous test program exists, and the vectors are not yet cyclized for ATS. Even though this enhancement is not expected to create a legal .TGO file for some cases, it will be a first pass that makes the test engineer's job easier in terms of modifying the vectors to fit within the constraints of the GenRad tester. Once vectors are modified to meet tester resource allocation constraints, they must be resimulated to provide an accurate nodal file that is used to re-create a new .CAP and .CTX file. This is made possible by use of the $T^2$-6 tool shown in Figure 4 and described previously. This is a major advantage, as during test integration it is almost always necessary to alter simulation models to emulate "real world" implementations. Without the TISSS tools described, any minor .TGO manipulations would have to be painstakingly traced back to the simulation stimulus file and altered manually. This is never an easy process and leaves much room for human error.

SUMMARY AND CONCLUSIONS

The current TISSS effort is significant in that it has proven out the ability to capture design information in non-proprietary, open standard formats and then create information necessary to develop test program sets. The information standards used are VHDL, WAVES, and TDL (TDL is not a formal standard, however). Further significance of this demonstration lies in the fact that use of the VHDL and WAVES standards for test development is a goal of the "A Broad Based Environment for Test" (ABBET) program. The prototype tools developed under the described effort are a first step in enabling the test engineer to take information captured in standard formats and successfully develop test program sets. This represents potential savings in that the test engineer no longer has to develop new models for a tester specific simulator, nor does he have to translate a vector set captured in WAVES before it can be applied in simulation.

Lessons learned on the project included understanding the differences in the design simulation models and test simulation models. Further, a great deal of time was expended on fully understanding tester specific formats, as the information captured in VHDL, WAVES, etc. as well as outputs of the VHDL test bench, still had to be translated to tester specific formats. Information captured in non-proprietary formats does not alleviate the need to know the tester. Nor does it eliminate the need to translate information from non-proprietary to proprietary. There is still the need for translation tools such as those developed under the described effort.

Finally, because this project demonstrated the usefulness of VHDL and WAVES for test development, other programs, such as Virtual Test (V-TEST), ABBET, the Navy's Technology Independent Representation of Electronic Products (TIREP) program, or any current and future efforts that rely on capture of information in VHDL and WAVES, now have a means to develop test programs directly from this information. While this effort is only a starting point, it represents an excellent baseline of tools and
procedures to build on to solve future problems.

REFERENCES


