System Testability Assessment for Integrated Diagnostics

WILLIAM R. SIMPSON
JOHN W. SHEPPARD
Arinc Research Corp.

In part 3 of their series on integrated diagnostics, the authors present techniques for analyzing the testability of a system. Based on the information flow model detailed in the second article, the measures presented here identify testability problems involving ambiguity, feedback, the test set, and multiple failures.

In this series of articles, we are interested in the ability to diagnose failures as part of an overall maintenance architecture. Testability is a means to that end. Its only purpose is to improve system maintenance and repair. Testability is a yardstick by which we measure our success in achieving design goals for various aspects of field maintenance. As we shall demonstrate, testability is not a single issue but comprises several issues involved in maintaining complex systems.

In the first article of the series,[1] we presented an overview of the problem of analyzing testability and conducting diagnosis for complex systems. In our second article,[2] we described in detail the form of an information flow model, including test representations, conclusions, logical interdependencies, groupings, and special logical constructs. Several tools[3,4] incorporate this modeling approach, and we will use the information flow model as the basis of discussion in this and future articles. We now describe several measures that are useful in analyzing system testability.

Model representation

A key element in the computation of testability is the information flow model, which represents the flow of diagnostic information by means of a network of logical constructs. For illustration we use the case study of an antitank missile launcher, introduced in our previous article. Figure 1 is a dependency model of the missile launcher system, and Figure 2 on p. 42 is a matrix representation of the same system. Because the representation is a bit matrix, we can replace any letter with a 1 and any empty area with a 0. The letters represent the method by which the relationships between elements (indicated by the row and column) were derived:

- f: first-order (or input) relationships
- h: higher order relationships derived by transitive closure
- l: higher order relationships derived by logical closure
- n: higher order relationships derived by incremental closure

With one exception (feedback analysis, which we will discuss later), it is not important what the letter designation is, only that a relationship exists.

Terminology

Before we can describe the mathematics, we must define the terminology.

0740-7475/92/0300-0040$03.00 © 1992 IEEE

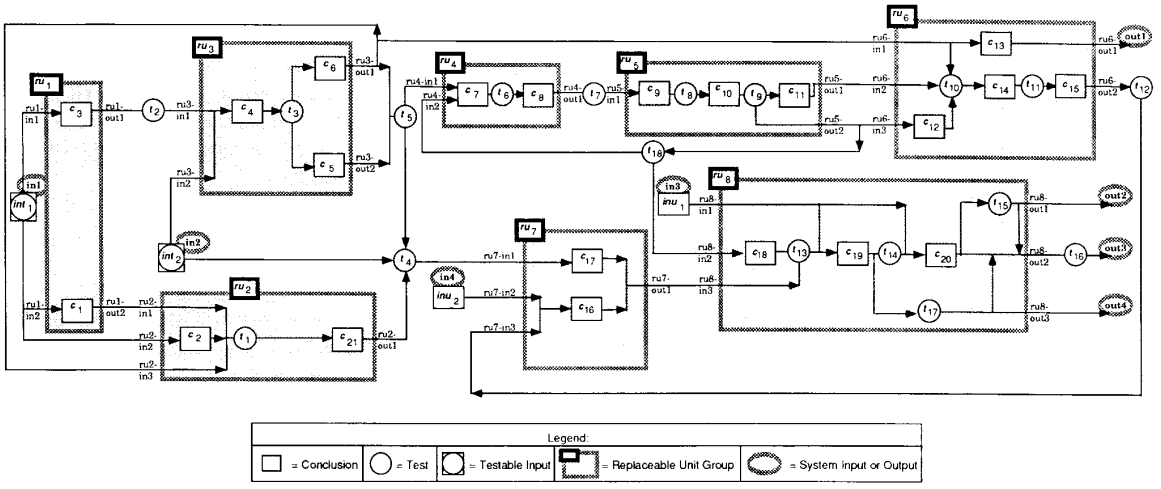

Figure 1. *Dependency diagram of antitank missile launcher case study.*

In general, we use a lowercase letter to denote an individual member of a set and give the letter a subscript to indicate which member; thus, c_{15} is the 15th member of the conclusion set. An uppercase letter denotes the entire set, and the cardinality symbol (for example, $|\mathbf{X}|$) indicates the number of set members. Several sets are of interest:

- **C**: the set of fault isolation conclusions (not including inputs or *No Fault*).
- **T**: the set of tests that can be evaluated (not including testable inputs).
- **IN**: the set of inputs to the system being evaluated. There are two input sets: **INU** is the set of untestable inputs, and **INT** is the set of testable inputs ($\mathbf{IN} = \mathbf{INU} \cup \mathbf{INT}$).
- **NF**: the set containing the element *No Fault* and for which the cardinality, $|\mathbf{NF}|$, is 1.
- **I**: the set of information sources, including the tests and testable inputs ($\mathbf{I} = \mathbf{T} \cup \mathbf{INT}$).
- **F**: the set of fault isolation conclusions, including conclusions, inputs, and *No Fault* ($\mathbf{F} = \mathbf{C} \cup \mathbf{IN} \cup \mathbf{NF}$).
- **E**: the set of elements in the model, including all the elements in **F** and **I** ($\mathbf{E} = \mathbf{F} \cup \mathbf{I}$). Note that the cardinality of **E**, $|\mathbf{E}|$, cannot be computed as a sum of the cardinalities of **F** and **I** ($|\mathbf{E}| \neq |\mathbf{F}| + |\mathbf{I}|$) because testable inputs belong to both **F** and **I** (that is, $\mathbf{F} \cap \mathbf{I} = \mathbf{INT}$).
- **U**: the set of elements in the model that have unique properties. There are two unique sets: **UI** is the set of unique information sources, and **UF** is the set of unique fault isolation conclusions ($\mathbf{U} = \mathbf{UI} \cup \mathbf{UF}$).
- **RU**: the set of replaceable unit groups in the model.

To define uniqueness, we must first define two vectors: failure signature ($\vec{\mathbf{SF}}$) and test signature ($\vec{\mathbf{ST}}_j$).

A failure signature is a vector associated with a specific element in **F** that indicates all the tests that depend on f_j. The vector corresponds to specific rows in the test-to-conclusion dependency matrix. Thus, from row c_3 in Figure 2,

$$\vec{\mathbf{SF}}_{c_3} = \begin{pmatrix} 1,1,1,1,1,1,1,1,1,1, \\ 1,1,1,1,1,1,1,1,0,0 \end{pmatrix} \quad (1)$$

and from row c_{11} in Figure 2,

$$\vec{\mathbf{SF}}_{c_{11}} = \begin{pmatrix} 0,0,0,0,0,0,0,0,0,1, \\ 1,1,1,1,1,1,1,0,0,0 \end{pmatrix} \quad (2)$$

We call this vector a failure signature because if the tests corresponding to row entries in the test-to-conclusion dependency matrix were to fail, we would expect the element corresponding to that row to fail. That is, if c_3 fails,

$$c_3 \supset t_1 \wedge t_2 \wedge t_3 \wedge t_4 \wedge t_5 \wedge t_6 \wedge t_7 \wedge t_8 \wedge t_9 \wedge t_{10} \wedge t_{11} \wedge t_{12} \wedge t_{13} \wedge t_{14} \wedge t_{15} \wedge t_{16} \wedge t_{17} \wedge t_{18} \quad (3)$$

and if c_{11} fails,

$$c_{11} \supset t_{10} \wedge t_{11} \wedge t_{12} \wedge t_{13} \wedge t_{14} \wedge t_{15} \wedge t_{16} \wedge t_{17} \wedge t_{18} \quad (4)$$

Tests may also have failure signatures. For example,

$$\vec{\mathbf{SF}}_{t_1} = \begin{pmatrix} 0,0,0,1,0,0,0,0,0,0, \\ 0,0,1,1,1,1,1,1,0,0 \end{pmatrix} \quad (5)$$

or

$$t_1 \supset t_4 \wedge t_{13} \wedge t_{14} \wedge t_{15} \wedge t_{16} \wedge t_{17} \quad (6)$$

Say that we allow the orientation of both dependency matrices to define an

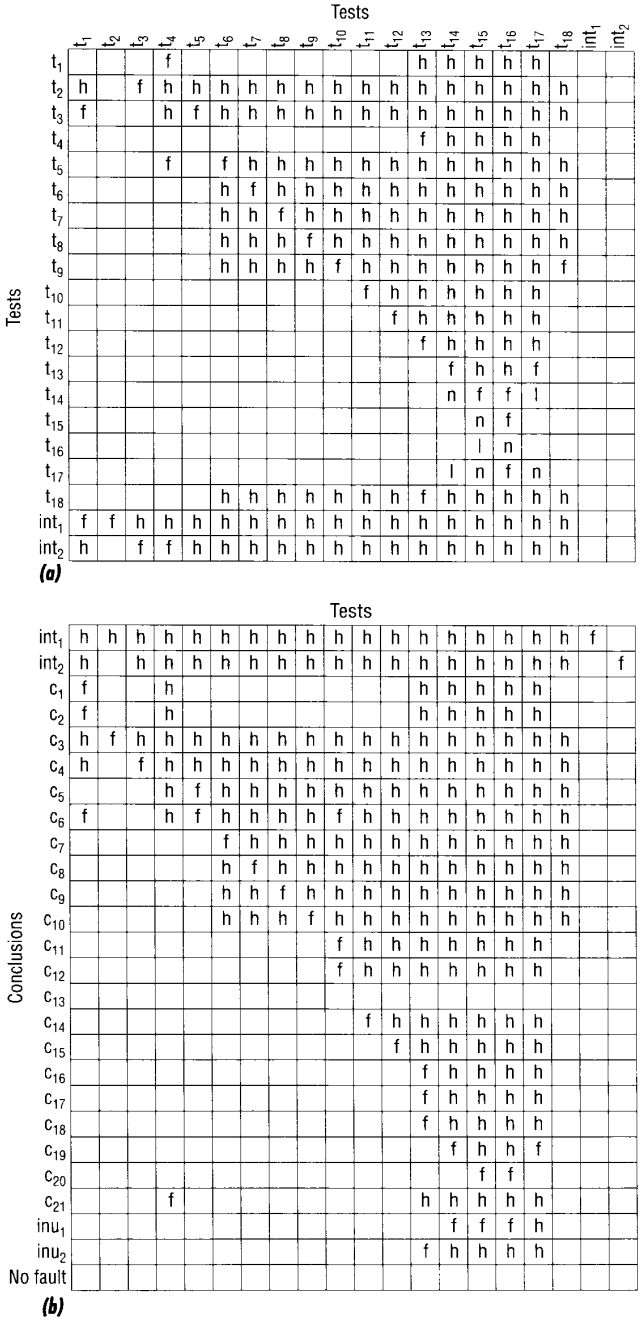

Figure 2. Closed dependency matrices of case study system: test-to-test dependency matrix (a) and test-to-conclusion dependency matrix (b).

ordering on **F** and **I**. Then any element in **F** is a member of **UF** if the element is not preceded by another member having a failure signature equal to that of the element:

$$f_j \in \mathbf{UF} \text{ iff } \vec{\mathbf{SF}}_k \neq \vec{\mathbf{SF}}_j, \quad \forall k \in (0, j), f_k, f_j \in \mathbf{F} \tag{7}$$

Under this formulation, the first occurrence of any failure signature is unique, and subsequent occurrences are not unique. Nonunique elements may be associated with the unique element whose signature they match. We do not consider the corresponding case for tests except when considering feedback.

A test signature is a vector associated with a specific element in **I**. It is a mapping of a specific column in both matrices. Thus, from Figure 2,

$$\vec{\mathbf{ST}}_{t_7} = \begin{pmatrix} 0,1,1,0,1,1,1,1,1,0,0,0,0,0,0, \\ 0,0,1,1,1,1,1,1,0,1,1,1,1,1,1, \\ 1,1,0,0,0,0,0,0,0,0,0, \\ 0,0,0,0,0,0 \end{pmatrix} \tag{8}$$

Any element in **I** is a member of **UI** if the element is not preceded by another member having a failure signature equal to that of the element:

$$i_j \in \mathbf{UI} \text{ iff } \vec{\mathbf{ST}}_k \neq \vec{\mathbf{ST}}_j, \quad \forall k \in (0, j), i_k, i_j \in \mathbf{I} \tag{9}$$

Under this formulation, the first occurrence of any test signature is unique, and subsequent occurrences are not unique. Nonunique elements may be associated with the unique element whose signature they match.

The construction of matrix **D** will determine the order in which we evaluate the dependency structures and thus which member of a group of nonunique elements we declare to be unique. However, the construction of the matrix does not affect the members of the group or the total number of unique dependency structures. Table 1 lists the

Table 1. Set memberships in the case study.

Set Label	E	I	F	C	T	IN	INU	INT	NF	U	UI	UF
Cardinality	|E| = 44	|I| = 20	|F| = 26	|C| = 21	|T| = 18	|IN| = 4	|INU| = 2	|INT| = 2	|NF| = 1	|U| = 28	|UI| = 14	|UF| = 16
Set Members	$e_i \in$ E	$e_i \in$ I	$e_i \in$ F	$e_i \in$ C	$e_i \in$ T	$e_i \in$ IN	$e_i \in$ INU	$e_i \in$ INT	$e_i \in$ NF	$e_i \in$ U	$e_i \in$ UI	$e_i \in$ UF
	t_1 c_1	t_1	int_1	c_1	t_1	int_1	inu_1	int_1	*No Fault*	t_1 c_1	t_1	int_1
	t_2 c_2	t_2	int_2	c_2	t_2	int_2	inu_2	int_2		t_2 c_3	t_2	int_2
	t_3 c_3	t_3	c_1	c_3	t_3	inu_1				t_3 c_4	t_3	c_1
	t_4 c_4	t_4	c_2	c_4	t_4	inu_2				t_4 c_5	t_4	c_3
	t_5 c_5	t_5	c_3	c_5	t_5					t_5 c_6	t_5	c_4
	t_6 c_6	t_6	c_4	c_6	t_6					t_6 c_7	t_6	c_5
	t_7 c_7	t_7	c_5	c_7	t_7					t_{10} c_{11}	t_{10}	c_6
	t_8 c_8	t_8	c_6	c_8	t_8					t_{11} c_{12}	t_{11}	c_7
	t_9 c_9	t_9	c_7	c_9	t_9					t_{12} c_{14}	t_{12}	c_{11}
	t_{10} c_{10}	t_{10}	c_8	c_{10}	t_{10}					t_{13} c_{16}	t_{13}	c_{13}
	t_{11} c_{11}	t_{11}	c_9	c_{11}	t_{11}					t_{14} c_{18}	t_{14}	c_{14}
	t_{12} c_{12}	t_{12}	c_{10}	c_{12}	t_{12}					t_{15} c_{19}	t_{15}	c_{15}
	t_{13} c_{13}	t_{13}	c_{11}	c_{13}	t_{13}					int_1 c_{20}	int_1	c_{16}
	t_{14} c_{14}	t_{14}	c_{12}	c_{14}	t_{14}					int_2 c_{21}	int_2	c_{19}
	t_{15} c_{15}	t_{15}	c_{13}	c_{15}	t_{15}							c_{20}
	t_{16} c_{16}	t_{16}	c_{14}	c_{16}	t_{16}							c_{21}
	t_{17} c_{17}	t_{17}	c_{15}	c_{17}	t_{17}							
	t_{18} c_{18}	t_{18}	c_{16}	c_{18}	t_{18}							
	int_1 c_{19}	int_1	c_{17}	c_{19}								
	int_2 c_{20}	int_2	c_{18}	c_{20}								
	c_{21}		c_{19}	c_{21}								
	inu_1		c_{20}									
	inu_2		c_{21}									
	No Fault		inu_1									
			inu_2									
			No Fault									

set memberships as they apply to the case study.

Groups are subsets of the set of elements that are not otherwise assigned to one of the labeled sets. They are mapped in accordance with the second article of this series. Group types include test, replaceable unit, multiple-failure, ambiguity, redundant-test, and feedback.

Computing testability

Using the information flow model, we can compute values for a number of measures associated with the ability to diagnose failures. Some of these measures concern maintenance factors observable in the field and were previously available only after a fault tree was developed. Because the information flow model incorporates the required system maintenance data, we can compute these measures without developing a fault tree. The following paragraphs examine measures that address ambiguity, feedback, the test set, and multiple failure. (Recall that our analysis assumes a single failure.[1])

Ambiguity measures

Ambiguity exists when the tests provided in the information flow model cannot distinguish between two or more conclusions. Given that a group may contain one or more elements, an ambiguity group of cardinality 1 does not contribute to an ambiguity problem. The ability to distinguish among conclusions in the conclusion set is related to the failure signature $\vec{SF}_j$. Ambiguous conclusions have identical failure signatures. Therefore, no combination of existing tests can distinguish among ambiguous conclusions.

Figure 3 on the next page shows two example ambiguity groups in the case study. Table 2 lists all the ambiguity groups in the case study. There are six ambiguity groups, each of which may or may not be significant (that is, require

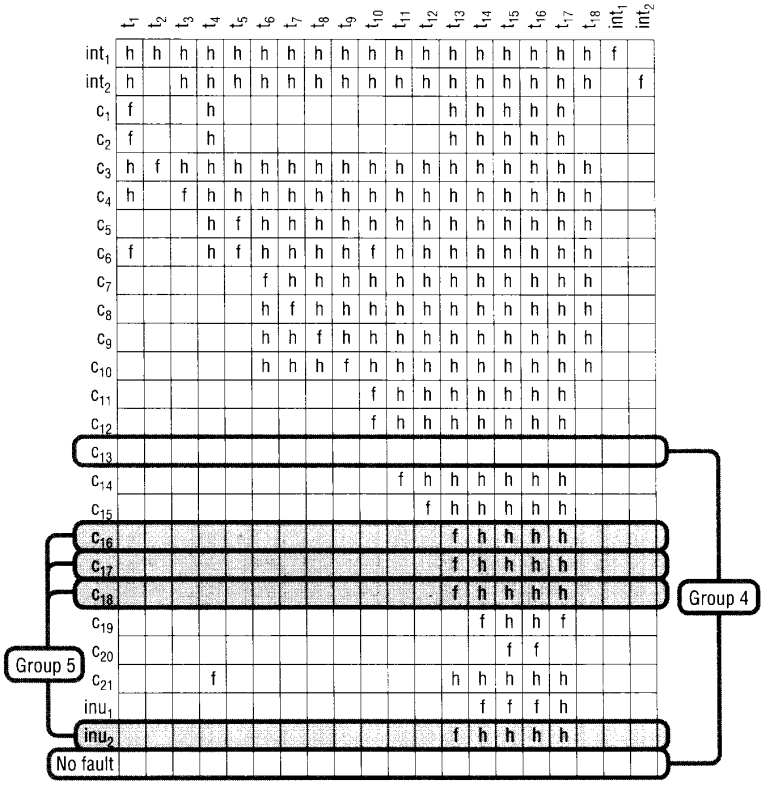

Figure 3. Case study ambiguity analysis.

design changes) for meeting maintenance requirements. We have derived a number of measures to indicate the amount and type of ambiguity.

Isolation level. IL is the ratio of the number of isolatable groups to the number of isolatable elements. In our definition of unique conclusions, we defined the first element of each ambiguity group as being unique. Thus, the number of isolatable groups is $|\mathbf{UF}|$, and

$$IL = \frac{|\mathbf{UF}|}{|\mathbf{F}|} \tag{10}$$

The ideal value of IL would be 1.0000. For the case study, IL = 16/26 = 0.6154. Roughly 62% of the conclusions available can be drawn uniquely, which may or may not be a problem. If isolation to the element level is the design goal, there are serious problems in this case study. But if isolation to the group level is the goal, it may not be important that we achieve an isolation level of only 61%. The next measure clarifies the difference between the element level and the group level.

Operational isolation. A system's operational isolation level (OI[n]) is the percentage of observed faults that result in isolation to n or fewer replaceable units. To compute this measure, we must determine the number of replaceable units associated (ambiguous) with each conclusion in the model (α). For fault isolation conclusion f_i,

Table 2. Ambiguity groups in the case study.

Group*	Members
1	c_1 c_2
2**	c_7 c_8 c_9 c_{10}
3	c_{11} c_{12}
4	c_{13} *No Fault*
5	c_{16} c_{17} c_{18} inu_2
6	c_{19} inu_1

*Group data structure is in accordance with Sheppard and Simpson.[2]
**Ambiguity resulting from feedback.

$$\alpha_i = \sum_{k=1}^{|\mathbf{RU}|} \beta_k;$$
$$\beta_k = \begin{cases} 1; & \exists f_j \in \mathbf{RU}_k \ni \tilde{\mathbf{SF}}_j = \tilde{\mathbf{SF}}_i \\ 0; & \text{otherwise} \end{cases} \tag{11}$$

Here $|\mathbf{RU}|$ represents the cardinality of the set of replaceable unit groups, and $\mathbf{RU}_k$ is the kth replaceable unit group. As shown in Figure 1, we have 13 replaceable unit groups (eight are the shaded defined groups, and five are the ungrouped conclusions: int_1, int_2, inu_1, inu_2, and *No Fault*.) Table 3 provides the data necessary to compute α_i. The operational isolation is

$$OI[n] = \frac{\sum_{i=1}^{|\mathbf{F}|} w_i \gamma_i}{|\mathbf{K}|};$$
$$\gamma_i = \begin{cases} 1; & \alpha_i \geq n, \forall f_j \in \mathbf{K} \\ 0; & \text{otherwise} \end{cases} \tag{12}$$

Here w_i is a weighting factor associated with each fault isolation conclusion (usually the probability of occurrence), and $\mathbf{K}$ is a subset of $\mathbf{F}$ determined on the basis of the type of analysis being performed. A project office may specify operational isolation or something similar as part of the design criteria. The ideal value of OI[n] would be 1.0000 for every definition of operational isolation. Table 4 shows variations of the operational

isolation measures for the case study. Which operational isolation value is used depends on several factors, including the wording of specifications. For example, the last column of Table 4 shows values that include failure rate weighting but exclude inputs and *No Fault*. Thus, failures of inputs are not the responsibility of system testability, and nondetections (discussed in the next section) are not included in the calculation (they may be penalized separately).

Nondetection (ND). Of the six ambiguity groups in the case study (listed in Table 2), the fourth is of interest for nondetections. The ambiguity between c_{13} and *No Fault* indicates that we cannot detect the failure of c_{13} with the defined set of tests. Because no test depends on *No Fault*, $\bar{\mathbf{SF}}_{No\ Fault} = (0, 0, ..., 0)$. Because any conclusion ambiguous with *No Fault* must have the same failure signature as *No Fault*, no tests in our test set detect a failure of c_{13}. Thus, c_{13} is a nondetection item. We obtain a measure of nondetection by enumerating the occurrences of nondetections:

$$ND = \frac{\left(\sum_{i=1}^{|\mathbf{F}|}\delta_i\right) - 1}{|\mathbf{F}|};$$

$$\delta_i = \begin{cases} 1; \ \bar{\mathbf{SF}}_i = \bar{\mathbf{SF}}_{No\ Fault} \\ \quad \text{and } f_i \in \mathbf{F} \\ 0; \ \text{otherwise} \end{cases} \qquad \textbf{(13)}$$

Table 3. *Replaceable unit ambiguity groups in the case study.*

Failed element	Isolation ambiguity	Replaceable unit groups	α_i (Equation 11)	Failure frequency*
int_1	int_1	int_1	1	0.0010
int_2	int_2	int_2	1	0.0010
c_1	c_1 c_2	ru_1 ru_2	2	0.0005
c_2	c_1 c_2	ru_1 ru_2	2	0.0005
c_3	c_3	ru_1	1	0.0100
c_4	c_4	ru_3	1	0.0100
c_5	c_5	ru_3	1	0.0100
c_6	c_6	ru_3	1	0.0100
c_7	c_7 c_8 c_9 c_{10}	ru_4 ru_5	2	0.0005
c_8	c_7 c_8 c_9 c_{10}	ru_4 ru_5	2	0.0005
c_9	c_7 c_8 c_9 c_{10}	ru_4 ru_5	2	0.0005
c_{10}	c_7 c_8 c_9 c_{10}	ru_4 ru_5	2	0.0005
c_{11}	c_{11} c_{12}	ru_5 ru_6	2	0.0005
c_{12}	c_{11} c_{12}	ru_5 ru_6	2	0.0005
c_{13}	c_{13} ***No Fault***	ru_6 ***No Fault***	2	0.0005
c_{14}	c_{14}	ru_6	1	0.0100
c_{15}	c_{15}	ru_6	1	0.0100
c_{16}	c_{16} c_{17} c_{18} inu_2	ru_7 ru_8 inu_2	3	0.0005
c_{17}	c_{16} c_{17} c_{18} inu_2	ru_7 ru_8 inu_2	3	0.0005
c_{18}	c_{16} c_{17} c_{18} inu_2	ru_7 ru_8 inu_2	3	0.0005
c_{19}	c_{19} inu_1	ru_8 inu_1	2	0.0005
c_{20}	c_{20}	ru_8	1	0.0100
c_{21}	c_{21}	ru_2	1	0.0100
inu_1	c_{19} inu_1	ru_8 inu_1	2	0.0010
inu_2	c_{16} c_{17} c_{18} inu_2	ru_7 ru_8 inu_2	3	0.0010
No Fault	c_{13} ***No Fault***	ru_6 ***No Fault***	2	0.9095**

*Values taken from Sheppard and Simpson[2] in units of failures per 10,000 hours.
**Analysis goal of operational performance check (expected frequency of occurrence).

Table 4. *Operational isolation values in the case study.*

Operational isolation	w_i = uniform K = F	w_i = uniform K = F – NF	w_i = failure prob.* K = F	w_i = failure prob.* K = F – NF	w_i = failure prob.* K = F – IN	w_i = failure prob.* K = F – IN – NF
OI(1)	0.3846	0.4400	0.0811	0.9061	0.0808	0.9364
OI(2)	0.8462	0.8400	0.9975	0.9724	1.0000	1.0000
OI(3)	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

*Failure probability computed as $\lambda_i \left(\sum_{i=1}^{|\mathbf{K}|}\lambda_i\right)^{-1} = P_i$ where P_i is the failure probability of the ith conclusion and λ_i is the failure frequency of the ith conclusion.

The 1 in the numerator excludes the *No Fault* conclusion, which is only technically a nondetection.

The ideal value of ND would be 0.0000. For the case study, ND = (2 – 1)/26 = 0.0385.

In analyzing the impact of nondetections on the other measures, note that they will result in ambiguity between two or more replaceable unit groups, assuming *No Fault* is treated as a separate replaceable unit group. The case study illustrates the effect of nondetection on operational isolation. As shown in Table 4, when the probability of a conclusion's being drawn is $1/|\mathbf{F}|$, removing the *No Fault* conclusion from the analysis set **K** only slightly affects the operational isolation. However, when a high probability of drawing the *No Fault* conclusion exists, a significant difference in operational isolation values exists. The exact form of operational isolation is critical when we check for specification compliance, and we should control the computation to meet the letter of the specification.

Feedback measures

Feedback in an information flow model represents a logical circularity. Such circularities may result from physical feedback, information flow feedback, or modeling errors. When we analyze information flow models, topological circularities most often correlate to physical feedback loops. We can determine the topological circularities for the case study by examining Figure 2. From the matrix representation, t_i is in feedback if and only if $\hat{\mathbf{D}}_{ii} \neq 0$, and it is in topological feedback if and only if $\hat{\mathbf{D}}_{ii}$ = "h" or "f." For all other cases, $\hat{\mathbf{D}}_{ii} \neq 0$ indicates the test is in logical feedback. Thus, t_6, t_7, t_8, t_9, and t_{18} are in topological feedback, and t_{14}, t_{15}, t_{16}, and t_{17} are in logical feedback. Further, the feedback loop, or circularity, has a unique failure signature. Therefore, any element in **F** having the same failure signature as a test in feedback is a member of the same feedback loop as the test. In addition, any two tests in feedback having the same failure signature are members of the same feedback loop. Table 5 shows the circularity groups detected in the case study. Group 1 corresponds to topological feedback, which is identified by the "h" in the diagonal cell of $\hat{\mathbf{D}}$ for each of the member tests in the table. We can verify the members of this feedback loop, including the conclusion elements, by carefully inspecting Figure 1.

Table 5. Circularity groups for the case study.

Group*	Failure signature ($\bar{SF}_i$)	Members
1**	(0,0,0,0,0,1,1,1,1,1, 1,1,1,1,1,1,1,1,0,0)	t_6 t_7 t_8 t_9 t_{18} c_7 c_8 c_9 c_{10}
2***	(0,0,0,0,0,0,0,0,0,0, 0,0,0,1,1,1,1,0,0,0)	t_{14} t_{17} c_{19} inu_1
3	(0,0,0,0,0,0,0,0,0,0, 0,0,0,0,1,1,0,0,0,0)	t_{15} t_{16} c_{20}

* *Group data structure is in accordance with Sheppard and Simpson.*[2]

** *Circularity is topological feedback.*

*** t_{13} *has the same signature, but* $\hat{\mathbf{D}}_{13,13} = 0$.

The following measures indicate the amount and type of circularity. Because of the close relationship between topological circularity and physical feedback, we have restricted these measures to topological circularity.

Test feedback dominance. TFD is the fraction of information sources involved in topological feedback.

$$\text{TFD} = \frac{\sum_{i=1}^{|\mathbf{I}|} \rho_i}{|\mathbf{I}|}; \quad \rho_i = \begin{cases} 1; & \hat{\mathbf{D}}'_{ii} = 1; \\ & i_i \in \mathbf{I} \\ 0; & \text{otherwise} \end{cases} \tag{14}$$

where $\hat{\mathbf{D}}'$ is the dependency matrix following transitive closure but before logical closure (see our previous article[2] for a complete discussion of the closure algorithms).

The ideal value of TFD would be 0.0000 (no feedback). For the case study, TFD = 5/20 = 0.2500. In other words, 25% of the information sources are tied up in feedback. In this case, only one feedback loop exists.

Component feedback dominance. CFD is the fraction of conclusions involved in topological feedback:

$$\text{CFD} = \frac{\sum_{i=1}^{|\mathbf{F}|} \xi_i}{|\mathbf{F}|};$$

$$\xi_i = \begin{cases} 1; \exists j \in \mathbf{I}\left(\bar{\mathbf{SF}}_i = \bar{\mathbf{SF}}_j\right) \wedge \\ \quad \hat{\mathbf{D}}'_{jj} = 1; f_i \in \mathbf{F} \\ 0; \text{ otherwise} \end{cases} \tag{15}$$

CFD is important because it is a measure of ambiguity caused by feedback. Recall that each failure signature is the same in a circularity; that is also the definition of ambiguity.

The ideal value of CFD would be 0.0000 (no feedback). For the case study, CFD = 4/26 = 0.1538. In other words, 15% of the conclusions are tied up in feedback. Again, only one feedback loop exists.

Feedback modifications. Although feedback is a testability problem, physi-

cal feedback often is necessary for the system to perform. It is important that changes made to a system to improve testability do not affect system performance. Several options are available to eliminate the effect of circularity that results from physical feedback: saturating the feedback circuit, conducting test measurement before feedback occurs, inserting feedback loop breaks that are engaged only during a test cycle, and repackaging feedback loops into a single, replaceable element. We can modify the basic measures to reflect these options. For example, we modify the isolation level to ignore ambiguity resulting from feedback, as follows:

$$FMIL = \frac{|UF|}{|F| - \sum_{i=1}^{|F|} \xi_i + L} \quad (16)$$

where FMIL is the feedback-modified isolation level, ξ is as defined in Equation 15, and L is the number of feedback loops. The summation removes all elements in feedback, and L adds back one element for each feedback loop.

The ideal value of FMIL would be 1.0000. For the case study, FMIL = 16/(26 − 4 +1) = 0.6957. The difference between FMIL and IL indicates that we can achieve an 8% improvement in component uniqueness by repackaging the feedback loop.

Test set measures

To utilize an effective mix of test resources, we must consider several issues related to determining tests for a system, including test adequacy, sufficiency, consistency, and efficiency. The following measures provide insight into the usefulness of the test set in relation to these issues.

Test leverage. TL measures the robustness of the test set—that is, the relative capability of the test set to determine system health:

$$TL = \frac{|I|}{|F|} \quad (17)$$

For the case study, TL = 20/26 = 0.7692.

We derive theoretical limits of TL to provide a basis for comparing the actual test set with an ideal set. These limits, TLMAX and TLMIN, let us determine how appropriately we have specified the test set. In other words, is the test set overspecified (overtesting), underspecified (undertesting), or appropriate for the system?

Overtesting. We assign an upper limit to the test leverage by specifying one test for each conclusion. The absence of a failed-test outcome indicates *No Fault*, so under this extreme the number of tests would be |**F**| − |**NF**| or |**F**| − 1. The maximum test leverage (TLMAX) would then be

$$TLMAX = \frac{|F| - 1}{|F|} \quad (18)$$

If TL exceeds this value, the test set has been overspecified.

For the case study, TLMAX = (26 −1)/ 26 = 0.9615. Note that for extremely large systems, |**F**| − 1 ≅ |**F**|, and TLMAX ≅ 1.0:

$$\lim_{|F| \to \infty} \left(\frac{|F| - 1}{|F|} \right) = 1 \quad (19)$$

Undertesting. To determine the minimum amount of testing needed to accomplish a diagnosis, we consider some fault isolation theory. For single failures and binary (two-value) tests, a test partitions the set of fault isolation conclusions **F** into two subsets.[1] (This assumes binary-outcome tests. Multiple-outcome tests may actually partition **F** into many subsets. The modeling approach considers mostly binary-outcome tests.) The subsets include elements that are still feasible after a test outcome and elements that are not feasible after a test outcome. Because we do not know the outcome of a test before we execute it, the most efficient partition results from a test that divides **F** into two equal subsets.

The next test is most efficient if it again divides the feasible subset in half. This process, called the half-interval technique, is one method by which we can accomplish fault isolation. Real systems can rarely follow half-interval fault isolation, but the half-interval case represents the lowest number of tests for which we can accomplish diagnosis to the element level. Under this assumption, conducting one test reduces the feasible set from |**F**| to |**F**|/2, conducting two tests reduces the set to (|**F**|/2)/ 2, and so on. Conducting n tests reduces the feasible set to $|F|/2^n$. Fault isolation results when $|F|/2^n = 1$, so we want the minimum number of tests for fault isolation, $n = \log_2 |F|$. From this, we specify the lower bound of TL, or TLMIN:

$$TLMIN = \frac{\log_2 |F|}{|F|} \quad (20)$$

Any value of TL that is less than this value indicates that the test set has been underspecified.

For the case study, TLMIN = ($\log_2$ 26)/ 26 = 0.1808.

Test leverage modifications. We modify TL to prevent some elements in the model from affecting the TL value. For example, we compute input-modified test leverage (IMTL) as

$$IMTL = \frac{|I| - |INT|}{|F| - |IN|} \quad (21)$$

Further, we compute feedback-modified test leverage (FMTL), which excludes tests and conclusions involved in feedback, as

$$FMTL = \frac{|I| - \sum_{i=1}^{|I|} \rho_i + L}{|F| - \sum_{i=1}^{|F|} \xi_i + L} \quad (22)$$

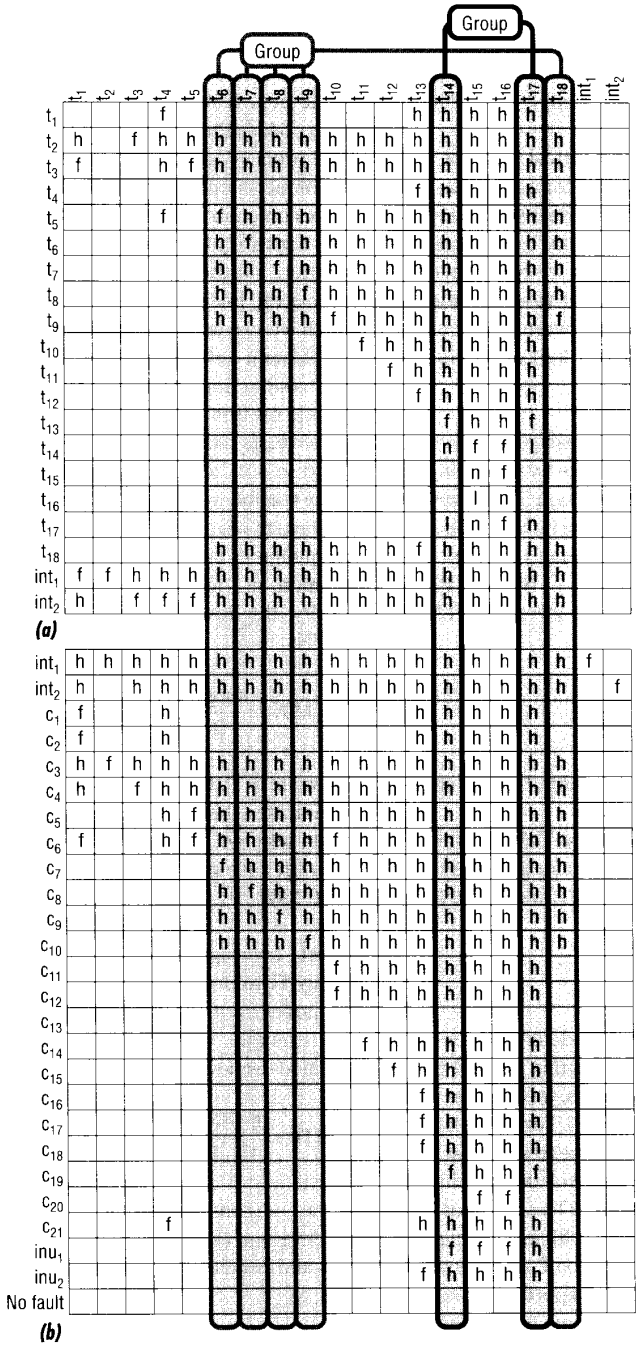

Figure 4. Case study redundancy analysis: test-to-test dependency matrix (a); test-to-conclusion dependency matrix (b).

where ξ is as defined in Equation 15. We can also modify TL for redundancy; this is discussed later.

Test uniqueness. TU measures the degree to which the tests in the test set provide unique information:

$$TU = \frac{|\mathbf{UI}|}{|\mathbf{I}|} \tag{23}$$

The ideal value of TU would be 1.0000. For the case study, TU = 14/20 = 0.7000.

Recall that the first occurrence of a configuration of $(\vec{\mathbf{ST}}_i)$ is considered unique, so that each redundancy group is represented by one element in the set **UI**.

Test redundancy. We identify test redundancy (TR) whenever two or more test signatures are identical—that is, whenever $\vec{\mathbf{ST}}_i = \vec{\mathbf{ST}}_j$. Redundancy simply means that the evaluation of any member of the test redundancy group will provide the same information as the evaluation of any other member of that group. Figure 4 highlights two redundancy groups in the case study. Table 6 lists all the redundancy groups, which are identical to the circularity groups (Table 5), as is generally the case.

TR is the complement of test uniqueness. It is a measure of the percentage of tests that we can consider for elimination due to redundancy:

$$TR = 1 - TU = 1 - \frac{|\mathbf{UI}|}{|\mathbf{I}|} \tag{24}$$

The ideal value of TR would be 0.0000. For the case study, TR = 1 − 14/20 = 0.3000.

Redundancy modifications. Test redundancy may be desirable, and we can modify certain measures to ignore redundant tests. For example, the test leverage may indicate that the test set is overspecified because of desirable redundant tests. A new measure, nonre-

dundant test leverage (NRTL), may be a more reasonable value to compare with TLMAX:

$$\text{NRTL} = \frac{|\mathbf{UI}|}{|\mathbf{F}|} \qquad (25)$$

For the case study, NRTL = 14/26 = 0.5385.

Excess-test candidates. In assessing the value of the test set, one major objective is to minimize the number of tests required. Because each test must be specified, developed, documented, and validated, reducing the number can reduce costs significantly. Frequently, ad hoc methods of developing system diagnostics result in overtesting. A natural question for an analyst to ask is which tests can be eliminated without creating additional ambiguities among the conclusions. These tests are excess-test candidates, which can be individually eliminated if the analyst is satisfied with the system's single-failure testability and does not anticipate false alarms (discussed later in this article).

Using the matrix formulation, we identify ambiguity when conclusions have equal failure signatures. In considering a test for excess-test candidacy, we want to determine whether ambiguities will occur when that test is eliminated. If a new ambiguity is created, the test is not an excess-test candidate; if no new ambiguity is created, the test is a candidate. The former situation is shown in Figure 5 for t_2 and t_{11}. Neither test can be considered an excess-test candidate because the elimination of either will create new ambiguity groups.

Table 7 on the next page shows the result of eliminating each test in the test set. Testable inputs are not usually considered in the analysis, but they could be. If they are considered, both int_1 and int_2 are excess-test candidates.

We define excess-test measures in terms of either conclusions or replace-

Table 6. *Redundancy groups for the case stydy.*

Group*	Test signature ($\overline{ST}_j$)**	Members
1	(0,1,1,0,1,1,1,1,1,0, 0,0,0,0,0,0,0,1,1,1 + 1,1,0,0,1,1,1,1,1,1,1, 1,0,0,0,0,0,0,0,0,0, 0,0,0,0,0)	t_6 t_7 t_8 t_9 t_{18}
2	(1,1,1,1,1,1,1,1,1,1, 1,1,1,0,0,1,1,1,1 + 1,1,1,1,1,1,1,1,1,1, 1,1,10,1,1,1,1,1,1, 0,1,1,1,0)	t_{14} t_{17}
3	(1,1,1,1,1,1,1,1,1,1, 1,1,1,1,1,1,1,1,1,1 + 1,1,1,1,1,1,1,1,1,1, 1,1,1,0,1,1,1,1,1,1, 1,1,1,1,0)	t_{15} t_{16}

* *Group data structure is in accordance with Sheppard and Simpson.*[2]

** *For ease of cross-reference, + denotes the transition point between the test-to-test and test-to-conclusion matrices (Figure 4a, b).*

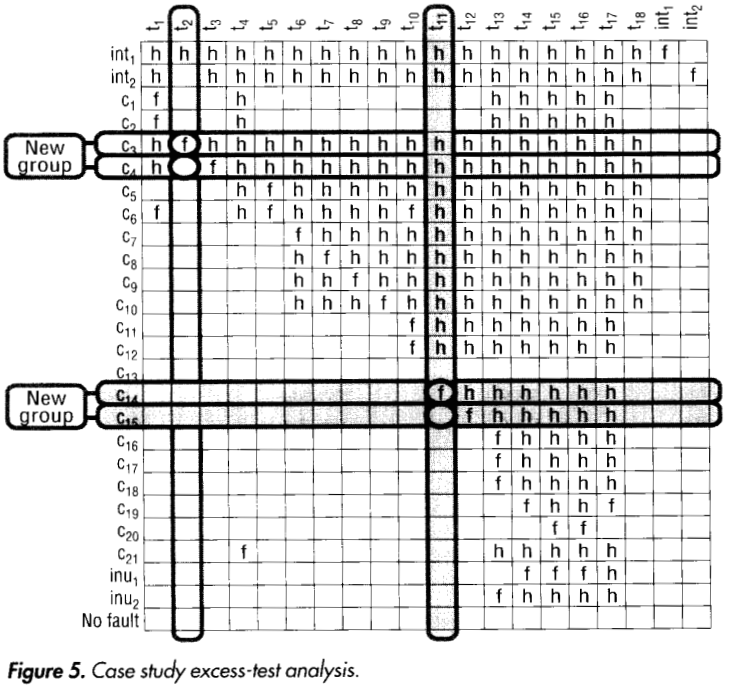

Figure 5. *Case study excess-test analysis.*

able unit groups. The excess-test measure for conclusions is

$$XMC = \frac{\sum_{i=1}^{|\mathbf{T}|} \psi_i}{|\mathbf{T}|};$$

$$\psi_i = \begin{cases} 0; \text{ iff given } \mathbf{T} - \{t_i\}, \\ \quad \forall f_j, f_k \in \mathbf{UF}(j \neq k), \\ \quad \mathbf{SF}_j \neq \mathbf{SF}_k \\ 1; \text{ otherwise} \end{cases} \quad (26)$$

The excess-test measure for replaceable unit groups is

$$XMR = \frac{\sum_{i=1}^{|\mathbf{T}|} \theta_i}{|\mathbf{T}|};$$

$$\theta_i = \begin{cases} 1; \ \psi_i = 1 \text{ or RUA} \\ \quad \text{not increased} \\ 0; \text{ otherwise} \end{cases} \quad (27)$$

where ψ_i is as defined in Equation 26 and RUA is replaceable unit ambiguity. The excess-test measure for conclusions including inputs is

$$XMIC = \frac{\sum_{i=1}^{|\mathbf{I}|} \psi_i}{|\mathbf{I}|} \quad (28)$$

where ψ_i is as defined in Equation 26. The excess-test measure for replaceable unit groups including inputs is

$$XMIR = \frac{\sum_{i=1}^{|\mathbf{I}|} \theta_i}{|\mathbf{I}|} \quad (29)$$

where θ_i is as defined in Equation 27.

Table 7. Ambiguity groups created by eliminating individual tests.

Test eliminated	New isolation ambiguity	Replaceable unit groups	Excess-test candidate conclusion replaceable unit groups
t_1	c_5 c_6	ru_3 ru_3	No/yes
t_2	c_3 c_4	ru_1 ru_3	No
t_3	c_4 c_6	ru_3 ru_3	No/yes
t_4	c_{18} c_{21}	ru_2 ru_8	No
t_5	None	None	Yes
t_6	None	None	Yes
t_7	None	None	Yes
t_8	None	None	Yes
t_9	None	None	Yes
t_{10}	c_{12} c_{14}	ru_6 ru_6	No/yes
t_{11}	c_{14} c_{15}	ru_6 ru_6	No/yes
t_{12}	c_{15} c_{16}	ru_7 ru_6	No
t_{13}	c_{18} c_{19}	ru_8 ru_8	No/yes
t_{14}	None	None	Yes
t_{15}	None	None	Yes
t_{16}	None	None	Yes
t_{17}	None	None	Yes
t_{18}	None	None	Yes

*Excess test candidate at either the conclusion level (column 2) or replaceable unit level (column 3)

The ideal value for XMC, XMR, XMIC, and XMIR would be 0.0000, without multiple-failure or false-alarm considerations. For the case study, XMC = 0.5560, XMR = 0.8890, XMIC = 0.6000, and XMIR = 0.9000, indicating that we can make significant reductions in the test set.

Excess-test analysis. The analysis we have just performed to determine which tests are excess-test candidates is not sufficient for recommending which tests we will actually eliminate. Clearly, the elimination of one test may affect our ability to eliminate another. For example, in Figure 4, we can see that t_{14} and t_{17} are redundant. But careful examination of Figure 4b indicates that if we eliminate both t_{14} and t_{17}, c_{19} and c_{20} become ambiguous. For the analysis to recommend which tests to eliminate, it must first consider each test in terms of weighting criteria defined by the analyst. We will address multiple-criteria analysis in detail in an article on fault isolation later in this series; here we give only an overview of the use of multiple criteria to evaluate excess-test candidates.

First, we compute desirability values for each test, based on the criteria and weights determined by the analyst. We then rank the tests in decreasing order of desirability of having the test performed. The idea is to consider less desirable tests first for elimination. The desirability values may be based on multiple criteria, including (but not limited to) the supplied weights. Finally, we examine the tests serially. We eliminate the first test. If that increases the ambiguity of the system, we return the test to the model; otherwise, we update the model to reflect the elimination of the test. Then we go on to the next test, stopping the process when all tests have been considered.

When we apply this process to the case study, using the failure frequency data of Table 3 to compute weighting values and limiting isolation to replace-

able unit groups, the analysis recommends t_3, t_5, t_6, t_7, t_8, t_9, t_{11}, t_{16}, and t_{17} for elimination. Many of these tests are recommended for elimination because they are redundant. For example, the algorithm saves t_{18} from redundancy group 1, t_{14} from redundancy group 2, and t_{15} from redundancy group 3, thereby recommending elimination of the rest (see Table 6 for the redundancy groups). The algorithm eliminates t_5 because it is excess. The algorithm eliminates t_{11} and t_3 because they cause no redundancy in replaceable unit groups and are therefore excess at the replaceable unit level. However, the algorithm makes no recommendation about t_1, t_2, t_{12}, and t_{13}—all of which are candidates in Table 7—because the interactions of eliminations may preclude elimination of a specific test.

False alarms. False alarms are usually associated with built-in test, although they may occur in any type of diagnostic testing. As defined by military standards, a false alarm is an indication of failure in a system where no failure exists.[5]

False alarms result from imperfect testing. The better we understand a process or technology, the more accurate the testing becomes. False alarms generally become a problem when system complexity becomes great or the design pushes the state of the art. Because we cannot actually measure false alarms in the field,[6,7] specifications should be based on *cannot duplicate* (CND) events instead of false alarms. We will address this issue in detail later in this series.

The following are four viable solutions to false-alarm problems:

- *Improving test science.* We can avoid false alarms by sampling more often, modeling in greater detail, and accounting for a greater number of variables. In the case of built-in test, these steps create an increased software requirement and may require the addition of sensors to the built-in test equipment.
- *Increasing test tolerances.* We can avoid false alarms by making the test less sensitive to anomalous behavior. Unfortunately, this may reduce the test's ability to detect real failures.
- *Conducting repeat polling.* In repeat polling, we try to avoid false alarms by executing a test repeatedly. Each time the test is evaluated, the test algorithm uses the results to confirm any previous executions. Repeat polling is intended to allow transient characteristics to work their way through the system without triggering a failure indication. Repeat polling requirements are usually written as, for example, "three or more indications within 250 milliseconds." As this solution also may lead to missed detections, a better approach is to recognize transient characteristics by means of the first solution, improved test science.
- *Cross-correlating test information.* We can correlate an anomalous indication with other testing to either confirm or deny the original information. The information flow model can analyze this technique to assist in planning for false-alarm prosecution.

False-alarm tolerance. False-alarm tolerance (FAT) is a measure of our ability to perform test-to-test cross-checking. The test-to-test matrix in Figure 2a is a complete map of the higher order interrelationships between the tests. Recall that this map is generated in closure. FAT is the average percentage of tests in the test set that we can use as verifiers. For example, we can verify an anomalous outcome of t_{13}, using t_{14}, t_{15}, t_{16}, and t_{17} (the values in $\overline{\mathbf{ST}}_{t13}$). False-alarm tolerance then is given by

$$\mathrm{FAT} = \frac{\sum_{i=1}^{|\mathbf{I}|}\sum_{j=1}^{|\mathbf{I}|}\phi_i}{|\mathbf{I}|\{|\mathbf{I}|-1\}};$$
$$\phi_i = \begin{cases} 1; & \tilde{\mathbf{D}}'_{ij} = 1,\ i \neq j \\ 0; & \text{otherwise} \end{cases} \qquad \textbf{(30)}$$

There is no ideal value for FAT, but a fully tested serial system will have a value of 0.5000. For the case study, FAT = 183/380 = 0.4816.

FAT typically decreases as systems become larger and excess tests are removed. We have found that real systems with FAT values below 10% should be carefully analyzed. One way to maintain a high false-alarm tolerance is to retain redundant and excess tests. Of course, merely having these tests available is not sufficient; they must be used in the diagnostic strategy, which means that additional testing will be specified. We will address this point in a future article in this series.

Multiple failure measures

Multiple failure is both a testability and a diagnosis problem. As we will discuss in future articles on diagnosis, the basic problem formulation is amenable to both single- and multiple-failure diagnostic paradigms. However, there are two cases in which we must be careful to provide testability in a multiple-failure situation because diagnosis cannot solve the corresponding problems. The first case is hidden failures, the masking of symptoms that occurs in most systems. The second is false failures, situations in which symptoms lead to wrong conclusions.

Hidden failures. The failure of one element may prevent the test set from determining that a multiple failure even exists. For example, if c_{11} in Figure 2 were to fail, the failure would be detected by t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, and t_{17}. This is the failure signature of c_{11} ($\overline{\mathbf{SF}}_{c11}$) discussed earlier in this article. In addition,

Table 8. Hidden failures in the case study.

Failed Element	Potential Hidden Failures in a Multiple-Failure Situation
int_1	c_1 c_2 c_3 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{20} c_{21} c_{19} inu_1
int_2	c_1 c_2 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{20} c_{21} c_{19} inu_1
c_1	c_2 c_{16} c_{17} c_{18} inu_2 c_{20} c_{21} c_{19} inu_1
c_2	c_1 c_{16} c_{17} c_{18} inu_2 c_{20} c_{21} c_{19} inu_1
c_3	c_1 c_2 c_4 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{20} c_{21} c_{19} inu_1
c_4	c_1 c_2 c_5 c_6 c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{20} c_{21} c_{19} inu_1
c_5	c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{20} c_{21} c_{19} inu_1
*c_6	c_1 c_2 c_5 c_7 c_8 c_9 c_{10} c_{11} c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{20} c_{21} c_{19} inu_1
c_7	c_8 c_9 c_{10} c_{11} c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{20} c_{19} inu_1
c_8	c_7 c_9 c_{10} c_{11} c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{20} c_{19} inu_1
c_9	c_7 c_8 c_{10} c_{11} c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{20} c_{19} inu_1
c_{10}	c_7 c_8 c_9 c_{11} c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{20} c_{19} inu_1
c_{11}	c_{12} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{19} inu_1 c_{20}
c_{12}	c_{11} c_{14} c_{15} c_{16} c_{17} c_{18} inu_2 c_{19} inu_1 c_{20}
c_{13}	none
c_{14}	c_{15} c_{16} c_{17} c_{18} inu_2 c_{19} inu_1 c_{20}
c_{15}	c_{16} c_{17} c_{18} inu_2 c_{19} inu_1 c_{20}
c_{16}	c_{17} c_{18} inu_2 c_{19} inu_1 c_{20}
c_{17}	c_{16} c_{18} inu_2 c_{19} inu_1 c_{20}
c_{18}	c_{17} c_{16} inu_2 c_{19} inu_1 c_{20}
c_{19}	inu_1 c_{20}
c_{20}	none
c_{21}	c_{16} c_{17} c_{18} inu_2 c_{19} inu_1 c_{20}
inu_1	c_{19} c_{20}
inu_2	c_{16} c_{17} c_{18} c_{19} inu_1 c_{20}
No Fault	none

*** Indicates potential false failure**
Boxed elements represent an ambiguity group
Shaded boxed elements are in ambiguity with indicated failure

if c_{19} were to fail, it would be detected by t_{14}, t_{15}, t_{16}, and t_{17}. A joint failure of c_{11} and c_{19} would generally exhibit all the symptoms of each failure, which are just the symptoms of c_{11} (that is, all the symptoms of c_{19} are contained in c_{11}). There is no symptom to separate this multiple failure from either of the single failures. Even this situation is not generally a problem. Repair of c_{11} and subsequent testing will isolate c_{19}. This is called peeling and is one of the rationales used in developing single-failure isolation techniques. Peeling works well except where the hidden failure is the cause (sometimes referred to as a root cause) of the isolated failure. Then, the repair of c_{11} is ineffective because the failure in c_{19} will again cause c_{11} to fail, and subsequent diagnosis will again identify c_{11}. This situation occurs whether we use single- or multiple-failure diagnostic approaches.

Table 8 lists all the hidden failures for each conclusion in the case study. Analysis of this list will determine if root cause relationships exist. Through improved testability or maintenance-manual directions to repair the combination of components (that is, repair both c_{11} and c_{19}), we can diagnose any failure in the second column that is a root cause of a failure in the corresponding row of the first column. If we do not improve testability, diagnosis will fail to identify the complete problem. We obtain a measure of the size of the analysis set by computing the average percentage of failures that may be hidden by any conclusion reached (HF):

$$\mathrm{HF} = \frac{\sum_{i=1}^{|\mathbf{F}|}\sum_{j=1}^{|\mathbf{F}|}\Phi_{ij}}{\left\{|\mathbf{UF}|-2\right\}^2}; \qquad \Phi_{ij} = \begin{cases} 1; & f_i,\ f_j \in \mathbf{UF}(i \neq j), \\ & \mathbf{SF}_i \subset \mathbf{SF}_j \\ 0; & \text{otherwise} \end{cases} \tag{31}$$

Here Φ_{ij} is the actual existence of a hidden-failure relationship. Note that we count only unique elements in **F**. The 2 in the denominator accounts for the fact that two elements have no hidden failures (*No Fault* and at least one other element).

For the case study, HF = $94/14^2$ = 0.4796.

On average, about 48% of the reachable conclusions are hidden from a given failure. We can modify HF to exclude inputs; inputs generally hide a large number of other failures and are not subject to root cause:

$$\text{IMHF} = \frac{\sum_{i=1}^{|\mathbf{F}|}\sum_{j=1}^{|\mathbf{F}|}\hat{\Phi}_{ij}}{\left\{|\mathbf{UF}|-n-2\right\}^2}; \qquad \hat{\Phi}_{ij} = \begin{cases} 1; & \left(\Phi_{ij,}=1\right)\wedge f_i \in \mathbf{IN} \\ 0; & \text{otherwise} \end{cases} \tag{32}$$

where Φ_{ij} is as defined in Equation 31 and n is the number of inputs that do not belong to any ambiguity group of a size greater than 1.

For the case study, IMHF = 68/[(16 – 2 – 2)²] = 0.4722.

False failures. The second class of multiple failures that will render ineffective both single- and multiple-conclusion diagnosis is the false failure. A false-failure indication occurs when the combined symptoms of two or more failures are identical to the symptoms presented by a single failure that is not a member of the failure set. We compute false failure by examining the union of failure signatures for a conclusion's hidden failures to see if there is a match. Figure 6 illustrates the process for pairs of failures. We repeat this process for three failures, four failures, and so on. The only conclusion in the case study that has a potential false-failure problem is c_6. The associated failure pairs are (c_1, c_5) and (c_2, c_5). Mathematically, the false-failure condition exists if the following equality holds:

$$\mu \equiv \left(\bar{\mathbf{SF}}_j = \bigvee_{i=1}^{|\mathbf{F}|}\chi_i\right); \qquad \chi_i = \begin{cases} \bar{\mathbf{SF}}_i; & \text{iff } \bar{\mathbf{SF}}_i \subset \bar{\mathbf{SF}}_j \\ \Gamma; & \text{otherwise} \end{cases} \tag{33}$$

where μ is a logical variable and Γ denotes the null vector. We can compute a measure that provides the extent to which false failures may be a problem:

$$\text{FF} = \frac{\sum_{i=1}^{|\mathbf{F}|}\upsilon_i}{|\mathbf{F}|-2}; \quad \upsilon_i = \begin{cases} 1; & \text{iff } \mu \text{ is true} \\ 0; & \text{otherwise} \end{cases} \tag{34}$$

Here FF is the fraction of elements potentially falsely identified, and μ is as defined in Equation 33. Two elements are removed for the *No Fault* and minimum-failure signature (other than *No Fault*) in the denominator.

The ideal value of FF would be 0.0000. For the case study, FF = 1/24 = 0.0417. We can modify this measure to exclude inputs:

$$\text{IMFF} = \frac{\sum_{i=1}^{|\mathbf{F}|}\hat{\upsilon}_i}{|\mathbf{F}|-2-|\mathbf{IN}|}; \qquad \hat{\upsilon}_i = \begin{cases} 1; & \text{iff } \left(\upsilon_i = 1\right)\wedge f_i \notin \mathbf{IN} \\ 0; & \text{otherwise} \end{cases} \tag{35}$$

where υ_i is as defined in Equation 34.

The ideal value of IMFF would be

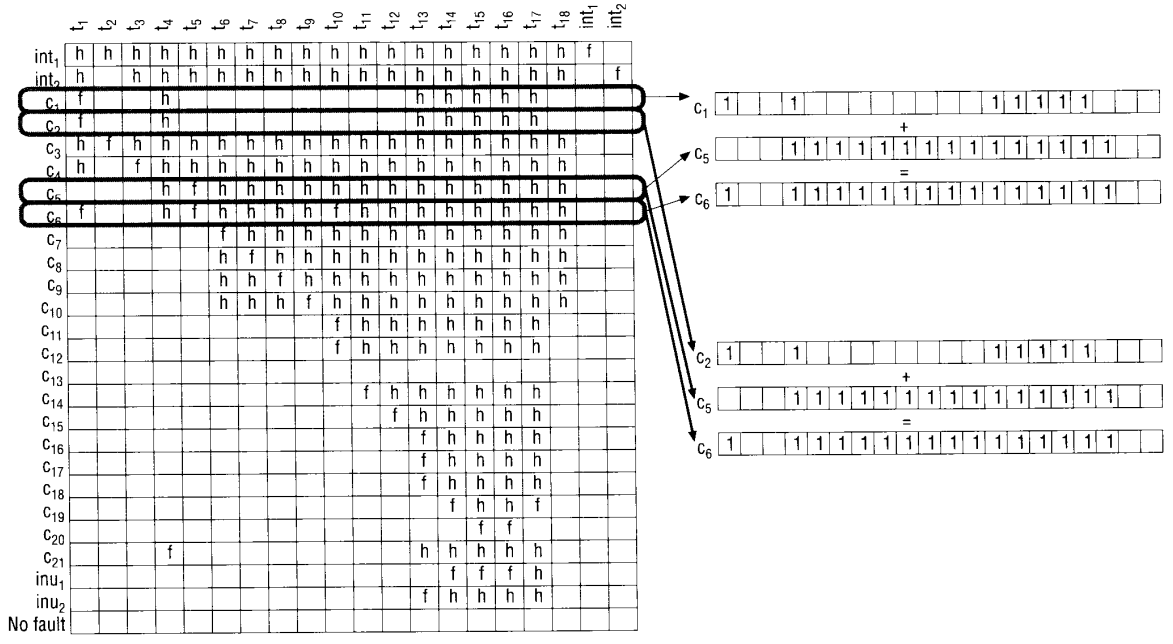

Figure 6. Case study false-failure analysis.

0.0000. For the case study, IMFF = 1/20 = 0.0500.

The mathematical formulation of diagnostic information flow allows us to closely examine and assess the testability of complex systems. The analysis demonstrated in this article is an initial analysis because it only demonstrates where the deficiencies in the system are located.

In developing any complex system, we can minimize maintenance costs if we identify testability problems and correct them early in system design. After applying the testability assessment techniques, our next step is to interpret the results and recommend design changes to improve system testability. Such improvements might include eliminating or reorganizing existing tests, developing new tests (and maybe test points), repackaging components to improve operational isolation, locating feedback loops, implementing cross-checks for false alarms, and improving the observability of some failures to minimize the effects of root causes or false failures.

We've explained how to identify the problem areas in a system design. We must point out, however, that the measures defined are *indicators* of testability and have no predetermined thresholds for use in the analysis. Analysts must use their own engineering judgment in the context of the problem being studied to determine what the indicators mean. We will demonstrate how to apply the testability analysis to redesigning the case study system in the next article in this series. D&T

References

1. W. Simpson and J. Sheppard, "System Complexity and Integrated Diagnostics," *IEEE Design & Test of Computers*, Vol. 8, No. 3, Sept. 1991, pp. 16-30.
2. J. Sheppard and W. Simpson, "A Mathematical Model for Integrated Diagnostics," *IEEE Design & Test of Computers*, Vol. 8, No. 4, Dec. 1991, pp. 12-25.
3. F. Johnson and R. Unkle, "The System Testability and Maintenance Program (STAMP): A Testability Assessment Tool for Aerospace Systems," *Proc. AIAA/NASA Symp. on Maintainability of Aerospace Systems*, AIAA, New York, 1989.
4. J. Sheppard and W. Simpson, "Incorporating Model-Based Reasoning in Interactive Maintenance Aids," *Proc. 42nd Nat'l Aerospace and Electronics Conf.*, IEEE Press, New York, 1990, pp. 1238-1242.
5. *Definitions of Terms for Test, Measurement, and Diagnostic Equipment, MIL-STD-1309B*, Washington, D.C., May 1975.
6. W. Simpson et al., "Prediction and Analysis of Testability Attributes: Organizational-Level Testability Prediction," RADC-TR-85-268, Rome Air Development Center, Griffis AFB, N.Y., Feb. 1986.
7. E. Gilreath, B. Kelley, and W. Simpson, "Predictors of Organizational Level Testability Attributes," 1511-02-2-4179, Arinc Research Corp., Annapolis, Md., Nov. 1986.

Direct questions or comments on this article to either author at Arinc Research Corp., Advanced Research and Development Group, 2551 Riva Rd., Anapolis, MD 21401. Sheppard's e-mail address is sheppard @csjhu.edu.

John W. Sheppard is a senior research analyst in the Advanced Research and Development Group at Arinc Research Corp. He is also pursuing a PhD in computer science at Johns Hopkins University. His research interests include applying AI techniques to fault diagnosis, machine learning, neural networks, and nonstandard logic. He has developed algorithms to diagnose system failures, verify knowledge bases, and classify software. He was also a principal developer of Pointer, an intelligent, interactive maintenance aid, and assisted in the development of a prototype expert system that diagnoses system failures and reconfigures the system to keep functioning. He holds a BS from Southern Methodist University and an MS from Johns Hopkins University, both in computer science.

William Simpson, a research fellow in the Advanced Research and Development Group at Arinc Research Corp., works with testability and fault diagnosis. He helped develop the System Testability and Maintenance Program, which is based on an information flow model. He was also a principal developer of Pointer. He holds a BS from Virginia Polytechnic Institute and State University and an MS and a PhD in aerospace engineering from Ohio State University.