A HIERARCHICAL APPROACH TO SYSTEM-LEVEL DIAGNOSTICS

William R. Simpson
John W. Sheppard

ARINC Research Corporation

Summary

As systems have become more complex, diagnosing system problems has become much more difficult. In the past, the development of test protocols to diagnose systems was undisciplined. The system designer often used intuition and expertise to diagnose system anomalies. As system complexity increased, more structured approaches were developed. Small systems of intermediate complexity are currently being modeled and simulated at the component level. However, larger, more complex systems often cannot be simulated, or simulation at the required level of detail is too costly. Therefore, a hierarchical approach to diagnosis of complex systems is required.

Because the growing complexity of full-scale systems has outpaced the ability to provide detailed simulations for fault analysis, ARINC Research developed a hierarchical approach to system-level diagnosis. This approach sets aside the detailed physical knowledge of the system and analyzes the information flow through the system using information theory and dependency modeling. Diagnostic tests are considered information sources, and possible failures are considered conclusions. Tests are chosen from the model for evaluation on the basis of the amount of information they provide. Test results are then combined in a data fusion algorithm so that diagnostic conclusions may be drawn. This hierarchical approach can be applied to any level of system complexity.

The techniques used in the hierarchical approach are embodied in the ARINC Research System Testability and Maintenance Program (STAMP®) and have been applied to a large number of complex systems with outstanding results. This paper provides an overview of system-level testability and diagnosis, describes STAMP and its application in testability and fault isolation, and summarizes the successful uses of the STAMP analysis approach.

Introduction

Field maintenance statistics for complex avionic systems indicate that system maintenance requirements are not being met. System and test design has resulted in false "pull" rates of 40% or higher—the consequence of high ambiguity and labor-intensive test procedures. In addition, false alarms have consumed excessive maintenance resources. Studies of the CH-53[1] and F-16[2] aircraft show that troubleshooting actions can consume as much as 50% of the total labor-hours spent for repair. Data for the scheduled airlines[3] show similar trends for complex electronics. These figures suggest a large potential return on investment if system testability and fault-isolation procedures can be improved.

In the early 1980s, a number of initiatives were undertaken to help keep pace with the growing complexity of maintenance. From these initiatives, programs such as modular automatic test equipment (MATE),[4] intermediate forward test equipment (IFTE),[5] and the consolidated automated support system (CASS)[6] are now being developed, spawning renewed interest in testability. The military has even developed a testability specification, MIL-STD-2165.[7]

Since the mid-1980s, testability has been recognized as a valid and viable engineering discipline. Equipment has good testability when existing faults can be confidently and efficiently identified.[8] Confidence is achieved by frequently identifying only the failed components or parts without removing good items. Efficiency is achieved by limiting the resources required (including labor power, labor-hours, test equipment, and training).

Testability at the System Level

The literature generally describes two different types of testability: *inherent* testability and *achieved* testability. Inherent testability refers to the way in which the system is designed and encompasses the ability to observe system behavior under a variety of stimuli. Inherent testability is defined by the location, accessibility, and sophistication of tests and test points that may be included in the system. Achieved testability refers to how the system is maintained. It is defined by the results of the maintainability process (such as false alarms, ambiguities, incorrect isolations, and no faults found). Achieved testability has inherent testability as an upper limit with no potential lower limit except zero. For achieved testability to be maximized, an analysis of inherent testability is invaluable.

At the design phase, the testability analysis should provide answers to the following questions concerning the inherent testability of the system:

- ***Ambiguity Groups***—Which components are, and which components are not, uniquely identifiable in the current system or test configuration?
- ***False Failures***—When multiple failures occur, are there combinations that can provide the same symptoms as an unrelated single failure?
- ***Hidden Failures***—When multiple failures occur, are they related, and is the root cause of the failure hidden?
- ***Information Feedback Loops***—Where does feedback occur? Feedback loops typically cause isolation problems and

CH2929-8/90/0000-0254 $1.00 © 1990 IEEE

result in larger-than-acceptable ambiguity groups. Mapping of these feedback loops is one step in improving testability through reducing the ambiguity.

- ***Undetectable Component Failures***—Which failure modes of which components are not observed by any of the available tests?

- ***Test Disposition***

  — Which tests are not needed? This information can be used to reduce maintenance complexity and test program set (TPS) test times.

  — Where are extra tests needed? This information is essential for improving testability.

- ***Tolerance to False Alarms***—Does the system require special provisions to identify and handle potential false alarms?

- ***Operational Isolation***—What percentage of the time can one expect to isolate to n or fewer replaceable units? This information is critical for logistic planning.

Diagnosis at the System Level

As with testability, the term diagnosis is often used to represent more than one concept. For the purposes of this paper, we identify three basic terms that are related to diagnosis: *detection*, *localization*, and *isolation*.

Detection refers to the ability of a test, a combination of tests, or a diagnostic strategy to determine that a failure in some system element has occurred. Detection is often associated with built-in tests (BITs) and may actually be the design requirement for BIT.

Localization means that a fault has been restricted to some subset of possible causes. Localization is also associated with a combination of tests or a diagnostic strategy. Clearly, all BITs that can detect do localize (to at least one of all possible faults). If the localization is sufficient in most cases to undertake repair, we often refer to the BIT as *smart* BIT. BIT, however, is not the only diagnostic technique that localizes. Often automatic test equipment (ATE) and manual fault-isolation techniques use a diagnostic strategy that localizes the fault to a degree sufficient to undertake repairs.

The third term, isolation, is frequently misused. It is often used to mean that localization has been achieved to a degree consistent with a single repair unit. However, isolation means that, through some test, combination of tests, or diagnostic strategy, the specific cause of a fault has been identified.

At the implementation phase, the diagnostic strategy should provide a limited set of items:

- A procedure that brings the achieved testability up to the level of the inherent testability (This should be apparent; however, a glitch in the implementation often causes the achieved testability to fall short of the inherent testability.)

- A procedure that can fault-isolate (localize) the system quickly, cheaply, with low skill level and high safety, on nonpremium shifts with minimum schedule disruption (This attribute list can become quite extensive.)

A Hierarchical Approach

It is assumed that, at any analysis level, if we were to write a full-scale physical simulation of the entire system at a specific level of detail, we would then be able to answer all of the testability questions by meticulously tracing stimuli through the system to observed responses. We can do this for exhaustively modeled faults and determine such items as detection and ambiguity. The problem is that the sheer volume of computations required at higher levels of complexity or by a larger system is too great for this approach to be practical. For example, let us suppose that we have a VLSI chip of some 10,000 gates, any one of which may be "stuck open" or "stuck at," yielding 20,000 faults to model. Further, suppose that one of four such chips is on a board with other components, and six such boards make up the subsystem in a system that has 23 such subsystems. We would then have to model and simulate approximately 2.8 million faults for this system.

To derive a less computationally intense testability process, we want to construct an analysis method that is hierarchical and considers only the information necessary to make a diagnosis. The first step is to strip the test of its details of stimulus response and treat it strictly as an information carrier. This is not to say that the details of how the test is conducted are unimportant. In fact, they are essential to actually performing the test. We simply will not carry them along in our analysis. The second step is to set aside the physical details of the hardware and look at functions. This will give us a hierarchical formulation because functions can be aggregated from combinations of other functions, and we can proceed functionally to any level in the analysis. A function, of course, carries with it an aggregation of hardware or a piece of hardware. This in turn provides a way to "repair" functions.

What have we lost? A great deal. We cannot use this model to provide the stimulus-response details. The computer-aided drawing (CAD) file can no longer be used directly for input, although we may be able to manually enter some of the details. The result may be that our model provides a much grosser level of localization than does a simulation model.

What have we gained? A great deal. We can now perform our testability analysis in a hierarchical manner. We can hypothesize information sources without concerning ourselves with the details of stimulus-response—unless and until we want to actually perform the test. We can conduct what-if and trade-off analyses at a much simpler modeling level. And we have a full range of information theoretic tools to help us answer the basic testability and fault-isolation questions.

The software that implements this approach is called STAMP, which runs on HP-1000 A/900 workstations. The software has recently been hosted on a 386-based personal computer.

STAMP Testability

STAMP computes measures of testability and synthesizes fault-isolation strategies on the basis of a dependency model of the system under analysis. The vehicle for dependency modeling is a block diagram that represents the information flow of a given system. For example, consider the simple diagram shown in Figure 1 (c_i = component i, t_j = test j).

We see that test t_4 depends on component c_4 and test t_3. Also, t_3 depends on c_2 and t_1. These are referred to as first-order dependencies. By inference, t_4 also depends on c_2 and t_1. This exemplifies a higher order dependency. Although the higher order dependencies could be directly determined for this simple system, computer approaches are often necessary to determine higher order dependency relationships of complex designs.

The elements shown in the model of Figure 1 are the minimum data input to the process. Although they are labeled as components, the functional blocks of the dependency model may be pieces of components, groups of components, or nonphysical entities, such as timing of buses. The labeled tests are, in this case, functional tests.

We have identified five basic types of tests for inclusion in dependency models: functional test, special test, conditional test, asymmetric test, and linked-outcome test.

- ***Functional Test***—A functional test measures the functional health of the system. The outcome of this test depends on all components and signals (elements) that "feed" the test. A good test verifies all components and signals that feed the test; a bad test implies that one or more of these elements is bad.

- ***Special Test***—A special test is any test that is not a functional test. It does not characteristically monitor the functional health of a system; rather, it provides insight into the possible failures within a system.

- ***Conditional Test***—A conditional test may be either functional or special and is characterized by having more than one set, or list, of dependencies. These dependency lists, in turn, depend on (that is, are conditioned by) an event. Such an event might be a scale setting, bit select, mode of operation, or other parameter.

- ***Asymmetric Test***—An asymmetric test is a special case of the conditional test in that it uses the dependency list as a function of the test outcome. There are three types of asymmetric tests: positive inference, negative inference, and fully asymmetric.

  - Positive Inference—A test that provides usable information only when the outcome is *good*. A *bad* outcome yields no information.

  - Negative Inference—A test that provides usable information only when the outcome is *bad*. A *good* outcome provides no information.

  - Fully Asymmetric—A test that, when the test outcome is *good* (inference drawn from one dependency list), provides different information than when the outcome is *bad* (inference drawn from another dependency list).

- ***Linked-Outcome Test***—The outcomes of some tests may be linked within STAMP. For example, when a trouble light is on (indicating that the test outcome is *bad*), we know that the light bulb and voltage source are *good*. This being the case, when the trouble light outcome is *bad*, the inference engine triggers a *good* test outcome for the bulb test. This cross-linkage differs from the normal dependency inference where a *good* test outcome yields inferences of other *good* elements. Often this type of linkage helps to distinguish between system faults and test equipment faults.

Given these data, we employ algorithms to identify all higher order dependencies. A full range of testability measures and tables is then produced to answer the testability questions (see Testability at the System Level). The details as they apply to the STAMP analysis are discussed in references 9 and 10, which include example computations.

STAMP Fault Isolation

A fault-isolation strategy is a road map of how to use the available tests to determine what in the system, if anything, has failed. A number of different search strategies for building road maps are available. Several such strategies are shown in Table 1.

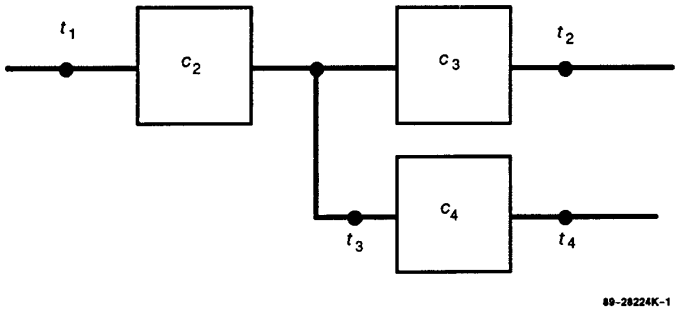

Figure 1. Functional Dependency Diagram

Table 1. Search Strategies

Type of Search	Ordering Required	Strategy
Directed	Yes	Evaluate all tests sequentially from right to left, beginning with the first known fault. Skip those tests where results can be inferred from previous tests.
Half-Interval	Yes	Evaluate at the midpoint of the remaining tests. Go right to left after a good test and left to right after a bad test, repeating the half-interval search. Skip those tests where results can be inferred from previous tests.
Combination of Half-Interval and Directed	Yes	Evaluate at the midpoint of the remaining tests until a bad test is encountered. Use directed search from left to right for those tests that depend on bad tests. Skip those tests where results can be inferred from previous tests.
Exponential	Yes	Select the test closest to 63% partition from left to right of the remaining tests. Continue testing with 63% partition.
Combination of Exponential and Directed	Yes	Select the test closest to 63% partition from left to right of the remaining tests. Continue until a bad test is encountered, then use directed search from right to left.
Adaptive or Information Theoretic	No	For each test, ask how much information can be inferred from either good or bad tests. Select tests to optimize the answer.
Random	No	Use uniformly distributed random numbers to choose tests.

Fault isolation can be mathematically described as a partition process. Let $\mathbf{C} = c_1, c_2, \ldots, c_n$ represent the set of components. After the jth test, a fault-isolation strategy partitions C into two classes:

$\mathbf{F}^j = (c_1^j, c_2^j, \ldots, c_m^j)$ = the set of components that are still failure candidates after the jth test (feasible set).

$\mathbf{G}^j = \mathbf{C} - \mathbf{F}^j$ = the set of components found to be good after the jth test (infeasible set).

By using this partition process, a failure will have been isolated when $\mathbf{F}^j$ consists of a single element or a component ambiguity group. The directed search strategy may reduce the size of $\mathbf{F}^j$ by only one component at a time for a serial system. The half-interval technique may reduce $\mathbf{F}^j$ by approximately 50% after each test, an obvious advantage.

It can be proved that for a well-ordered* system the half-interval technique will provide the minimum number of tests. However, as was pointed out earlier, such an ordering rarely exists. The STAMP approach uses an adaptive, information-based strategy, which reduces to the half-interval technique for well-ordered systems.

With an information model available, it is possible to compute the information content of each test. If a test is performed, the set of dependencies allows us to draw conclusions about a subset of components. The process of drawing conclusions about the system from given information is called inference. For any test sequence, STAMP allows us to compute $(c_1^j, c_2^j, \ldots, c_m^j)$ and the set of remaining failure candidates, namely $\mathbf{F}^{j1}, \mathbf{F}^{j2}, \ldots, \mathbf{F}^{jk}$. A complicated but effective algorithm has been developed to look at the information content of all remaining tests so that the number of remaining tests that have to be performed to isolate faults is minimized over the set of potential failure candidates. This adaptive approach embodies several artificial intelligence algorithms, including inference and pattern recognition. It can be described mathematically as follows:

Let **D** represent the set of dependency relationships between components and test points.

Let $\mathbf{S}_k$ be a sequence of k tests, $t_{j1}, t_{j2}, \ldots, t_{jk}$.

Let $\mathbf{F}^k$ be the feasible failure candidate set associated with $\mathbf{S}_k$.

*The distinction between partially ordered and well-ordered systems is one of uniqueness. A partially ordered set may have several permutations by which ordering is satisfied; a well-ordered system has only one ordering.

We then compute an information measure for each remaining (unperformed) test (j), which is a function of the dependency relationship and the remaining candidate failure class, say, $\mathbf{I}_k^j = f(\mathbf{D}, \mathbf{F}^k)$. The test sequence $\mathbf{S}_k$ that is derived is obtained by optimizing at each decision point. That is, the next test in the sequence is taken as the test that maximizes $\mathbf{I}_k^j$ for the conditions imposed by each previous test outcome and is based on an unknown current outcome. The sequence ends when adequate information is derived for fault isolation.

STAMP Effectiveness

In a number of applications, the adaptive or information theoretic approach has provided the mean and variance of the required number of tests under all failure conditions, either equal to or lower than those resulting from other procedures examined and often approaching the theoretical minimum values. This is illustrated in Table 2, which lists the fault-isolation performance values for a number of search strategies for a partially ordered system consisting of 17 components and 18 test points.* For this system, the theoretical minimum number of tests, 4.09, is approached by three of the methods considered: exponential, exponential-directed, and adaptive, with the last providing the lowest mean. The adaptive method has a variance considerably lower than the other alternatives. That quality offers advantages in planning labor power, facility, and equipment resources for maintenance.

Table 2. Performance Values for Search Strategies for a System with 17 Components and 18 Test Points

Search Strategy	Average Number of Tests Required	Test Variance
Directed	6.86	5.26
Combination of Half-Interval and Directed	5.14	1.26
Exponential	4.43	1.24
Combination of Exponential and Directed	4.50	1.68
Adaptive	4.35	0.23
Random	5.71	4.49
Theoretical Limit	4.09	—

STAMP's effectiveness is best demonstrated by how it has been applied to actual systems. STAMP has been applied to more than 50 systems, each time improving field performance or design characteristics. Table 3 lists some successful applications of STAMP.

Table 3. Results of STAMP Applications

System	Customer	Results
AN/ALR-67 Countermeasures Set	NADC/USN	Developed test procedures for test requirements documents.
AN/APG-63 Radar	ALC/USAF	Evaluated BIT test configuration.
AN/ALR-62 Warning Receiver Processor Group	ALC/USAF	Made recommendation for reducing ambiguity groups by over 40%.
Air Pressurization System	International Fuel Cells	Improved unique isolation by over 100%.
AN/MSQ-103C TEAMPACK	EW/RSTA/USA	Reduced required testing by 87%; portable maintenance aid developed.
Mk 84 60/400-Hz Static Frequency Converter	NAVSEA/USN	Reduced required testing by 70%; portable maintenance aid developed.
UH-60A (Blackhawk) Stability Augmentation System	ATL/USA	Reduced mean time to fault-isolate by a factor of 10; reduced maintenance complexity by a factor of 3.
ALQ-131 Podded EW System	ASD/USAF	Reduced mean time to fault-isolate by 75%.
ALQ-184 Podded EW System	AFLC/USAF	Reduced false alarm rate by a factor of 10; developed unit under test (UUT) software procedures.
B-2 Bomber DFT Program	USAF/Northrop	Improved specification compliance at the shop replaceable unit (SRU) level by 80%.

*The example system was derived from Cramer, et al.[11] No weighting of failure rate, costs, or other factors was considered.

Conclusion

STAMP has been applied to a wide variety of systems with outstanding results. The applications have encompassed a large number of system technologies at varying points in the life cycle. STAMP emphasizes maintenance at the system level, which differs from most other testability analysis tools that operate at the gate or, at most, board level. This system-level emphasis forces STAMP to be used hierarchically and to approach the testability problem from an information flow, rather than electronic simulation, standpoint. An additional result is that the approach is independent of the underlying technology, thus allowing the analysis of most systems, including hybrids.

References

1. Cook, Thomas N., et al. "Analysis of Fault Isolation Criteria/Techniques." In *Proceedings Annual Reliability and Maintainability Symposium*, San Francisco, California, January 1980.

2. Labit, M. L., et al. *Special Report on Operational Suitability (OS) Verification Study Focus on Maintainability.* 1751-01-2-2396, Annapolis, Maryland: ARINC Research Corporation, February 1981.

3. Aeronautical Radio, Inc. *Avionics Maintenance Conference Report—San Diego, 1987.* 87-087/MOF-34, Annapolis, Maryland: Aeronautical Radio, Inc., August 1987.

4. Cross, G. "Third Generation MATE—Today's Solution." In *Proceedings of the 1987 IEEE AUTOTESTCON Conference*, San Francisco, California, September 1986.

5. Espesito, C. M., et al. "U.S. Army/IFTE Technical and Management Overview." In *Proceedings of the 1986 IEEE AUTOTESTCON Conference*, San Antonio, Texas, September 1986.

6. Najaran, Captain M. T. "CASS Revisited — A Case for Supportability." In *Proceedings of the 1986 IEEE AUTOTESTCON Conference*, San Antonio, Texas, September 1986.

7. Naval Electronic Systems Command (ELEX-8111). *Testability Program for Electronic Systems and Equipment.* MIL-STD-2165, Washington, D.C.: Naval Electronic Systems Command, January 26, 1985.

8. Simpson, W. R. "Definitions Used in the STAMP Technical Note Series." STAMP Technical Note 312, Annapolis, Maryland: ARINC Research Corporation, July 1986.

9. Simpson, W. R. "The Application of the Testability Discipline to Full System Analysis." In *Proceedings of the IEEE Automatic Test Program Generation Workshop*, San Francisco, California, March 1985.

10. Simpson, W. R., et al. "Experiences Gained in Testability Design Trade-Offs." In *Proceedings of the 1984 IEEE AUTOTESTCON Conference*, Washington, D.C., November 1984.

11. Cramer, M. L., et al. *Logic Model Analysis and Standard Maintenance Information Display System (SMIDS).* USAAVRADCOM-TR-81-D-45, Fort Eustis, Virginia: U.S. Army Applied Research and Technology Laboratories, April 1982.