PREDICTING SOFTWARE WITH PARAMETER EVALUATION

INTRODUCTION

Software reliability measurement and prediction are used to evaluate model parameters in *advance* of applying the model. Measurement involves collecting and analyzing data about the observed reliability of software, from which the parameters are estimated, for example, the occurrence of failures during test. Prediction is using a model to forecast future software reliability, for example, failure rate during operation. Once the parameters are estimated, using the methodology we will demonstrate, we can rank the software releases by their *projected* relative reliability in order to rationally allocate resources to prediction and testing activities.

In Reference 1, it is stated that there is no way, a priori, to determine whether software reliability model. A will produce more accurate predictions than model B. They use the prequential likelihood ratio (PLR) to determine, *after predictions are made*, which model produced the more accurate predictions. They do this by defining the PLR as follows:

$$\text{PDR} = \prod_{i=1}^{n} (\ f_i(A))/(\ f_i(B))$$

where $f_i(A)$ are the probability density functions (pdfs) of predictions for Model A and $f_i(B)$ are the pdfs of predictions for Model B. If Model A is more accurate, the trend of PDR would increase; conversely, if Model B is more accurate, the trend would decrease.

Although the authors' statement may be true about *comparing model predictions*, we have developed a methodology for ranking, a priori, the relative reliability of releases of a software system, using the failure rate parameters of a *given model*. In addition, the ranking methodology allows us to allocate critical resources, such as test time, to the releases *prior* to making predictions.

SCHNEIDEWIND MODEL OBJECTIVES (2)

To demonstrate the prediction methodology, we must use a software reliability model. As the Schneidewind model has been used on the NASA Shuttle flight software for reliability predictions (3), and we have a considerable amount of Shuttle failure data, we use the model and data to demonstrate our methodology however, it is important to note that it is not the particular model that is important in this article. Rather, it is the *methodology* that is key. The approach articulated in this article could be applied using any one of a number of other models (4).

The objectives of this model are to estimate or predict the following software product attributes:

$D(T_L)$ Predicted failure count in the range [l,∞]; maximum failures over the life of the software
$D(T)$ Predicted failure count in the range [l,T]
MTTF Mean time to failure
$p(t)$ Fraction of remaining failures predicted at time t
$r(t)$ Remaining failures predicted at time t
$T_F(t)$ Time to next failure(s) predicted at time t
R_i Function of parameters α and β for allocating resources to predicting and testing OI i

PARAMETERS USED IN THE PREDICTIONS (2)

α Failure rate at the beginning of interval s
β Negative of derivative of failure rate divided by failure rate (i.e., relative failure rate)
s Starting interval for using observed failure data in parameter estimation
t Test time; interval of observed failure data; current interval
T Future time of predicted reliability metrics
Δt Increment of time between *future time of predicted reliability metrics* and *test time* = $T - t$

OBSERVED QUANTITIES (2)

X_{s-1} Observed failure count in the range [1, $s - 1$]
$X_{s,t}$ Observed failure count in the range [s,t]
X_t Observed failure count in the range [1,t]
T_j Time between failure j and $j + 1$
$N(t)$ Cumulative failure count in the range (0,t)

DEFINITIONS

OI Operational increment: NASA Space Shuttle software release. In another application, this could be any software *object* such as a module, subsystem, etc.
i Operational increment identifier.

BASIC PHILOSOPHY (2)

The basic philosophy of this model is that, as testing proceeds with time, the failure detection process changes. Furthermore, recent failure counts are usually of more use than earlier counts in predicting the future. Suppose there are t intervals of testing and f_i failures were detected in the ith interval, one of the following can be done:

- Ignore the failure counts completely from the first $s - 1$ time intervals ($1 \leq s \leq t$), and only use the data from intervals s through m.

SCHNEIDEWIND MODEL RELIABILITY RANKING APPLICATION

The purpose of this application is to rank reliability on the basis of the parameters α and β, *without making a prediction*. This is important because, with this ranking, we can rationalize the allocation of time and effort to the prediction and testing functions, resulting in two benefits: 1) Prioritize prediction and testing activities based on need (e.g.,. assign the highest priorities to the lowest reliability

software), and 2) conserve human and machine resources by not wasting them on software that *a priori* is judged to be high reliability. The parameters α and β are, respectively, the initial value of the failure rate and the rate at which the failure rate changes. It is desirable to have the largest ratio of β to α for high reliability because this will yield the fastest decrease in failure rate combined with the smallest initial failure rate. Thus, after estimating α and β, using a tool such as SMERFS (5) or CASRE (6), rank reliability without, or before, making predictions. This procedure is useful for doing an initial reliability screening of projects to determine, for example, which projects might meet reliability specifications and which require reliability improvements.

We will use the ratio β/α in three ways: 1) Make several reliability metric predictions and plots to see whether increasing β/α corresponds to a *decreasing* reliability trend; 2) if this is the case in 1), use β/α to allocate a given amount of test time to the OIs; and 3) see whether a fit can be made to the plots in 1) to produce regression equations for predicting reliability metrics as a function of β/α.

In the analysis that follows, all failures are treated as having equal severity. Although this is not strictly the case, it is an appropriate assumption because the Shuttle development process requires the correction of all faults no matter how minor the failure caused by the fault.

Applying Ranking to Predicted Fraction Remaining Failures (2)

First, staring to use β/α in the first way, compute a *fraction of remaining failures* predicted to occur at time t in equation (2):

$$p(t) = \frac{r(t)}{D(T_L)} \tag{1}$$

The ranking obtained by using equation (3) is shown in Fig. 1. According to the criterion of equation (8 see the ranking equation below), OI2 would be allocated the most test time t and OI5 would receive the least amount of test time.

Applying Ranking to Predicted Remaining Failures (2)

First, compute *remaining failures* predicted to occur at time t in equation (3):

$$r(t) = (\alpha/\beta) - X_{s,t} \tag{2}$$

Then, recognizing that it is wise to use more than one type of prediction when doing the ranking in order to not base the test time decision on a single result that could be a statistical fluke, we produce Fig. 2. Happily, we find the same ordering of ranks in Fig. 2 as were obtained in Fig. 1.

Applying Ranking to Predicted Total Failures (2)

A third example is obtained by predicting total failures over the life of the software using equation (4):

$$D(T_L) = \frac{\alpha}{\beta} + X_{s-1} \tag{3}$$

With this prediction in hand, we produce Fig. 3 that, again, has the same ranking as in the two preceding cases. In addition to Fig. 3 providing a ranking, we demonstrate the second way of applying the β/α ratio. For example, we obtained an accurate fit (e.g., R^2 = .9316) with the actual data. Equation (5) can be used for predicting total failures for OIs not in the original data set. This is significant because, with equation (5), we can predict total failures with *no prior* knowledge of model parameters other than α and β.

$$D(T_L) = 37.382\, e^{-10.45(\beta/\alpha)} \tag{4}$$

For example, we would predict $D(T_L)$ = 37.382 $e^{10.45(0.079378)}$ = 16.31 failures for OI8, which is not a member of Fig. 4. The actual failure count = 15 or a relative error of .0873.

Applying Ranking to Predicted failure count $D(T)$ in the range [1,T] and to MTTF

From Reference 2, we have equation (6):

$$D(T) = (\alpha/\beta)[1 - e^{-\beta(T-s+1)}] + X_{s-1} \tag{5}$$

Next we compute the MTTF for the OIs in order to rank this metric of reliability. We compute MTTF by considering the total number of failures that have occurred in the time $T = t + \Delta t$ (see definitions). Thus, we have

$$\text{MTTF} = T/D(T) \tag{6}$$

Again, in Fig. 4, we have confirmation that reliability parameters—namely $D(T)$ and MTTFr—can be accurately ranked by the ratio β/α. We also note that, because *decreasing* $D(T)$ corresponds to *increasing* reliability and *increasing* MTTF corresponds to *increasing* reliability, there is a downward trend for $D(T)$ and a upward trend for MTTF.

Ranking of Test Time Results

Now that we have confirmed the correct ranking by β/α, we can develop the function for the *inverse* allocation of resources to an OI as given by equation (8):

$$R_i = \frac{1 - [(\beta_i/\alpha_i)/\sum_{i=1}^{n}(\beta_i/\alpha_i)]}{\sum_{i=1}^{n} 1 - [(\beta_i/\alpha_i)/\sum_{i=1}^{n}(\beta_i/\alpha_i)]} \tag{7}$$

Then, in particular, to apply equation (8) to test time t, we multiply equation (8) by t:

$$R_i t$$

Figure 5 portrays the ranking achieved of test time t = 300 days for the Shuttle. This is just one example of many rankings that could be done using equation (2). For example, the labor time of testers could also be assigned with this algorithm. We note the fact that OI2 has the worst reliability in Figs. 1–3 and one of the worst in Fig. 4. Thus, it is assigned a relatively large amount of test time in Fig. 5. Converesly, OI5 has the best reliability in Figs. 1–3 and is assigned the least amount of test time in Fig. 5.

Table 1 summarizes the parameter evaluation results. The reliability metric or error value that is the worst in each row is bolded. In general, the worst OIs are 2, 6, and 7. These OIs would be given priority attention, for example,

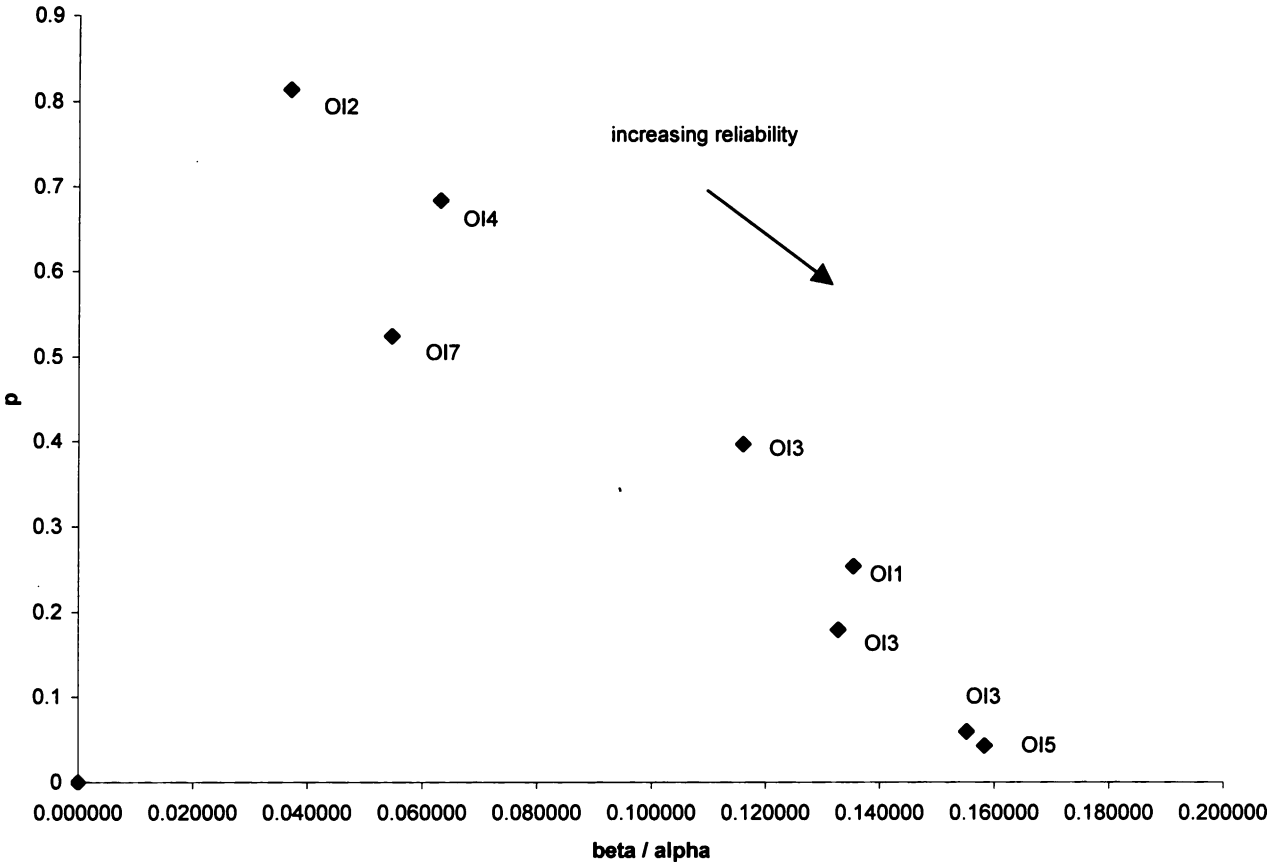

Figure 1. NASA Space Shuttle fraction of failures remaining p vs. parameter ratio (β/α) for OIs.

Table 1. Summary of Reliability Parameter and Metric Analysis Results

Parameter/Metric	OI1	OI2	OI3	OI4	OI5	OI6	OI7	OI8
α	0.695865	0.922051	0.863540	0.316453	1.895283	1.000007	1.461051	0.530851
β	0.094273	0.034311	0.134070	0.020048	0.300182	0.000001	0.080146	0.042138
β/α	0.135475	0.037211	0.155256	0.063354	0.158384	0.000001	0.054855	0.079378
$p(t)$	0.254	**0.814**	0.059	0.683	0.043	could not obtain a prediction	0.524	0.559
t	14	11	25	24	14	13	12	18
$r(t)$	2.38	**21.87**	0.44	10.78	0.31	could not obtain a prediction	13.23	7.60
$r(t)$ relative error	0.278	0.191	1.564	1.157	0.937	could not obtain a predicdtion	2.307	**2.799**
T	103	25	36	45	17	17	28	81
$D(T)$	9.381	13.343	7.340	8.706	7.186	14.000	**21.560**	13.064
$D(T)$ relative error	0.107	**1.369**	0.049	0.436	0.327	1.251	0.768	0.758
$D(T_L)$	9.381	**26.874**	7.440	15.784	7.314	could not obtain a prediction	25.230	13.598
$D(T_L)$ relative error	0.042	**2.839**	actual failure count unknown	0.578	0.437	could not obtain a prediction	0.768	0.700
$T_F(t)$	88.833	1.364	18.576	4.494	4.494	could not obtain a prediction	**0.981**	3.349
$T_F(t)$ relative error	0.869	0.892	0.615	0.791	0.791	could not obtain a prediction	**0.943**	0.922
$R_i t$ allocation of 300 days of test time	20.86	21.66	20.38	22.60	20.31	24.13	22.81	22.22

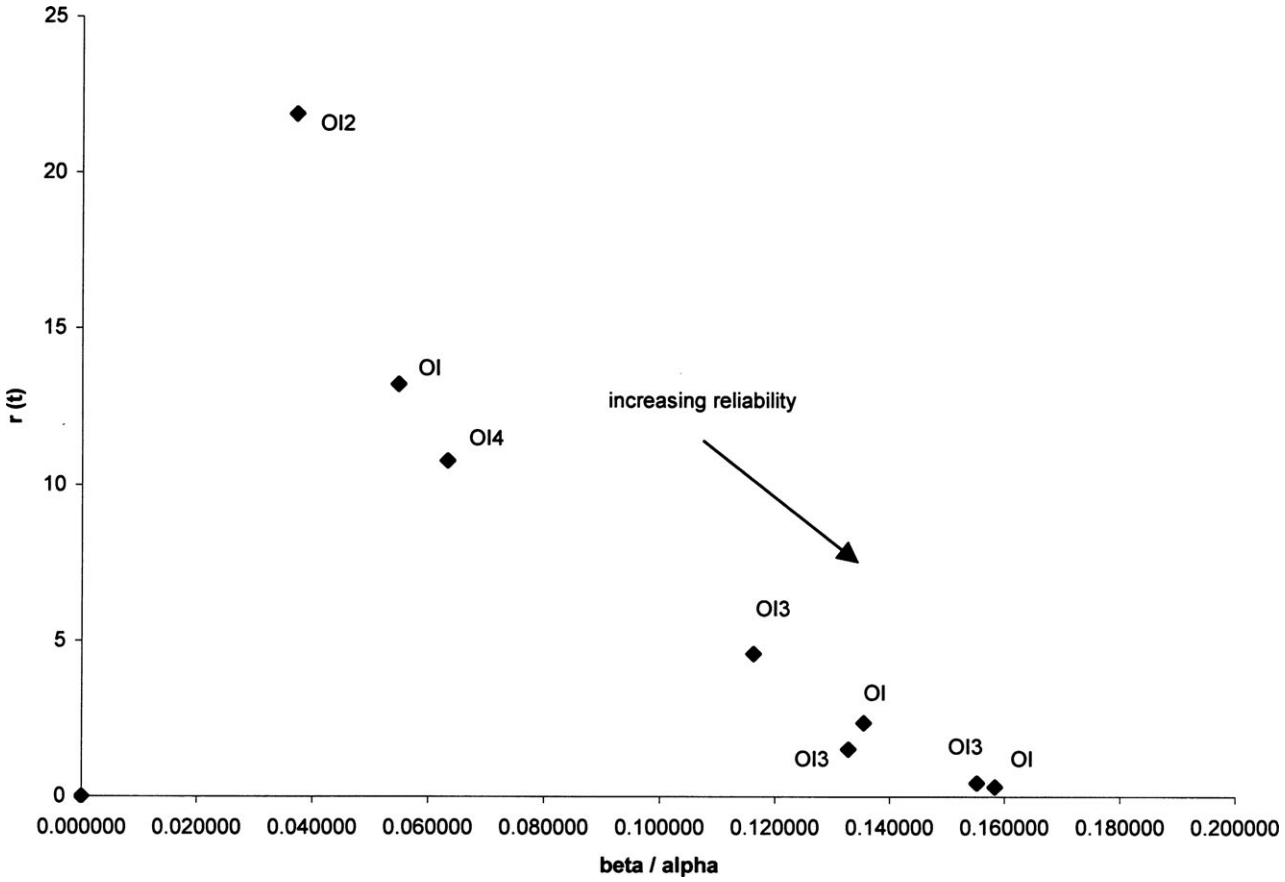

Figure 2. NASA Space Shuttle remaining failures $r\,(t)$ vs. parameter ratio (β/α) for OIs.

in allocating test time in Fig. 5. Conversely, in general, OI8 is the best in Table 1 and, correspondingly, receives the least test time allocation in Fig. 5.

Definitions:

α:	Failure rate at the beginning of interval s
β:	Negative of derivative of failure rate divided by failure rate
$p(t)$:	Fraction of remaining failures predicted at time t
t:	Test time; last interval of observed failure data; current interval
$r(t)$:	Remaining failures predicted at time t
T:	Future time of predicted reliability metrics
$D(T)$:	Predicted failure count in the range $[1,T]$
$D\,(T_L)$:	Predicted failure count in the range $[1,\infty]$
$T_F(t)$	Time to next failure predicted at time t

SOFTWARE RELIABILITY TREND ANALYSIS

Another way, a priori, to gauge the nature of predictions based on analyzing historical failure data are the *Arithmetic Mean Test* and the *Laplace Test* described below.

Arithmetic Mean Test [SWA]

This test consists of computing the arithmetic mean $\tau\,(i)$ of the observed interfailure times. An increasing sequence indicates reliability growth and a decreasing sequence indicates reliability decay.

$$\tau(i) = \frac{1}{i}\sum_{j=1}^{i} T_j \tag{8}$$

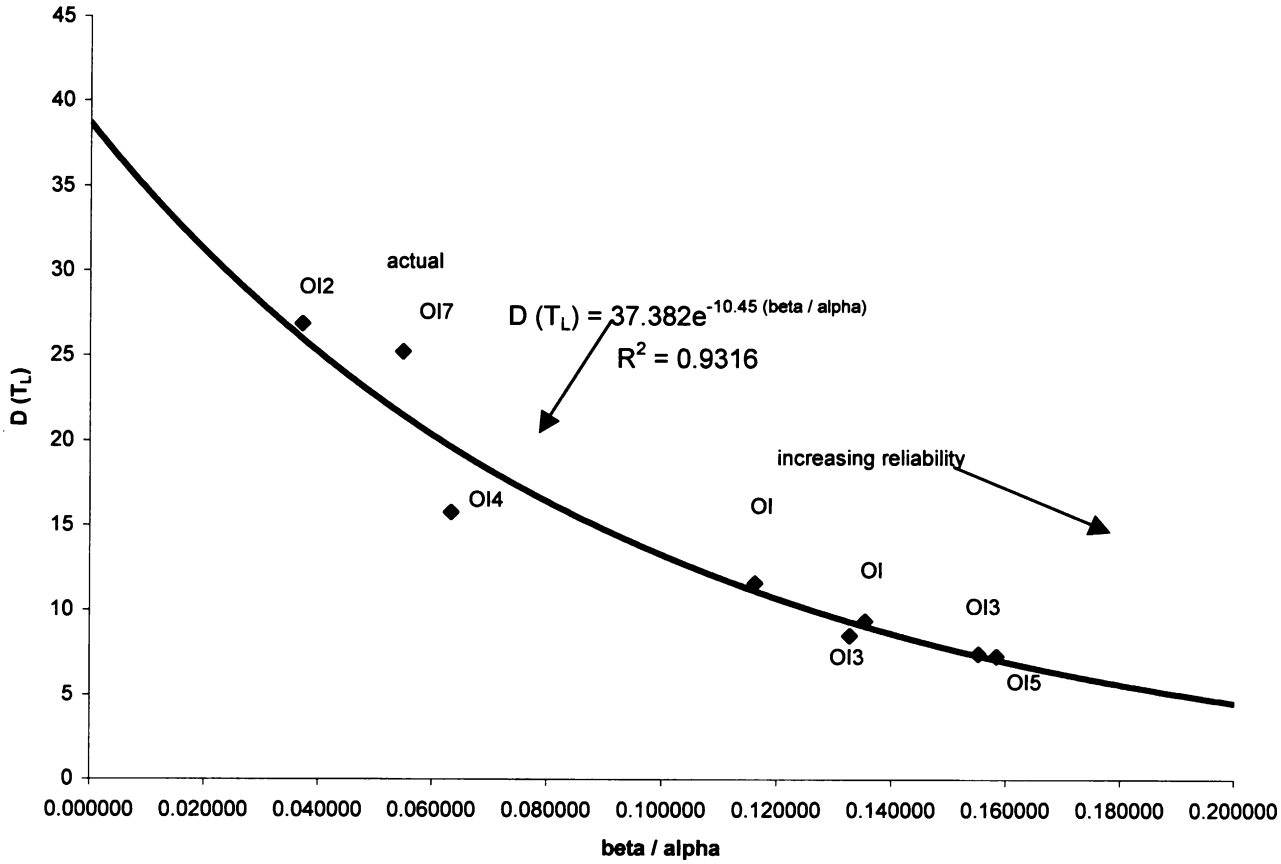

Figure 3. NASA Space Shuttle total failures $D(T_L)$ vs. parameter ratio (β/α) for OIs.

Laplace Test (7)

The Laplace test is superior from an optimality point of view and is recommended for use when the nonhomogeneous Poisson process assumption is made (e.g., Schneidewind model).

$$1(t) = \frac{\frac{1}{N(t)} \sum_{n=1}^{N(t)} \sum_{j=1}^{n} t_j - \frac{t}{2}}{t\sqrt{\frac{1}{12N(t)}}} \tag{9}$$

The Laplace factor can be interpreted as follows:

- Negative values indicate a decreasing failure intensity and, thus, reliability growth.
- Positive values indicate an increasing failure intensity and, thus, a decrease in the reliability.
- Values between −2 and +2 indicate stable reliability.

To test whether there is reliability growth or reliability decay, as produced by equations (10) and (11), we use the *empirical* failure rate given by equation (12) to investigate the trend:

$$\text{Cumulative failure count/length of count interval} = f(t) = N(t)/t \tag{10}$$

Figure 6 shows plots of the three equations for OI7 plotted against the *actual* failure times T. We see that the criteria for reliability growth of the arithmetic mean test and the Laplace test are satisfied, and there is confirmatory evidence of this situation because the failure rate trend is decreasing, which is suggestive of reliability growth. This analysis could be performed for all OIs to judge in advance of detailed prediction and testing the priority of these activities (i.e., give high priority to OIs that do not meet the criteria of reliability growth). However, it is important to note that we performed the same analysis on OI6 that passed the arithmetic means test and the Laplace test, but this OI did not demonstrate reliability growth, as given by equation (12). Therefore, we conclude that using the parameter ratio β/α is superior for ranking the relative reliability a priori for a set of objects (e.g., OIs).

SUMMARY

A methodology has been presented for judging the relative reliability of software in advance of performing detailed predictions and testing. The reason for this methodology is to conserve valuable human and machine resources dedicated to the prediction and testing activities. Our desire is to give priority to allocating resources to the software objects that need it the most—the lowest reliability software. We found that model parameter ratio β/α can be used to do an accurate job of ranking Space Shuttle operational increments. Various reliability metrics were shown to be highly related to β/α (i.e., high values of β/α were correlated with high reliability, and low values were correlated with low

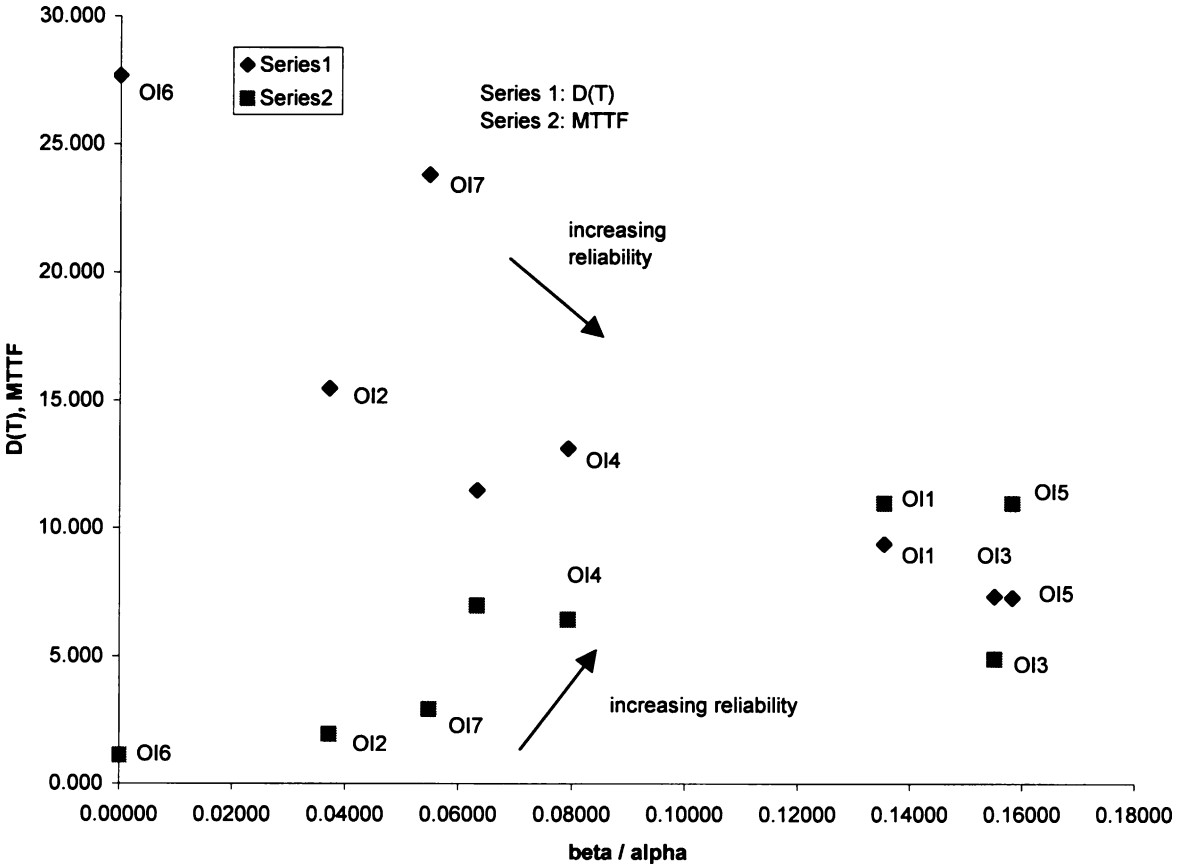

Figure 4. NASA Space Shuttle predicted total failures at time [T, $D(T)$], and MTTF vs. parameter ratio (β/α) for OIs.

reliability).

In addition, we used trend analysis, namely the arithmetic means test and the Laplace test, to investigate the trends in the historical failure data. The purpose was to see whether the *historical* trends would be indicative of reliability growth or reliability decay in the *future* reliability of the software. These methods did not provide consistent predictive results. Thus, we conclude that the parameter ratio method is superior for a priori software reliability evaluation—at least for the Space Shuttle. We believe that other reliability models, applied to other applications, would produce similar results.

BIBLIOGRAPHY

1. Broklehurst, S.; Littlewood, B. In *Handbook of Software Reliability Engineering*, Lyu, M. R., Ed., IEEE Computer Society Press: New York, 1996, Ch. 4.
2. Schneidewind, N. F. Reliability Modeling for Safety Critical Software. *IEEE Trans. Reliability*; 1997, **46**,pp 88–98.
3. Keller, T.; Schneidewind, N. F. A Successful Application of Software Reliability Engineering for the NASA Space Shuttle. *Software Reliability Engineering Case Studies, International Symposium on Software Reliability Engineering, Albuquerque*, November 4, 1997, pp 71–82.
4. IEEE/AIAA P1633/Draft 5, Draft Standard for Software Reliability Prediction, Prepared by the Software Reliability Engineering Working Group of the Definitions and Standards Committee of the Reliability Society, November 2006.
5. Far, W. H.; Smith, O. D.Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) Users Guide. NAVSWC TR-84-373, Revision 2, Naval Surface Warfare Center, Dahlgren, VA.
6. Nikora, A.CASRE, Open Channel Foundation. http://www.openchannelfoundation.org/projects/CASRE_3.0.
7. Gokhaleand, S. S.; Trivedi, K. S. Log-Logistic Software Reliability Growth Model. *Proc. of the Third IEEE International High-Assurance Systems Engineering Symposium*, Washington, DC, 1998, pp 34–41.

NORMAN F. SCHNEIDEWIND
IEEE Congressional Fellow
2005 US Senate

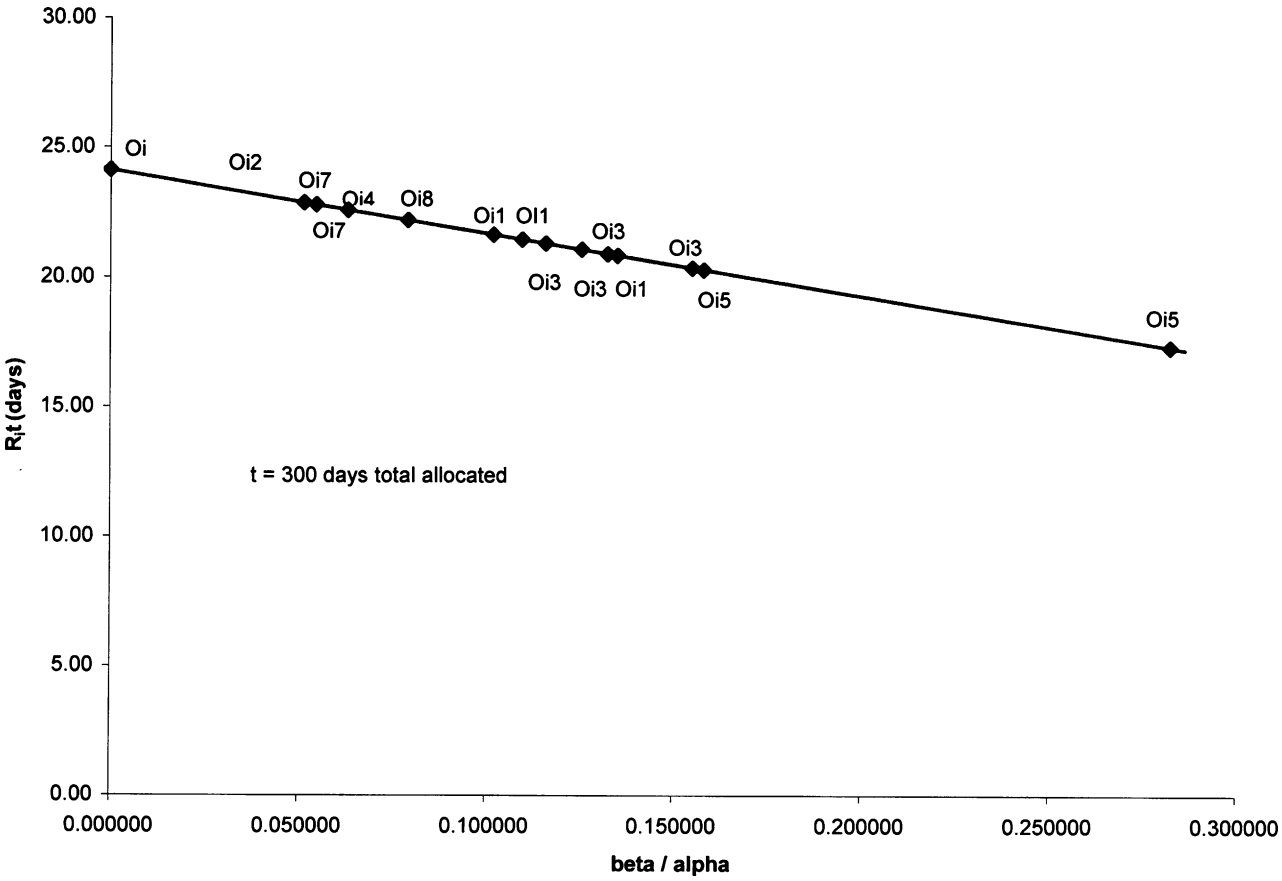

Figure 5. NASA Space Shuttle test time allocation R_i t vs. parameter ratio (β/α) for OIs.

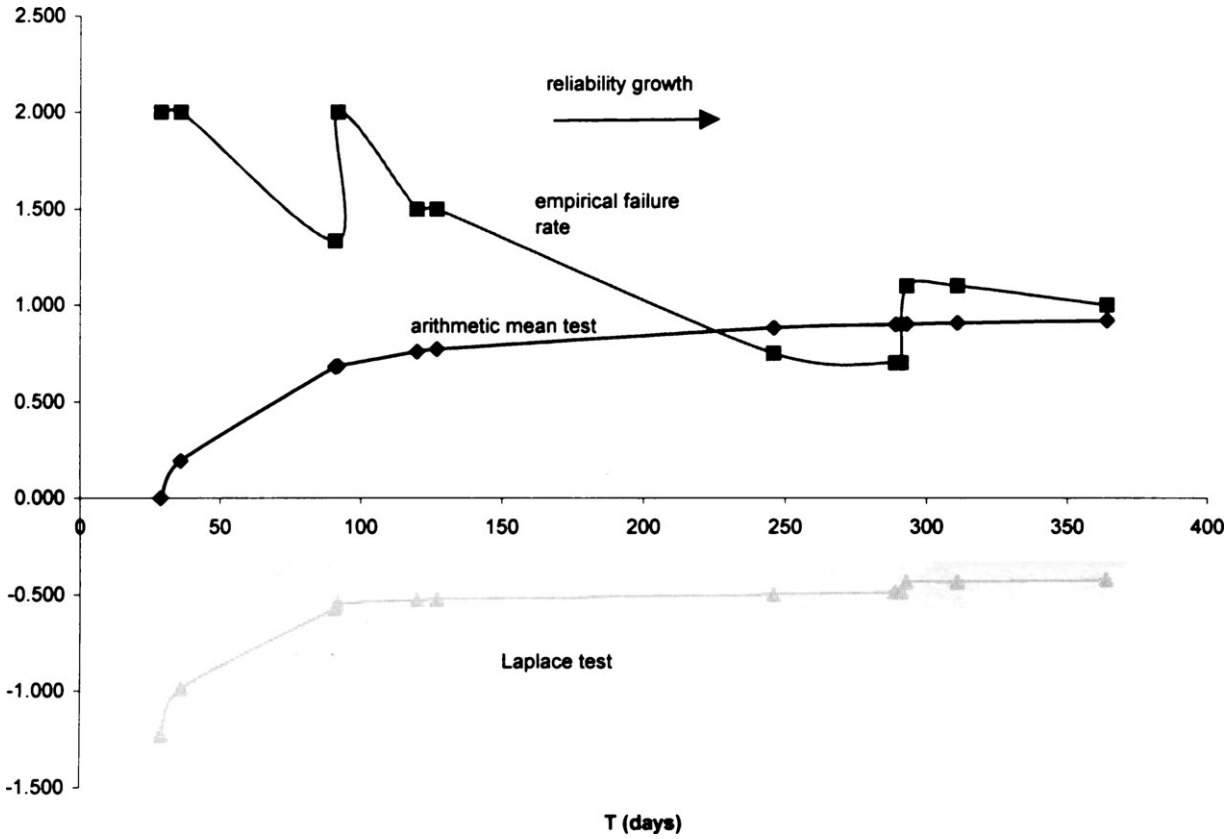

Figure 6. NASA Space Shuttle software failure trend analysis and failure rate vs. time of failure occurrence T for OI7.