RESEARCH INITIATION PROGRAM (RIP)

Sponsored by the

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
Bolling, AFB Washington, D. C.

Conducted by the

Universal Energy Systems, Inc.

FINAL REPORT

Comparison of Testability Analysis Tools for USAF

Prepared by:	Dr. Sundaram Natarajan Bradley K. Herman
Academic Rank:	Associate Professor Graduate Student
Department and	Center for Electric Power
University:	Tennessee Technological University
Research Location:	Tennessee Technological University
Date:	February, 1991
Contract No:	F49620-88-C-0053/SB5881-0378

COMPARISON OF TESTABILITY ANALYSIS TOOLS FOR USAF

by
Dr. Sundaram Natarajan
and
Bradley K. Herman

ABSTRACT

As electronic systems become more complex, testability is becoming an increasingly important design consideration. To incorporate testability properly during the design phase, an appropriate testability analysis tool must be selected. In this report, we address the necessity for including testability in the design phase and develop a set of criteria to compare testability analysis tools. We analyze several tools with respect to the established criteria. Finally, we conclude with the summary and the recommendations regarding the selection of a suitable testability analysis tool for design for testability (DFT).

Acknowledgements

We thank the Air Force Office of Scientific Research for sponsoring this research. We also acknowledge Universal Energy Systems and the Center for Electric Power for their help and concern in all the administrative aspects of this research.

We also wish to thank thefollowing individuals for contributing their time and products to our research: Mr. James Patterson of Automated Technology Systems corporation for providing I-CAT, Mr. Melvin Nunn of Naval Ocean Systems Center for providing CAFIT, Mr. Ralph DePaul of Detex Systems, Inc. for providing STAT, Dr. Randy Simpson of Arinc Research Corp. for the work associated with STAMP, Mr. Gene Biagi of BITE, Inc. for providing ASTEP, and Mr. Dean DeMeyer of WRALC for providing the circuits. Without their cooperation and contributions, this research would not have been possible.

I. INTRODUCTION

Recent USAF programs have sought to address the enhancement of both built-in and off-line testing of new systems early in the acquisition phase. This requires the inclusion of testability concepts in the design phase. It has the goal of eliminating interim contractor support (ICS) and thus reducing the life-cycle cost (LCC) of the system. In addition to reducing the LCC of the system, the incorporation of testability in the design phase also increases the reliability, maintainability, and availability (RM&A) of the system. The potential monetary savings and the increased quality of the system make the inclusion of testability in the design phase very attractive. Testability must be included at the lowest level of design to be effective. For electronic systems, this means that the testability criteria should be applied at the circuit card assembly (CCA) level. This is becoming increasingly more important with the increased CCA complexity of modern electronic systems. The task of including testability in the design phase of the CCAs lends itself to computer-aided design (CAD) tools. To ensure an accurate verification of testability requirements, a proper CAD tool must be chosen that is suitable for the job at hand. The previous studies of testability analysis tools have dealt mainly with system level analysis, but not with the board level analysis required by modern electronic systems. The other board level studies were performed before most of today's CAD tools were even in existence. Also the growing number and added maturity of CAD tools requires a new study to be performed.

In this report we first establish the objectives of our research and provide support for the research in the form of an overview of the design for testability philosophy. We then state and support the criteria and method by which we evaluate each testability tool. A section is devoted to each tool that briefly describes how each software meets the established criteria. These descriptions should not be considered complete overviews of the tools. For details beyond the scope of this report, any interested individuals should contact the individual vendors. We conclude with a summary of the research and our recommendations.

II. OBJECTIVES OF THE RESEARCH EFFORT

The objective of our research was to analyze and compare several testability analysis tools as to how they perform a board level testability analysis of electronic systems. This involved evaluating several aspects of each tool from ease of input to clarity and accuracy of the outputs. Another main goal of this research was to determine one or more Figures of Merit (FOM) that can be used to set testability requirements. Since there is no standard method for obtaining an FOM, the resulting FOM from each tool is unique. The proper FOM mould be one which describes the testability of a CCA in the fewest possible parameters with the most insight into its testability. To get an accurate evaluation, the CAD tools were tested with a variety of circuits, namely digital, analog, and hybrid circuits. For our study to be realistic, it was carried out using actual CCAs selected from the E-3 AWACS aircraft.

III. DESIGN FOR TESTABILITY

Testability addresses the issues involved in achieving the inherent ability in a system to be testable so that desired levels of fault detection and isolation can be achieved in an accurate, timely, and cost-effective manner. The testing of an item can be influenced by the physical design of the item, the electrical design of the item, and procedures and equipment used to test the item. With the introduction of sophisticated test equipment, lacking physical designs can be overcome with guided probe and bed-of-nails techniques. However, this becomes ineffective in test cost and time required to test. Even with the sophistication of the test equipment, the system must still be designed in such a manner that it is testable. This is known as being inherently testable. As systems become more costly and more complex with higher part density on each CCA, the ability to test and maintain these systems becomes increasingly difficult. Only if a system is highly testable can confidence be placed in the fact that it is fault free. Once a system is known to contain a fault, the ability to isolate the fault is directly related to the inherent testability of the system. The testability of the system is one of the leading factors influencing the LCC of the system. This is because the maintenance of a system is influenced by its testability and it can account for 50% of its LCC.

The most common method of system design places the test engineer at the end of the design phase. Typically, at this point, the hardware has been designed and is at some point of production and the test engineer must find a way to improve the testability of the system through probe tests and other intrusive methods. This method is time consuming and quite often the test requirements cannot be met due to a lack of inherent testability of the design. The decision must then be made whether to redesign the hardware or relax the test requirements. Since the hardware is being produced in some fashion at this point, the redesign of the hardware would be very costly and time consuming. On the other hand, the relaxation of the test requirements would reduce the testability of the system and thus increase its maintenance cost and hence its LCC. The most logical solution to this is to include testability in the initial design phase so that test requirements can be met before the system enters the production phase. The test requirements can then be treated as additional design requirements for the system. This is called design for testability (DFT). Addressing testability issues in the design phase enhances the testability of a system thus reducing its LCC.

As the complexity of the system increases, the ability of a designer to incorporate testability by the old time-consuming manual methods becomes impossible. Thus MIL-STD-2165 becomes more of a guideline than a method of analyzing testability. At the early stages of the design phase, the designs can change on a daily basis. Therefore, some method of incorporating these changes rapidly into the testability analysis becomes a necessity. This problem lends itself quite well to computer-aided design (CAD) systems. Therefore CAD tools have become necessary for proper testability analysis. There are several such tools available. Some of the popular ones are: CAFIT, I-CAT, STAMP, ASTEP, and STAT. Our main concern was to compare these tools for the USAF. ASTEP was considered in the original proposal, but was removed from this study by request because it is primarily a system level testability analysis tool and it is not usable for testability analysis at the CCA level.

IV. CRITERIA FOR TESTABILITY ANALYSIS TOOLS

The first criteria to evaluate was the input requirements of each testability tool as to how much effort is required to prepare the design information in a form suitable for the software. This involved evaluating the modeling method used for each program. Other aspects such as the flexibility of the modeling procedure and the time required to do the modeling should also be included in the criteria. The next criteria evaluated was the ease with which the modeling information can be entered into the program. This included the user interface as well as the form in which the information is presented back to the user. The data should be presented in such a way as to allow the user to debug the information quickly. The error checking ability and the clarity of the error messages presented to the user are two criteria that must be addressed for tasks of the size being considered (i.e. large CCAs such as the ones we used). The run-time of the programs is also very important since one would not wish to wait overnight for the analysis of a single CCA.

The primary criteria for the evaluation of the testability analysis tools is the information in the form of a FOM from the output of each tool. The output should be concise for the quick assessment of the inherent testability of the CCA. It should also be detailed enough to provide insight into the details of the testability in a specific portion of the CCA. This information should include such aspects as the various ambiguity groups, feedback loops, types of testing strategy to apply, etc. In addition to the quantitative outputs, some qualitative outputs such as suggestions to improve the testability are also desirable. These include where to break feedback loops, where to add test points, test points to eliminate, and any other suggestions.

In addition to these criteria, some other aspects of the individual programs were also evaluated. These were primarily features regarding the testability analysis unique to a particular program.

Finally, to summarize the evaluation criteria, they are in the following order of importance:

(1) A concise FOM yielding a measure of testability
(2) Suggestions to improve testability through the breaking feedback loops
(3) Suggestions to improve testability through the addition or elimination of test nodes
(4) Information on ambiguity groups and feedback loops
(5) Testing strategy or procedure
(6) Data input requirements
(7) Amount of modeling required
(8) User interface.

V. METHOD OF EVALUATION

Our primary concern in the evaluation of the testability tools was how the tools meet the criteria established in the previous section. The three circuits chosen were selected to give a feel for how the tools met the criteria on a variety of real-world electronic circuits. Another criterion that was used in selecting the circuits was the experience of the WRALC personnel in developing the depot for these circuits. The actual numerical results are of little consequence since each circuit, that any user will analyze, will have its own unique characteristics. How the tools met the challenge of evaluating different types of circuits was of a greater importance. Since every circuit will yield unique results, the numerical results themselves cannot be used to evaluate the tools. Each tool was scored on a scale of 0 to 10 as to how effectively each of the criteria was met. Each criteria was assigned a weighting factor based on its importance to design for testability (DFT).

The first criteria is the most important since it characterizes and summarizes the testability of the CCA and it received a weight of 10. The second and third criteria were both given weights of 9 since they are the primary means of improving testability through design changes. The fourth criteria was assigned a weight of 8. This is because the detailed information can be used to improve testability even though no direct suggestions are made. The fifth criteria is more limited in the information that it can provide with regard to the design procedure so it received a weight of 6. The last three criteria do not deal directly with the testability analysis and the design of a CCA. However, they are still important because even an effective design tool may not be practical in its use. Thus, each of the last three criteria was given a weight of 5.

Once all the scores were assigned, each tool had its weighted total score calculated from the assigned scores and weights. Any evaluation is bound to be subjective, but we believe that our independence from any government agency or private corporation limits the subjectiveness as much as possible.

VI. CAFIT

CAFIT stands for Computer-Aided Fault Isolation and Testability Model. CAFIT was developed by ATAC of Mountain View, CA. It is a government owned testability analysis tool. CAFIT was designed to enhance the testability of electronic circuits, both digital and analog. It provides outputs, after performing several forms of analysis, that aid in designing circuits to be inherently testable. For this study, CAFIT was run on an IBM-PC AT compatible. It requires an EGA compatible monitor and occupies approximately 2 Mbytes of hard disk space. An additional 1.4 Mbytes of disk space is taken up by the library, but this will vary depending on the number of device models used. It can also be run on Mentor-Graphics workstations.

CAFIT provides a library utility that builds and maintains a model library of parts. This library contains all of the logic, attribute, failure rate, and pin information for each device in the library. The user can modify and add devices easily to the library with the help of the reference manual. Digital devices can be entered easily from the data sheet of most devices with slight modifications into the format required by CAFIT. For the digital devices, the model contains the actual function of the device in terms of HIGH, LOW, TRI-STATE, etc. For analog devices, no functional information is used. Thus the analog model is purely topological. This mixture also allows for the modeling of hybrid components such as D/A and A/D converters. Care must be taken when using analog devices to ensure that the proper dependency model is used.

CAFIT is designed to accept schematic information from a netlist provided by a CAE package such as the CT-2000 software package from Case Technologies, Inc. If such a package is not available, the connectivity information from the schematic can be entered into CAFIT through an ASCII file in the proper format. We used this method since the CAE tool was not available for our use. CAFIT takes this information along with the dependency model information previously entered to do the testability analysis. If the CAE system is available, results are shown in color format on the schematic. Otherwise the output files provide the means to view the results. Input for the ASCII file is easily obtained directly from the schematic and this does not require any special knowledge of the inner workings of the circuit if the circuit is purely digital. However, if analog components are used, then the knowledge of the operation of the circuit is essential to get the proper topological models. The only constraint on the labeling of nodes is in the labeling of test nodes, input signals, and output signals. Test node labels must have a TS prefix while signal input and output signals must have a SG prefix. Overall, CAFIT requires very little preparation of the input information if the models have already been created within CAFIT. This will most often be the case once a library of devices has been established.

Performing a basic testability analysis with CAFIT for an existing design is relatively easy. A basic analysis attempts to locate structures inherent in the design that will inhibit the testing of the design. These structures include negative reconvergence, logic redundancy and, most importantly, the information on feedback loops present in the design. Negative reconvergence is a design condition in digital systems where a portion of logic prevents an input from controlling an output. This condition is not necessarily a design flaw, but it can seriously impede the testing of the circuit. Logic redundancies, which inhibit the testing of the circuit, are also identified. With logic redundancy, the observability of the circuit is reduced. The feedback loop portion of the report is probably the most useful. This provides a detailed output listing of all the components involved in each loop as well as nodes at which to break each loop. The next type of analysis provided by CAFIT allows the user to select parameters about the test nodes and perform a detailed testability analysis with regard to signal controllability and observability.

If the test nodes have already been selected, an analysis can be done to determine signal coverage with only these nodes. The signal coverage of a device is a measure of whether or not stuck-at faults in device can be tested for. This is a function of controllability, observability, and the number of tests permitted. Controllability and observability figures using the signal input and output and test sites are given along with the signal coverage for each device. In addition to the selected test nodes, an additional set of nodes can be chosen by the user. CAFIT then selects the nodes that will yield the highest signal coverage. If 100% coverage can be achieved with fewer test nodes, then the lowest number of nodes are chosen. The test node selection is probably one of the most powerful aspects of the analysis that CAFIT provides. This allows the user to experiment with various configurations to determine the optimal number of test nodes for a given CCA. The number of test signals is an additional parameter that can also be varied. This constraint is becoming less prevalent as sophisticated test equipment becomes commonplace. This is also more applicable to digital circuits where CAFIT has the knowledge of the various states of the circuit from the model description. Analog circuits may require only one or possibly several tests to be performed on a single node, depending on the circuit.

The outputs from CAFIT can be used for the selection of test nodes, the selection of points to break feedback loops, and the estimation of controllability and observability. However, most testability requirements are in terms of fault isolation parameters and ambiguity group sizes. Even though a node is controllable and observable, it may not have a unique test that identifies only that node, leading to the creation of ambiguity groups. This is especially true when dealing with feedback loops. The entire loop is an ambiguity group even though each device may by controllable and observable. This is especially important in the testing of analog and hybrid circuits where feedback networks are invariably used. The information provided through controllability and observability is not enough to describe the testability of an analog or a hybrid circuit thoroughly.

The user interface for CAFIT is one of the easiest to use and understand. It is designed in such a way that even an inexperienced user can use the program. On screen help may be called up throughout the program. CAFIT provides information on the screen without cluttering it up. The error messages provided were descriptive and aided in the debugging of the various circuits tested. The only shortfall in the user interface is having to prepare an ASCII file if the CAE system is not available. It is acknowledged that CAFIT was designed to be used with an accompanying CAE system, but a provision for entering the schematic information directly into CAFIT and having CAFIT manage the files would have been helpful. CAFIT also writes the output data to one file only without the option to change the name of the file. Thus, after each run, it overwrites the existing file making it necessary to continually rename the output file after each run. This is not a serious drawback, but it can be frustrating after waiting 45 minutes for results and then realizing previous results were accidentally overwritten.

CAFIT is primarily geared toward the analysis of digital circuits. It can analyze analog and hybrid circuits also. However, the terminology used by the program and the user's manual suggest that the analog portion has been added as an afterthought. The lack of fault isolation and ambiguity group parameters limits the usefulness of the output to verify whether the desired levels of testability are being met. However, the test node selection and feedback loop breakpoint analysis make CAFIT a tool for choosing test nodes and making changes in the circuit to increase its inherent testability.

CRITERIA and SCORES for CAFIT:

(1)	5:	The controllability and observability figures alone are not enough to fully characterize the testability of a circuit.
(2)	8:	The suggestions for selecting feedback loop breakpoints are essential.
(3)	8:	The time spent selecting appropriate test nodes is greatly reduced with CAFIT suggesting the most effective test nodes.
(4)	3:	The information on feedback loops is helpful, but there is a total lack of information on ambiguity groups.
(5)	0:	No outputs concerning testing procedure.
(6)	5:	The existence of the interface with the CAE tool can make data input simple. However, the alternative of having to format an ASCII file with all of the circuit information is tedious.
(7)	6:	Minimal modeling is required once a library of devices is established. However, modeling of analog circuits must still be done on an individual basis.
(8)	8:	The user interface is simple and easy to use and has plenty of on-line help.

VII. STAT

STAT is a product of Detex Systems, Inc. of Orange, CA. STAT, along with its predecessor LOGMOD, is a testability analysis tool designed to enhance the testability of systems. This applies to the enhancement of the testability of new systems through their design and for the maintenance of existing systems. Our study will focus on the design aspects of its capability. STAT runs on a variety of PC platforms, but a minimum of a 286-based PC is recommended with a sufficiently large hard drive to store the STAT program and all of the user's databases. It requires approximately 2.5 Mbytes of disk space for the program itself. The basis for the analysis that STAT performs is the dependency model information supplied by the user.

A dependency model is one that simply shows the relationships (dependencies) between the various nodes and items in a system. This can be interpreted as describing the information flow within a general system and as a signal flow for an electronic system. The topological information from a circuit is entered into STAT completely in the form of dependency models, not as the connections between the actual electronic components. This type of modeling has advantages as well as disadvantages. The main advantage for electronic circuits is that it bridges the gap between the descriptions of digital and analog circuits. Another advantage of the dependency modeling for electronic circuits is that it provides considerable flexibility to describe a particular circuit. This includes what aspects of a particular signal are important as well as the level of modeling for each portion of the circuit. For example, it allows the user to choose whether or not to consider packages containing more than one device to be considered as individuals or as a single item. Being able to describe the particular aspects of one signal as individual signals is particularly in analog circuits where biasing and actual signal information must be considered separately. The disadvantage of this type of modeling is that it requires an intimate knowledge about the operation of a particular circuit in order for it to be accurately modeled. Modeling a complex analog or hybrid circuit can be a time consuming and frustrating task even when the circuit is familiar to the user. However, the proper modeling up front will help to reduce future problems and the time required to perform a proper testability analysis using STAT.

The entering of the dependency model information into STAT is very straight-forward. The information is entered in an interactive fashion with STAT updating and maintaining the databases transparently. Each node in the dependency model is entered as a test node. One aspect of STAT, that is particularly frustrating, is that all node labels must have a 'T' prefix and all item labels must have an 'I' prefix. This requires the user to maintain some type of translation table independently. Being able to use descriptive labels would aid in interpreting the results of the testability analysis. Descriptions can be added to the dependency information as it is entered. The dependency model information is displayed back to the user in a manner that allows errors to be seen visibly and corrected. Once the basic dependency information has been entered, the model can then be processed to find and correct any additional errors. The error messages supplied are descriptive and they can be corrected easily.

The evaluation of a model with STAT is not a one step process, but rather a series of processes. When a model is first processed, all nodes are considered as test nodes. This becomes the base model for all subsequent evaluations. This is the point at which STAT allows the user to perform a variety of "What if?" type scenarios. This is done by first creating a new model that is identical to the base model with a simple keystroke. The user can then modify the characteristics of each node within the model. Most nodes will be chosen as non-test nodes. However, the aspects of the chosen test nodes can also be further modified. The type of access required to reach a node, such as external or probe accessible, can be specified. The cost and time to test each point can also be specified. The failure rate and replacement time and cost of individual items can also be modified. Other factors can also be modified and weighted. The weighting allows the user to perform an analysis based on individual specifications.

One aspect of the operation of STAT deserves special mention. Along with identifying feedback loops within the model, STAT has an interactive feature called the

Feedback Loop Breaker. This feature identifies feedback loops, then recommends the nodes at which to break the feedback loops and the new characteristics resulting from the breakage. The resulting ambiguity groups can be viewed immediately. This allows the user to continue to modify the loops until the required ambiguity group requirements are met. This is just one aspect of a very workable user interface. Once all the terminology associated with STAT's modeling is understood, it is easy to maneuver through many screens and perform a testability analysis.

The outputs from STAT take on a variety of forms that range from the concise and informative to the long and drawn out. The longer outputs are more applicable to the maintenance aspects of testability and not to the design process. Each output report contains a summary section and a detailed section. This is very helpful when trying to locate a particular piece of information without turning through perhaps hundreds of pages. The management model report contains a graphical representation of the topological dependencies for a particular model. This allows the user to more clearly see the loops that are not easily identifiable in the original circuit diagram. The feedback loop indicator report contains a detailed information on all feedback loops in a model. This includes information on the relative complexity of the feedback loops and information on the sizes of the ambiguity groups resulting from the breaking of the loops. The topological indicator report contains the most information to determine the inherent testability of a circuit. The fault isolation level is given for each ambiguity group size present in the circuit. This FOM is a primary means of verifying whether testability requirements are being met. All the characteristics of each ambiguity group, test, and item are given in the detail section. This information is in an easy to read form, but for large circuits, the detailed section can become quite large. The last report produced is the fault isolation indicator report. The processing of this report provides a means for determining the number and types of tests to use. The user has options regarding the acceptable size, time to replace, and cost to replace for ambiguity groups. The user can also select the number and type of tests to consider. The report then lists the suggested tests to use and tests that are not used. This report also includes a detailed flow diagram for the isolation of faults. This diagram includes the cost and time spent testing to reach a particular point in the isolation procedure. The first three reports, namely the management model, feedback loop indicator, and topological reports, are better suited to evaluate the testability of a particular design.

Using the dependency modeling, STAT can analyze any circuit regardless of whether it is analog, digital, or hybrid. The only difference is in the amount of modeling required for a particular CCA. The multitude of output information allows the evaluation of just about any aspect of the testability. The most important and useful of which are the detailed fault isolation and ambiguity group parameters.

CRITERIA and SCORES for STAT:

(1) 9: The presentation of the fault isolation levels in a simple table is the ideal result to express the testability of a circuit.

(2) 10: The feedback loop breaker provides information directly on the results of breaking loops. The interactive nature of this feature makes dealing with feedback loops extremely easy.

(3) 9: Provides informative, detailed suggestions regarding the placement of test nodes.

(4) 9: STAT provides very detailed information on all aspects of ambiguity groups and feedback loops.

(5) 9: Very detailed descriptions of test strategy and suggestions as to what type of test to use.

(6) 6: The interactive data editor makes data entry easy, but having to have the proper prefixes for all nodes and items is irritating.

(7) 4: The development of the dependency model for complex circuits can become very tedious and time consuming.

(8) 8: The descriptive names of all the choices presented and the on-line help make STAT an easy program to operate.

VIII. STAMP

STAMP, which stands for System Testability and Maintenance Program, is a tool designed for the analysis of system testability and fault-isolation strategies. STAMP is a program from ARINC Research Corporation of Annapolis, MD. It runs on an HP-1000 mini-computer and recently a PC-based version has been announced. STAMP is for the use of ARINC employees only and primarily for government contracts. It is unfortunate that, at this time, it is not available to the end user directly. For a tool to be effective in the design phase, it must be readily available to the design personnel. Then, when design changes in the circuit occur, their effect on the circuit's inherent testability can be quickly determined and suggestions made back to the design personnel on how to improve the circuit's testability. We developed the models for the three circuits and sent them to ARINC to be processed. The quick turn-around for receiving the results would be sufficient when doing an analysis of a previously designed circuit for testing purposes only. However, for a design work, the results must be available as quickly as possible. STAMP performs its analysis based on the first-order dependency model.

The discussion on the dependency model in the previous section applies to STAMP as well except for the differences in terminology for tests and items. STAMP refers to these as events and elements, respectively. The use of the dependency model is essential for modeling hybrid and analog circuits properly. Again the main disadvantage of dependency modeling is the large amount of time required to model analog and hybrid circuits. The user has several options for inputting the dependency information. The dependency information can be entered directly into STAMP through an interactive interface, through an ASCII file, or through the wire-list output of the popular schematic drawing program Schema. We produced our dependency lists which were then entered directly into STAMP manually. STAMP has default labels for all the aspects of the dependency model. However, these labels may be modified so that the labels on the output information can be related directly back to the original schematic. In addition to the first-order dependency information, a wide range of modifiable parameters are also available. These include failure rates, different types of test, test time and cost, test grouping, component grouping, etc. The ease with which the data can be entered is not a criteria that we can judge fairly since we did not actually run STAMP. Likewise, the options available when obtaining output cannot be fairly judged. However, the extensive outputs can be analyzed and they do contain the information on testability parameters.

The outputs are separated into two groups, namely, the testability analysis and the fault-isolation analysis. In addition, the lists of all components, dependencies, and a wide variety of other lists are also available from the output. The amount of paper consumed can become excessive, but the reports can be printed separately. Our primary interest is in the testability analysis section which provides 24 different measures of testability. All the results are normalized to provide the user with an idea of how the measures compare relatively. Most of the measures are slightly esoteric, but several provide very information on what to concentrate and on how to improve the inherent testability. The outputs, that are the most informative, are the operational isolation measures. These are commonly referred to as fault-isolation levels. These are one of the means of stating the required testability specifications in the SOW of government contracts. They are the primary FOM for inherent testability. The two primary results give the fault-isolation for a given ambiguity group size. One output considers the weighting of the failure rates while the other does not. This allows for additional flexibility for the specifications to achieve desired testability levels. STAMP provides information on ambiguity groups such as number, size and component members. One type of output, that is worth noting, is the listing of excess tests and redundant tests. This information is very useful when the number of test nodes available is limited by some other factor such as connector pins. Components and tests that comprise feedback loops are also listed. The user's manual states that, in addition to listing the loops, additional tests are identified that could be in dealing with the loops. We could not find these suggestions in any of the outputs that were supplied to us. These are the most useful of the testability analysis outputs. Additional outputs are available that describe a variety of other conditions and parameters.

The fault-isolation analysis provides the information that a technician would use in the isolation of a fault. Lists are given which identify the tests that will be bad given a failure in a particular component. Probabilities of failure figures are also given for all components and tests. A tabular decision tree and a graphical tree can be produced that detail the steps a technician would follow in the fault isolation procedure. The procedure can be affected by different weightings and options associated with test time, test cost, test groupings, component groupings, etc. Finally, a summary of the testing procedure in terms of the number of steps required to isolate faults is given.

With the use of dependency modeling, STAMP can also analyze any type of circuit regardless of whether it is digital, analog, or hybrid. The drawback to this is the extensive amount of modeling that must be performed to properly analyze analog and hybrid circuits. The major feature missing from STAMP is the lack of information on breaking up feedback loops. The most useful output provided is the data on fault isolation levels related to ambiguity group size. The extensive output information for about every conceivable seems excessive, but may find application in different situations.

NOTE: STAMP will only be an effective tool for incorporating testability in the design phase if it is available for use by the design personnel. The time needed to obtain the results and the need to experiment with many different testing scenarios makes this a necessity.

CRITERIA and SCORES for STAMP:

(1) 10: The operational isolation measures for both the consideration or exclusion of failure rates is the most result.

(2) 0: No suggestions are provided for the breaking of feedback loops. This is a major drawback since the existence of feedback loop is a primary contributor to poor testability. Invariably all real world CCAs will have feedback loops.

(3) 6: The listing of redundant and excessive tests is very useful for large designs where all possible paths to the output tests are not always clear. However, no suggestions are made regarding the placement of additional tests.

(4) 9: STAMP provides detailed listings of every aspect of the dependencies, ambiguity groupings, etc.

(5) 8: Detailed test procedures are given in decision tree form.

(6) 8: The existence of a CAD interface with Schema can make data input easier. The flexibility of the labeling of components and tests makes data entry and verification easier.

(7) 4: The development of the dependency model for complex circuits can become very tedious and time consuming.

(8) N/A: We cannot evaluate this aspect of STAMP since we did not actually use the program.

IX. I-CAT

I-CAT, which stands for Intelligent Computer Aided Testing, is a model based expert system designed for fault-diagnosis applications. I-CAT is a software program from Automated Technology Systems Corporation. We ran I-CAT under SunView on a Sun SPARCstation 1 with 8 MBytes RAM installed and a color monitor. Although I-CAT is primarily designed for fault-diagnosis, it has several features that make it attractive for performing testability analysis. I-CAT also requires essentially the same type of models as those required by STAT and STAMP. The only difference is that I-CAT refers to the model as a functional model rather than a dependency model, although both contain the same information.

The model information is entered into I-CAT through a graphical interface. I-CAT defaults to starting all component labels with a 'C'. This can be altered to anything that is considered appropriate by the user. When test nodes are placed in the diagram, they can have any label that has not already been used. Since the model can be seen, it is easier to debug the input data with I-CAT than other programs. However, this is not always the case with large, cumbersome models. Even when using the grid to aid in the alignment of connecting lines, it is often difficult to make the connections if the view is not zoomed in. Any open lines resulting from these bad connections have the same effect as in a real circuit since the signal path is broken. These are sometimes difficult to spot unless the view is zoomed in quite close. Trying to maneuver around a large model can be frustrating due to the lack of a dynamic scroll. However, I-CAT encourages the use of nested models to make the model easier to create, edit, and view. Any component can contain a substructure that can consist of another model. I-CAT contains a built-in syntax checker that verifies the graphical model for connectivity errors. This is essential since the graphical interface can sometimes be deceiving. Along with the functional model, various other parameters may be entered, such as component cost, component failure rate, and test cost.

I-CAT's development tools generate a wide variety of outputs that have many uses. We focus primarily on the testability analysis data. Several pertinent input options are available to the user that have an effect on the testability analysis, namely, whether or not to allow multiple faults, whether or not to use only the assigned test nodes, and whether or not to test inside all the substructures. If the assigned test nodes are not used, the number of test nodes to use may be entered. The primary outputs of the testability analysis are the line tree and analysis report. The line tree provides a graphical representation of the testing procedure. This is also given in other forms such as a bar tree and a flow chart. The analysis report contains statistical information on the cost of diagnosis, the replacement costs, and the ambiguity group sizes. Statistics such as variance and standard deviation are given. The ambiguity group information contains no information about which components belong to which group. This makes it difficult to modify a design to have smaller ambiguity groups. Information is also provided on feedback loops and test nodes. I-CAT identifies feedback loops and the components that comprise each loop. No suggestions are given about where to break the loops. The test node data is the most useful of the testability analysis outputs. It details which nodes are used, not used, and any additional nodes that may be required. The last output is only given if the assigned test nodes are ignored. It is not clear what testability parameter of the model is being maximized by the choice of test nodes. There are no outputs that give the testability of the model in terms of fault coverage and ambiguity group sizes. These two aspects are the important ones for DFT and it is unfortunate that these are not available from I-CAT. I-CAT can also produce diagnostic programs in various languages, test requirement documents (TRD), netlist information, and a Personal ATLAS Workstation (PAWS) data base.

I-CAT also has several additional features that deserve mention, although not directly related to design for testability. These are a troubleshooter, fault simulator, and a rule editor. The troubleshooter is an interactive graphical interface for diagnostics. The user enters components known to be good and the results of tests. The troubleshooter can direct a technician on how to proceed with the testing of a device. This is all accomplished through extensive dialog boxes that are full of information previously entered into I-CAT. The symptoms reported at test nodes can also be entered into I--CAT. The user can even step backward through a diagnostic session. Perhaps the most important aspect of the troubleshooter is that the failure information can be saved to create a history of components that fail. The fault simulator can be used to compare the testing strategies used by technicians and the strategies offered by I-CAT. I-CAT randomly introduces a fault into the model and it is up to the user to find its location. The user can keep track of the number of steps required to isolate a fault and then let I-CAT find the fault. This way a technician can train on a particular circuit without having to damage the actual hardware.

The rule editor is probably what distinguishes I-CAT from the other testability tools. Information on all aspects of the tests and the testing procedure is entered through the rule editor. The parameters associated with tests range from the dimensions of the measured quantities to the probabilities of test results. These quantities along with the failure history of the model from the troubleshooter can be used to increase the quality of the testing procedure. These features are primarily concerned with manual testing or any type of testing where operator intervention is necessary.

With functional modeling, I-CAT can also analyze all types of circuits whether they are analog, hybrid, or digital. Once again the drawback to this is the extensive amount of modeling required for analog and hybrid circuits. Although I-CAT is primarily a fault diagnosis tool, the information on test node selection and feedback loops is informative . However, the lack of fault isolation parameters and details about ambiguity groups make I-CAT better suited for establishing testing procedures once the circuits are designed.

CRITERIA and SCORES for I-CAT:

(1)	0	: No final testability measure is given.
(2)	0	: No information on where to break feedback loops.
(3)	9	: The details concerning test nodes are the most useful of all the outputs with regard to design for testability.
(4)	4	: Only limited information on ambiguity groups and feedback

NOTE

Before installing the STAT I Program on your PC, you must review and verify these five simple checklist steps. Please follow these installation instructions implicitly or you may experience difficulties in operating the STAT Program.

I. VERIFY THAT YOUR P. C. HAS AT LEAST 540 K-BYTES OF AVAILABLE RAM SPACE AFTER THE P. C. HAS BEEN BOOTED. This is of prime concern, since this is the minimum amount of FREE RAM required for execution of the STAT I Program. Remember, a certain amount of RAM is always allocated to DOS and resident programs. This can be verified through the use of the use of the "CHKDSK" or "MEM", MS-DOS commands and observing the "free space available".

For those installations being performed on machines whose operating system software has been upgraded to MS-DOS Version 5.0, it is necessary to verify that your system has been properly configured to operate STAT I. This should be done in the following steps:

A: Verify that your system's CONFIG.SYS file contains the statement DEVICE=C:\DOS\SETVER.EXE, where C:\DOS represents the directory on your system that contains the executable file SETVER.EXE. If it does not contain this statement, then please properly modify it now.

B: If it was necessary for you to modify your CONFIG.SYS file, then you must re-boot your system. This can be done by depressing the Contrl-Alt and DEL keys simultaneously.

C: Copy the DSI supplied batch file SV.BAT resident in the root directory of the last disk of the STAT I Installation Diskettes, to your PC's root directory (C:\). This Batch file appends STAT's file names to your systems SET VERSION file. After executing this command you must once again re-boot your system.

II Verify that your P. C.'s hard disk drive has had proper periodic housekeeping and that there are **NO** UN-CONTROLLED BAD SECTORS, LOST CHAINS or CLUSTERS. This is easily determined and remedied through the use of the MS-DOS command "CHKDSK". Please consult your Computer Systems Manager or your DOS operator's guide.

III Verify that your P. C.'s "CONFIG.SYS" file in the "root" directory of your P.C. has 'files' set to at least 40 and 'buffers' set to the following table depending on the storage capacity of your hard disk drive:

HARD DISK SIZE	***BUFFER SIZE***
Less than 40 MB	20
40 through 79 MB	30
80 through 119 MB	40
More than 120 MB	50

There is an exception to this rule, and that is the case where your system is capable of utilizing cache memory". This caching is usually accomplished through the use of a caching program such as that which comes with MS-DOS 5.0 known as ,SMARTDRV In this case you should probably reduce the number of buffers to 10 or 20.