THINKING BEYOND THE GROUP SIZE FETISH: TOWARDS A NEW TESTABILITY

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

Danver Hartop
DSI International
1574 N. Batavia, #3
Orange, CA 92867
714-637-9325
dhartop@dsiintl.com

Abstract – This paper exposes some major deficiencies inherent within current methods for assessing design Testability—critical shortcomings that not only might cause adherence to contracted Testability requirements to be in conflict with long-term maintenance goals (such as Life Cycle Cost and Operational Availability), but could also result in evaluations that fail to predict the actual diagnostic behavior of the system or device. In recent years, the need to accurately forecast diagnostic performance has become more essential as more development projects are contractually linked to the maintenance of the fielded product (as witnessed by the recent emergence of maintenance warrantees and the combined contracting of development and maintenance efforts). What is needed are Testability procedures that can better serve long-term maintenance and support objectives, yet remain true to the discipline's original intent of providing diagnostics-based feedback in early phases of the development cycle. Hoping to foster a less conflicted Testability practice, this paper proposes some alternatives to current quantitative methods of Testability assessment that can more accurately predict diagnostic behavior and more consistently reflect the relationships between a system or device's diagnostic capability and its Life Cycle Cost and Operational Availability.

INTRODUCTION: THE STATISTICAL HYPOCRISY OF CURRENT TESTABILITY PRACTICE

For nearly three decades, the centerpiece of most contracted Testability requirements has been that isolation must be achieved to fault groups containing a certain number of components a certain percentage of the time, given the ability to detect a certain minimum percentage of failures. The rationale for this requirement was simple—if technicians could achieve better resolution of failures, then replacement cost and time would be reduced, ultimately translating into a lower Life Cycle Cost (LCC) and a greater Operational Availability (A_O) for the device or system in question. It has long been recognized, however, that this thinking is inherently flawed. The additional testing necessary to achieve isolation to smaller group sizes has its own cost and time associated with it and therefore the implementation of more extensive diagnostics does not in all cases serve a project's maintenance goals (not to mention the cost of developing additional testing). By blindly adopting the erroneous assumption that isolation to smaller ambiguity groups necessarily implies improved Maintainability, Testability analyses are currently unable to recognize cases of diminishing returns and can thereby easily subvert their own goals.

The fact that Testability as it is practiced today is little more than a statistical means to an end is recognized upon occasion and contracted Testability requirements may be subsequently modified or waived in deference to more fundamental objectives (such as LCC, A_O, or development cost). Frequently, however, the possibility of conflict between long-term goals and short-term requirements goes unnoticed and designers resort to a variety of means—including, but certainly not limited to, design modifications and the development of more extensive diagnostics—to achieve their contracted Testability obligations.

One frequently employed way of improving Testability numbers without the introduction of design changes or additional testing has been to treat several separate components as if they were a single replacement unit. This can often be a legitimate way to improve both Testability and Maintainability, such as when several inexpensive and easily replaced parts have relatively high failure rates. However, because the increase in

the cost and time to replace this combined group is not reflected in the isolated group sizes, this technique can be misused. For example, if a group of components designated as comprising a single replacement unit were to have a relatively high failure rate and be either expensive or time-consuming to repair, then the improvement in calculated Testability numbers that would result from this grouping would be misleading. (When this technique is used in this manner, it is frequently rationalized as being merely a partitioning issue.) The result would be more attractive Testability figures without a corresponding progress toward the very maintenance goals that Testability analysis is intended to support.

Another shocking disconnect between assessments of Testability and Maintainability exists in the fact that it is possible for Testability figures to be improved by selectively utilizing less reliable components. Because most Testability indicators take reliability data into consideration, the introduction of a component with a high failure rate that can be unambiguously isolated (that is, that can be isolated in a fault group containing only itself), may result in a substantial improvement in Fault Isolation (Resolution) statistics. The result of this maneuver is a Testability assessment that is inversely proportional to the Reliability of the design—another instance of Testability at cross-purposes with itself.

Wittingly or not, current Testability methods repeatedly encourage this statistical hypocrisy—the ravaging of various maintenance goals in the name of improved Maintainability. It happens even when Testability is pursued with the best intentions and in the most conscientious manner possible.

For example, when a design does not meet contracted Testability requirements, one possible solution is the introduction of additional BIT hardware to facilitate fault detection and/or isolation. What is typically overlooked (sometimes intentionally) is that this BIT hardware is frequently more prone to failure than the functional hardware that it tests. When contract requirements are interpreted as applying only to the operational or functional elements of a design, Testability analysis will show the diagnostic benefits of the additional testing capability without depicting the negative impact of the BIT hardware upon design Reliability. In this case, the strict adherence to contract Testability requirements (as they are typically expressed today) could actually result in a system or device that fails more frequently and, unless the BIT is equipped with the capacity to test itself, has larger ambiguity groups. In other words, the ability to achieve maintenance goals has been hampered for the sake of Testability compliance.

The exclusion of BIT hardware from examination under the rigorous eye of Testability not only exemplifies the current inability of the discipline to express conclusions in terms that demonstrate the effect of design changes upon maintenance goals. It is also a perfect example of what might be called the *imaginary world* approach to Testability. This approach assumes that analyses need not reflect the real world behavior of a system or device under test because any improvement in Testability will necessarily indicate a corresponding improvement in Maintainability. However, as we have already demonstrated (in the case where BIT hardware is excluded from Testability analysis), the adoption of *imaginary worlds* can actually conceal the relationship between a design's inherent diagnostic capability and its ability to be effectively maintained.

Common phrases that often betray the assumption of an *imaginary world* include:

- operational hardware – implies an *imaginary world* that excludes all hardware exclusively associated with testing
- fault universe – implies an *imaginary world* in which certain types of failures are excluded from diagnostic coverage
- detected failures – implies an *imaginary world* in which faults that are not immediately observable will never need to be repaired
- single-point failure – implies an *imaginary world* in which no more than one component can be malfunctioning as diagnostics are performed.
- probability-weighted – implies an *imaginary world* in which the relative failure frequencies of different components or functions remain constant over any arbitrary interval of time

The adoption of *imaginary worlds* in order to reduce the complexity of certain analysis tasks has been sanctioned by tradition to the point where they are often assumed even when not explicitly legislated. Furthermore, each of these *imaginary worlds* has its own legions of adherents who will argue tooth and nail to maintain the legitimacy of their *world* and to insure its acceptability as the basis for their analyses. What is often forgotten is that most of these *imaginary worlds* originated as a means to facilitate tasks that were too complex to be performed on paper. With the nearly universal adoption of computer-based engineering tools, the computational complexities associated with real (non-imaginary) world analyses are no longer cost or time prohibitive. Yet the fact remains that many

Testability efforts continue to systematically reduce their workload by the unquestioned adoption of one or more of these *imaginary worlds*. Contracts are either assumed to require or interpreted as permitting the adoption of these *imaginary worlds*—even when this contradicts with other provisions of the contract. When a contract does explicitly call for the adoption of an *imaginary world*, it is seldom called into question, for many companies are all too ready to satisfy contract requirements to the letter, even when doing so conflicts with their actual intent.

It is precisely in order to reduce this risk that many new contracts contain provisions that attempt to insure that a fielded system or device can be as effectively maintained as the contract requirements demand and the development team's predictions promise. For example, many new contracts contain maintenance warrantees. If diagnostic performance does not meet spec, the developers will be held responsible. If, for example, actual operational availability were to fall short of the level specified within the contract, the developer could be required to provide additional units free of charge (this is an expensive mistake in the case of a billion dollar ship or aircraft). Likewise, if a fielded system fails to satisfy contracted requirements for a certain maintenance or diagnostic measure—such as the Mean Time to Repair (MTTR) or False Alarm Rate—the developers would be obliged to correct the problem at their own cost.

As an alternative to maintenance warrantees, some projects ensure that maintenance goals are met—at least in terms of cost—by including several years of maintenance (at a fixed price) within the development contract. The developer then has an interest in insuring that the system or device can really be maintained as advertised, since any failure to do so would have a direct effect upon profit and loss.

Perhaps not surprisingly, this trend toward greater accountability is not yet reflected in the metrics that developers use to evaluate (i.e. predict) compliance with contracted maintenance requirements. In order for Reliability, Testability or Maintainability indicators to be useful as predictions, they must be based upon realistic assumptions. They must recognize that BIT hardware can and does fail, that any type of failure can occur and therefore must be able to be diagnosed, and that more than one thing can be wrong with a system as it is tested.

Furthermore, Testability, Reliability and Maintainability assessments must also take into consideration the fact that a device or system will fail differently over time—the relative frequency of different failures may be completely different during the first few years of deployment than near the end of the system's useful life. Knowledge of how system maintenance statistics will change over time may help the customer to make informed decisions about how that system can be best maintained throughout its lifetime. If maintenance data gathered during the first two years of deployment were to be better than the development estimates and those numbers were then used to reallocate support budgets and reduce the number of spare parts, then, as the system matures, its availability could be severely impacted. This can be avoided if the maintainer is informed that the diagnostic behavior of the system will change over time. Suppose, to give another example, that the actual MTTR for a device during its first few years of deployment were to be significantly higher than the estimates provided during development. The diagnostic capability of that design may then have to be augmented (requiring further expenditure that would have been unnecessary if development analyses were to have predicted diagnostic behavior across multiple phases of the device's useful life).

Current Testability metrics not only fail to account for differences in behavior within different time intervals, they actually assume infinite deployment (or, rather, an arbitrary time interval that is large enough for the relative frequencies of all failures to have approached their respective means). Statistics calculated in this manner are inherently skewed, since a large number of the potential malfunctions that can be maintained in a cost and time-effective manner may be highly unlikely to occur during the product's expected lifetime. This problem is only exacerbated by the fact that the component reliability numbers upon which current Testability statistics are based are means—numbers that by themselves are only useful for describing failure frequencies across an arbitrarily large time interval. Testability measures, as they are calculated today, completely ignore the specific distributions into which the failures fall for each particular component. When metrics are calculated for a specific time interval, consideration must be given not only to the mean time between failures (MTBF) for each component, but also the specific distributions into which that component's failures fall.

FOUR CHALLENGES TO THE TESTABILITY COMMUNITY

If Testability is to remain useful as a discipline, it must evolve not only in order to overcome its own current shortcomings (as they become apparent), but also so that it remains applicable within shifting development paradigms. In order to throw down the metaphorical gauntlet before all those who might be concerned with

the future of the discipline—analysts, contractors, customers, providers of Testability analysis tools, and especially those working to standardize the metrics employed throughout the discipline—we have codified the problems discussed in the previous section into the following four challenges:

Challenge #1: Testability metrics should be expressed in terms directly related to maintenance goals, rather than as abstract entities that constitute good practice.

Perhaps the Testability requirement that has been included in the most contracts over the last few decades has been the percentage of detected failures that can be isolated to a single failed component. Statistics used to satisfy this requirement have been introduced under a variety of names. In two government documents that have often been invoked as guidelines for Testability assessment, this statistic is called Fault Resolution (MIL-STD-2165) and Percent Isolation to a Group of Replaceable Items (MIL-HDBK-472). Within DSI's own tools, these figures have been referred to as Fault Isolation Levels (LogMod), Ambiguity Group Isolation Probabilities (STAT), and Fault Group Size Isolation Probabilities (e*Xpress*). These different metrics are equivalent, but not identical. What is common to all of these statistical measures is the quantification of Testability in terms of an abstract entity—*ambiguity* or fault group size—rather than in terms of specific maintenance considerations.

When Testability is evaluated in terms unrelated to those in which the maintenance goals for a system or device have been defined, then the analysis is suspect since it will be possible for Testability to vary inversely with respect to Maintainability. It is imperative that the enforcement of good Testability as a "rule of thumb" not be counterproductive to the project's maintenance and support goals.

Of course, when Testability analysis is applied in early design phases of a project, it is not likely that the maintenance data will be available in order to ensure the meaningful link between Testability indicators and defined maintenance goals. This does not in any way challenge the need to establish this link. When these metrics are calculated in the absence of meaningful maintenance data, then intelligent defaults should be used (every test takes the same amount of time to perform, etc.), rather than resorting to abstractions that are themselves equally meaningless without recourse this same maintenance data. Testability analysis should be thought of not as a one-time evaluation, but rather as an iterative process that produces better statistics as the initial defaults are replaced by more accurate estimates. Some of the results that emerge from these early analyses may be relatively conclusive and the resulting recommendations can be acted upon even though they are not based upon actual support data. Other conclusions may be deferred until more accurate data is available. This procedure should not extend the scope of most analysis tasks—if Testability analysis is performed in early stages of development, it is likely that Testability is already being viewed as an iterative task within the overall development process.

Challenge #2: Testability metrics should transparently represent the tradeoffs between different maintenance considerations.

One recurring trend in the assessment of inherent Testability is the attempt to embody all Testability information within a single metric or, as it is sometimes called, Figure of Merit. If this metric is based on a single diagnostic attribute, such as repair time, then it risks being at cross-purposes with actual maintenance goals, which often take into consideration multiple criteria. If, on the other hand, a Testability Figure of Merit (or TFOM) were to be a combination of different considerations, it would become, in effect, a reduction of potentially useful knowledge into an abstraction that can no longer be immediately mapped to meaningful characteristics of the design. One of the strengths that distinguishes Testability from its sister disciplines (the other "ilities") is in the ease with which its conclusions can be translated into specific recommendations to improve the physical design or diagnostic strategy. When results are incorporated into a single composite TFOM, Testability loses this useful transparency and risks transforming itself from a proactive engineering discipline (one that can influence the development of the design) into a purely reactive analytical discipline.

Another important advantage of expressing Testability as a set of several easily compared metrics (rather than a single combined metric) is that the different tradeoffs that result from design and diagnostic modifications can be easily evaluated. For example, the introduction of additional BIT into a design may not only reduce the MTTR (which is good), but also result in a reduction in the system MTBF (which is not so good). Rather than blindly suggesting that this BIT be implemented because it improves Testability, the analyst should be able to compare the advantages with the disadvantages in order to arrive at a more measured recommendation. Of course, this type of tradeoff analysis is impossible if the real benefits and penalties are not discernable—which leads us to our next challenge.

Challenge #3: Testability metrics should be able to be derived with sufficient accuracy that they can serve as predictions.

Imaginary worlds beget imaginary recommendations. Testability metrics that are based on anything other than real-world scenarios can potentially obscure the relationships between the inherent diagnostic capacity of a system or device and the ability to achieve long-term maintenance goals for that design.

For example, if Testability analysis were to be based on single point failures (adopting an *imaginary world* in which only one component can be malfunctioning as the system is diagnosed), then the resulting metrics would never indicate the maintenance implications that would result when multiple malfunctions must indeed be diagnosed. Combinations of multiple, simultaneous malfunctions that mask a diagnostic strategy's ability to isolate unambiguously could result in time-consuming diagnosis or exorbitant replacement costs. If this is not taken into consideration when generating Testability statistics, these statistics would have little predictive value and could not be verified using data gathered from the field.

The same holds true when analysis is restricted to a subset of all possible failures—such as those which can be automatically registered within the system (detected failures), are not exclusively associated with BIT (operational hardware), or result from a prescribed type of failure mode (fault universe). The Testability statistics that would result from these exclusions would not reflect the inadequacies that are potentially associated with isolating these *other* malfunctions—endemic and perhaps even inevitable inadequacies that may very well have prompted the exclusion of these failures from analysis in the first place. If responsible recommendations are to be made, however, these diagnostic inadequacies must also be incorporated into the evaluative metrics.

Imaginary worlds are not the only possible impediment to the predictive accuracy of Testability metrics. These predictions should take into consideration not only the isolation sequence and resulting fault groups but also knowledge about the way in which the system will be maintained. Such considerations include, but are not limited to, the types of maintenance to be employed (preventative, scheduled, condition-based) and the way in which each isolated fault group is to be repaired (block replacement, serial/prioritized replacement, or some combination of the two). Furthermore, predictive Testability metrics should be based on realistic estimates of the test and repair times and cost that will be required to maintain the system.

Once again, it is unlikely that this information will all be available when Testability is analyzed in early phases of product development. As we have already indicated, the solution to this dilemma is not to ignore the potential impact of maintenance considerations upon design Testability, but rather to employ intelligent estimates that can then be replaced with more accurate data as the process is iterated. By mandating the use of this information when it is available (and requiring the use of reasonable defaults when it is not), maintenance-based Testability metrics will circumvent the enabling conditions for what we earlier called statistical hypocrisy.

Challenge #4: Testability metrics should be calculated over specific (and perhaps multiple) time intervals.

In order for Testability to be meaningful in a predictive capacity, metrics must be calculated not over infinite time, but rather over meaningful intervals for the device or system in question. If nothing else, the interval over which these metrics are calculated should be restricted to the useful lifetime of the system or device. This allows analyses to properly take into account the fact that certain failures may be highly unlikely during the expected interval of deployment.

In addition to metrics that describe the maintenance and support behavior (such as MTTR) over the product's expected lifetime, it would also be helpful to provide statistics that describe how maintenance might change over time. It would be useful to know, for example, whether or not a maintenance characteristic (such as the MTTR) remains constant or whether it vacillates during different stages of the device or system's lifetime. This would be of particular interest if the MTTR were to reach unacceptable levels at some stage of product maturation (a new breed of contract requirements might mandate that the Expected MTTR cannot exceed a certain limit during the first twenty years of deployment). Metrics that depict changes in maintenance characteristics over time could also be used by the maintainer to optimize the maintenance plan so that it too changes over time. This will be particularly helpful for determining budgets, allocating resources and fixing bin levels for spare parts. These metrics would also be useful in determining the impact upon maintenance if a system or device were to remain in deployment beyond its intended lifetime.

If we are to meet this challenge we must question one of Testability's most deep-rooted practices—the use of failure rates in probability-weighted Testability metrics. Statistics in which the frequencies of failure are based only on failure rates (or MTBF) must be calculated over an arbitrarily long deployment. These measures are

unable to predict diagnostic performance within a specific interval of time or show how diagnostic and maintenance behavior changes over time. As we shall see, to generate failure frequencies for multiple time intervals, a diagnostic simulation should be employed that uses not only the estimated means (MTBF), but also the specific failure distributions—when known—for each component.

PROPOSED METRICS AND TECHNIQUES

Acknowledging that Testability analysis is perhaps the most effective way to improve the diagnostic capability of a design during all development stages, we would now like to propose some metrics and techniques that specifically respond to the four challenges proffered in the previous section.

First of all, it is important that Testability Statistics be expressed in terms directly related to maintenance goals (such as cost & time), rather than abstract rules of "good practice" (such as reduction of ambiguity). The challenge lies in the fact that most of the statistics traditionally used in Testability Analysis are expressed in varying units that are often not comparable to contractual goals.

In the early 90's, DSI International developed a Testability metric called *Isolation Effectiveness* as a way to compare different sets of Fault Isolation statistics using a single value. The primary advantage of this new statistic was that it expressed the diagnostic capability of a system or device in terms of deviation from a goal. The following formula represents DSI's original Isolation Effectiveness metric:

$$IE = 1 - \frac{\sum_{i=1}^{N} P_i (S_i - 1)}{\sum_{i=1}^{N} P_i S_i}$$

where N = the number of ambiguity group sizes

S_i = the *ith* ambiguity group size

P_i = The probability that a fault will be isolated to an ambiguity group of size *i*

This metric measures how well a particular diagnostic sequence can isolate to a single failed component (in other words, how much fault isolation deviates from the goal of isolating to an ambiguity group of size one). Because this metric represents a ratio, the actual units (dollars, hours, etc.) of the data from which the metric is computed are removed. This means that we can compare the Isolation Effectiveness of different diagnostic scenarios and even completely different designs for the purpose of considering alternatives.

Although, in its original form, Isolation Effectiveness is not directly related to either maintenance or support goals, this metric can be easily modified to express the fault isolation capability of a given diagnostic sequence with respect to any type of data. The following formula, then, gives the general form of this metric:

$$IE_x = 1 - \frac{\sum_{i=1}^{N} P_i \Delta_i}{\sum_{i=1}^{N} P_i V_i}$$

where Δ_i = the distance from goal *x* associated with the diagnosis and/or replacement of the *ith* ambiguity group

V_i = the computed value for ambiguity group *i* that is to be compared with goal *x*

P_i = the percentage, either actual or predicted, of isolations to ambiguity group *i*

N = the total number of ambiguity groups

This formula differs from the first one primarily in that the expression (S_I – 1) has been replaced by the variable Δ_i. The key to this metric is to characterize diagnostic performance as the computed difference (delta) between the individual attribute values and the maintenance goal that corresponds to those attributes. This modification now allows us to compute Isolation Effectiveness for any single design characteristic. To see this, let's compute Isolation Effectiveness with respect to Repair Time. For this example, we will be calculating how closely maintenance will meet a 1-hour turn-around time. Here are the Repair Times (V_i) and Relative Failure Probabilities (P_i) for each of the three ambiguity groups that can be isolated for this design:

AG #1: V_1=90 minutes, P_1=10%
AG #2: V_2=50 minutes, P_2=30%
AG #3: V_3=60 minutes, P_3=60%

Therefore,

$IE_{\text{Repair Time <= 60}}$ = 1 – [0.1*(90-60) + 0.3*(0) + 0.6*(0)] / [0.1*90 + 0.3*50 + 0.6*60]

= 1 – (3 / 60)

= 0.95

In this case, the calculated Isolation Effectiveness with respect to Repair Time is 0.95. This design could therefore be said to be 95% effective at isolating to ambiguity groups whose Repair Time is less than or equal to 60 minutes. Notice that an IE of 1.0 represents full compliance, and that for those repair actions that could be performed under the goal of one hour, delta is set to 0. By doing so, we are ensuring that we do not take an arbitrary benefit for those repair actions that exceed the goal. While this may at first seem to be harsh, it alleviates the misleading behavior of an MTTR when the deviation is great. For example, three repair actions of 1 minute, 1 minute, and 3 hours can produce an MTTR that appears to be compliant, yet fails miserably to meet contract goals stating no repair action longer than 1 hour.

There are of course some considerations when applying this formula to other metrics. For example, should the goal not be reduction but rather an increase, as would be the case for availability and reliability, it is important that the delta be computed by subtracting in the opposite order. Provided that the delta is always positive, this equation can be used with any diagnostic attribute to create an appropriate measure for Testability, Reliability or Maintainability.

The key to the acceptance of this metric, we believe, is in an attractive presentation to the customer. If new statistics demonstrate more accuracy or more insight into the design, customers are likely to include requirements using these types of calculations in future contracts. While many vendors (analysts) may wish to argue that they do not need any more requirements than those presently used in most contracts, there are numerous benefits even for the vendor. The following graphs demonstrate how Testability can use Isolation Effectiveness to recommend improvements. Although each of these graphs was generated using the same model, the actual numbers here are less important than the concepts and how the results of this metric might be applied.

First of all, we'll compute repair time and cost so that we can determine our level of compliance. In this case, we are required to have all faults repaired in 30 minutes or less at a cost of no more than $250.

Fault Probability	Time to Repair (minutes)	Delta
0.2188	71	41
0.2878	59	29
0.0504	83	53
0.0504	43	13
0.0504	57	27
0.1691	60	30
0.0621	64	34
0.0490	76	46
0.0490	16	0
0.0087	36	6
0.0043	31	1

$$IE_{\text{Repair Time <= 30}} = 0.48$$

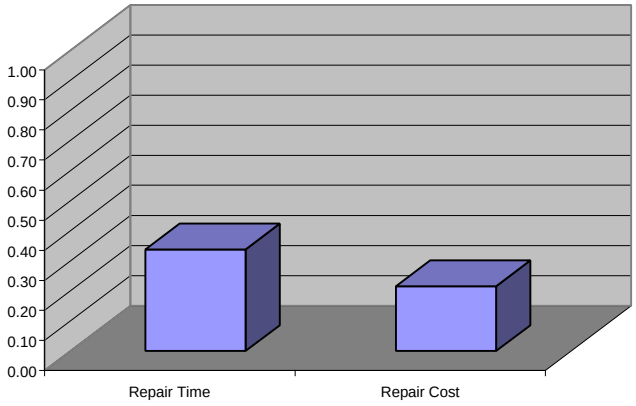
Notice that, although only one ambiguity group meets the goal of being repaired within 30 minutes, we have nevertheless 0.48 compliance with this requirement. This demonstrates the qualitative ability of this metric as a measure of progress towards a goal. If the Time to Repair for the first ambiguity group were reduced from 71 to 36, (a number that still does not meet the 30-minute goal), the Isolation Effectiveness would improve to 0.55—even though the percentage of compliant ambiguity groups has not changed.

Fault Probability	Cost to Repair	Delta
0.2188	149	0
0.2878	905	655
0.0504	106	0
0.0504	110	0
0.0504	127	0
0.1691	30	0
0.0621	212	0
0.0490	71	0
0.0490	194	0
0.0087	49	0
0.0043	31	0

$$IE_{\text{Repair Cost <= \$250}} = 0.45$$

All but one of the isolated ambiguity groups meet the goal of a Repair Cost less than 250 dollars. The poor Isolation Effectiveness results from the single, highly-probable isolation that costs $905 to repair. This again demonstrates this metric's ability to quantify progress towards a goal. If we were to reduce the Repair Cost for the second ambiguity group to $305 (only $55 more than the goal), the IE would rise to 0.91. Now, instead of changing the Cost to Repair (it will remain at $905) let's reduce the Fault Probability for that group (to keep the calculation neat, let's simply swap probabilities with the third group so that the second ambiguity group is now isolated only 5% of the time). The new IE would be 0.78, since our seriously out-of-spec ambiguity group would be isolated less frequently.

Using our original values, here is a graph that depicts two Isolation Effectiveness metrics—one calculated with respect to Repair Time and the other with respect to Repair Cost:

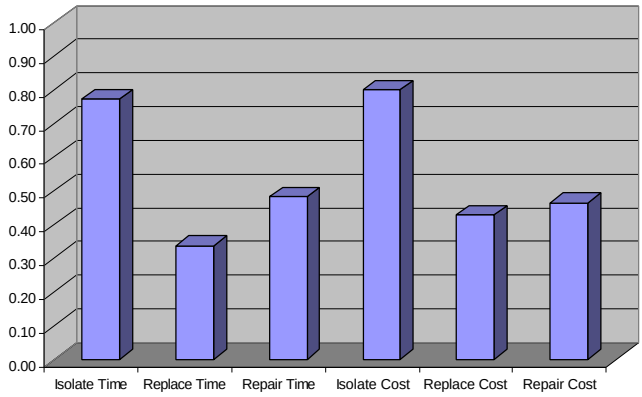


While this is valuable information, it does not provide us with a direction to pursue in order to meet our goals. We must break the two repair calculations into their constituent parts:

RepairTime = IsolationTime + ReplacementTime

RepairCost = IsolationCost + ReplacementCost

If we calculate separate Isolation Effectiveness metrics for Isolation Cost and Time, as well as Replacement Cost and Time, we can include them (along with our Repair Cost and Time metrics) in a single graph. This graph now shows us what has caused Isolation Effectiveness with respect to Cost and Time to Repair to be out of compliance.

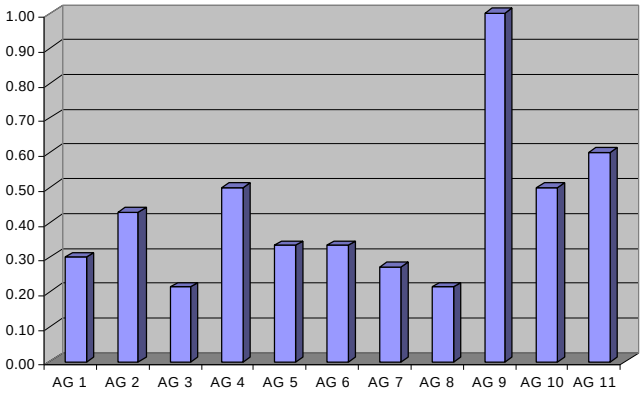


From this graph, it is clear that poor Replacement Time and Costs are what are preventing Repair Time and Cost from meeting their respective goals. This clearly demonstrates the importance of not overly reducing statistics into an overall figure, especially for purposes of Testability and design recommendations.

This example also demonstrates the advantage that the Isolation Effectiveness metric offers over expected values and sample means (such as MTTR). Because Isolation Effectiveness is ultimately a measure of compliance, it produces values within the same range (between 0 and 1), regardless of the attribute on which it is based. This is what allows us to look at the first two columns of this graph and immediately notice that the Time to Isolate is significantly more compliant than is Replacement Time.

The final step (one that allows our Testability analysis to suggest where we can improve the design's diagnostic capacity) is to enumerate one step further and display effectiveness on a per-fault group basis.

The following graph depicts Isolation Effectiveness with respect to Replacement Time.



The usefulness of this graph is immediately apparent. The most problematic ambiguity groups are AG3 and AG8 and we can see their relative impact on Isolation Effectiveness versus the other ambiguity groups. That is, AG2 is approximately twice as compliant as AG3, AG4 twice AG1, etc. Using this technique one can begin to quantify exactly how much benefit can be achieved and how to go about achieving it.

We could even go further by displaying the least compliant fault group in terms of its constituent items—and once again to show its constituent functions. While all of this may be important to the Testability engineer in increasing overall design compliance, it should not be overlooked that the same calculation that was used to calculate the low-level recommendations was also used to obtain the high-level compliance statistics that might be expected in contractual requirements. This allows the Testability engineer to make specific recommendations that lead toward compliance with a goal expressed in terms of maintenance data.

Now that we have seen how Isolation Effectiveness can be applied across any range of statistics, we would like to introduce a new concept that further refines this calculation. Specifically, we will be discussing how to calculate our same effectiveness statistic over specific intervals of time.

So far, all statistics provided have been computed over the entire set of ambiguity groups, which are each probability-weighted. Although these statistics provide valuable information, they are often overused as an indication of what one can expect from the system on a year-to-year basis. The inherent flaw in this logic is that these statistics are based on infinite time—that is, the ambiguity groups sizes, repair costs, etc. have all converged on an expected value. When this is then used for budgeting, however, it is possible that the cost or time allocated could be grossly wrong.

To accurately calculate effectiveness or expected values (MTTR, for example) over time, it is vital that a simulation engine be used. The power of a simulation engine is in its ability, over a large number of simulated lifetimes, to accurately produce only those failures that would occur in the field during the time interval in question. Failures that occur earlier in the system's life will bias the statistics accordingly. If run long enough, the simulation's expected values will converge on the expected values that were computed directly using the failure rates from the different ambiguity groups.

While the full details of a simulation engine are beyond the scope of this paper, here are a few of the features and restrictions associated with the simulation engine that we used to calculate the upcoming statistics:

- Some components have been modeled with exponential failure distributions, while others use a standard normal distribution. At the heart of any simulation engine of this type is a random number generator that produces faults in accordance with the component reliability and a failure distribution curve. Our simulation is somewhat advanced in its ability to use mixed types of distribution curves.

- Faults are instantaneously detected, thereby eliminating almost any possibility of multiple failures during isolation.

- The system is assumed to be taken offline while testing and repair is performed. In other words, new failures are not allowed to occur, nor do parts age during maintenance.

- When a fault is isolated, the entire ambiguity group is replaced with new parts.

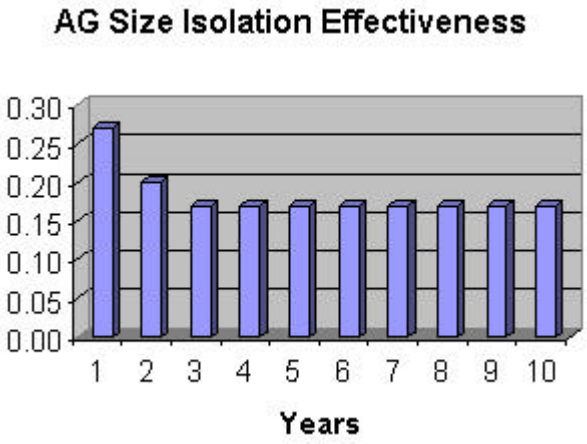
Before we proceed, it is important that we respond to the argument that changes in maintenance philosophy (such as using serial replacement instead of block replacement) will eliminate many of the effects that we are about to demonstrate. This argument is short-sighted if it leads to the belief that a simulation engine is not required in these cases. In researching the statistics shown here, we have found that extremely subtle changes can have dramatic and sometimes unexpected results. The only way to accurately predict the system behavior is using a time-based simulation.

Before we look at the simulation results, let's look at the predictions generated using standard (traditional)

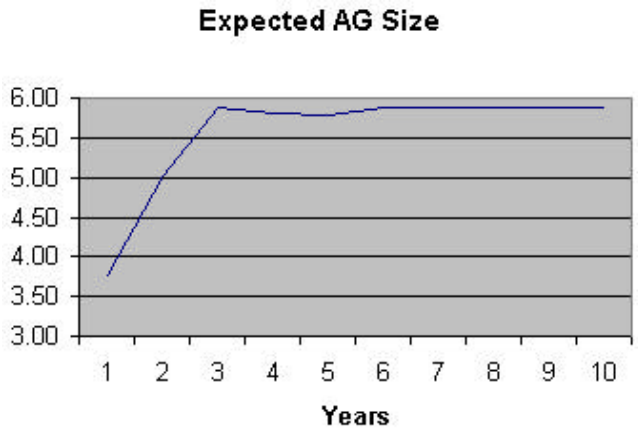
Testability metrics. We can expect any simulation to converge on these numbers since they represent the expected values at near-infinite time. In fact, running a simulation can help confirm whether the system being tested converges fast enough to be accepted as meeting the goals. Obviously a system that takes 30 years to converge is likely to be questioned when the support costs are nowhere near the goals for the first 5-10 years. In such cases, customers may require a redesign in order for the system to meet contracted maintenance requirements. Here are Testability and Maintainability numbers computed using failure rates (means) to represent relative frequencies of failure:

AG Isolation Effectiveness:	0.17
Expected AG Size:	5.87
Expected Cost to Isolate (MCTI):	$8.66
Expected Cost to Replace:	$388.29
Expected Cost to Repair (MCTR):	$396.94
Expected Time to Isolate (MTTI):	37 minutes
Expected Time to Replace:	178 minutes
Expected Time to Repair (MTTR):	215 minutes

Next, lets analyze several of the simulation statistics (each over a ten year time span) in comparison to these projections. First up is Isolation Effectiveness with respect to ambiguity group size (the goal being isolation to a group size of 1).

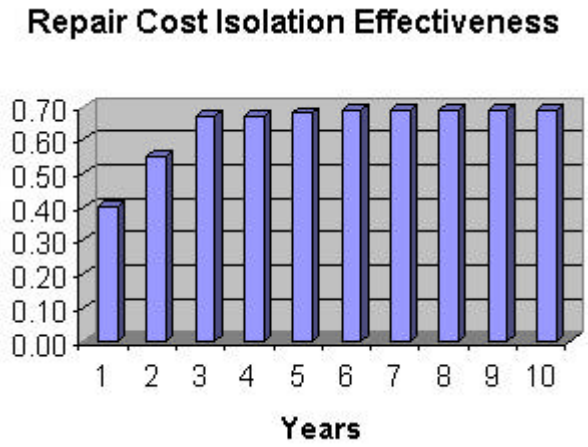


This graph indicates that, during the first few years, we are able to isolate to a single component more effectively than in later years. The metric converges rapidly to our mean, however. Our next graph confirms this in terms of the expected AG size. Note how it converges to our original (traditional) calculation of 5.87.



What is interesting about this particular system is that our new technique, if used in this limited way, would say that this system is fantastic. During the first few years, it is actually easier to isolate to smaller groups, whereas future years simply approach what was originally predicted—better performance than even traditional Testability statistics would imply.

Unfortunately, if for this system we were to assess Testability solely in terms of isolated ambiguity group sizes, we would be sadly misled about the system's diagnostic capability. The following graph depicts the Isolation Effectiveness with respect to Repair Cost for this same system.

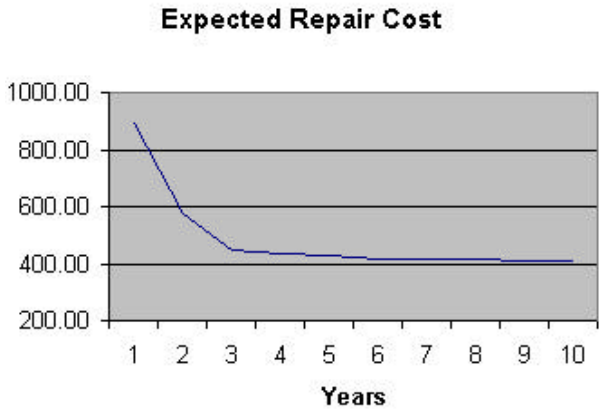


Keep in mind that not reaching an effectiveness of 1.0 doesn't mean we haven't met contract goals that only call for an expected repair cost. Isolation Effectiveness is more sensitive in that it indicates when any single repair action has not met the specification.

Notice that the Isolation Effectiveness with respect to Repair Cost is significantly lower in the first couple years, and converges by about the fourth year. Our

goal was $400, and the cost of testing was not a significant amount of the repair cost.

Our next graph shows Repair Cost in dollars, and we can actually see that although Isolation Effectiveness converged in a few years, the Expected Repair Cost takes several decades to converge to our original calculation. In fact, the Expected Repair Cost is has only reduced to $401 at the 30-year mark—a clear indication of just how long a time frame the standard testability statistics are based on. With this kind of difference, almost any customer would question the accuracy of the original testability predictions.



What is particularly disturbing for anyone who is unaware of this trend, is that using the Mean Cost to Repair (MCTR) of $396.94 to derive a yearly per-unit support cost, could easily use up the entire year's budget in the first few months of deployment. In fact, this system will be severely under-budgeted for at least the 5-10 years! Since the exact over-budget amount depends on how many units are fielded, the next table shows the per-unit cost which used in this analysis:

Year of Deployment	Expected Repair Cost ($)
1	894.17
2	576.26
3	446.74
4	437.92
5	426.63
6	417.64
7	418.05
8	416.09
9	412.34
10	411.44

Now, If we were to have 1,000 units with a budget of roughly $396 per unit (the traditional Testability prediction), we would have a severe budgeting crisis during the first decade of deployment. The following table shows precisely how much over budget we would be during the first ten years:

Year of Deployment	Total Amount Over Budget ($)
1	497K
2	677K
3	726K
4	767K
5	797K
6	817K
7	838K
8	857K
9	873K
10	888K

Thus, despite dropping to within $40 of our original prediction in the first four years, our 1,000 units are nearly $1 million over-budget by the tenth year!

One of the most interesting things learned from simulation is that there are very often small deviations even after the numbers have apparently converged. For instance, in the first table, year 7 has a higher support cost than year 6. While it is small, this could represent a significant change in what components are failing. A different system might not have a much more substantial change towards higher cost, but might require a far more sophisticated sparing strategy to account for the different faults encountered during this year. What we have determined, is that one really never knows what effect a subtle difference might produce until a full simulation has been run. Some of the most dramatic changes in numbers can arise from the smallest of changes. Increasing the cost by only a couple of percent for components which factor heavily in the final statistics can easily mean the difference of compliance or not.

CONCLUSIONS

This paper has attempted to expose and address several severe problems with the statistical methods currently employed by Testability and Maintainability analyses. We hope that we have also demonstrated that the benefits of simulation-based metrics need not be confined to development analyses. This could be the start of a new era in diagnostic development—one in which model-based diagnostic analysis, because it can accurately predict support costs, times and even spares on a year-by-year basis, can become an essential input to year-by-year allocation procedures, replacing decade-old predictions based on "best-guess" reliability means.

If, at the very least, the Testability metrics described in this paper were to be computed for the first 20 years of a system or device's useful life, these predictions would be more useful than current measures for determining sparing and costing strategies. Whereas traditional Testability statistics help us formulate long-term expectations, they are far less meaningful when used to establish specific maintenance procedures and budgets.

Finally, it must be remembered that, when contract conformance is measured using standard Testability techniques and metrics, these calculations statistically include many failures that are not likely to occur during the expected lifetime of the system or device in question. Furthermore, because these assessments are expressed in terms of isolated ambiguity group sizes, the relationship between Testability and Maintainability is, at best, ambiguous. If Testability is to fulfill its promise and become an essential and unequivocal development practice, the entire development community must respond to the challenges presented within this paper. We hope that the metrics and techniques that we have proposed provide a good starting point for discussions about the future of Testability as a discipline.