MODELING IT BOTH WAYS: HYBRID DIAGNOSTIC MODELING AND ITS APPLICATION TO HIERARCHICAL SYSTEM DESIGNS

Eric Gould
DSI International
1574 N. Batavia #3
Orange, CA 92867
(714) 637-9325
egould@dsiintl.com

Abstract – Hybrid Diagnostic Modeling (HDM) is an extension of diagnostic dependency modeling that allows the inter-relationships between a system or device's tests, functions and failure modes to be captured in a single representation (earlier dependency modeling approaches could represent the relationships between tests and either functions or failure modes). With Hybrid Diagnostic Modeling, the same model can be used for early evaluations of a design's diagnostic capability, creation of hierarchical FMECAs, prediction of diagnostic performance, and generation of actual run-time diagnostics. This paper examines issues associated with the application of HDM to hierarchical systems, including: the types of diagnostic inference used to interpret the relationships between functions and failure modes, the correlation of functional and failure-based reliability data, and diagnostic assessment using Hybrid Diagnostic Models.

INTRODUCTION

Dependency modeling was first developed in the 1950s in response to the need for a more rigorous and formal method of developing diagnostics for military equipment and systems. By the 1970s, dependency modeling was employed not only as a technique for diagnostic development, but also as a method for assessing the diagnostic capacity of a design as it was developed. In their earliest manifestations, dependency models represented the relationships between a design's testable events and the functions of the design that are responsible for those events. In later decades, alternative approaches to dependency modeling appeared in which tests were mapped to specific failure modes, rather than to functions, effectively bridging the gap between FMECA analysis and run-time diagnostics. However, because failure-based dependency models cannot be developed until relatively late in the design process (when design implementation details become available), they proved to be less useful at providing early, iterative feedback to diagnostic engineers [1]. By the mid 1990s, diagnostic analysts had begun using both types of dependency models on the same project. Functional dependency models were created in early development phases and iteratively used to assess and improve a proposed diagnostic design. Later, when implementation details were available, the functional dependency models would be converted into failure-based models and then used to predict diagnostic performance, document the diagnostic strategies and, in some cases, generate (failure-oriented) run-time diagnostics [2].

One major drawback of this dual-model approach was that there was no traceability between the two dependency models. Although analysts could refer to the functional model when developing the failure mode model, the process still required two separate modeling efforts. Because there was no direct link, both models had to be updated whenever the design changed (including changes to test definitions). If, on the other hand, the first model were to be abandoned once the second one was developed, then there was the risk that it would have to be redeveloped if, at a later time, the system/device were upgraded or redesigned. Another limitation of conventional (non-hybrid) dependency modeling was that, because all tests had to be defined using the model's constituent elements, test coverage could be defined in terms of functions or failure modes, but not both. This could lead to some tortuous test definitions when system diagnostics were comprised of both failure and function-oriented testing (such as when a system used both embedded diagnostics and external maintenance equipment). Solutions using conventional modeling techniques often required the analyst to assume that the presence of a function is equivalent to the absence of a set of failure modes. While functionality can be thought of as a lack of failure, this relationship becomes more problematic when failure modes can affect more than one function of a component or device.

HYBRID DIAGNOSTIC MODELING

Recognizing the need to address both functional and failure-based testing within a single diagnostic model, DSI International began developing Hybrid Diagnostic Modeling (HDM) techniques in the late 1990s. By 2000, this capability was available in DSI's diagnostic modeling tool ***eXpress***—the first commercial modeling tool to feature HDM (the ***eXpress*** failure mode definition panel is shown in figure 1). Whereas, over the years, there have been various attempts by analysts to include both functions and failure modes within a single dependency model, HDM represents not only the relationships between functions/failure modes and the tests used during diagnostics, but also the causal inter-relationships between failure modes and their affected functions.

In a hybrid diagnostic model, each failure mode definition is comprised of the following data:

- failure mode name
- the percentage of the component failure rate associated with that failure mode
- the functions of that component that are impacted when that failure mode occurs
- the relationship of the failure mode to each affected function (*always affects* vs. *sometimes affects*)

Notice that, in addition to listing the functions affected by each failure mode, a hybrid diagnostic model must also specify the relationships between the failure mode and each of its affected functions. One failure mode may *always affect* a set of functions (in which case, the existence of that failure mode may always be determined by observing *any* of those functions), where another failure mode may *sometimes affect* a set of functions (in which case, *all* of the functions must be observed before that failure mode can be ruled out). The ability to specify that a given failure mode *sometimes affects* a function is particularly useful in situations where the detailed information about the physics of failure is not available—such as for a black box or a Commercial-off-the-Shelf (COTS) device for which BIT coverage percentages are provided, but not a mapping of BIT to functionality.

With HDM, tests can be defined in terms of functions, failure modes, or a combination of the two. This is useful when developing hierarchical system designs. In early top-down models, tests are nearly always defined in terms of functions (since detailed failure mode data is generally not available until the later phases of product design). These functional models can be used to perform iterative assessments of the diagnostic capability of the system *as it is developed*, thereby providing useful design feedback when it is most profitable. As the design matures and implementation details become available, failure modes can be added to models at lower levels of the design and tests defined in terms of those failure modes can be inherited (bottom up) into higher design levels.

One task to which HDM is particularly well suited is the development of system-level testing to supplement a system's Built-in Test (BIT). Most of the large-scale systems developed today contain

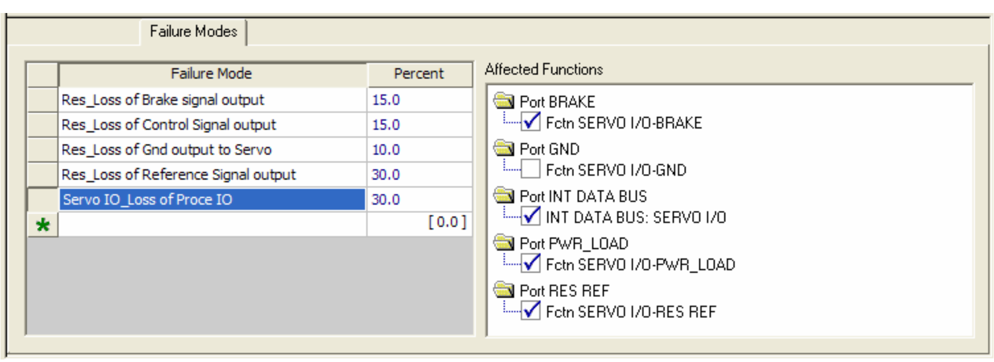

Figure 1. The failure mode definition panel in ***eXpress*** version 5.9

large amounts of self-testing electronics. When the entire system is put together, however, the testing capability of the electronics is expected to provide most, but not all of the desired diagnostic capability. Additional tests need to be developed to account for the areas (both within and without the electronics) not fully tested by system BIT. Hybrid Diagnostic Modeling, by allowing a full functional description of the system to be integrated with failure-oriented BIT test definitions, not only provides a way to determine functional areas of the system that remain untested, but also helps analysts identify the specific test points that are most useful in developing the additional tests.

DIAGNOSTIC REASONING USING HYBRID DIAGNOSTIC MODELING

When applied to hierarchical system designs, a diagnostic reasoner must be able to correlate—at multiple levels of the system design—the diagnostic conclusions associated with one or more tests. In order to support HDM, however, a diagnostic reasoner must also be able to perform inferences between related functions and failure modes. As the reasoner “rules out” the existence of certain failures, it can derive knowledge about the “goodness” of the functions that are affected by those failures. Conversely, if a function is exonerated (determined to be good) during diagnostics, knowledge may be gained about the failure modes associated with that function.

Hybrid Diagnostic Inferences

There are five types of diagnostic inference that are uniquely associated with HDM. Although each rule is relatively simple on its own, they are quite powerful when they are all employed together in a hierarchical diagnostic inference engine. These rules can be grouped into two categories—inferences from failure modes and inferences from functions. We will look at each HDM inference rule individually, using the sample component depicted in Figure 2.

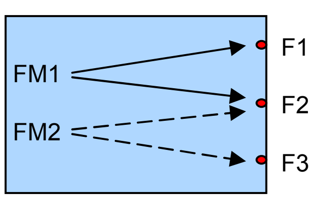

Figure 2. Sample HDM Component

Each of this component’s two failure modes affects two of the component’s three functions. The arrow type (solid or dashed) indicates the relationship between the failure mode and its affected functions. Here, FM1 *always affects* (solid arrows) functions F1 and F2, where FM2 *sometimes affects* (dashed arrows) F2 and F3.

Hybrid Inferences from Failure Modes

When test outcomes during diagnostics result in one or more failure modes being either indicted (called into suspicion) or exonerated (ruled out), an HDM compatible diagnostic reasoner should determine the status of all functions that are directly associated with those failure modes. There are two inference rules that can be used to draw conclusions about functions based on knowledge of failure modes.

HDM Inference Rule #1

When a failure mode is indicted, all unproven functions that are directly affected by that failure mode should also be indicted.

This inference rule states that a function that has not yet been proven good should be considered suspect any time one of the failure modes that directly affect that function is called into suspicion. For the sample component depicted in Figure 2, this rule could be applied as follows:

FM2 is indicted → F2 & F3 are indicted

FM1 is indicted → F1 & F2 are indicted

HDM Inference Rule #2

When all failure modes that directly affect a function are ruled out during diagnostics, then the function should be inferred to be good.

This rule states that a function can be inferred to be good once all failure modes that can directly affect that function have been eliminated from suspicion. This holds true regardless of the relationship (*always affects*, *sometimes affects*) between the function and its failure mode causes. Applying this rule to our sample component, the following inferences are possible:

FM1 is ruled out → F1 is inferred to be good

FM2 is ruled out → F3 is inferred to be good

FM1 & FM2 are ruled out → F1, F2 & F3 are inferred to be good

Hybrid Inferences from Functions

There are three inference rules that a diagnostic reasoner that supports HDM should use to draw conclusions about the status of failure modes based on knowledge of functional status.

HDM Inference Rule #3

When a function is indicted, all unproven failure modes that directly affect that function should also be indicted.

According to this inference rule, a failure mode that has not yet been eliminated from suspicion should be considered suspect any time one of its affected functions is called into suspicion. For the component in Figure 2, the following alternative inferences are possible:

F2 is indicted → FM1 & FM2 are indicted

F1 is indicted → FM1 is indicted

F3 is indicted → FM2 is indicted

HDM Inference Rule #4

*When a function is determined to be good during diagnostics, all failure modes that **always affect** that function should be eliminated from suspicion.*

This inference rule states that the existence of a failure mode can be ruled out when a function that is *always affected* by that failure mode is either proven or inferred to be good during diagnostics. Using our sample component, this rule could be applied as follows:

F1 is proven good → FM1 is ruled out

F2 is proven good → FM1 is ruled out

Notice that FM2 is not inferred to be good when F2 is proven to be good (since FM2 does not *always affect* F2). For the same reason, this inference rule does not apply when F3 is proven good (since both of its related failure modes only *sometimes affect* that function).

HDM Inference Rule #5

*When all functions that are **sometimes affected** by a failure mode are determined to be good during diagnostics, then that failure mode should be eliminated from suspicion.*

This rule states that a failure mode can be ruled out if all functions that are *sometimes affected* by that failure mode have been either proven or inferred to be good during diagnostics.

Using this rule, the following inference is possible for our sample component:

F2 & F3 are proven good → FM2 is ruled out

Chained Hybrid Inferences

For some designs, the addition of failure modes to a functional model may result in differences in the tests used by diagnostics. Tests previously used for fault detection, for instance, may no longer be used at all (even though the test definitions have not been modified). This phenomenon—which can perplex analysts unfamiliar with HDM diagnostic reasoning—results from chained inferences.

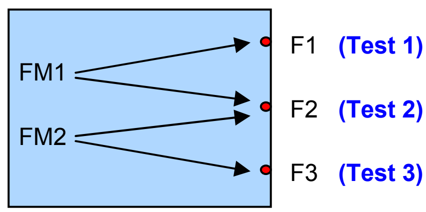

Figure 3. Chained Hybrid Inferences

Consider the component depicted in Figure 3. Prior to adding failure modes, this component would require all three tests to fully diagnose all possible failures (one test per function). Once the two failure modes have been added to the model, however, diagnostics may no longer need all three tests. If Test 1 passes, for instance, diagnostics can determine that F1 is good and infer (inference rule #4) that FM1 has not occurred. If Test 3 were performed next, the diagnostics would learn that F3 is good and rule out FM2. At this point, since both failure modes have been ruled out, the diagnostic reasoner should realize that F2 does not need to be tested (rule #2). Test 2, which was previously needed by the diagnostics, would no longer be necessary. If, on the other hand, diagnostics were to begin with Test 2 and that test were to pass, both failure modes would be ruled out (rule #2) and the other two tests would not be needed for fault detection (although they could still be useful in isolating a fault when Test 2 fails).

Chained hybrid inferences may be difficult to identify in large hierarchical systems, where the diagnostic reasoner performs both hierarchical and hybrid inferences. A function proven good at a relatively high level of the design will result in lower-level "child" functions being inferred good, which in turn may cause both failure modes to be ruled out and other functions to be inferred good, ultimately resulting in seemingly unrelated functions being inferred good at a higher level. This chaining of diagnostic inferences tends to occur in designs where failure modes *always affect* functions, since inference rule #4 is only applied in this situation. When failure modes *sometimes affect* functions, the less aggressive inference rule #5 is used.

The Correlation of Failure Rates for Functions and Failure Modes

As an HDM compatible diagnostic engine derives knowledge about the status of individual functions and/or failure modes, it must update failure rates to reflect that knowledge. If, for instance, it is determined that a given failure mode does not exist, the failure rates of all functions directly affected by that failure mode (and which have not yet been inferred to be good) are updated to reflect their reduced likelihood of having failed. Conversely, if a function is proven good during diagnostics, the failure rates of failure modes that affect that function (and which have not been eliminated from suspicion) should be updated to reflect their reduced likelihood of having occurred.

Hybrid diagnostic models (particularly those that represent hierarchical designs) frequently contain separate sets of reliability data for functions and failure modes. Functional failure rates may have been derived as apportionments of local or higher-level component failure rates, or as roll-ups of lower-level component failure rates. Failure mode reliability figures, on the other hand, are typically calculated as a percentage of a component failure rate. Because the two sets of reliability data come from different sources, it is possible for them to be in conflict. Consider the example in Figure 4:

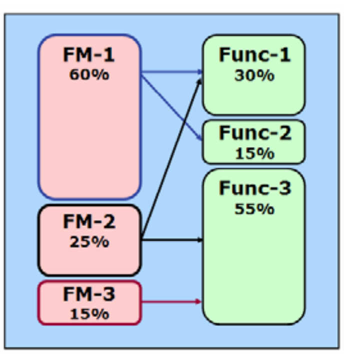

Figure 4. Conflicting Reliability Data

Notice that, for this component, failure mode FM-1 represents 60% of the component failure rate, yet the two functions (Func-1 and Func-2) affected by that failure mode collectively only represent 45% of the component failure rate. Furthermore, FM-1 is responsible for only part of the failure rate for Func-1, since FM-2 also affects that function. Before HDM-based diagnostics can update failure probabilities for this component, it must have a way of correlating this conflicting reliability data.

In DSI's ***eXpress*** software, analysts can select from three methods of mapping between failure mode and functional failure rates. These three methods—*failure mode apportionment, failure mode precedence* and *functional precedence*—are representative of the different ways in which failure mode and functional reliability data might be correlated by an HDM compatible diagnostic reasoner.

Failure Mode Apportionment

When failure rates are correlated using *failure mode apportionment,* the functional reliability values are recalculated by splitting up the failure mode rates equally among all of their affected functions (the rates for the failure modes are left unchanged). Although this is the simplest of the three approaches, the original functional reliability data is completely ignored—the adjusted function probabilities are derived entirely from the failure modes that affect them. Figure 5 depicts the relative percentages that would result if the failure rates for the item in Figure 4 were to be adjusted using *failure mode apportionment*. The failure mode rates have been equally split among their affected functions. Thus Func-1 is now allotted 42.5% of the component failure rate—30% from FM-1 and 12.5% from FM-2 (the dotted lines in this figure show the portion of the functional failure rate that is contributed by each failure mode).

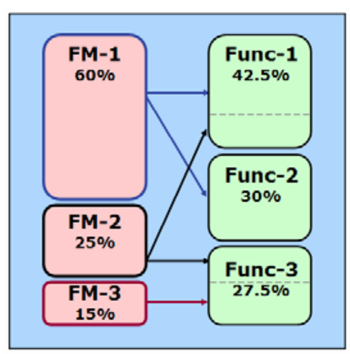

Figure 5. Failure Mode Apportionment

This approach is particularly useful for low-level hybrid models in which functional reliability data has not been developed. If, however, a model contains functional failure probabilities (including those rolled up from lower-level models in a hierarchical design), then the analyst may wish to employ one of the other two methods, each of which takes both failure mode and function failure rates into consideration.

Failure Mode Precedence

When the *failure mode precedence* method is used to correlate failure rates, the failure mode rates are left unchanged, whereas the functional reliability values are adjusted so that they can be mapped to the probabilities of the failure modes that affect them. Although the ratios between functional reliability values are taken into consideration, the failure mode reliability data takes precedence. Figure 6 depicts the relative percentages that would result if the failure rates for the component in Figure 4 were adjusted using *failure mode precedence*.

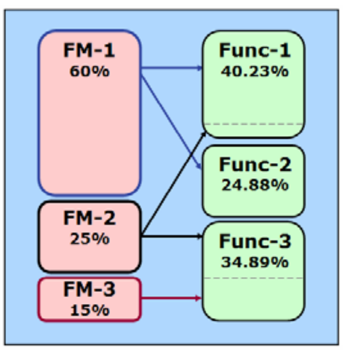

Figure 6. Failure Mode Precedence

Notice that, with this method, a larger portion of the failure rate was allocated to Func-3 than was with *failure mode apportionment* (34.89%, rather than 27.5% in Figure 5). F*ailure mode precedence* takes into consideration the original ratios when adjusting the function failure rates (Func-3 was originally allocated 55% of the item failure rate). Conversely, Func-2 (which originally represented only 15% of the failure rate) has been adjusted to 24.88% (rather than 30% in Figure 5).

Note also the different percentages of the function failure rates contributed by each failure mode (indicated by the dotted lines). FM-2, for example, contributes only a tiny portion of the failure rate for Func-1. There are two reasons for this. First of all, FM-2 constitutes a small portion of the overall failure rate in comparison to FM-1, the other contributor to Func-1. Secondly, the majority of the failure rate for FM-2 is allocated to Func-3, which had a higher initial failure ratio (55%) than did Func-1 (30%).

Failure mode apportionment can be performed using the following steps:

1. Compute, if necessary, the raw failure rate for each function and failure.
2. Compute distributed failure rates by splitting up the raw functional failure rates among the failure modes that affect them (using the total failure rate of each failure mode to determine the proportions).
3. Compute partial failure rates by scaling the distributed failure rates (for all functions affected by a given failure mode) so that they add up to the failure rate of that failure mode.
4. Compute the adjusted functional failure rate by adding up all of the partial failure rates associated with that function.

If we assume, for simplicity's sake, that the component in Figures 4, 5 and 6 has a failure rate of 100.0 (100 failures per million hours), the raw functional and failure mode failure rates can be easily calculated (step 1). The functional failure rates are next distributed among the failure modes that affect them (step 2), using the full failure rate of each failure mode to determine the proportion. The results are shown in Table 1.

Functions (w/ Failure Rates)	FMs Affecting Function (w/ Failure Rates)	Relative Pctgs.	Distrib. Failure Rates
Func-1 (30.0)	FM-1 (60.0)	70.59%	21.1765
	FM-2 (25.0)	29.41%	8.8235
Func-2 (15.0)	FM-1 (60.0)	100.0%	15.0000
Func-3 (55.0)	FM-2 (25.0)	62.50%	34.3750
	FM-3 (15.0)	37.50%	20.6250

Table 1: Failure Mode Precedence (steps 1–2)

Based on the ratio between the failure rates of FM-1 and FM-2 (the two failure modes that can affect Func-1), 70.59% of Func-1's failure rate is distributed to FM-1, whereas 29.41% is distributed to FM-2. The next step (step 3) is to re-scale the distributed failure rates so that the values for each failure mode add up to the failure rate of that failure mode. The results are depicted in Table 2.

Failure Modes (w/ FR)	Affected Functions	Distrib. Failure Rates	Partial Failure Rates
FM-1 (60.0)	Func-1	21.1765	35.1220
	Func-2	15.0000	24.8780
FM-2 (25.0)	Func-1	8.8235	5.1064
	Func-3	34.3750	19.8936
FM-3 (15.0)	Func-3	20.6250	15.0000

Table 2: Failure Mode Precedence (step 3)

The distributed failure rates for FM-1 (21.1765 & 15.0000) have been re-scaled, keeping the same proportions, so that they add up to the failure rate of that failure mode (35.1220 + 24.8780 = 60.0). The final step is to sum the partial failure rates to get the adjusted functional failure rate (Table 3).

Functions	FMs Affecting Function	Partial Failure Rates	Adjusted Failure Rates
Func-1	FM-1	35.1220	40.2284
	FM-2	5.1064	
Func-2	FM-1	24.8780	24.8780
Func-3	FM-2	19.8936	34.8936
	FM-3	15.0000	

Table 3: Failure Mode Precedence (step 4)

Failure mode precedence should be used when the analyst wishes to consider the failure mode ratios to be more accurate than the design's functional failure rates, yet does not wish to discard all of the model's knowledge about the relative reliability of the different functions. If the analyst prefers functional reliability data over the failure mode ratios, then a third correlation method should be used—*functional precedence*.

Functional Precedence

Functional precedence, unlike the previous two approaches to failure rate correlation, does not modify the functional failure rates. Instead, this method adjusts the failure mode reliability values so that they can be mapped to the probabilities of their affected functions. Although the failure mode reliability data is taken into account, the functional failure rates take precedence. Figure 7 depicts the use of *functional precedence* to adjust the failure rates for the component depicted in Figure 4.

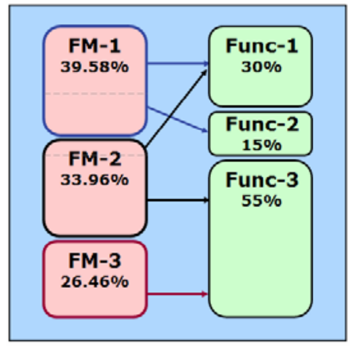

Figure 7. Functional Precedence

The following steps can be used to adjust failure rates using *functional precedence*:

1. Compute, if necessary, the raw failure rate for each function and failure mode.
2. Compute distributed failure rates by splitting up the raw failure mode rate among its affected functions (using the total failure rate of each function to determine the proportions).
3. Compute partial failure rates by scaling the distributed failure rates (for all FMs that can affect a given function) so that they add up to the functional failure rate.
4. Compute the adjusted failure mode rate by adding up all partial failure rates associated with that FM.

Failure Probabilities in FD/FI Metrics

When Hybrid Diagnostic Modeling is used to predict the diagnostic performance of a system, fault detection and isolation (FD/FI) metrics are calculated using a combination of full and partial failure probabilities. For failure modes that *always affect* functions, FD/FI probabilities are calculated using the failure mode's entire failure probability. Failure modes that *sometimes affect* functions, however, are only partially implicated when one of those functions is called into suspicion (and only partially ruled out when one of those functions is proven good). Here, FD/FI metrics are based on the partial failure rates (the fourth column of Table 2) that correspond to that function-failure pairing.

REFERENCES

[1] "A Short History of Diagnostic Modeling." http://www.Testability.com

[2] "STAT User's Group'94." *STAT Newsletter*, Vol. 4, No. 3, December 1994, pp. 4-5.