Reliability Modeling for Safety-Critical Software

Norman F. Schneidewind, Fellow IEEE
Naval Postgraduate School, Monterey

Key Words — **Software reliability prediction, Safety-critical software, Risk analysis.**

Summary & Conclusions — **Software reliability predictions can increase trust in the reliability of safety critical software such as the NASA Space Shuttle Primary Avionics Software System (Shuttle flight software). This objective was achieved using a novel approach to integrate software-safety criteria, risk analysis, reliability prediction, and stopping rules for testing. This approach applies to other safety-critical software. We cover only the safety of the software in a safety-critical system. The hardware and human-operator components of such systems are not explicitly modeled nor are the hardware and operator-induced software failures. The concern is with reducing the risk of all failures attributed to software. Thus, *safety* refers to software-safety and not to system-safety. By improving the software reliability, where the reliability measurements & predictions are directly related to mission & crew safety, we contribute to system safety.**

Remaining failures (RF), maximum failures, total test time (TTT) required to attain a given fraction of RF and time to next failure (TTNF) are shown to be useful reliability measures & predictions for:

- **providing assurance that the software has achieved safety goals;**
- **rationalizing how long to test a piece of software;**
- **analyzing the risk of not achieving RF & TTNF goals. Having predictions of the extent that the software is not fault free (RF) and whether it is likely to survive a mission (TTNF) provide criteria for assessing the risk of deploying the software. Furthermore, 'fraction of RF' can be used as both an operational-quality goal in predicting TTT requirements and, conversely, as an indicator of operational-quality as a function of TTT expended.**

Software reliability models provide one of several tools that software managers of the Shuttle flight software are using to assure that the software meets required safety goals. Other tools are inspections, software reviews, testing, change control boards, and perhaps most important — experience & judgement.

1. INTRODUCTION[1]

Two categories of software reliability measurements (observed failure data used for model parameter estimation) and predictions (forecasts of future reliability using the parameterized model) are used together to assist in assuring the safety of the software in safety-critical systems like the Shuttle flight software. The two categories are:

1. measurements & predictions that are associated with residual software faults and failures;
2. measurements & predictions that are associated with the ability of the software to survive a mission without experiencing a serious failure.

In category #1 are: RF, maximum failures, fraction of RF, and TTT required to attain a given number or fraction of RF. In category #2 are: TTNF, and TTT required to attain a given TTNF. In addition, the risk associated with not attaining the required RF and TTNF is defined. A quantity from the 'fraction of RF' (operational quality) is derived.

The benefits of predicting these quantities are that they provide:

- assurance that the software has achieved safety goals,
- a means of rationalizing how long to test a piece of software (stopping rule).

Having predictions of the extent that the software is not fault free (RF) and its ability to survive a mission (TTNF) are meaningful for assessing the risk of deploying safety-critical software. In addition, with this type of information, a software manager can determine whether more testing is warranted or whether the software is sufficiently tested to allow its release or unrestricted use. These predictions, in combination with other methods of assurance, such as inspections, defect prevention, project control boards, process assessment, and fault tracking, provide a quantitative basis for achieving safety & reliability goals [3].

Risk, in Webster's New Universal Unabridged Dictionary, is defined as:

- "The chance of injury, damage, or loss" [19].

Some authors have extended the dictionary definition [19] to:

- "Risk Exposure = [Probability of an Unsatisfactory Outcome] × [Loss if the Outcome is Unsatisfactory]" [2].

Such a definition is frequently applied to the risks in managing software projects such as budget & schedule slippage. In contrast, this paper's application of the dictionary definition [19] pertains to the risk of executing the software of a safety-critical system where there is the chance of —

- injury (*eg*, astronaut injury or fatality),
- damage (*eg*, destruction of the Shuttle), or
- loss (*eg*, loss of the mission),

if a serious software failure occurs during a mission. Risk criterion metrics are developed to quantify the degree of risk associated with such an occurrence.

Lockheed-Martin, the primary contractor on the Shuttle flight software project, is experimenting with a promising algorithm which involves the use of SSRM to compute a parameter: fraction of RF as a function of the archived failure history during test & operation [10]. The prediction

[1]Acronyms, nomenclature, and notation are given at the end of the Introduction.

methodology in this paper uses this parameter and other reliability quantities to provide bounds on TTT, RF, operational quality, and TTNF, that are necessary to meet Shuttle safety requirements. This paper shows that there is a pronounced asymptotic characteristic to the TTT and to operational quality curves that indicate the possibility of big gains in reliability as testing continues; eventually the gains become marginal as testing continues. This prediction methodology is feasible for the Shuttle and other safety-critical systems.

This paper covers only the safety of the software in a safety-critical system. The hardware & human-operator components of such systems are not explicitly modeled nor are the hardware- & operator-induced software failures. However, in practice, these hardware-software interface and human operator-software interface failures can be very difficult to identify as such; these failures might be recorded as software failures. The concern here is with reducing the risk of all failures attributed to software. Thus, *safety* refers to software-safety and not to system-safety.

RF has been discussed in general as a type of software reliability prediction [13]. Various stopping rules for testing have been proposed, based on costs of testing and releasing software [4, 5, 8, 17], failure intensity [12], and testability [18]. Our approach is novel because it integrates software-safety criteria, risk analysis, reliability prediction, and a stopping rule for testing. For a system like the Shuttle where human lives are at risk, economic or time-to-market criteria can not be used to determine when to deploy the software. Although failure intensity has proven useful for allocating test effort and for determining when to stop testing in commercial systems [12], this criterion is not directly related to software safety. In a safety-critical system, the 'prediction of RF' and 'identification of the faults which cause them' is more relevant to ensuring safety than the trend of failure intensity over time. The latent faults must be found and then removed through additional testing, inspection, or other means, if mission safety is not to be jeopardized. Furthermore, as shown, RF along with TTNF can be used as risk criteria. It is not clear how failure intensity could be a meaningful safety criterion.

Because testability attempts to quantify the failure probability if the code is faulty [18], this criterion has a relationship with reliability if we know that the code is faulty. However in the Shuttle and other safety-critical software, the purpose is to predict whether the code is faulty. For safety-critical software, reliability measurements & predictions must be used to assess whether safety & mission goals are likely to be achieved.

Two criteria for software safety are defined, and then applied to risk analysis of safety-critical software, using the Shuttle flight software as an example. Next, definitions and brief derivations are provided for a variety of prediction equations that are used in reliability prediction and risk analysis; included is the relationship between TTNF and 'reduction in RF'. This is followed by an explanation of the principal of 'optimal selection of failure data' that involves selecting only the most relevant set of failure data for reliability prediction, with the result of producing more accurate predictions than would be the case if the entire set of data were used. Then it is shown how the prediction equations can be used to integrate testing with reliability & quality. An example shows how the risk analysis and reliability predictions can be used to decide whether the software is safe to deploy. Validation results are shown for a variety of predictions.

Acronyms[2]

OI	Shuttle operational increment
OI-X	OI for $X \in$ [A, B, C, D]
MSE	mean square error
RCM	risk criterion metric
RF	remaining failure(s)
SSRM	Schneidewind software reliability model [1, 14, 15, 16]
TTNF	time to next failure
TTT	total test time.

Assumptions [1]

1. Faults that cause failures are removed.
2. As more failures occur, and as more faults are corrected, RF are reduced.
3. For those OI that were executed for extremely long times (years) with no additional failure reports:
   a. the RF are zero;
   b. 'maximum failures' = 'total observed failures'.
4. The number of failures detected in one interval is *s*-independent of the failure count in another.
5. Only "new "failures are counted, *ie*, 'failures that are repeated as a consequence of not correcting a fault' are not counted. ◄

Definitions

- Failure: The inability of a system or system-component to perform a required function within specified limits [1].
- Fault: A defect in the code that can be the cause of one or more failures [1].
- Interval: An integer time unit t of constant length defined by

$$t-1 < t < t+1,\ t > 0;$$

failures are counted in intervals (*eg*, one failure occurred in interval 4) [1, 7].

- Number of Intervals: The number of contiguous integer time units t of constant length represented by a positive real number (*eg*, the predicted TTNF is 3.87 intervals).
- OI: A software system comprised of modules, and configured from a set of builds to meet Shuttle mission functional requirements.
- Time: Continuous CPU execution time over an interval range. ◄

[2]The singular & plural of an acronym are always spelled the same.

Severity Codes

1. Severe vehicle or crew performance implications.
2. Affects ability to complete mission (not a safety issue).
3. Work-around available, minimal effect on procedures.
4. Insignificant[3] (paperwork, *etc*).
5. Not visible to user. ◄

Nomenclature

- Predicted at time t: A prediction made in the interval t.
- Safety: Software safety; not system safety.

Notation

α	failure rate at the beginning of interval s
β	negative of 'derivative of failure rate' divided by 'failure rate' (relative failure rate)
γ	α/β
$F(i)$	predicted failure count in the range $[1, i]$; used in computing MSE_r
$F_{i,j}$	observed failure count during interval j since interval i; used in computing MSE_T
$F(t)$	predicted failure count in the range $[1, t]$
F_t	given number of failures to occur after interval t; used in predicting $T_F(t)$
$F(t_1, t_2)$	predicted failure count in the range $[t_1, t_2]$
$F(\infty)$	predicted failure count in the range $[1, \infty]$: maximum failures over the life of the software
i	current interval
j	next interval, $j > i$, where $F_{i,j} > 0$
J	maximum $j \leq t$, where $F_{i,j} > 0$
MSE_F	MSE criterion for selecting s for failure-count predictions
MSE_r	MSE criterion for selecting s for RF predictions
MSE_T	MSE criterion for selecting s for TTNF predictions
$p(t)$	fraction of RF predicted at time t
$Q(t)$	operational quality predicted at time t: $1 - p(t)$; degree to which software is free of remaining faults (failures)
t_t	TTT (observed or predicted)
t_m	mission duration (end-time − start-time); used in computing RCM $T_F(t_t)$
r_c	critical value of RF; used in computing RCM $r(t_t)$
$r(t)$	RF predicted at time t
$\Delta r(T_F,t)$	reduction in RF that would be achieved if the software were executed for T_F, predicted at time t
$r(t_t)$	RCM for RF at t_t
$T_F(t_t)$	RCM for TTNF at t_t
s	starting interval for using observed failure data in parameter estimation
s^*	optimal s, as determined by MSE criterion
t	cumulative time in the range $[1, t]$; last interval of observed failure data; current interval
$T_F(\theta)$	TTNF, predicted at time θ
$T_F(\Delta r,t)$	time to next N failures that would be achieved if RF were reduced by Δr, predicted at time t
$T_{i,j}$	time since interval i to observe $F_{i,j}$ during interval j; used in computing MSE_T
X_θ	observed failure count in the range $[1, \theta]$
$X_{s,\theta}$	observed failure count in the range $[s, \theta]$.

Other, standard notation is given in "Information for Readers & Authors" at the rear of each issue.

[3]This word is used in its ordinary meaning, not in its statistical meaning.

2. CRITERIA FOR SAFETY

If 'safety goal' is defined as the reduction of failures that would cause loss of life, loss of mission, or abort of mission, to an acceptable level of risk [11], then for software to be ready to deploy, after having been tested for t_t, the following 2 criteria must be satisfied:

$$1.\ r(t_t) < r_c, \tag{1}$$

$$2.\ T_F(t_t) > t_m. \tag{2}$$

For systems that are tested & operated continuously like the Shuttle, the t_t, $T_F(t_t)$, t_m are measured in execution time. As with any methodology for assuring software safety, we cannot guarantee safety. Rather, with these criteria, we seek to reduce the risk of 'deploying the software' to an acceptable level.

2.1 RF, Criterion #1

Using assumption #1 (as for the Shuttle), criterion #1 specifies that the residual failures & faults must be reduced to a level where the risk of operating the software is acceptable. As a practical matter, I suggest $r_c = 1$. That is, the goal is to reduce the s-expected RF to less than one before deploying the software. The reason for this choice is that one or more RF is an unacceptable risk for safety-critical systems. This is the threshold used by the Shuttle software managers. One way to specify r_c is by failure severity level (*eg*, severity level 1 for life threatening failures). Another way, which imposes a more demanding safety requirement, is to specify that r_c represents all severity levels. For example, $r(t_t) < 1$ would mean that $r(t_t)$ must be less than one failure, independent of severity level.

If we predict $r(t_t) \geq r_c$, we would continue to test for a TTT $t_t' > t_t$ that is predicted to achieve $r(t_t') < r_c$, using assumption #2 that we will experience more failures and correct more faults so that the RF will be reduced by $r(t_t) - r(t_t')$. If the developer does not have the resources to satisfy criterion #1 or is unable to satisfy criterion #1 through additional testing, the risk of deploying the software prematurely should be assessed (see section 2.2). The Dijkstra dictum states that we can not demonstrate the absence of faults [6]; however we can reduce the 'risk of failures occurring' to an acceptable level, as represented by r_c, as shown in figure 1. Case A of figure 1 predicts $r(t_t) < r_c$ and the mission begins at t_t. Case B of figure 1 predicts $r(t_t) \geq r_c$ and postpones the mission until we test for t_t' and predict $r(t_t') < r_c$.

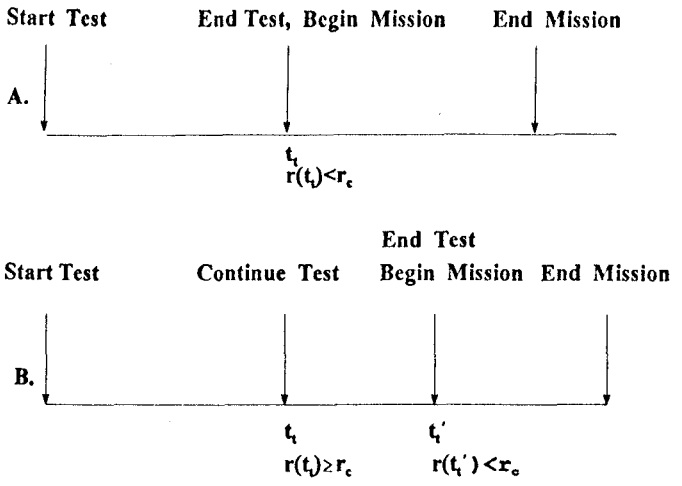

Figure 1. RF (Criterion #1) Scenario

In both cases criterion #2 must also be satisfied for the mission to begin.

2.2 TTNF, Criterion #2

Criterion #2 specifies that the software must survive for a time greater than the mission duration. If we predict $T_F(t_t) \leq t_m$, then we continue to test for $t_t'' > t_t$ that is predicted to achieve $T_F(t_t'') > t_m$, using assumption #2 that we will experience more failures and correct more faults so that the TTNF will be increased by $T_F(t_t'') - T_F(t_t)$. Again, if it is infeasible for the developer to satisfy criterion #2 because of lack of resources or failure to achieve test objectives, the risk of deploying the software prematurely should be assessed (see section 3). Figure 2 shows this scenario. Case A of figure 2 predicts $T_F(t_t) > t_m$ and the mission begins at t_t. Case B of figure 2 predicts $T_F(t_t) \leq t_m$ and postpones the mission until we test for t_t'' and predict $T_F(t_t'') > t_m$. In both cases, criterion #1 must also be satisfied for the mission to begin. If neither criterion #1 nor #2 is satisfied, test for $\max(t_t', t_t'')$.

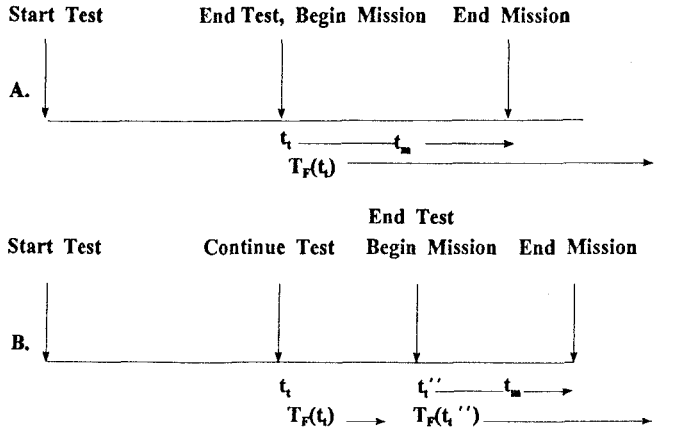

Figure 2. TTNF (Criterion #2) Scenario

3. RISK ASSESSMENT

The t_t can be considered a measure of the degree to which software reliability goals have been achieved, particularly for systems like the Shuttle where the software is subjected to continuous & rigorous testing for several years in multiple facilities, using a variety of operational & training scenarios (*eg*, by Lockheed-Martin in Houston, by NASA in Houston for astronaut training, and by NASA at Cape Kennedy). If t_t is viewed as an input to a risk-reduction process, and $r(t_t)$ & $T_F(t_t)$ as the outputs, then the process is shown in figure 3, where r_c & t_m are shown as 'risk criteria levels' of safety that control the process. While TTT is not the only consideration in developing test strategies and while there are other important factors (*eg*, consequences for reliability & cost, in selecting test cases [20]), nevertheless, for the foregoing reasons, TTT has been found to be strongly positively *s*-correlated with reliability growth for the Shuttle [15].

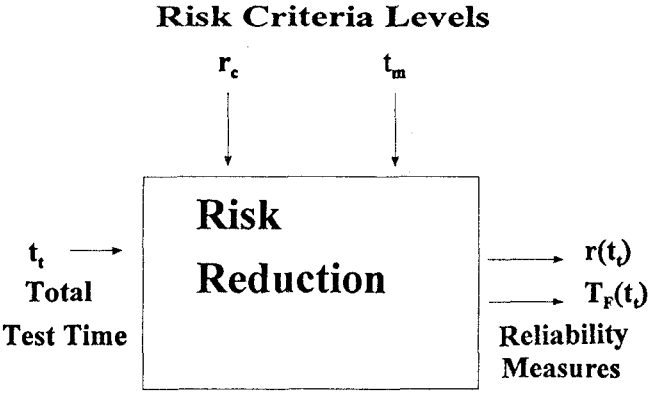

Figure 3. Risk-Reduction Process

3.1 RF, Criterion #1

$$r(t_t) = (r(t_t) - r_c)/r_c = (r(t_t)/r_c) - 1. \quad (3)$$

Figure 4 plots (3) as a function of t_t for $r_c = 1$, where RCM $r(t_t)$ —

- $> 0 \Rightarrow r(t_t) > r_c \Rightarrow$ UNSAFE (above the X-axis: $r(t_t) >$ safe value);
- $= 0 \Rightarrow r(t_t) = r_c \Rightarrow$ NEUTRAL (on the X-axis: $r(t_t) =$ safe value);
- $< 0 \Rightarrow r(t_t) < r_c \Rightarrow$ SAFE (below the X-axis: $r(t_t) <$ safe value).

Figure 4 is for OI-D. In this example at $t_t \approx 57$, the risk transitions from the UNSAFE region to the SAFE region.

3.2 TTNF, Criterion #2

$$T_F(t_t) = (t_m - T_F(t_t))/t_m = 1 - (T_F(t_t))/t_m. \quad (4)$$

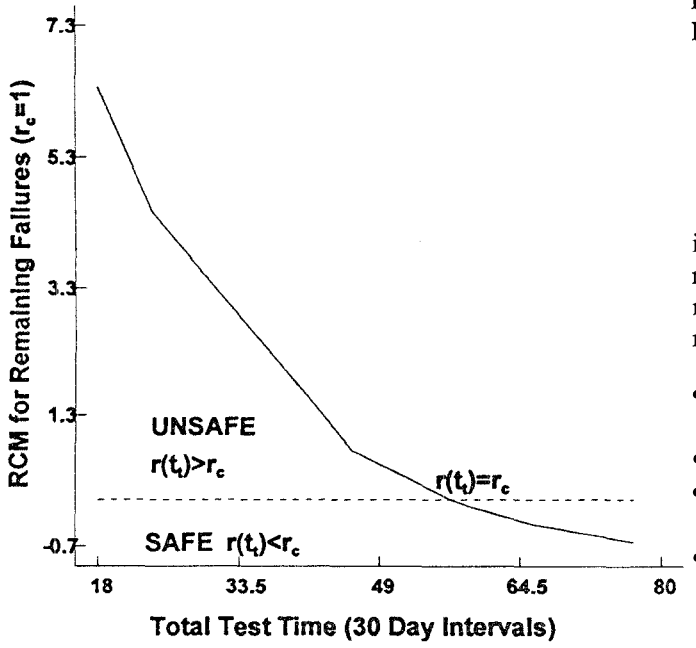

Figure 4. RCM for RF, OI-D

Figure 5 plots (4) as a function of t_t for t_m=8 days (a typical mission duration for this OI), where RCM $T_F(t_t)$ —

- $> 0 \Rightarrow T_F(t_t) < t_m \Rightarrow$ UNSAFE (above the X-axis: $T_F(t_t)$ < safe value);
- $= 0 \Rightarrow T_F(t_t) = t_m \Rightarrow$ NEUTRAL (on the X-axis: $T_F(t_t)$ = safe value);
- $< 0 \Rightarrow T_F(t_t) > t_m \Rightarrow$ SAFE (below the X-axis: $T_F(t_t)$ > safe value).

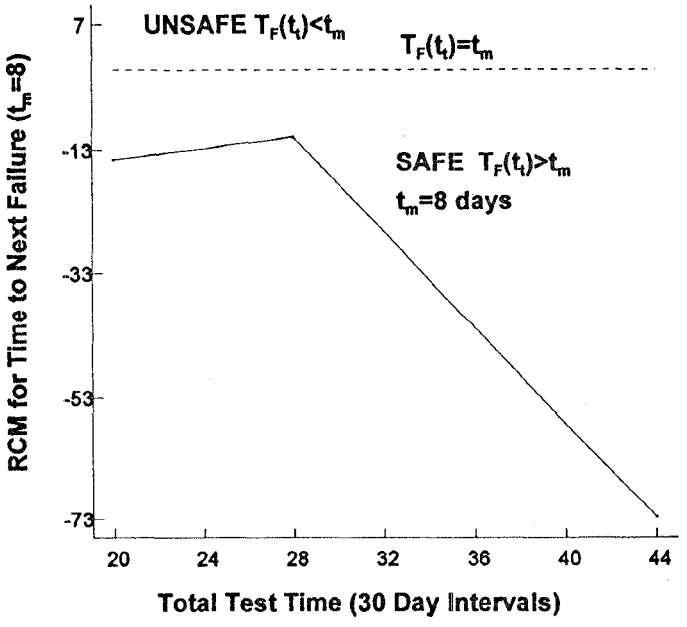

Figure 5. RCM for TTNF, OI-C

Figure 5 is for OI-C. In this example at all values of t_t, the RCM is in the *SAFE* region.

4. APPROACH TO PREDICTION

To support the safety goal and to assess the risk of deploying the software, various reliability & quality predictions are made. These predictions are used for tradeoff analysis between reliability and TTT. Thus, this approach is to use a software reliability model to predict:

- maximum failures, RF, and operational quality (as defined in section 5);
- TTNF (beyond the last observed failure);
- TTT necessary to achieve required levels of RF (fault) level, operational quality, and TTNF;
- tradeoffs between increases in levels of reliability & quality with increases in testing.

5. PREDICTION EQUATIONS

The prediction equations in this section:

- are all for mean values;
- are based on the SSRM, one of the 4 models recommended in the AIAA Recommended Practice for Software Reliability [1];
- use assumptions #1 - #5 in the Introduction;
- are derived in section 6;
- are applied to analyze the reliability of the Shuttle flight software. ◄

Because the flight software is run continuously, around the clock, in simulation, test, or flight, *time* refers to continuous execution time, and TTT refers to execution time for testing. Failure-count intervals are 30 days of continuous execution time. This interval is long because the Shuttle software is tested for several years; a 30-day interval length is convenient for recording failures for software that is tested this long.

Figure 6 shows these 'failure-count interval relationships' and t_t. Failures are counted against OI. Data from four Shuttle OI, designated OI-A, OI-B, OI-C, OI-D, are used in this analysis.

5.1 Cumulative Failures

Using maximum likelihood estimates for α & β, with s as the starting interval for using observed failure data,

$$F_{s,t} = \gamma \cdot [1 - \exp(-\beta \cdot (t - s + 1))]. \quad (5)$$

If X_{s-1} is added,

$$F(t) = \gamma \cdot [1 - \exp(-\beta (t - s + 1))] + X_{s-1}. \quad (6)$$

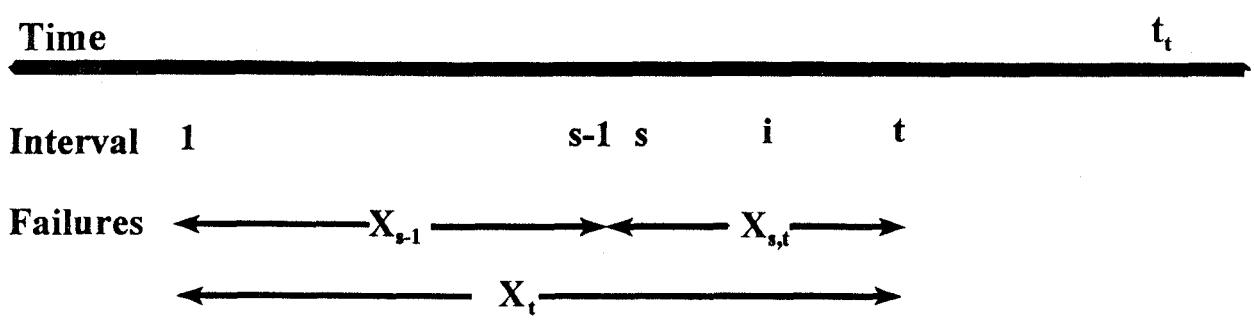

Figure 6. Failure Count Interval Relationships

5.2 Failures in an Interval Range

Let $t \equiv t_2$; subtract $X_{t1} = X_{s-1} + X_{s,t1}$. Then from (6),

$$F(t_1, t_2) = \gamma \cdot [1 - \exp(-\beta \cdot (t_2 - s + 1))] - X_{s,t_1}. \quad (7)$$

5.3 Maximum Failures

Let $t \to \infty$ in (6).

$$F(\infty) = \gamma + X_{s-1}. \quad (8)$$

5.4 RF

To obtain $r(t)$, subtract $X_t = X_{s-1} + X_{s,t}$ from (8):

$$r(t) = \gamma - X_{s,t} = F(\infty) - X_t. \quad (9)$$

The $r(t)$ can be expressed as a function of t_t by substituting (5) into (9), and setting $t \equiv t_t$:

$$r(t_t) = \gamma \cdot \exp[-\beta \cdot (t_t - s + 1))]. \quad (10)$$

5.5 Fraction of RF

Divide (9) by (8):

$$p(t) = r(t)/F(\infty). \quad (11)$$

5.6 Operational Quality

Operational quality of software is the degree to which software is free of remaining faults (failures), using assumption #1.

$$Q(t) = 1 - p(t). \quad (12)$$

5.7 TTT to Achieve Specified RF

The $r(t_t)$ is obtained from (10) by solving for t_t:

$$t_t = \beta^{-1} \cdot \log[\gamma / r(t_t)] + (s - 1). \quad (13)$$

5.8 TTNF

Substitute $t_2 = t + T_F(t)$ in (7); set $t_1 \equiv t$; define $F_t = F(t, t + T_F)$; solve for $T_F(t)$:

$$T_F(t) = \beta^{-1} \cdot \log(\gamma / \Psi_{13}) - (t - s + 1), \quad (14)$$

for $\Psi_{13} > 0$,

$$\Psi_{13} \equiv \gamma - (X_{s,t} + F_t).$$

$t \equiv$ current interval.

5.9 Discussion

Consider (5) - (11), (14) to be predictors of reliability that are related to safety; (13) represents the predicted TTT required to achieve stated safety goals. If a quality requirement is stated in terms of 'fraction of RF', the definition of Q as Operational Quality (12), is consistent with the IEEE definition of quality: the degree to which a system, component, or process meets specified requirements [9]. For example, let a reliability specification require that software is to have no more than 5% RF ($p = 0.05$, $Q = 0.95$) after testing for t_t intervals; then a predicted $Q = 0.90$ indicates the degree to which the software meets specified requirements.

5.10 Relating Time to Next N Failures and RF Predictions

The risk analysis and prediction equations for RF & TTNF are shown separately. It is useful to combine them in one equation so that we can predict the effect on one quantity for a given change in the other. In particular we want to predict, at time t, the $T_F(\Delta r, t)$, that would be achieved if RF were reduced by Δr. Use assumption #1 ($N = \Delta r$). When $N = 1$, we have the familiar TTNF. When $N > 1$, $T_F(\Delta r, t)$ is interpreted as cumulative execution time for the N failures to occur. Conversely, we want to predict, at time t, the $\Delta r(T_F, t)$ that would be achieved if the software were executed for a time T_F. This relationship is derived using (10) and setting:

$$\Delta r = r(t_1) - r(t_t),$$

$$t_t = t_1 + \Delta t,$$

$$t_1 \equiv t,$$

and solving for $\Delta t \equiv T_F(\Delta r, t)$:

$$T_F(\Delta r, t) = (-1/\beta) \cdot \log(1 - \Psi_{15}), \quad (15)$$

for $\Psi_{15} < 1$,

$$\Psi_{15} \equiv (\Delta r/\gamma)\cdot\exp(\beta\cdot(t-s+1)).$$

Eq (15) is analogous to (14); Δr in (15) is analogous to F_t in (14), if assumption #1 is true. Eq (14) & (15) produce the same result for the same parameter values. Eq (15) has simpler computation because it does not require $X_{s,t}$ which is used in (14). Also, (15) is convenient to use for trading-off 'time to next N failures' against 'reduction in RF' AND 'effort and TTT implicit in making the reductions'.

Invert (15) to solve for the 'reduction in RF' that would be achieved by executing the software for T_F.

$$\Delta r(T_F,t) = \gamma\cdot\exp(-\beta\cdot(t-s+1))\cdot[1-\exp(-\beta(T_F))]. \quad (16)$$

6. CRITERION FOR OPTIMALLY SELECTING FAILURE DATA

The first step in identifying s^* is to estimate α & β for each value of s $\in$ [1,t] where convergence can be obtained [1, 14, 16]. Then the MSE criterion is used to select s^*, the failure count interval that corresponds to the minimum MSE between predicted & actual failure counts: MSE_F, MSE_T, or MSE_r — depending on the type of prediction. The first two were reported in [14]. This paper develops MSE_r which is also the criterion for $F(\infty)$ & t_t because the two are functionally related to $r(t)$; see (9) & (13). MSE_T is shown because it is used in predictions that involve TTNF: $T_F(t)$, $T_F(\Delta r,t)$, $\Delta r(T_F,t)$. Once α, β, and s are estimated from observed counts of failures, the foregoing predictions can be made. MSE is used to evaluate which triple, (α, β, s), is best in the range [1,t] because research shows that: because the product & process change over the life of the software, old failure data ($s=1$) are not as representative of the current state of the product & process as the more recent failure data ($s>1$) [14]. The s^* used in the risk analysis and prediction examples are shown in tables 1-4.

SMERFS [7] is used for all predictions except t_t, $T_F(\Delta r,t)$, and $\Delta r(T_F,t)$, which are not implemented in SMERFS.

6.1 MSE Criterion for RF

Although we can never know whether additional failures might occur, nevertheless we can form the difference between two equations for $r(t)$: (9) which is a function of 'predicted maximum failures' and 'observed failures', and (10) which is a function of TTT; then apply the MSE criterion.

$$MSE_r = \left[\sum_{i=s}^{t}(F(i)-X_i)\right]/(t-s+1). \quad (17)$$

6.2 MSE Criterion for TTNF

From [14]:

$$MSE_T = \left[\sum_{i=s}^{J-1}[[\beta^{-1}\cdot\log(\gamma/\Psi_{18}) - (i-s+1)] - T_{i,j}]^2\right]/(J-s), \quad (18)$$

for $\Psi_{18} > 0$,

$$\Psi_{18} \equiv \gamma - (X_{s,i} + F_{i,j}),$$

$t \equiv$ the last interval of observed failure data.

7. RELATING TESTING TO RELIABILITY & QUALITY

7.1 Predicting TTT & RF

Use (8) to predict $F(\infty) = 11.76$ for Shuttle OI-A. Using given values of p and (11), and setting $t \equiv t_t$, predict $r(t_t)$ for each value of p. The values of $r(t_t)$ are the predictions of RF after the OI has been executed for t_t. Then use the values of $r(t_t)$ and (13) to predict corresponding values of t_t. Figure 7 shows the results: $r(t_t)$ & t_t are plotted against p *for OI-A. The* t_t rises very rapidly at small values of p & $r(t_t)$. The maximum value of p on the plot corresponds to $t_t=18$; smaller values correspond to future values of t_t ($t_t > 18$).

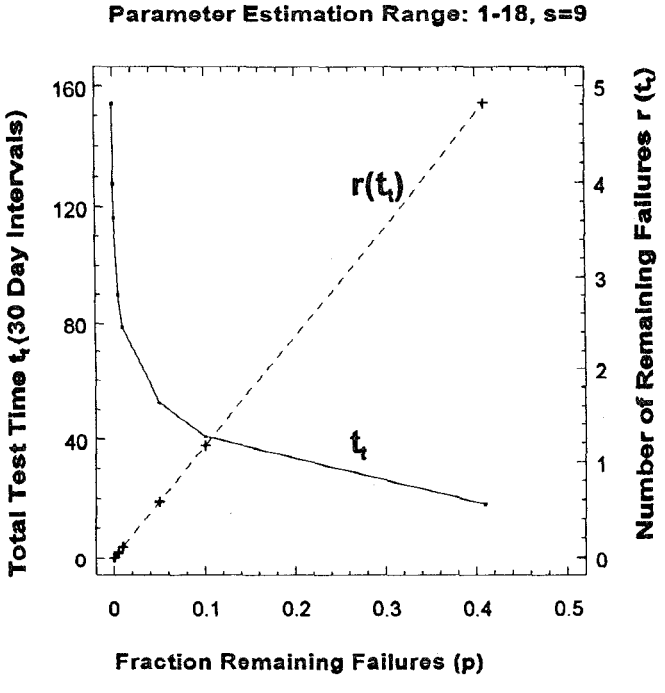

Figure 7. 'TTT & RF' *vs* 'Fraction RF', OI-A

7.2 Predicting Operational Quality

Eq (12) is a useful measure of the operational quality of software because it measures the degree to which faults have been removed from the software (using assumption #1), relative

to predicted maximum failures. This is operational quality (based on software execution) to distinguish it from static quality (*eg*, based on software complexity).

Using given values of p, and (11) & (12), and setting $t \equiv t_t$ — compute $r(t_t)$ & Q. The values of $r(t_t)$ are then used in (13) to compute t_t. Figure 8 plots the corresponding values of Q & t_t for OI-A. Observe the asymptotic nature of the testing relationship in the great amount of testing required to achieve high levels of quality.

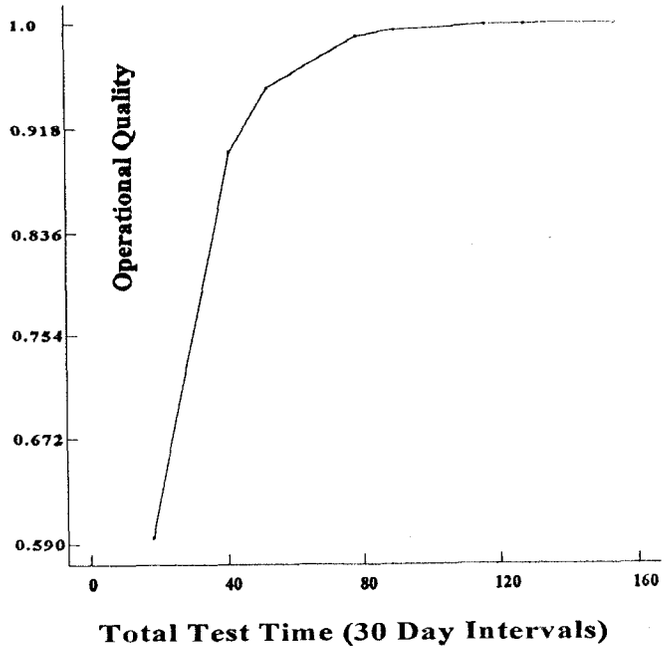

Figure 8. Operational Quality *vs* TTT, OI-A

7.3 Predicting TTNF

Figure 9 shows the actual TTNF for OI-A on the solid curve that has occurred in the execution-time range $t=[1,18]$, where:

- 1 failure occurred at $t = 4, 14, 18$,
- 2 failures occurred at $t = 8, 10$.

All failures were Severity Level #3. The way to read the graph is:

- Take a given failure, *eg*, Failure 1; it occurs at $t=4$. Therefore, at $t=1$ the TTNF is $3 = 4 - 1$. At $t=2$ the TTNF is $2 = 4 - 2$. At $t=4$, Failure 1 occurs, so the TTNF is $4 = 8 - 4$, which now refers to 'Failure 2'; *etc*.
- Using (14), predict the TTNF $T_F(18)$ to be 4 (3.87 rounded) on the dashed curve.

Based on the foregoing, this prediction indicates we should continue testing if,

$$T_F(18) = 3.87 \leq t_m.$$

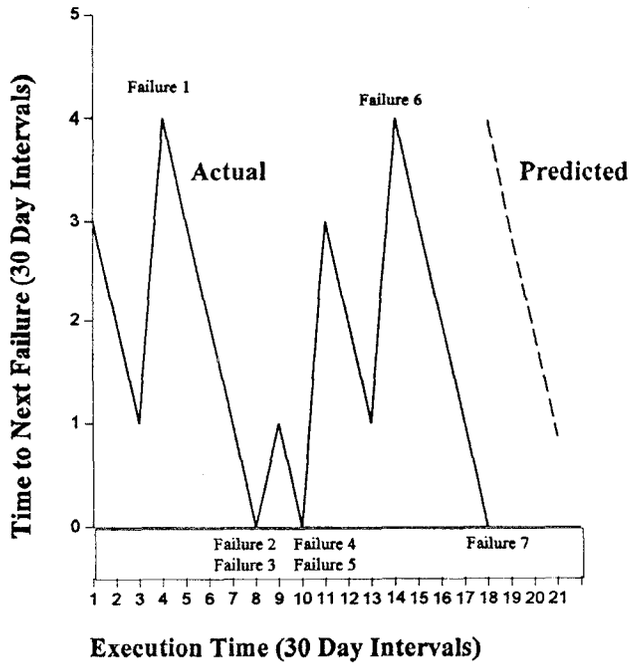

Figure 9. TTNF *vs* 'Execution Time', OI-A

7.4 Predicting Tradeoffs of 'Time to Next N Failures' with 'Reduced RF'

By using (15), predict $T_F(\Delta r,t)$ as a function of Δr. Figure 10 shows this for OI-A, where, *eg*, with $\Delta r=1$, we predict $T_F(1,18) = 3.87$ (a 'reduction in RF' of 1 corresponds to achieving a TTNF of 3.87 intervals from the current interval 18). Conversely, by using (16), we predict $\Delta r(T_F,t)$ as a function of T_F. Figure 11 shows this for OI-A, where, *eg*, with $T_F=3.87$, we predict $\Delta r(3.87,18) = 1$ (executing OI-A for a TTNF of 3.87 intervals from the current interval 18 corresponds to achieving a 'reduction in RF' of 1). Section 8 elaborates further on these graphs.

8. MAKING SAFETY-DECISIONS

To decide about t_t, apply the safety criteria and risk assessment approach. Table 1 illustrates the process. For $t_t=18$ (when the last failure occurred on OI-A), $r_c=1$, and $t_m=8$ days (0.267 intervals), then we show RF, RCM for RF, TTNF, RCM for TTNF, and operational quality. These results indicate that safety criterion #2 is satisfied but not criterion #1 (UNSAFE with respect to RF); also operational quality is low.

Figure 10 and table 1 show that if we reduce RF $r(18)$ by 1 from 4.76 to 3.76 (non-integer values are possible because the predictions are mean values), the predicted TTNF that would be achieved is $T_F(18)=3.87$ intervals. These predictions satisfy criterion #2:

$$T_F(18) = 3.87 > t_m = 0.267;$$

but not criterion #1

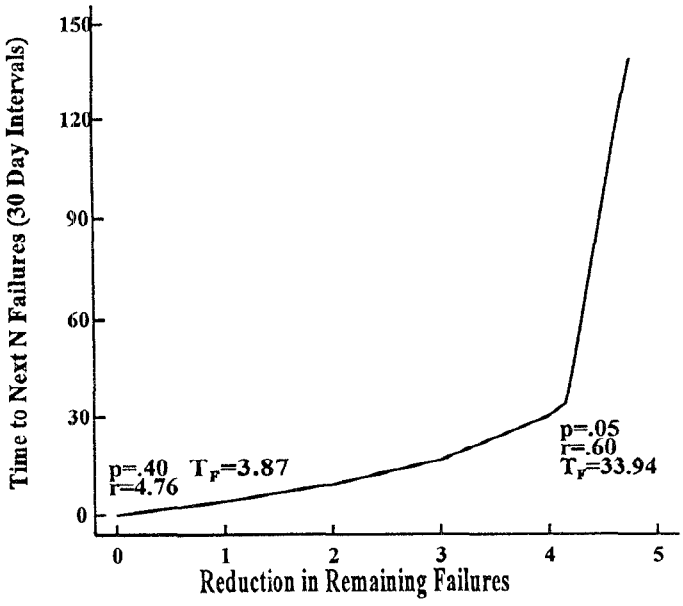

Figure 10. 'Time to Next *N* Failures' *vs* 'Reduction in RF', OI-A

Table 1. Safety Criteria Assessment, OI-A
[$r_c = 1$, $t_m = 8$ days
30-day TTT & TTNF Intervals]

t_t	α	β	s^*	$r(t_t)$	RCM	s^*	$T_F(t_t)$	RCM	Q
18	.534	.061	9	4.76	3.76	9	3.87	-13.49	.60
52	.534	.061	9	.60	-.40	9	*	*	.95

*Cannot predict because 'predicted RF' < 1

$r(18) = 4.76 > r_c = 1.$

Figure 10 and table 1 show that 'fraction of RF':

$p = 1 - Q = 0.40$ at $r(18) = 4.76.$

Now, if testing continues for $t_t = 52$ intervals, as shown in figure 10 and table 1, and RF reduces from 4.76 to 0.60, the predicted 'time to next 4.16 failures' that would be achieved is 33.94 (34 rounded) intervals. This corresponds to:

$t_t = 18 + 34 = 52$ intervals.

That is, if we test for an additional 34 intervals, starting at interval 18, we anticipate getting 4.16 failures. These predictions now satisfy criterion #1 because:

$r(52) = 0.60 < r_c = 1.$

In figure 10 and table 1, the 'fraction of RF':

$p = 1 - Q = 0.05$ at $r(52) = 0.60.$

Using the converse of the relationship in figure 10, provides another perspective, as shown in figure 11, where if we continue to test for an additional $T_F = 34$ intervals, starting at interval 18, the predicted 'reduction in RF' that would be achieved is 4.16, or r(52) = 0.60.

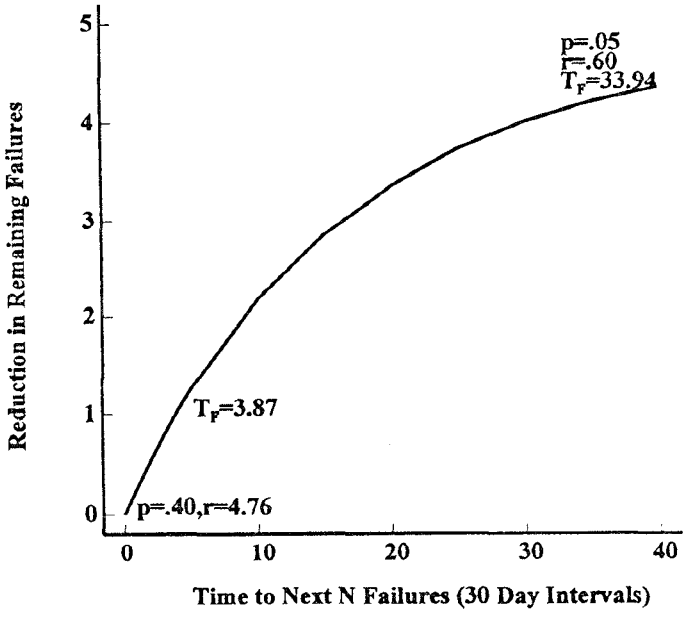

Figure 11. 'Reduction in RF' *vs* 'Time to Next *N* Failures', OI-A

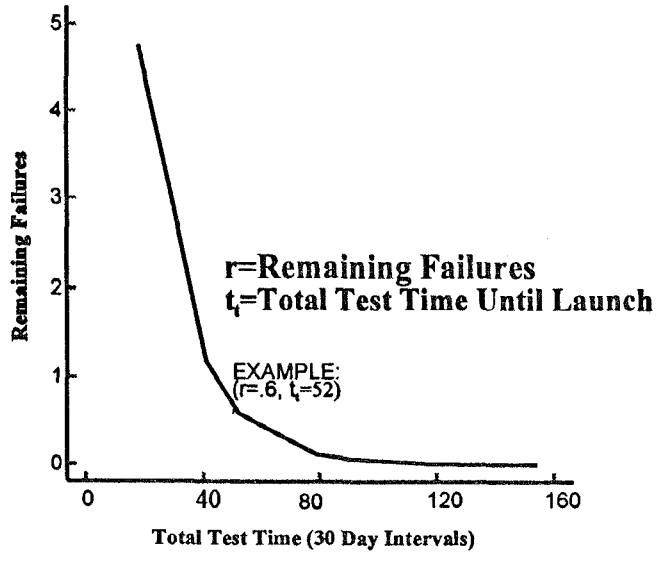

Figure 12. Launch Decision: RF *vs* TTT, OI-A

Figure 12 shows the Launch Decision, relevant to the Shuttle, (or, generically, the Deployment Decision), where RF are plotted against TTT for OI-A. With these results, the software manager can decide whether to deploy the software, depending on factors such as predicted RF as shown in figure 12, along with other factors such as the 'trend in reported faults over time', and 'inspection results'. If testing were to continue until $t_t = 52$,

the predictions in figure 12 and table 1 would be obtained. These results show that criterion #1 is now satisfied (SAFE) and 'operational quality' is high. Figure 12 shows that at this value of t_t, further increases in t_t would not result in an important increase in reliability & safety. At $t_t=52$ it is not feasible to predict $T_F(52)$ because the 'predicted RF' < 1.

9. SUMMARY OF PREDICTIONS AND VALIDATION[4]

9.1 Predictions

Table 2 summarizes RF & maximum-failure predictions compared with actual failure data, where available, for OI-A, OI-B, OI-C, OI-D. Because we do not know the actual RF & maximum-failures, we use assumptions #3a & 3b.

Table 2. Predicted RF & Maximum-Failures *vs* Actuals
[30-day TTT Intervals]

	t_t	s^*	α	β	$r(t_t)$	Actual r	$F(\infty)$	Actual F
OI-A	18	9	.534	.061	4.76	?[A]	11.76	7[A]
OI-B	20	1	1.69	.131	0.95	1[B]	12.95	13[B]
OI-C	20	7	1.37	.126	1.87	2[C]	12.87	13[C]
OI-D	18	6	.738	.051	7.36	4[D]	17.36	14[D]

Time of last recorded failure:

A. No additional failures reported after 17.17 intervals.
B. The last recorded failure occurred at 63.67 intervals.
C. The last recorded failure occurred at 43.80 intervals.
D. The last recorded failure occurred at 65.03 intervals.

Table 3 summarizes TTT and TTNF predictions compared with actual 'execution time' data, where available, for OI-A, OI-B, OI-C, OI-D.

Table 3. Predicted TTT & TTNF *vs* Actuals
[30-day TTT & TTNF Intervals]

	s^*	$t_t(r=1)$	Actual t_t	t	s^*	$T_F(t)$	Actual T_F
OI-A	9	43.59	?	18	9	3.9	?
OI-B	1	*	63.67	20	*	*	43.67
OI-C	7	24.98	27.07	20	5	4.2	7.63
OI-D	6	56.84	58.27	18	5	6.4	6.2

*Cannot predict because 'RF failures' < 1.

Additional Predictions for OI-D:

[These are additional predictions of TTT for OI-D that are not listed in table 3] $t_t(r=2) = 43.35$, Actual=45.17; $t_t(r=3) = 35.47$, Actual=23.70.

Table 4 summarizes the predictions of TTNF for a given 'reduction in RF' of 1 and the predictions of 'reduction in RF' for given TTNF compared with actual execution time and failure data (where available), for OI-A, OI-B, OI-C, OI-D.

Table 4. Predicted Tradeoffs of TTNF with 'Reduced RF' *vs* Actuals
[30-day TTT & TTNF Intervals]

	t	s^*	α	β	$T_F(\Delta r=1,t)$	Actual	(T_F,t)	$\Delta r(T_F,t)$	Actual
OI-A	18	9	.534	.061	3.87	?	3.87	1.00	?
OI-B	20	1	1.69	.131	*	43.67	43.67	.95	1.0
OI-C	20	5	1.34	.096	4.16	7.63	7.63	1.58	1.0
OI-D	18	5	1.61	.137	6.35	6.20	6.20	.99	1.0

*Cannot predict because 'predicted RF' < 1.

9.2 Validation

A total of 18 predictions were made across tables 2 - 4, where there was an actual value to compare: 3 $r(t)$, 4 $F(\infty)$, 4 t_t, 2 $T_F(t)$, 2 $T_F(\Delta r,t)$, 3 $\Delta r(T_F,t)$. The mean relative error, [mean of (actual−predicted)/actual] of prediction is 22.92% and the standard deviation is 27.61%. In making these predictions we note both the sparsity of post-delivery failures and the extremely long test times for Shuttle flight software, as summarized in table 5. The appendix lists the failure data.

Despite the fact that SSRM uses optimal selection of failure data, and thus less than the full set of data, there must be a minimum number of failures to start the parameter estimation process, understanding that the model will then select s^*. Thus, given the sparsity of the data, all failures in table 5 were used in parameter estimation, regardless of their severity. Furthermore, as described earlier, a more conservative risk assessment is produced if all categories of failures are included in the analysis.

Table 5. Failure Distribution by Severity Code (SC)
[30-day TTT Intervals]

	SC-2	SC-3	SC-4	Max Failures	TTT
OI-A	0	7	0	7	18
OI-B	5	8	0	13	64
OI-C	3	6	2	13*	44
OI-D	6	8	0	14	66

*Unknown Severity for two failures

There are no post-delivery SC-1 or SC-5 failures in the OI.

APPENDIX

Observed Failure Counts

[Interval i = 30-days execution time]

[4]The number of significant figures is not intended to imply any accuracy in the estimates, but to illustrate the arithmetic.

i	OI-A	OI-B	OI-C	OI-D
1	0	1	0	0
2	0	1	0	0
3	0	1	0	0
4	1	2	0	0
5	0	1	0	3
6	0	0	2	1
7	0	0	1	0
8	2	2	3	1
9	0	1	1	0
10	2	0	0	1
11	0	2	0	1
12	0	0	0	0
13	0	1	1	2
14	1	0	1	0
15	0	0	0	0
16	0	0	0	0
17	0	0	1	0
18	1	0	0	1
19		0	0	0
20		0	1	0
21		0	0	0
22		0	0	0
23		0	0	0
24		0	0	1
25		0	0	0
26		0	0	0
27		0	0	0
28		0	1	0
29		0	0	0
30		0	0	0
31-63	0			
64	1			
31-43		0		
44		1		
31-45				0
46				1
47-58				0
59				1
60-65				0
66				1
Totals:	7	13	13	14

ACKNOWLEDGMENT

I am pleased to acknowledge the support provided for this project by Dr. William Farr, Naval Surface Warfare Center; Ms. Alice Lee of NASA; US Marine Corps Tactical Systems Support Activity; and Mr. Ted Keller and Ms. Patti Thornton of Lockheed-Martin. I thank the referees and Associate Editor for their helpful comments.

REFERENCES

[1] *Recommended Practice for Software Reliability*, R-013-1992, 1993; ANSI/AIAA.

[2] B.W. Boehm, "Software risk management: Principles and practices", *IEEE Software*, vol 8, 1991 Jan, pp 32-41.

[3] C. Billings *et al*, "Journey to a mature software process", *IBM Systems J*, vol 33, num 1, 1994, pp 46-61.

[4] S.R. Dalal, A.A. McIntosh, "When to stop testing for large software systems with changing code", *IEEE Trans. Software Engineering*, vol 20, 1994 Apr, pp 318-323.

[5] S.R. Dalal, A.A. McIntosh, "Some graphical aids for deciding when to stop testing", *IEEE J. Selected Areas in Communications*, vol 8, 1990 Feb, pp 169-175.

[6] E.W. Dijkstra, "Structured programming", *Software Engineering Techniques* (J.N. Buxton, B. Randell, *Eds*), 1970 Apr, pp 84-88; NATO Scientific Affairs Division.

[7] W.H. Farr, O.D. Smith, *Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) Users Guide*, NAVSWC TR-84-373, rev 3, 1993 Sep; Naval Surface Weapons Center.

[8] W. Ehrlich *et al*, "Determining the cost of a stop-test decision", *IEEE Software*, 1993 Mar, pp 33-42.

[9] *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std 610.12.1990, 1990; IEEE.

[10] T. Keller, N.F. Schneidewind, P.A. Thornton, "Predictions for increasing confidence in the reliability of the space shuttle flight software", *Proc. AIAA Computing in Aerospace 10*, 1995 Mar 28, pp 1-8; San Antonio.

[11] N.G. Leveson, "Software safety: What, why, and how", *ACM Computing Surveys*, vol 18, 1986 Jun, pp 125-163.

[12] J.D. Musa, A.F. Ackerman, "Quantifying software validation: When to stop testing?", *IEEE Software*, vol 6, 1989 May, pp 19-27.

[13] J.D. Musa *et al*, *Software Reliability: Measurement, Prediction, Application*, 1987; McGraw-Hill.

[14] N.F. Schneidewind, "Software reliability model with optimal selection of failure data", *IEEE Trans. Software Engineering*, vol 19, 1993 Nov, pp 1095-1104.

[15] N.F. Schneidewind, T.W. Keller, "Application of reliability models to the space shuttle", *IEEE Software*, vol 9, 1992 Jul, pp 28-33.

[16] N.F. Schneidewind, "Analysis of error processes in computer software", *Proc. Int'l Conf. Reliable Software*, 1975 Apr 21-23, pp 337-346; IEEE Computer Society.

[17] N.D. Singpurwalla, "Determining an optimal time interval for testing and debugging software", *IEEE Trans. Software Engineering*, vol 17, 1991 Apr, pp 313-319.

[18] J.M. Voas, K.W. Miller, "Software testability: The new verification", *IEEE Software*, vol 12, 1995 May, pp 17-28.

[19] *Webster's New Universal Unabridged Dictionary* (2nd ed), 1979; Simon & Shuster.

[20] E.J. Weyuker, "Using the consequences of failures for testing and reliability assessment", *Proc. Third ACM SIGSOFT Symp. Foundations of Software Engineering*, 1995 Oct 10-13, pp 81-91; Washington DC.

AUTHOR

Dr. Norman F. Schneidewind; Code SM/Ss; Naval Postgraduate School; Monterey, California 93043 USA.
Internet (e-mail): schneidewind@nps.navy.mil

Norman F. Schneidewind is Professor of Information Sciences and Director of the Software Metrics Research Center at the Naval Postgraduate School. He is the developer of the Schneidewind software reliability model which is used by NASA to assist in predicting software reliability of the Space Shuttle. This model is one of the models recommended by ANSI & AIAA, *Recommended Practice for Software Reliability*. Dr. Schneidewind is a Fellow of the IEEE, elected for "contributions to software measurement models in reliability and metrics, and for leadership in advancing the field of software maintenance". He was awarded a certificate for outstanding research achievements in 1992 by the Naval Postgraduate School. He was Chair'n of the Working Group that produced the *IEEE Standard for a Software Quality Metrics Methodology*, published in 1993 Mar. In 1993 he was given the IEEE Computer Society *Outstanding Contribution Award* for "work leading to the establishment of IEEE Standard 1061-1992, Standard for a Software Quality Metrics Methodology".

Manuscript TR95-179 received 1995 December 5; revised 1996 September 15

Responsible editor: M.A. Vouk

Publisher Item Identifier S 0018-9529(97)03035-2 ◄TR►