LOGIC MODELING AS A TOOL FOR TESTABILITY

Presented at
AUTOTESTCON '85
Long Island, New York
October 22 - 24, 1985

AUTHOR: R. A. De Paul Jr.
DETEX Systems, Inc. Villa Park, Ca. 92667
(714) 637-9325
JULY,1985

ABSTRACT

Logic Modeling as discussed in this paper refers to a disciplined method of communicating the NATURAL knowledge base or scheme of any single or multiple generic equipment organization. It is not limited to specific methods of digital logic design, display, or simulation, nor is it a drawing system.

The Logic Model (frequently referred to as LOGMOD or Design Disclosure Format) has been identified and evaluated as a bone fide method of deriving Testability by Government and Industrial institutions for about twenty years. This test of time proves that Logic Modeling is a viable design disclosure technique and a friendly user communication tool. Recent refinements have contributed to its continued success in translation from design to support applications.

There is an ever increasing requirement for a single data base capable of multi-discipline use. Therefore it is incumbent upon anyone developing any data base that the data base be capable of communicating objective quantitative parameters. Some of these parameters are: Figure of Merit, BIT/BITE candidates for design, determination of test strategies before hardware commitment, exhaustive test strategies to structure Test Program Sets, and technically accurate delooping schemes. Logic Modeling more than fulfills these requirements and has truly become a mature system for use throughout the design and support disciplines.

INTRODUCTION

In any system development program, interface must be made between Customer and Contractor at a point of maximum impact, and the best impact can be made before building pre-production models for test. Therefore, if any of the concepts which result from analysis have to be exercised, it should be done at the proper points in time.

Currently computerized analytic techniques of COMPONENT models have been developed and used. The shortcoming is that most of our computer techniques lack sufficient and disciplined input data to assess testability at the SYSTEM level, equally as well as at the COMPONENT level.

"There exists a tremendous gap between the concept stage of design and the operational stage, and this is where most of the decisions by the technical people are made. In this gap there is a lack of documentation which defines the system as it is developed by the contractor. The lack of documentation imposes a real barrier to anyone sitting back with a new test concept, or anyone with a new micro device who wants to introduce it into the test system. The barrier exists because communication cannot be made with the system design. Therefore the first problem is to translate our knowledge of a system into a working document (or data base). This we may term Design Disclosure. Formalizing this documentation or data base into a Design Disclosure Format (DDF) will produce the guidelines necessary to providing thorough system analysis."

These ideas are as pertinent today as they were in 1965 (April 27, 28). A breakthrough was taking place in communicating operational equipment design to the test engineer. Logic Modeling was the breakthrough. Its generation was discussed at the first NMSE (Naval Material Support Establishment) System Performance Effectiveness Conference, in the Civil Service Commission Auditorium, Washington D.C. A portion of the ideas expressed in the previous paragraph were taken from the Welcoming Address by RADM C.A. Curtze, Assistant Chief, Bureau of Ships. Other portions were excerpted from Mr. Paul Giordano, System Effectiveness Branch Head, U.S. Naval Applied Science Laboratory, Brooklyn, NY.

Here we are, more than twenty years later asking ourselves the same questions. This time there is an advanced computer technology able to develop Logic Models thousands of times faster and have the models accurately quantifying almost all the parameters called for in MIL-STD-2165. An example of how the Logic Model has been enhanced in the past twenty years is perhaps best stated by Mr. Hiroshi Satake (NAVELEXCEN, Vallejo, CA) in his report discussing LOGMOD (or LOGic MODel) "LOGMOD.MMO" dated February 28, 1985. "The power of Logical Modeling rests on its simplicity: 1) the analyst doesn't have to be an engineer; 2) LOGMOD generates products of a very general utility; and 3) LOGMOD can be hosted on a microcomputer. The two general principles with which LOGMOD simplifies fault tree generation for testing also are the source of LOGMOD being dismissed as too simplistic. I choose to call LOGMOD elegant and realizable."

THE LOGMOD CONCEPT

The LOGMOD Concept illustrated in Figure 1 and outlined below is:

* A set of procedures to unambiguously assess the testability of hardware/software during any phase of its development from concept through production;

* The application of a rigorous set of analysis techniques to formulate a true multi-use Natural Intelligence knowledge base;

* The Cause and Effect philosophy developed as a pragmatic design and support tool;

* A means of automatically obtaining a comprehensive testability report containing, among other assessments, a Figure-Of-Merit (FOM) based upon design and support factors;

* A means of automatically obtaining Logic Test Structures (LTS). These are hardcopy printouts in a graphic form similar to flow charts, depicting the optimum testing strategy required to find a single malfunction or any combination of malfunctions in an equipment;

* A means of human interaction with an active or passive electronic maintenance aid to "WALK" a technician through optimized fault isolation activities. This replaces voluminous technical publications yet maintaining a near perfect confidence level in fault detection and isolation;

* A primary building block for developing an "EXPERT" support system;

* A graphic hard copy display depicting the true interrelationship of all system or equipment elements (known as a LOGIC MODEL or MANAGEMENT DIAGRAM);

* A means of allowing rapid and accurate Computer Aided Battle Damage Assessment and Repair (GABDR) using assessors of less expertise than conventionally required, and using the same knowledge base developed for ordinary maintenance tasks;

The animated illustration (Figure 2) serves two purposes. To bring an otherwise static block diagram into a dynamic form, and to portray the amount of task time associated with the interrelated Logic Modeling activities. Naturally the outputs would reach fruition by correct testing strategy. The SAMA is a Stand-Alone-Maintenance Aid and a forerunner to the hardware portion of an Expert System.

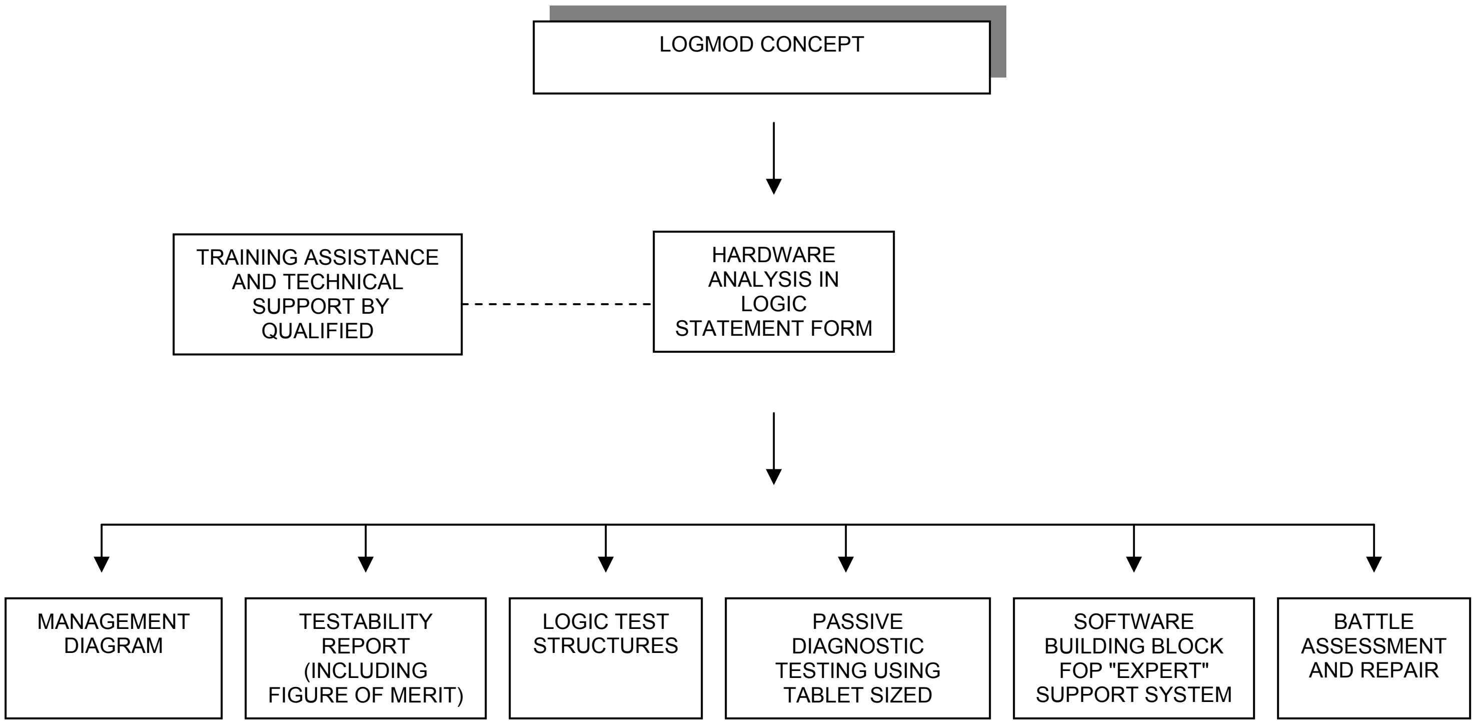
LOGMOD CONCEPT
TRAINING ASSISTANCE AND TECHNICAL SUPPORT BY QUALIFIED
HARDWARE ANALYSIS IN LOGIC STATEMENT FORM
MANAGEMENT DIAGRAM
TESTABILITY REPORT (INCLUDING FIGURE OF MERIT)
LOGIC TEST STRUCTURES
PASSIVE DIAGNOSTIC TESTING USING TABLET SIZED
SOFTWARE BUILDING BLOCK FOP "EXPERT" SUPPORT SYSTEM
BATTLE ASSESSMENT AND REPAIR

Figure 1

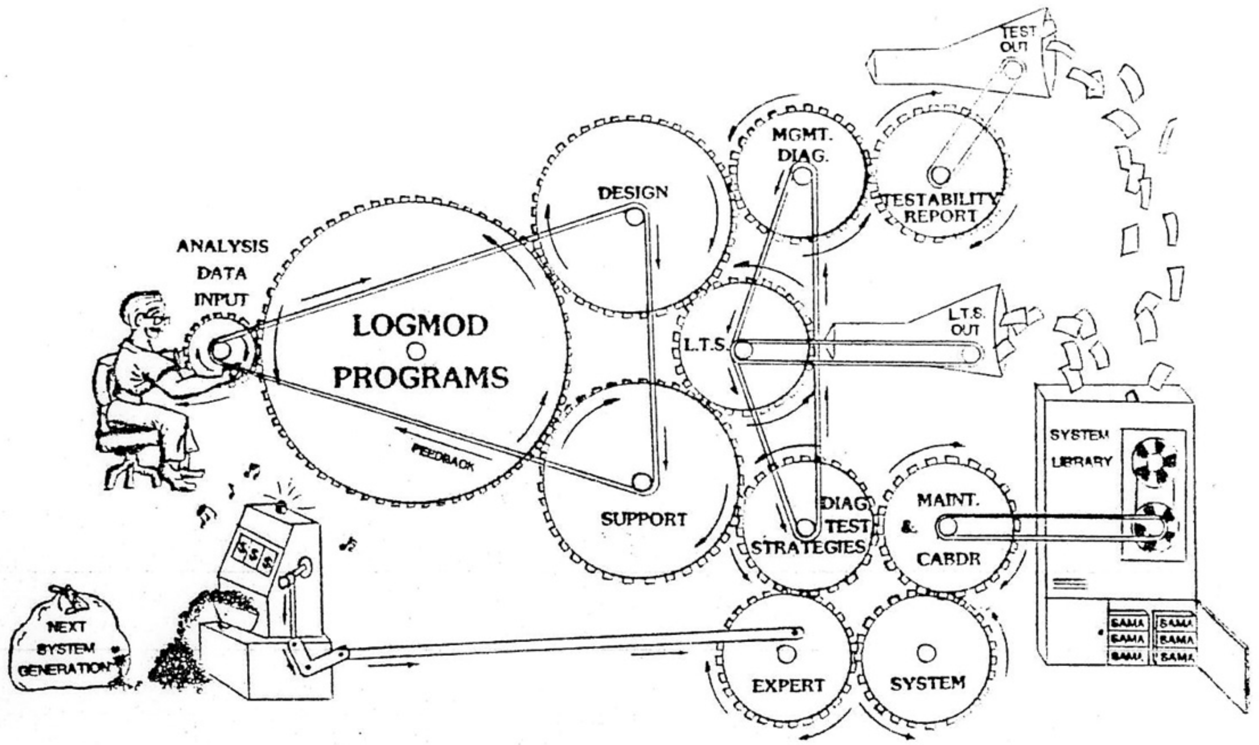

Figure 2

LOGMOD METHODOLOGY

There are two major principles underlying the reason for Logic Modeling. These were referred to by Mr. Satake in his report "LOGMOD.MMO". They are stated here for clarity.

LOGMOD PRINCIPLE I FAULT FLOW FOLLOWS SIGNAL.

LOGMOD PRINCIPLE II FAULT FLOWS REGARDLESS OF THE ELECTRICAL, MECHANICAL ETC. FUNCTIONS OF THE DISCRETE ENTITIES REPRESENTED.

The first principle simplifies analysis such that 90% (or more) of the time, in some instances the analyst is inputting a type of netlist description of the circuitry or components. In others, he describes the interrelationship between replaceable units and sub-systems. Not all phenomena are causal, but certainly most designed systems and all lumped electric circuits are. The validity of this principle rests on a fault buffering assumption: a fault affects forward signal paths, not backward signal paths. Classic exceptions to this fault buffering is the "blown fuse", bad switch, power supply loading, open relay coils, stuck-at-tri- state input cases and the redundant path case. In the 10% or so of the cases where fault modes are not buffered, special modeling rules are required which are only slightly more difficult to identify and apply. The analyst inputs schematics and system block diagrams into the LOGMOD Organizing and Structuring Program in a simple coded format. The resulting outputs are then formulated into specific forms of directed graphs and report data.

The second principle shows little regard for how an output depends on the workings of an item or its inputs. It generates its reports, fault trees, etc. based merely on the topology of the signal flow graph which it generates as a primary output. ATPG designers and FMEA analysts spend a lot of time performing an imprecise form of backward inference. This is a monumental task even in the simplest circuit cases: the number of paths grows exponentially. Also, most designers are trained analytically (by forward inference: "if A then B"). The discipline required to resolve a search strategy "if not B then maybe A or probably not C . . . " is currently practiced more as an art rather than as an engineering discipline. When the designer is confronted with resolving this strategy for each item output mode (especially at the system and sub-system level) and with having to understand how each item functions, false short cuts and errors are inevitable.

LOGMOD partitions the problem of "understanding and reliable test strategy generation" into two manageable steps: 1) the analysis required to know the "Goodness" of an output which is the forte of any designer and; 2) the generation of the test strategy for which the designer is ill equipped in training and time. The engineer, preferably working with a circuit simulator CAD tool, and a target system can devote his expertise to delooping complete test sequences by devoting his time to understanding the system for the purposes of analyzing what a "good" test point indication is rather than what a "bad" indication implies in the way of the next test point path.

A designer can obviously think of many exceptions to these simplifications in the real world. But these are the very principles used by electronic technicians (super-techs) to find faults. A good card swapper will find random faults faster than any designer especially for multiple faults. Moreover, simple modeling rules (sometimes termed "non intrusive") can be developed for these exceptions.

These principles are certainly very true of functional testing digital circuits but also apply to analog equipment. At the microscopic (transistor, bias, and coupling) level there is little buffering of faults, even in digital circuits. There are many more "blown fuse" cases in analog circuits. But at the macroscopic level there is forward direction to fault or signal flow from block to block: although the blocks may overlap or be "fuzzy".

LOGMOD MECHANICS

The Logic Model method transforms the circuit or block "logic statements" into an intermediate, internal "waterfall" dependency structure: structuring least dependent signals and items to the left, and most dependent to the right. This internal representation becomes the heart of the LOGMOD methodology. One of the important implementation traits of the LOGMOD programs are that every output has direct application to the designer, the logistician, or the test engineer. To gain a feeling for the power and validity of the LOGMOD Concept note how the different outputs fall out of a single original dependency structure.

LOGIC TEST STRUCTURES

The Logic Test Structures (LTS) are the automatic generations of Checklist and Independent fault finding test strategies printed out on hard copy for proofing or developing electronically embedded test strategies. Strategies are printed to a separately selectable output file.

Each of these outputs is a computerized "tree chart" approach to system testing and diagnostics EXCEPT that the LTS are based upon a mature and proven test strategy technique which explicitly provides exhaustive and interdependent tests for any equipment or system. Since the LTS is developed with LOGMOD, the conventional, ever-changing philosophical diagnostic approach is replaced with LOGMOD's scientific, objective and pragmatic test optimization.

The CHECKLIST Logic Test Structure output approaches testing and fault isolation from a performance test mode. In this mode, all known or implicit information is retained from any performance test to the following performance test(s), minimizing testing required for multiple or secondary failures.

The INDEPENDENT Logic Test Structure approaches testing and fault isolation from a failure observed, or what may be referred to as a fault indicator mode. In this mode, each fault indicator develops its own logic test structure completely independent of all others.

In either case, each output may be considered as a test "road map". In reading this "road map", "GOOD" test results move horizontally from left to right and "BAD" test results move down- ward. In the event that an entire logic test structure does not fit within the physical boundaries of an 8 1/2" x 11" sheet, continuation references appear. The actual length of each logic test structure is dynamic in nature. They may continue from one page to another until the entire structure is defined.

Any LTS can be constructed or reconstructed instantaneously with each new data input as its construction is determined by the LOGMOD Structuring program. Again, as typical with LOGMOD, there is NO ADDITIONAL PROGRAMMING REQUIRED. Test Program Set Developers find immediate use for the Logic Test Structure. This graphic hard-copy print-out allows both Customer and Contractor to visually compare the ease of malfunction isolation or assessment. The LOGMOD generated fault tree is optimal in the following respects:

MINIMUM TREE WIDTH (MINIMUM TESTS): The fault tree makes maximum use of previous test results. The goodness of dependent test points implies the probable goodness of their inputs. The exceptions to this can be handled by a small change. This applies not only to the GO-NO-GO confidence test points, but throughout the tree. No test point is tested unnecessarily. For each node (including the root node), a minimum number of test (including confidence test) branches is required.

MINIMUM HEIGHT (BISECTION): The fault tree uses true bisection to minimize the number of test nodes required to peg the fault. If a test result is bad, it tests the untested inputs. If the inputs are good, it tests the forward middle test points. If bad, it tests the backward middle test points.

Although LOGMOD diagnostics does not perform bisection in the classical sense, because of its internal dependency representation, it knows inputs, outputs, and middles of any fault without any additional memory or flagging requirements. Given a real world bus with multiple sources (which are wire- and'ed) and multiple destinations (which may become bad loads), LOGMOD will do a proper "card pull" bisection.

A very important characteristic of the LOGMOD test strategy is that it will find multiple faults. And it will find them with a minimum number of test points. Most test program sets will not guarantee fault isolation in the case of multiple faults. Again, this characteristic drops out of the waterfall dependency structuring of LOGMOD.

LOGIC TEST STRUCTURES AND DELOOPING FOR TESTABILITY

Thus far we have made reference to three outputs of the LOGMOD Computer programs, i.e., the waterfall, the CHECKLIST Logic Test Structure and the INDEPENDENT Logic Test Structure. Now we shall briefly describe the Testability Report output of the LOGM0D Computer Programs.

Those in the scientific community who have become aware of LOGMOD have ALWAYS found it as an answer to their needs in understanding HOW to build a more testable unit, and HOW it can be used to assist the diagnostician. So, now that we have described a real "cradle to grave" concept evaluated by the U.S. Government for at least 18 years, why is it still being evaluated? Primarily because it has its bases in simplicity and truth, while most people are looking for some very complicated and magic technique.

What is the real question in comparing Testability/Diagnostic techniques? Whatever manner we use to arrive at a quantitative measure for Testability must have a firm basis in the structure and organization of the input data.

Feedback loops play an important role in this structure. This is one of the great contributions LOGMOD makes to eliminating so many false removals. Now, one may say, "I know where all the loops exist." Past experience with Prime Equipment Contractors has indicated they have found how easy the LOGMOD techniques automatically find loops which are inherent in the operational equipment, yet the analyst or engineer did not notice them while documenting the equipment.

Since hardware analysis may be accomplished by several analysts working on different portions of the hardware, the resulting graphic interlaces common facts yet distinguishes unique information. The graphic representation of logic statements automatically highlights feedback loops that may or may not have been transparent to the analyst(s). It is absolutely necessary that the existence of these feedback loops be disclosed and completely eliminated through some sort of de-looping scheme prior to making any conclusion regarding the degree of testability residing in any system or equipment. Furthermore, since only meaningful tests can properly contribute to the isolation of a malfunction, tests appearing in a feedback loop must have their logic statements reviewed for accuracy. If implicated logic statements are correct, then the designer or management must be made aware of the loops and decide whether a change is required, or even possible to be made, to "de-loop" the hardware. It is important to emphasize that, AT TIMES DE-LOOPING CAN BE ACHIEVED BY ADDITION OF APPLICATION SOFTWARE WITHOUT ACTUAL HARDWARE MODIFICATION.

Let us pause for a moment and consider the fact that some equipments may have inter-related loops which form a mathematical union. In such an instance some of the loops may be transparent to the analyst while others are seemingly masked. In any event, however, a complete de-looping process must take place even though masked loops seem to complicate the de-looping process. The LOGMOD Algorithm can automatically deloop the entire Logic Model or it can provide the analyst outputs of intermediate steps in de-looping which allows the same type of decisions to be available at each step in de-looping as was originally available with transparent loops; namely, make a hardware modification, add application software, or condense tests (which may enlarge ambiguity groups).

In the event that a logic statement is correct but modification to the hardware is not desired or possible, the LOGMOD Concept includes a special de-looping computer program which automatically condenses the tests involved in a loop. This technique in no way affects the technical integrity or operation of the hardware. However, it does allow meaningful testability information to be derived by the LOGMOD Programs.

Somehow, any Testability/Diagnostic technique must account for all inherent loops; otherwise, any attempt at documenting a diagnostic strategy is at best a guess. Users of the LOGMOD Concept, without exception, have verified the actual existence of, as well as, the completeness of loop determination by the LOGMOD Algorithm. Without a completely effective treatment of this one small area, "loop handling", the quantitative Testability measures may be suspect.

LOGMOD TESTABILITY OUTPUTS

Following is a listing and brief description of those outputs of the LOGMOD Algorithm which directly influence Testability.

1. Figure of Merit

How often testing will reveal a combined set of ambiguity groups. The size of these sets are listed numerically beginning with a fault group of one (1) ambiguity group, and proceeding to each successively higher combination of ambiguity groups. This calculation is expressed as a percentage.

2. Performance Test List (Functional Operation Test List)

A list of the minimum (and exact) tests which must be performed to determine the complete functional operability of an equipment or system with 100% confidence.

3. Maintainability Information

A. The Test Strategy (and results) for each unique ambiguity group.

B. The testing time required to perform fault isolation to each ambiguity group (in units of 1/10th minute)

C. The cumulative testing and replacement time for each ambiguity group (in units of 1/10th minute).

D. Test Strategy Signature of a failed item. This is the combination of "good" and "bad" tests determined to be the most optimal.

4. Fault Isolation Times

A summation of all test times associated with the test strategy to find a malfunction if the performance check fails. These fault isolation times may be printed-out from the LOGMOD Structuring and Organizational program prior to completion of test procedure generation to influence the placement of sensors and test procedure language.

5. Hierarchy of BIT/BITE Candidates

A hierarchal listing of relative importance of each test in its ability to correctly and successively "split-halve" the system for testing. This list contains all nodes, actions and states disclosed by the equipment/system operational scheme. This portion of the report may become extremely interesting in that sometimes designers clearly show test points for signals of very little importance to fault isolate and yet do not provide points for very important test signals. Generally, test points are provided for the sake of the designer, not for fault isolation. This report provides a rational, prioritized list of test points to the designer.

6. Item (Ambiguity Group) Involvement Ratio

How often a specific ambiguity group is involved in all test actions. This calculation is expressed as a percentage.

7. Items Tested (an Optional Output)

A listing of all hardware related to each test, including performance tests.

OTHER LOGMOD OUTPUTS

VALIDATION

This file output is essentially a Failure Modes and Effects output. It correlates for each item, the effects of its being out of tolerance. It also identifies the test point path required to identify the out of tolerance item. It is not difficult to imagine how a collection of such paths can be used to generate a test strategy. The INDEPENDENT Logic Test Structure output in the form of a fault tree matches these paths. However, it is the structure of the test paths which makes this collection work as a "fault tree".

MTTI and MTTR REPORT

This report calculates a predicted MTTR and MTTI given simple estimates of time to measure each test point and repair each item (in units of 1/10th minute).

SUMMARY

Two LOGMOD principles provide the basis for dramatically increasing communication between design and support functions through the collective use of a single hardware/software knowledge base available from Natural Intelligence. A key ingredient is the automated generation of optimized fault paths (or test strategies) at the component level as well as the system level. This in turn allows Testability parameters to be more accurately predicted at the proper point of Customer/Contractor impact decreasing non-recurring Test Program development time in present systems while improving fault isolation performance (both in terms of percentages and isolation time). LOGMOD automatically generates the optimal fault, paths. Because the diagnostics can be targeted to functional automatic fault isolation test programs, usually only minimal changes will be required to the test stimulation primitives already available.

CONCLUSIONS

The Logic Modeling concept expressed in this paper is not new, nor are the reasons for its development. At least twenty years of tracking provides what we have known from nature since its beginning, namely: 1) anything that is material will breakdown; and 2) to provide some reasonably long lasting "fix" of any material object requires an organized and structured knowledge of the NATURAL makeup of that object. Taken to its limit, a long enough period of inadequate "fixes" to the Services' equipments could consume all the Services' resources in support. There may be no resources left for the development of new equipment without neglecting already fielded equipments. Logic Modeling offers the most natural basis of knowledge accumulation for use in Project Management, Design for Testability (at any level), and optimized test strategies for fault detection and isolation.

As in any system, there is always room for improvement. LOGMOD is no exception. If not already accomplished prior to the publication date of this paper, non intrusive testing, weighting of fault strategies, an Analyst's Guide, and additional printouts describing the tests and items involved in each loop, are output priorities which should be accomplished. Hooks for CAD are already in the LOGMOD Structuring and Organizing Program.