APPLYING RELIABILITY MODELS TO THE MAINTENANCE OF SPACE SHUTTLE SOFTWARE

Norman F. Schneidewind

Professor of Information Sciences &
Director of Laboratories

Code AS/Ss
Naval Postgraduate School
Monterey, CA 93943

(408) 646-2719/2471
FAX: (408) 646-3407
Internet: 0442p@cc.nps.navy.mil

Abstract

Software reliability models provide the software manager with a powerful tool for predicting, controlling and assessing the reliability of software during maintenance. We show how a reliability model can be effectively employed for reliability prediction and the development of maintenance strategies, using the Space Shuttle Primary Avionics Software Subsystem, as an example.

Keywords: Application of software reliability models, reliability prediction, maintenance strategies, Space Shuttle.

Allocating Test Resources

It is important for software organizations to have a strategy for maintenance; otherwise, maintenance costs are likely to get out of control. Without a strategy, each module you maintain may be treated equally with respect to allocation of resources. You need to treat your modules unequally! That is, allocate more test time during maintenance, effort and funds to the modules which have the highest predicted number of failures, **F(t1,t2)**, during the *interval* **t1,t2**, where **t1,t2** could be execution time or labor time (of maintainers) for a *single module*. In the remainder of this section, "time" means execution time. Use the convention that you make a prediction of failures *at* **t1** for a continuous interval with end-points **t1+1** and **t2**.

The following sections describe how a reliability model can be used to predict **F(t1,t2)**. **The maintenance strategy is the following:**

Allocate test execution time to your modules during maintenance in proportion to F(t1,t2).

You update model parameters and predictions based on observing the actual number of failures, $X_{0,t1}$, during **0,t1**. This is shown in **Figure 1**, where you predict **F(t1,t2)**, using the model and the observed failures $X_{0,t1}$. In this figure, t_m is total available test time for a *single module*. Note that you could have **t2** = t_m (i.e., the prediction is made to the end of the test period).

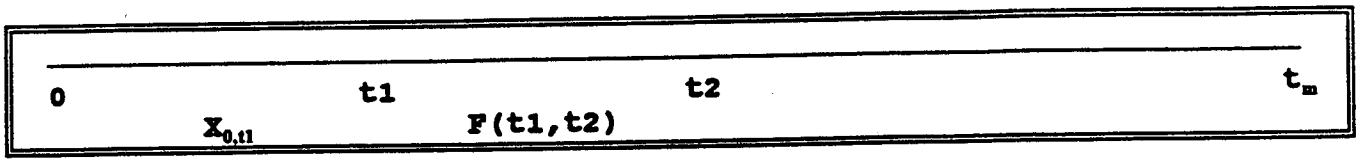

Figure 1. Reliability prediction time scale

Based on the updated predictions, you may want to reallocate your test resources during maintenance (i.e., test execution time). Of course, it could be disruptive to your organization to reallocate too frequently. So, you could predict and reallocate at major milestones (e.g., major upgrades). Using the Schneidewind software reliability model [2], the Space Shuttle Primary Avionics Software Subsystem, and failure data from the AIAA Software Reliability Database [3] as an example, the process of using prediction for allocating test resources is developed. Two parameters, α and β, which will be used in the following equations, are estimated by applying the model to $X_{0,t1}$ [2]. Once the parameters have been established, you can predict various quantities that will assist you in allocating test resources, as shown in the following equations:

- Number of failures during 0,t:

$$F(t) = (\alpha/\beta)[1 - \exp(-\beta(t-s+1)] \quad (1).$$

where $1 \leq s \leq t$ is the starting failure count interval determined by a mean square error criterion.

- Using (1) and **Figure 1**, you can predict number of failures during t1,t2:

$$F(t1,t2) = (\alpha/\beta)[1 - \exp(-\beta(t2-s+1))] - X_{0,t1} \quad (2).$$

- Also, you can predict maximum number of failures during the life ($t = \infty$) of the software:

$$F(\infty) = \alpha/\beta \quad (3).$$

- Using (3), you can predict the maximum remaining number of failures at t:

$$R(t) = (\alpha/\beta) - X_{0,t} \quad (4).$$

Given *n* modules, allocate test execution time *periods* T_i for each module *i* according to the following equation:

$$T_i = \frac{F_i(t1,t2) * (n)\,[t2-t1]}{\sum_{i=1}^{n} F_i(t1,t2)} \quad (5).$$

In (5), note that although predictions are made using (2) for a *single module*, the total *available* test execution time **(n)(t2 - t1)** is allocated for each module *i* across *n modules*. You use the same interval **0,20** for each module to estimate α and β and the same interval **20,30** for each module to make predictions, but from then on a variable amount of test time T_i is used depending on the predictions.

Tables 1 and **2** summarize the results of applying the model to the failure data for three Space Shuttle modules (operational increments). The modules are executed continuously, 24 hours per day, day after day. For *illustrative purposes*, each *period* in the test interval is assumed to be equal to 30 days. After executing the modules during **0,20**, the **SMERFS** [1] program was applied to the observed failure data during **0,20** to obtain estimates of α and β. The total number of failures observed during **0,20** and the estimated parameters are shown in **Table 1**.

Table 1

Observed Failures and Model Parameters

	X(0,20) failures	α	β
Module 1	12	1.6915	.1306
Module 2	11	1.7642	.1411
Module 3	10	1.3483	.1151

Equations (2), (3), (4) and (5) were used to obtain the predictions in **Table 2** during **20,30**. The prediction of **F(20,30)** led to the prediction of **T**, the allocated number of test execution time periods. The number of additional failures that were subsequently observed, as testing continued during **20,20+T**, is shown as **X(20,20+T)**. Since there may be remaining failures, **R(T)** is predicted from (4) and shown in **Table 2**. The predicted remaining failures indicate that additional testing is warranted. Note that the actual total number of failures **F(∞)** would only be known after *all* (i.e., extremely long test time) testing is complete and was not known at **20+T**. Thus you need additional procedures for deciding how long to test to reach a given number of remaining failures. A variant of this decision is the *stopping rule* (when to stop testing?). This is discussed in the following section.

Table 2

Allocation of Test Resources During Maintenance

	F(∞) failures	F(20,30) failures	R(T) failures	T periods	X(20,20+T) failures
Module 1					
Predicted	12.95	.693	.950	7.0	
Actual	13	0	1		0
Module 2					
Predicted	12.51	1.140	.507	11.6	
Actual	13	1	1		1
Module 3					
Predicted	11.65	1.125	.646	11.4	
Actual	14	1	3		1

Making Test Decisions During Maintenance

In addition to allocating test resources, you can use reliability prediction to estimate the *minimum* total test execution time t2 (i.e., interval 0,t2) necessary to reduce the *predicted* maximum number of remaining failures to R(t2). To do this, subtract equation (1) from (3), set the result equal to R(t2), and solve for t2:

$$t2 = \{\ln [(\alpha/\beta)/R(t2)]\}/\beta+(s-1) \quad (6).$$

where R(t2) can be established from:

$$R(t2) = (p)(\alpha/\beta) \quad (7).$$

where p is the desired fraction (percentage) of remaining failures at t2. Substituting (7) in (6) gives:

$$t2 = \{\ln [(1/p)]\}/\beta+(s-1) \quad (8).$$

Equation (8) is plotted for **Module 1**, **Module 2**, and **Module 3** in **Figure 2** for various values of p.

You can use (8) as a rule to determine when to stop testing *a given module* during maintenance.

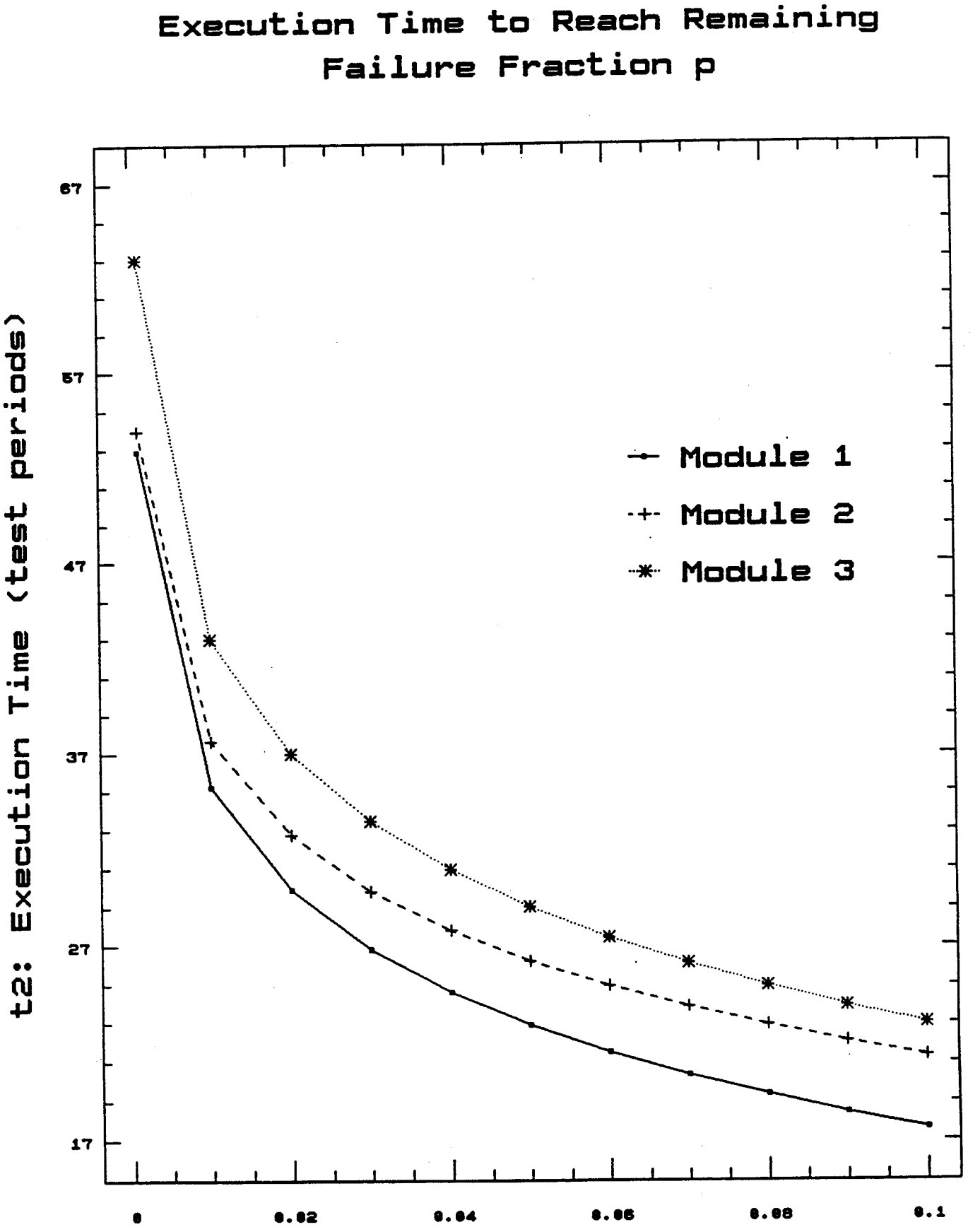

Figure 2. p: Remaining Failure Fraction

Using (8) and **Figure 2** you can produce **Table 3** which tells you the following: the total *minimum* test execution time **t2** from time **0** to reach essentially **0** remaining failures (i.e., at **p = .001 (.1%)**, predicted remaining failures are **.01295, .01251, .01165** for **Module 1, Module 2** and **Module 3**, respectively (see (7) and **Table 2**)); the additional test execution time beyond **20+T** shown in **Table 2**; and the *actual* amount of test time required, starting at **0**, for the "last" failure to occur (this quantity comes from the data and not from prediction). You don't know that it is necessarily the last; you only know that it was the "last" after 64 periods (1910 days), 44 periods (1314 days), and 66 periods (1951 days) for **Module 1, Module 2** and **Module 3**, respectively. So, **t2 = 52.9, 54.0** and **63.0** periods would constitute your *stopping rule* for **Module 1, Module 2** and **Module 3**, respectively. This procedure allows you to exercise control over software quality.

Table 3

Test Time t2 Required to Reach "0" Remaining Failures
p = .001

	t2	Additional Test Time	Last Failure Found
	periods	periods	periods
Module 1	52.9	45.9	64
Module 2	54.0	42.4	44
Module 3	63.0	51.6	66

SUMMARY

We have shown how to use a software reliability model for failure prediction, allocation of test resources during maintenance based on failure prediction, and a criterion for terminating testing based on prediction of remaining failures. These elements comprise a strategy for assigning priorities to modules for maintenance action.

REFERENCES

[1] William H. Farr and Oliver D. Smith, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) Users Guide, NAVSWC TR-84-373, Revision 2, Naval Surface Weapons Center, March 1991. (This program is available from Dr. William H. Farr, Strategic Systems Department, Naval Surface Warfare Center, Dahlgren, VA 22448, Tel: 703-663-4719).

[2] Norman F. Schneidewind and T.W. Keller, "Application of Reliability Models to the Space Shuttle", IEEE Software, July 1992 pp. 28-33.

[3] Recommended Practice for Software Reliability, R-013-1992, American Institute of Aeronautics and Astronautics, April 1992. (This document is available from Administrator, Standards, AIAA Headquarters, 370 L'Enfant Promenade SW, Washington, DC 20024-2518).

APPLYING RELIABILITY MODELS TO THE MAINTENANCE OF SPACE SYSTEM SOFTWARE

DR. NORMAN F. SCHNEIDEWIND

PROF. OF INFORMATION SCIENCES &
DIRECTOR OF LABORATORIES

CODE AS/SS
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

(408) 646-2719/2471

FAX: (408) 646-3407

INTERNET: 0442P@CC.NPS.NAVY.MIL

OUTLINE

- O PREDICT SOFTWARE RELIABILITY
- O DEVELOP MAINTENANCE STRATEGY
- O ESTIMATE MODEL PARAMETERS
  - SPACE SHUTTLE ON-BOARD SOFTWARE
- O PREDICT FAILURES
- O ALLOCATE TEST EXECUTION TIME
- O MAKE TEST DECISIONS DURING MAINTENANCE
  - DETERMINE WHEN TO STOP TESTING
- O SUMMARIZE

o THE MAINTENANCE STRATEGY IS THE FOLLOWING:

ALLOCATE TEST EXECUTION TIME TO YOUR MODULES DURING MAINTENANCE IN PROPORTION TO F(t1,t2).

o UPDATE MODEL PARAMETERS AND PREDICTIONS BASED ON OBSERVING THE ACTUAL NUMBER OF FAILURES, $X_{0,t1}$, DURING 0,t1. THIS IS SHOWN IN FIGURE 1, WHERE YOU PREDICT F(t1,t2), USING THE MODEL AND THE OBSERVED FAILURES $X_{0,t1}$.

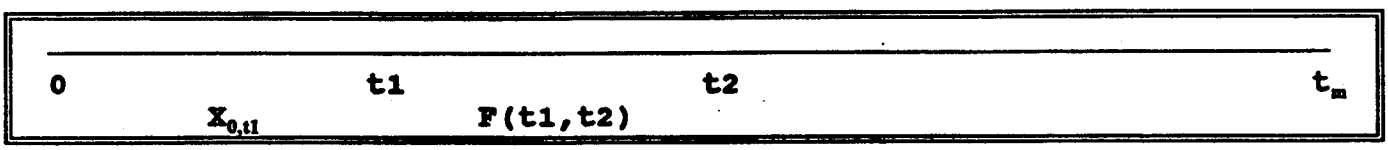

FIGURE 1. RELIABILITY PREDICTION TIME SCALE

ONBOARD PRIMARY SOFTWARE RELIABILITY PREDICTIONS

- Objective - To Predict Probability of Encountering a Serious Primary Software Error During Onboard Processing on the Next Shuttle Mission.

- Approach - Use Statistical Modelling of Error Detection History Data in the Configuration Management Data Base

  Given: Number of Failures Encountered During Execution* of Software

  - and -

  Failure Detection History for That Software

  Estimate: Mean Time Between Software Failure Encounters

  Model: Schneidewind Non-Homogeneous Poisson Distribution for Failure Detection (Encountered Due to Execution)

*Includes Test and Operational Use

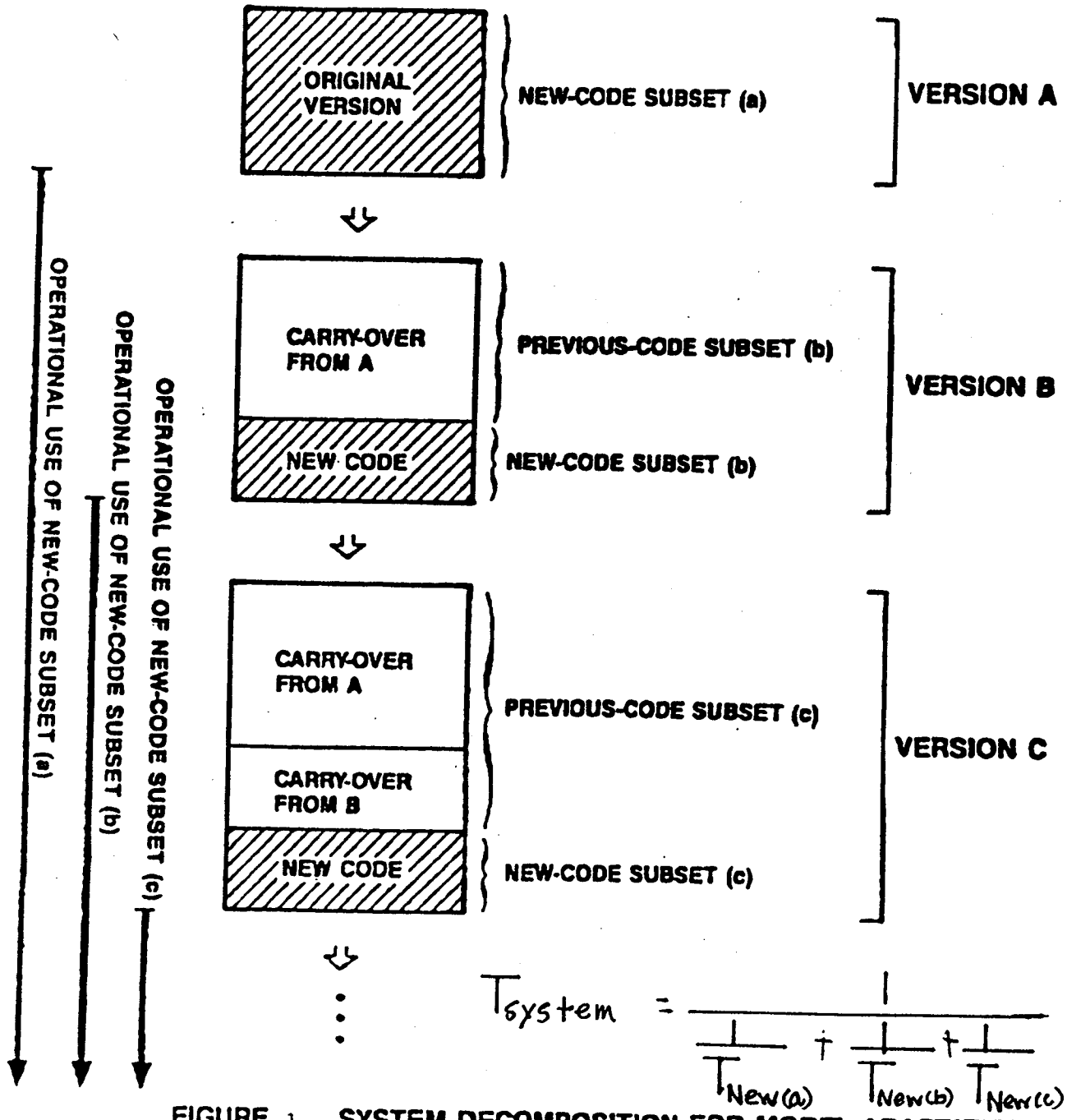

FIGURE 1 . SYSTEM DECOMPOSITION FOR MODEL ADAPTATION

THE TOTAL NUMBER OF FAILURES OBSERVED DURING 0,20 AND THE ESTIMATED PARAMETERS ARE SHOWN IN TABLE 1.

TABLE 1

OBSERVED FAILURES AND MODEL PARAMETERS

	x(0,20) FAILURES	α	β
MODULE 1	12	1.6915	.1306
MODULE 2	11	1.7642	.1411
MODULE 3	10	1.3483	.1151

YOU CAN PREDICT VARIOUS QUANTITIES THAT WILL ASSIST YOU IN ALLOCATING TEST RESOURCES, AS SHOWN IN THE FOLLOWING EQUATIONS:

o NUMBER OF FAILURES DURING 0,t:

$$F(t) = (\alpha/\beta)[1 - \exp(-\beta(t-s+1)] \quad (1).$$

WHERE $1 \leq s \leq t$ IS THE STARTING FAILURE COUNT INTERVAL DETERMINED BY A MEAN SQUARE ERROR CRITERION.

o USING (1) AND FIGURE 1, YOU CAN PREDICT NUMBER OF FAILURES DURING t1,t2:

$$F(t1,t2) = (\alpha/\beta)[1 - \exp(-\beta(t2-s+1))] - X_{0,t1} \quad (2).$$

o ALSO, YOU CAN PREDICT MAXIMUM NUMBER OF FAILURES DURING THE LIFE ($t = \infty$) OF THE SOFTWARE:

$$F(\infty) = \alpha/\beta \quad (3).$$

o USING (3), YOU CAN PREDICT THE MAXIMUM REMAINING NUMBER OF FAILURES AT t:

$$R(t) = (\alpha/\beta) - X_{0,t} \quad (4).$$

GIVEN *n* MODULES, ALLOCATE TEST EXECUTION TIME *PERIODS* T_i FOR EACH MODULE *i* ACCORDING TO THE FOLLOWING EQUATION:

$$T_i = \frac{F_i(t1,t2) * (n)\,[t2-t1]}{\sum_{i=1}^{n} F_i(t1,t2)} \quad (5).$$

- EQUATIONS (2)-(5) PREDICT FAILURES IN TABLE 2.
- PREDICTION OF F(20,30) LED TO THE PREDICTION OF T (ADDITIONAL TEST PERIODS PER MODULE DURING 20,30).
- NUMBER OF ADDITIONAL FAILURES OBSERVED, AS TESTING CONTINUED DURING 20,20+T, IS SHOWN AS X(20,20+T).
- TOTAL FAILURES IS SHOWN AS F(∞).
- THE PREDICTED REMAINING FAILURES R(T) INDICATE THAT ADDITIONAL TESTING IS WARRANTED.

TABLE 2

ALLOCATION OF TEST RESOURCES DURING MAINTENANCE

	F(∞) FAILURES	F(20,30) FAILURES	R(T) FAILURES	T PERIODS	X(20,20+T) FAILURES
MODULE 1					
PREDICTED	12.95	.693	.950	7.0	
ACTUAL	13	0	1		0
MODULE 2					
PREDICTED	12.51	1.140	.507	11.6	
ACTUAL	13	1	1		1
MODULE 3					
PREDICTED	11.65	1.125	.646	11.4	
ACTUAL	14	1	3		1

MAKE TEST DECISIONS DURING MAINTENANCE

- USE RELIABILITY PREDICTION TO ESTIMATE THE *MINIMUM* TOTAL TEST EXECUTION TIME t2 IN THE INTERVAL 0,t2 NECESSARY TO REDUCE THE *PREDICTED* MAXIMUM NUMBER OF REMAINING FAILURES TO p, WHERE p IS THE DESIRED FRACTION OF REMAINING FAILURES AT t2.

$$t2 = \{\ln [(1/p)]\}/\beta+(s-1) \qquad (8).$$

- EQUATION (8) IS PLOTTED FOR MODULES 1, 2 AND 3 IN FIGURE 2 FOR VARIOUS VALUES OF p.

YOU CAN USE (8) AS A RULE TO DETERMINE WHEN TO STOP TESTING *A GIVEN MODULE* DURING MAINTENANCE.

o USING (8) AND FIGURE 2, PRODUCE TABLE 3 WHICH GIVES THE FOLLOWING:

- THE TOTAL *MINIMUM* TEST EXECUTION TIME t2 FROM TIME 0 TO REACH p = .001 (.1%) REMAINING FAILURES (THE STOPPING RULE):

  * t2 = 52.9 PERIODS FOR MODULE 1
  * t2 = 54.0 PERIODS FOR MODULE 2
  * t2 = 63.0 PERIODS FOR MODULE 3

- ADDITIONAL TEST EXECUTION TIME BEYOND 20+T:

  * 52.9 - 7.0 (FROM TABLE 2) = 45.9 PERIODS FOR MODULE 1
  * 54.0 - 11.6 (FROM TABLE 2) = 42.4 PERIODS FOR MODULE 2
  * 63.0 - 11.4 (FROM TABLE 2) = 51.6 PERIODS FOR MODULE 3

TABLE 3

TEST TIME t2 REQUIRED TO REACH "0" REMAINING FAILURES

p = .001

	t2	ADDITIONAL TEST TIME	LAST FAILURE FOUND
	PERIODS	PERIODS	PERIODS
MODULE 1	52.9	45.9	64
MODULE 2	54.0	42.4	44
MODULE 3	63.0	51.6	66

SUMMARY

- SHOWN HOW TO USE A SOFTWARE RELIABILITY MODEL FOR FAILURE PREDICTION, ALLOCATION OF TEST RESOURCES DURING MAINTENANCE BASED ON FAILURE PREDICTION, AND A CRITERION FOR TERMINATING TESTING BASED ON PREDICTION OF REMAINING FAILURES.

- THESE ELEMENTS COMPRISE A STRATEGY FOR ASSIGNING PRIORITIES TO MODULES FOR MAINTENANCE ACTION.

WHERE TO OBTAIN SMERFS PROGRAM:

WILLIAM H. FARR AND OLIVER D. SMITH, STATISTICAL MODELING AND ESTIMATION OF RELIABILITY FUNCTIONS FOR SOFTWARE (SMERFS) USERS GUIDE, NAVSWC TR-84-373, REVISION 2, NAVAL SURFACE WEAPONS CENTER, MARCH 1991. (THIS PROGRAM IS AVAILABLE FROM DR. WILLIAM H. FARR, STRATEGIC SYSTEMS DEPARTMENT, NAVAL SURFACE WARFARE CENTER, DAHLGREN, VA 22448, TEL: 703-663-8388).

WHERE TO OBTAIN AMERICAN INSTITUTE OF AERONAUTICS AND ASTRONAUTICS (AIAA) "RECOMMENDED PRACTICE FOR SOFTWARE RELIABILITY":

ADMINISTRATOR, STANDARDS
AIAA HEADQUARTERS
370 L'ENFANT PROMENADE, SW
WASHINGTON, DC 20024-2518