Software Reliability Revisited

Paul B. Moranda

Book reviews normally appear in the Book Reviews section. Comments on book reviews—and on anything else in Computer—normally appear in the Letters to the Editor section. The following item arrived as a letter to the editor—but it’s really a book review of a book we’ve already reviewed.

All things considered, it seemed clear that Open Channel was the proper home for Mr. Moranda’s remarks.

I found the review by Dennis Geller of the Glenford J. Myers’ book, *Software Reliability: Principles and Practices* (Computer, October 1977, pp. 117-118), provided excellent coverage of the principal theme of the book, that being software, vis-a-vis reliability. While I concur with Mr. Geller on all of the points which he makes, both favorable and unfavorable, my own reading of the book focussed on the reliability aspects of the material presented.

From this perspective there are several comments on the book which I offer for consideration. I feel these comments are especially timely, since the Myers’ interpretation of reliability, presented in the first book on the subject, reinforces the erroneous concept that reliability “equates to” perfection.

It is necessary at the outset to point out that Myers’ book is not about software reliability as it has come to be defined. The following comments focus exclusively on this point.

The author seems to be apprehensive about the time aspects of software reliability. However in the only definition given for it in the book, time and probability are two of the principal parameters (with cost “weighting” the probability).

Myers discusses many techniques for developing error-free software, and although there are few quantitative metrics or gauges of the desired effects of the application of these techniques, the discussions are educational and form a good survey of both software and hardware. But with respect to the subject—reliability—the book is a total loss. The first egregious errors in this connection appear under a description of system testing (p. 233). There, under an enumeration of testing types, in an item of the list called, “Reliability/Availability Testing,” Myers makes some remarks that are disputable—and some that are patently wrong.

First, he notes that although “errors can be purposely injected into the system to test the detection, correction, and tolerance functions, other types of reliability are almost impossible to test.” Then he notes that if a system has a 250-hour MTTF objective, one has to operate much longer than 250 hours to establish statistically that this objective has been met. This is of course a credible remark only if there is no model for the failure mechanism.

Many hardware systems (e.g., those used in deep-space probes) have an MTTF design objective of several years, and in many cases this time exceeds the total time required to design, develop, and test the system. Even in the absence of sophisticated models, some estimates can usually be given as to the probability of achieving a 250-hour MTTF, on the basis of tests which truncate successfully, or otherwise, in much less time.

Next, Myers states that “showing that the system will contain at its time of delivery more software errors than the stated objective is impossible. for if you could count the number of remaining errors, you could correct them.”

This remark appears to ignore much of the published literature in probability, sampling, statistics, or estimation as applied to the error detection problem. *Estimating* errors, after all, is not the same as *counting* errors.

The indication of Myers’ “low” opinion of the applicability of probability-based models is manifested by his inclusion of the field under a part of his book called “Additional Topics.” Within this part, and in a section entitled, “A Reliability Growth Model,” Myers discusses some of the options available. He calls them “primitive techniques.” In this discussion Myers starts...
by improperly crediting M. L. Shooman as a co-developer of the primary model described in the section. This model was initially described in 1971 by Jelinski and Moranda (Myer’s Reference 18.1). Two years later, in 1973, Shooman, by distorting all of the definitions in his original model, found a way of “equating” the two.

After an initial discussion of the elementary reliability concepts, the author discusses the Jelinski-Moranda model. The basic equations for determination of the parameters are given, and the underlying assumptions are repeated. By use of a figure (Figure 18.1), the author provides a clarification of a point which has escaped many workers in the field: viz., the model has a different reliability curve for each error, and as errors are removed and time is reset to zero, the reliability curves show an improvement.

A description is then given of an “extension” to the basic model. This is a misnomer, since the rate curve illustrated (Figure 18.3) is one of the two used in the original (1971) Shooman model and is not relevant to a description of individual errors. In this “extension,” the error detection rate curve is a triangle, starting at zero, peaking at an initially unknown point, and ending at zero at project completion.

Myers then observes that the model has no predictive potential, since the error removal process has to be complete before the triangle can be determined. The “model” is not an extension of any model, it is merely an empirical description of how the errors occurred, and there is no description of an underlying mechanism. As noted above, Shooman uses the same error curve in the referenced paper (Reference 18.3) to determine the parameters for his “constant error rate” model (by averaging the triangle height over its base).

In the lead sentence of Myers’ critique of the primary growth model, he notes that the model “makes many assumptions and all of them are questionable.”

The first, which Myers considers the “basic assumption,” is that “all errors have equal severity.” As a matter of fact, this assumption is not only not basic, it is not even a part of the model: the purpose of the model is to determine the number of errors of all kinds, and the severity is not a consideration at all. If the user desires to focus on only the “severe” errors, then the model should be applied on a subcategory of the data.

A “second assumption” (which is actually one of three—not “many”—assumptions) is that “all errors are corrected immediately.” This is of course true, but in most applications which are made the “repair” time is small with respect to the testing period. Moreover, as shown in the illustrations of his suggested hazard rates (Figure 18A), all of them have this “defect.” The third Myers’ assumption is “that the program is not being altered.” This is again true, but one cannot conceive of any model which can know in advance the form which the program will take as new mods and revisions are made; that involves clairvoyance. So it can hardly be counted as a model-peculiar assumption, for all models are so “defective.” (The Mill’s error-seeding model mentioned later has this defect.) The “fourth assumption” is that all corrections “correct the detected error and do not introduce new errors.” This too is true but not uncommon; it is generally assumed in hardware reliability studies involving replacements that the new item is as good as the replaced item.

The analogy with software is apparent. Although introduction of an imperfect “repair” factor may some day be possible, it is not considered useful at this point in the development of models, because of a lack of historical data.

Myers then finds it objectionable that the model’s hazard rate has the stepdown nature of Figure 18.2. It is noted that the removal of each error reduces the rate by a constant amount. (This is indeed a basic assumption of the model.) Myers notes that this is probably a reasonable assumption “to start with” (hence not questionable). The “more interesting assumption” is that the hazard rate “is constant between errors.” He follows this with the comment: “Although it is reasonable to want to express software reliability in relation to time, one should recognize that it is really not a function of time.”

This comment seems sharply at variance with his own definition of software reliability (on p. 7), which states that it is a “probability . . . for a particular period of time.”

In the large context (and in what appears to be a non-sequitur to his objection registered against the assumption), he notes that software reliability is a “function of the number of errors, the severities and locations (sic) of those errors, and the way the system is being used.” It does not seem possible to uncover his meaning here. It would appear on the face of what he has written that Myers is “saying” that a measure of software reliability is never known: the number of remnant errors.
is not known, the severity of yet-to-be uncovered errors is hardly known, and the "location" (whatever that may mean) of future errors is, again, impossible to know; and finally, in their composite, the function connecting these elementary parts to a gauge of reliability is not known.

In arguing against the constant rate between errors, Myers introduces several other possible hazard functions (Figure 18.4), none of which have any accompanying parameters which would permit quantitative description of what changes take place as errors are detected.

It is noted, however, that Myers in all three cases implies an assumed "instant correction," since in the cases illustrated there is a zero-time jump in the hazard function.

Myers notes that "all of the guesses (sic) of the behaviour" of the hazard function show reliability growth, but that this "is not always the case in real programs." He quotes data from Craig (Reference 18.6) and Richards (Reference 18.7) for support. The latter argues that "each program has its own unique distribution of Z(t) and each installation of each program has a unique Z(t)." This is a literally miraculous finding, since if there is variability in the hazard function it is impossible to observe it from data on a single program. Incidentally the finding which Myers credits to Craig did not appear in the final report on the cited study (a TRW study contract with RADC).

If the Richard's remark has any credibility, under some unstated conditions, it is probably based on data (a record of error counts per unit time) from a total system, the configuration of which is almost never defined or recorded precisely across time. Generally the "unique Z(t)" is merely a picture of the unique way the system evolves and is tested.

Under a heading, "Other Probabilistic Models," Myers mentions some work by Schneidewind (References 18.8, 18.10, 18.11) in which it was found that the environment is a more significant factor than the number of remaining errors. This again shows a lack of understanding by Myers (and perhaps Schneidewind) of the primary model, for it is the detection of error which is being described by that model and if the "environment" is harsh the detection rate of errors will increase. The variability in these environmental effects can be accounted for by a transformation in the timing metric as explained in the original description of the model (Reference 18.1).

Myers next describes "An Error Seeding Model" which "makes no assumptions about the behaviour of the hazard rate Z(t) and is built on firm statistical ground." This model, attributed to Mills (of IBM) but originally applied by J. Neyman 30 years ago in a "fish-counting" experiment, assumes the seeded errors are as easy as or hard to detect as the "indigenous errors." By testing the program for "some amount of time," one can find the total number of indigenous errors on the basis of the numbers of seeded and indigenous errors found in the sample.

Whereas this model would appear to apply in some regimes, it has an obvious and admittedly weak point: the seeded errors must be representative of "typical" errors. Myers notes that "we do not yet understand programming well enough to know what these typical errors should be." He says that this seems to be a "solvable" and "relatively minor" problem. This may be so for trivial errors but certainly not for critical errors. It is certainly not clear how (or why) one would insert errors of major severity into a program. The method seems to be limited to errors of the kind normally edited out by utility programs.

A section titled "Simple Intuitive Models" describes an imagined experiment in which two "completely independent test groups" using "independent sets of test cases" (whatever this may mean) test the program for "some period of time." Then their two error counts, partitioned into those common to both and those unique to each group, are used to determine the total residual errors. This simple model is just that. Experience seems to indicate that one or the other group will almost surely incur one or more "crashes" which will stall the testing. Hence the implied assumption that equal time produces equal testing will (probably) not hold. The liability to slight changes in the number of common errors is also noteworthy; this is generally a small number, and slight changes produce a large effect on the estimate (a change from 1 to 2 doubles the estimate). The fact that the "commonly" found errors are probably easier to find than others dells the procedure.

*Myers notes the necessity to assume all errors have same chance of being detected, and states that this is "not entirely unreasonable": on the other hand, when the hazard rate models were described, the same assumption was deemed to be "questionable."