UNCLASSIFIED

AD NUMBER_____

TECOM PROJECT NO. 7-CO-R90-EPO-003

AD-A231 439

METHODOLOGY INVESTIGATION

FINAL REPORT

SOFTWARE MATURITY MODEL VALIDATION

PHASE II

CURTIS MASSIE

Software and Interoperability Division
Electronic Technology Test Directorate

U.S. ARMY ELECTRONIC PROVING GROUND
FORT HUACHUCA, ARIZONA 85613-7110

4 September 1990

Prepared for:
U.S. Army Test and Evaluation Command
Aberdeen Proving Ground, MD 21005-5055

Approved for public release;
Distribution unlimited.

UNCLASSIFIED

DISPOSITION INSTRUCTIONS

Destroy this report in accordance with appropriate regulations when no longer needed. Do not return it to the originator.

DISCLAIMER

Information and data contained in this document are based on input available at the time of preparation. Because the results may be subject to change, this document should not be construed to represent the official position of the United States Army Material Command unless so stated.

The use of trade names in this report does not constitute an official endorsement or approval of the use of such commercial hardware or software. This report may not be cited for purposes of advertisement.

DEPARTMENT OF THE ARMY
U.S. ARMY ELECTRONIC PROVING GROUND
FORT HUACHUCA, ARIZONA 85613-7110

REPLY TO
ATTENTION OF

STEEP-ET-S (70-10r)

MEMORANDUM FOR Commander, TECOM, ATTN: AMSTE-TC-M,
Aberdeen Proving Ground, MD 21005-5055

SUBJECT: Methodology Investigation Final Report

1. The final report for TECOM Project 7-CO-R90-EP0-003, "Software Maturity Model Validation, Phase II", is provided for review and approval.

2. Our point of contact for this action is Mr. Curtis Massie, AUTOVON 821-8176.

FOR THE COMMANDER:

3 Encls

For Eugene E. Mahoney
BRENDA J. TAYLOR
Director, Electronic
Technology Test Directorate

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

DEPARTMENT OF THE ARMY
HEADQUARTERS, U.S. ARMY TEST AND EVALUATION COMMAND
ABERDEEN PROVING GROUND, MARYLAND 21005

REPLY TO
ATTENTION OF

AMSTE-TC-D (70-10p)

6 DEC 1990

MEMORANDUM FOR Commander, U.S. Army Electronic Proving Ground, ATTN: STEEP-ET-S, Fort Huachuca, AZ 85613-, 7110

SUBJECT: Final Report of Methodology Investigation, Software Maturity Model Validation, Phase II, TECOM Project 7-CO-R90-EP0-003

1. Subject report is approved.

2. Point of contact at this headquarters is Mr. Richard V. Haire, AMSTE-TC-D, amstetcd@apg-9.apg.army.mil, DSN 298-3677/2170.

FOR THE COMMANDER:

FREDERICK D. MABANTA
Chief, Technology Development Div
Directorate for Technology

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: US Army Electronic Proving Ground; 6b. OFFICE SYMBOL (If applicable): STEEP-ET-S	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code): Fort Huachuca, AZ 85613-7110	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: US Army Test & Eval Cmd; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code): Aberdeen Proving Ground, MD 21005	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Methodology Investigation of Software Maturity Model Validation

12. PERSONAL AUTHOR(S)

Curtis Massie

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM _____ TO _____	1990/9/4	97

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Software Test and Software Reliability

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report covers phase II of an investigation to identify methods of quantitatively assessing software reliability. A set of software reliability models were used in estimating reliability parameters in a developmental testing environment. These methods demonstrated that software reliability models are one tool available to the tester for providing quantitative assessments of software reliability.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL
Mr. Curtis Massie	(602) 533-8204	STEEP-ET-S

Table Of Contents

LIST OF FIGURES

Table Of Contents (Continued).

LIST OF FIGURES

LIST OF TABLES

FOREWORD

This report on the methodology investigation of the software maturity model validation represents the completion of Phase II of the investigation, during which the software reliability models recommended in the Phase I report were applied to the Mobile Subscriber Equipment (MSE) system for validation. This investigation concerns only the software reliability aspect of software maturity. It does not address any other facet of software maturity.

This report has been developed in accordance with Test and Evaluation Command (TECOM) Reg 70-17 and consists of the following sections and appendices:

a. Section 1 is an executive summary of the investigation.

b. Section 2 contains the details of the investigation.

c. Section 3 consists of the following appendices:

Appendix A - Methodology Investigation Proposal

Appendix B - References

Appendix C - Acronyms and Abbreviations

Appendix D - SMERFS Models

Appendix E - Example Software Reliability Test Plan

Appendix F - Glossary

This Page Is Intentionally Left Blank

SECTION 1. SUMMARY

1.1 BACKGROUND

Software reliability is an essential component in the estimation of Command, Control, Communications, and Intelligence (C^3I) system reliability. Software complexity in development and maintenance is steadily increasing. This potentially decreases both software and system reliability. Test data shows that software errors occur more often than hardware errors. Many of the software errors go undetected until the system is tested in the field. As a software error goes undetected, the costs to correct it increase. These are some of the reasons why software reliability is becoming increasingly important (reference 1). Software reliability is probably the most important factor of software quality (reference 2). Software managers and engineers must be able to measure and predict software reliability, to reduce software development and maintenance costs, and to determine if a software system is sufficiently reliable for fielding.

System reliability is measured in stochastic terms. That is, system reliability is "the probability that the system performs its assigned functions under specified environmental conditions for a given period of time" (Reference 3). The weight given to reliability when rating a system's overall quality depends on the criticality of the system's mission. According to one Air Force study, "In the past, the approach to determining or predicting system reliability has been to look at the hardware components, calculate their combined reliability, assume software reliability was one, and use the hardware reliability number as the system reliability". That study exposes the inadequacy of this approach. That study indicates that "software is a significant contributor to system failures" and it identifies software reliability models as one dimension of research to improve system reliability prediction and estimation.

Software reliability may be characterized in terms that closely parallel the definition of reliability for hardware systems. One definition of software reliability is "the frequency and criticality of program failure where failure is an unacceptable effect or behavior under permissible operating conditions." Like hardware, software reliability can be represented by the rate at which errors are uncovered and corrected. Unlike hardware, there is less evidence that empirical error data (collected during testing and after release of the software) can be used to develop accurate predictive models of software reliability.

It is difficult to give a precise definition of software reliability. Many attempts have been made to standardize the definition; however, no one definition is accepted as standard (reference 1). Some might say that software is reliable if it is correct. Software is reliable if it meets its initial

specifications and performs as specified. This definition does not recognize the possibility that the software specifications may be incomplete and incorrect. This definition confuses software reliability with software correctness. Software reliability concerns any software failure, whereas software correctness concerns the degree to which software design and code conform to specifications and standards (reference 4).

Musa defines software reliability as "the probability of failure-free operation of a computer program for a specified time in a specified environment" (reference 2). This definition fits this methodology report since the concern of this investigation is to ascertain software reliability measures which can be combined with hardware reliability measures to determine system reliability. The definition is tied to the idea that the reliability of a software system (i.e., hardware, software, and manual operations) is dependent upon the reliability of its hardware and software components. By using this definition, the dependence of software reliability on other software quality factors such as correctness and maintainability is recognized.

In Musa's definition of software reliability, failure-free is defined as having no occurrence of a software failure. Software failure is defined as a deviation of the operation of a computer program from its requirements (reference 2). Software failure and software error are used interchangeably in this report. Software fault is defined for this report as any incorrectness induced in software through human error.

Musa's definition of software failure implies that software failure is a dynamic process. That is, the program has to be executing for a failure to occur. Software failures can be characterized in the following ways: time to failure, time interval between failures, cumulative failures experienced up to a given time, and failures experienced during a time interval.

Software faults cause software failures. A software fault is created when a programmer makes a coding error. Faults are also created when a systems analyst incorrectly specifies a requirement or when a programmer analyst produces erroneous program design language (PDL). Each of these latter instances can lead to seemingly correct code which, when executed, propagates an erroneous requirement or design.

Program size and complexity are at a point where it is impossible to check the extremely large number of logic paths through the code (reference 5). For example, large scale real-time embedded systems such as the Trident-I Fire Control System (TFCS) have an extremely large number of logic paths (reference 6). It is impractical to check every conceivable logic path in its computer code. To avoid checking every path, software reliability estimation researches ways to quantify the numbers of faults

remaining in a program. This form of estimation can tell us how often the faults cause failures. The primary objective of software reliability prediction is that given a software component, what is the probability that it will fail in a given time period, or equivalently, what is the expected time duration between failures? In hardware, if the mean time between failure (MTBF) is too small, then more reliable components or redundant components are used to achieve the required improvements. In software, the improved reliability is obtained by replacing erroneous code with debugged code.

Over the past two decades, many models and estimation procedures have been proposed to quantify the reliability of software. Examples of such models are known as software reliability models. Dr. William H. Farr of the Naval Surface Warfare Center (NSWC) conducted a survey of reliability modeling and estimation techniques in which he identified three categories of software reliability models (reference 6): error seeding models, data domain models, and time domain models. Dr. Amrit Goel of Syracuse University categorized software reliability models in a similar way through a survey of his own (reference 3).

a. Error seeding/tagging models involve the intentional introduction of faults into a piece of software. These models assume the distribution of original software faults is the same as the distribution of the seeded faults. The total number of software faults inherent in the original software are estimated from counts of software faults discovered during testing, using the ratio of discovered seeded faults to the total number of seeded faults.

b. Data domain models estimate a program's current reliability. It is based on the ratio of the number of successful runs observed to the total number of runs made. The estimated reliability is simply the total number of successful runs divided by the total number of test runs. Run is an arbitrary term generally associated with some function software performs (reference 2).

c. Time domain models have received the greatest emphasis in the literature and in real world applications. These models find their roots in hardware reliability modeling. The concepts of hardware reliability modeling were adapted for use in modeling software reliability. Some of these models have terms which do not have hardware counterparts (e.g., the number of remaining faults).

(1) Each model makes assumptions that can vary from model to model (reference 6). One of the assumptions made by the Geometric Poisson Model is that each discovered software fault is either corrected or not counted again. Brooks and Motley's Models (BAMMOD) assume that software faults can be reintroduced in the

software fault correction process. Musa's Execution Time Model assumes the software failure rate is constant and changes only at each software fault correction. Moranda's Geometric Model (GEOMOD) assumes the software failure rate is initially a constant which decreases in a geometric progression as software failures are detected.

(2) Each model makes certain assumptions about the independence of software failures. For example, Moranda's GEOMOD assumes the detections of software failures are independent. This assumption is needed to model software failure as an exponentially decaying process. The independence assumptions made by these models are often challenged. Considerable evidence shows the independence assumptions are valid (reference 2). The confusion arises from the argument that software faults can be related; not independent. The possibility of related faults does not imply that software failures are not independent.

Neither survey addresses the category of software reliability models based on the internal characteristics of the program. These models provide a priori estimates of software failures. They predict the number of software failures before operational data is available. Dozens of such models estimate the number of faults in a program based on static characteristics of the program. These are generally related to software complexity measures (reference 7). These models predict the total number of failures using the fault reduction ratio. This is the number of inherent faults in the program divided by the fault-reduction factor. The fault reduction factor is estimated from empirical data involving previous software development projects. Evidence shows this factor is project independent, although such a value is not known at present (reference 2).

The model designers do not propose use of models without checking the reasonableness of their assumptions. The current trend is to incorporate one or more time domain models in a software package which includes routines to perform statistical analysis of the models. These packages include options to check the reasonableness of the model assumptions by seeing how well the model fits the data (reference 5).

The Department of Defense (DoD) has issued a directive which addresses the reliability problem (reference 8). Department of Defense Directive (DoDD) 5000.3, "Test and Evaluation," authorizes the issuance and publication of DoD 5000.3-M-3 (reference 9), "Software Test and Evaluation Manual." This manual states that "[the Test and Evaluation Master Plan (TEMP)] must address the development process and the planning for Post Deployment Software Support (PDSS), as well as quantification of software maturity, system-level performance, and reliability, availability, and maintainability (RAM) parameters." The manual also states the TEMP must provide an approach to relate software reliability to system

reliability, and must describe how system-level RAM indicators integrate software and hardware perfcrmance characteristics. Consistent, quantifiable measures of software reliability are required to conform to this directive. Only then can system reliability be derived in a meaningful way.

1.2 PROBLEM

DoDD 5000.3-M-3 requires quantification of software reliability to help ascertain system reliability (reference 9). However, the suitability of various reliability models for providing quantitative reliability parameters with available test data had not been demonstrated. This investigation evaluated the suitability of candidate software reliability models for providing these quantitative estimates for test items requiring evaluation by TECOM.

1.3 OBJECTIVE

The objective of this investigation is to establish an accepted method of computing software reliability to help assess the maturity of software in embedded computer resources (ECR).

1.3.1 Completed Phase I goals Include:

a. Development of a set of candidate software reliability models to be screened for use in estimating software reliability.

b. Identification of approaches other than reliability models to estimate software reliability.

c. Completion of an evaluation of the set of candidate software reliability models and other approaches to determine if they can be applied to developmental testing (DT).

1.3.2 Phase II Goals Include:

a. Providing recommendations for a set of models or any other methods to help determine the status of software during DT.

b. Evaluation of these recommendations by conducting example reliability testing on the MSE system, for which field test data are readily available.

c. Providing recommendations for additional data collection for future C^3I system testing.

1.4 PROCEDURES

During Phase I, methods for assessing software reliability were identified. Once identified, the models and methods were evaluated based on the following criteria:

a. Does the model or method provide probabilistic and quantitative estimates of program reliability?

b. Is the model or method sufficiently documented?

c. Does the model or method have realistic data requirements?

d. Is the model or method readily available?

e. Is the model or method applicable to DT?

Phase I resulted in a report on these models and methods which recommended the use of a set of software reliability models known as the Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) package. SMERFS is available free of charge from the NSWC. For a complete description of the SMERFS package, see Appendix D.

Phase II applied software reliability models in the TECOM testing environment. MSE was chosen as a test case because, like most C^3I systems, its system configuration is comprised of multiple instances of the same software component. Unlike the case of a single software component on one machine, this test configuration required interleaving of failure data to obtain a true picture of software reliability.

The MSE test data was obtained from the Operational Test and Evaluation Agency's (OTEA's) Communications Reliability and Maintainability (COMRAM) data base and stored on floppy disks. Part of the data base was extracted based on the data base structure and the required inputs to the SMERFS models.

The eight reliability models in the SMERFS package were acquired. Four of these models were used to estimate the software reliability of the major MSE software components. The results of applying this methodology are described in the following section.

1.5 RESULTS

The data used for this test case came from the TECOM MSE testing. The test data used in this investigation includes software failures reported for the Node Center Switch (NCS) and the Large Extension Node (LEN) components of the MSE. These are composed of the same software and similar hosts. Software failure data from the four NCSs and one LEN which participated in the test

were aggregated to provide an overall reliability figure for the software associated with these components. Appendix E illustrates an example software reliability test plan.

The 85 aggregate software failures reported for all 5 components (4 NCSs and 1 LEN) were used. The total run time for each instance of the software, measured in wall clock time, was 71 days. Total run time for the software was (71 days/component)(5 components) = 355 days. A simple check of model output reasonableness is available by dividing total time by total failures as a rough estimator of MTBF. Thus (355 days)/(85 failures) = 4.2 days between failures.

Each of the models in SMERFS to which available data applied was used in this investigation.

The output for each model follows:

a. The LAVMOD maximum likelihood (ML) estimate was 6.6 days MTBF; the least squares (LS) estimate was 4.6 days MTBF.

b. The GEOMOD ML estimate was 4.4 days MTBF with 95% confidence that the MTBF lies between 2.54 and 6.33 days; the LS estimate was 4.9 days MTBF.

c. The Generalized Poisson Model (GPOMOD), which operates on the intervals between errors, provided a ML estimate of one software failure per five day period. The model predicted that there are 260 total errors within the software, with 95% confidence that the total number of software errors is between 252 and 267. The model estimated the number of errors remaining as 176, with 95% confidence that this estimate is between 168 and 183 remaining errors. The GEOMOD LS estimator predicted 341 total software errors, with 257 errors remaining undetected. This method also provided an estimate of one software failure per five day period.

d. The Norman Schneidewind Model (SDWMOD) provided an LS estimate of 1.2 errors per five day period.

All estimates of MTBF fall within the same range of 4 to 5 days, except the LAVMOD ML estimator. It was slightly higher. The 4 to 5 days MTBF is also in line with the 4.2 day MTBF estimate derived from (total time)/(total failures).

For a more detailed discussion of model estimates, see Section 2 of this report.

1.6 ANALYSIS

The determination of adequate software reliability requirements could pose a significant problem. Probabilistic, quantitative requirements for software and system reliability are required criteria against which to evaluate software systems through testing. While minimum criteria may be established through operational necessity, the feasibility of such requirements may be difficult to establish.

Acquiring the additional data needed to run both Central Processing Unit (CPU) time and wall clock time models could be done with reasonable effort. Since the additional data should be available during testing, it could be collected. Collecting such data could be required as part of software system tests, so the entire set of SMERFS models is available for use in system evaluation.

The output of reliability models is suitable for determining system reliability. The software reliability estimates presented in the results section, can be used to produce estimates of the software failure rate. The software failure rate, the hardware failure rate, and the knowledge of the configuration of the system under test, can be used to determine overall system reliability (Reference 2).

The SMERFS models provide quantitative, probabilistic estimates of software reliability. The estimates are used in the computation of system reliability, as required by DoDD 5000.3-M-3. The models are usable in the TECOM test environment, as demonstrated by the application of these models to test data collected as part of the MSE field test. Provision of test methods to collect CPU time between failure data would enhance the utility of these models, and provide for reasonable estimates of software reliability during TECOM testing.

1.7 CONCLUSIONS

The Phase II goals of the software reliability model investigation were achieved. The conclusions follow:

a. The SMERFS software reliability models are usable in the TECOM testing environment.

b. The software reliability estimates provided by the models satisfy the requirement to provide quantitative estimates of software and system reliability.

c. The collection of CPU time between failure data would enhance the utility of the SMERFS models.

1.8 RECOMMENDATIONS

The following items are recommended:

a. Procedures should be developed to define the explicit method of combining the hardware and software reliability estimates into a system reliability estimate.

b. The Army's test data collection should be enhanced to include CPU time expended between software error occurrences and starting and ending times of testing.

c. Quantitative requirements for software system reliability should be included as part of test plans such as the TEMP. Software and system reliability could then be used, with these quantitative criteria for system evaluation.

This Page Is Intentionally Left Blank

SECTION 2. DETAILS OF INVESTIGATION

Phase I of this methodology investigation identified a set of potential models and methods for use by TECOM in evaluation of software and system reliability. The models and methods included the United States Army Electronic Proving Ground (USAEPG) software Performance Parameter Assessment (SPPA) model, SMERFS, and three methods which estimate levels of software maturity. Software reliability estimation is the primary focus of this investigation. Maturity estimation methods were not recommended. The SPPA was not recommended because of unrealistic input data requirements. The SMERFS package was recommended because it met all of the evaluation criteria, described in Section 1.4 of this report.

This section will provide application details of the SMERFS models to MSE field test data, accomplished in Phase II. All data lists, plots, and results are from SMERFS output.

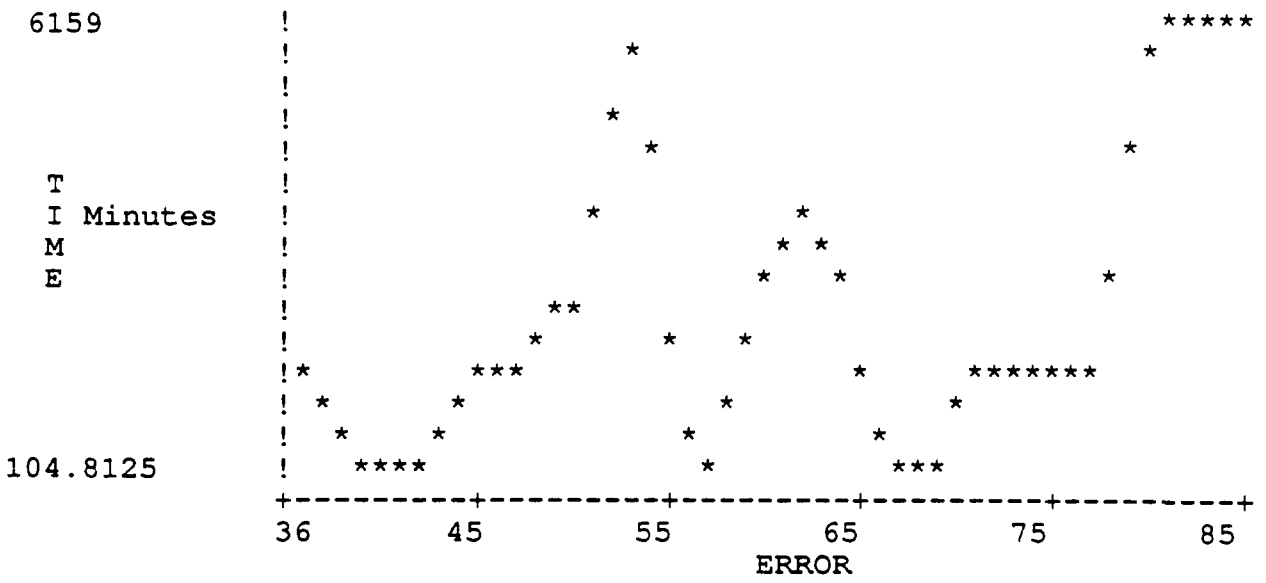
The raw data used for these tests is provided in Table 2-I. Smoothed raw data plots are provided as Figure 2-1. The graphs in Figure 2-1 show time to "failure n" (1 <= n <= 85), from "failure n-1". The LAVMOD and the GEOMOD model use the data in this form to determine mean time between software failures. Figure 2-2 presents some standard statistics on the test data.

Table 2-I. NCS and LEN Software Time Between Failure Raw Data (Minutes).

Error	Time Between	Error	Time Between
1	9078	27	2420
2	6180	28	1795
3	3041	29	5083
4	683	30	723
5	2921	31	3109
6	60	32	4557
7	8025	33	1135
8	755	34	1630
9	308	35	15510
10	30257	36	104
11	5588	37	108
12	41987	38	164
13	135	39	2590
14	575	40	1305
15	16	41	198
16	5748	42	362
17	2228	43	3614
18	4013	44	12
19	8836	45	4585
20	3075	46	1535
21	747	47	20
22	23613	48	365
23	2450	49	2309
24	1175	50	2965
25	3614	51	1595
26	4705	52	5525

Table 2-I. NCS and LEN Software Time Between Failure Raw Data (Minutes) (Continued).

Error	Time Between	Error	Time Between
53	22643	70	0
54	5540	71	16
55	1675	72	12872
56	490	73	1815
57	500	74	414
58	1456	75	2315
59	3935	76	1426
60	2158	77	150
61	3102	78	50
62	3580	79	5567
63	4740	80	98040
64	20660	81	7354
65	7360	82	4310
66	687	83	6159
67	225	84	7347
68	300	85	4895
69	111		



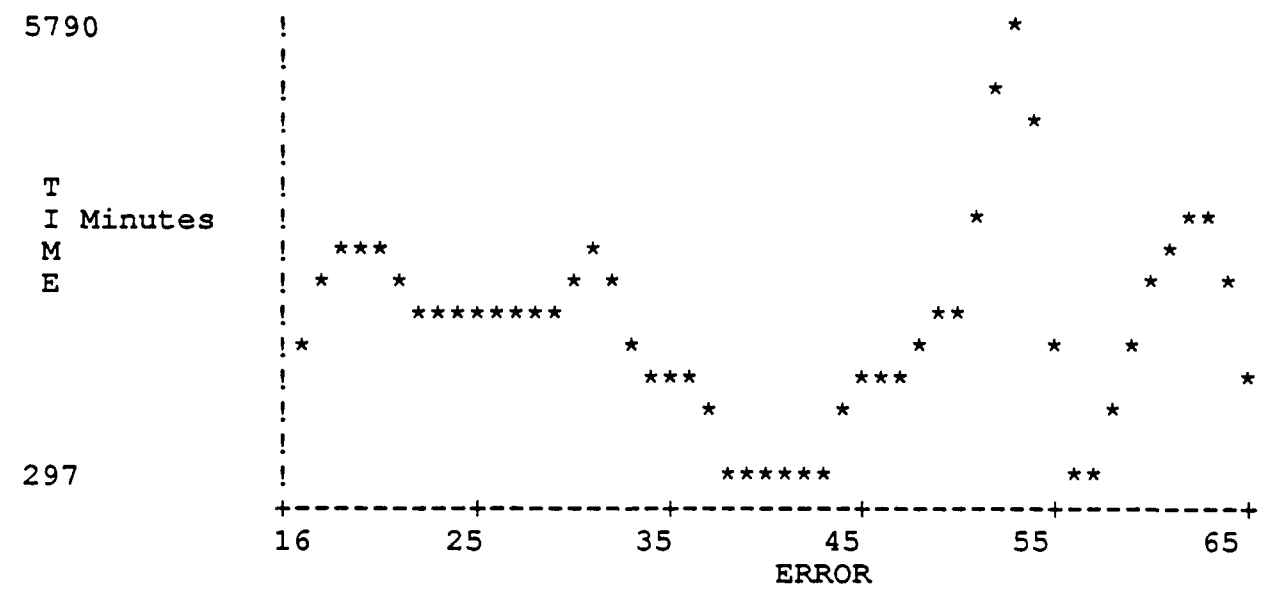


Figure 2-1. Time-To-Failure Raw Data.

```
WALL CLOCK TIME-BETWEEN-ERROR
WITH TOTAL TESTING TIME OF          495578.00

AND TOTAL ERRORED TIME OF           495578.00
                    ******************************************
   MEDIAN OF THE DATA   *              23090                 *
   LOWER & UPPER HINGES *      500.             5083         *
   MINIMUM & MAXIMUM    *      0                98040        *
   NUMBER OF ENTRIES    *                 85                 *
   AVERAGE OF THE DATA  *            5830.3294              *
   STD. DEV. & VARIANCE *      12831.117        164637563    *
   SKEWNESS & KURTOSIS  *      5.0730806        30.78382     *
                    ******************************************
```

Figure 2-2. SMERFS Summary Statistics.

2.1 LITTLEWOOD AND VERRALL BAYESIAN MODEL RESULTS

The LAVMOD assumes that successive times between software failures are independent, exponentially distributed random variables $x(i)$, $i = 1,2,\ldots,n$ with parameter $\mu(i)$. Each parameter represents the failure rate within one time interval. The varying parameters account for changing failure rates due to debugging and introduction of new errors into the software as a result of the debugging process. The $\mu(i)$ are assumed independent, Γ distributed variables with parameters α and $\pi(i)$. The functions $\pi(i)$ represent the quality of the programmer(s) and the programming task difficulty.

In other words, the model assumes that the software has a given failure rate at the start of testing. As bugs are discovered and fixed, the failure rate changes based on the quality of the programmer(s) and the difficulty of the task. Note: For the MSE test case, software failures were assumed fixed at discovery (not counted again). The test data is used to determine the α parameter and the $\pi(i)$ function coefficients (assuming either a linear or quadratic function), which define the a priori failure rate. The a posteriori failure rate is then determined using the a priori rate and the test data.

The results provided from this model, based on the test data in Table 2-I, are as follows:

Maximum Likelihood Estimator For $\pi(i)$ and α, Linear $\pi(i)$

```
INITIAL ESTIMATES FOR BETA(0) AND BETA(1).
                        1.0         0.5

ML MODEL ESTIMATES AFTER 35 ITERATIONS ARE:
     ALPHA   :  1.2926847
     BETA(0) :  2666.4551
     BETA(1) :  1.6041024

 THE EXPECTED TIME (To Failure) IS (5.99 days)
```

Using initial estimates closer to the estimates derived for the linear $\pi(i)$ function,

```
INITIAL ESTIMATES FOR BETA(0) AND BETA(1).
                      2000.0        1.0

ML MODEL ESTIMATES AFTER 37 ITERATIONS ARE:
     ALPHA   :  1.2925768
     BETA(0) :  2665.5628
     BETA(1) :  1.5356956

 THE EXPECTED TIME (To Failure) IS (6.01 days)
```

Maximum Likelihood Estimator For $\pi(i)$ and α, Quadratic $\pi(i)$

```
INITIAL ESTIMATES FOR BETA(0) AND BETA(1).
                      1.0          0.5

ML MODEL ESTIMATES AFTER 62 ITERATIONS ARE:
     ALPHA    :  1.2975483
     BETA(0)  :  2514.4319
     BETA(1)  :  .0041818599

   THE EXPECTED TIME (To Failure) IS (6.6 days)
```

Using initial estimates closer to the estimates derived for the coefficients of the $\pi(i)$ function,

```
INITIAL ESTIMATES FOR BETA(0) AND BETA(1).
                    2514.0       0.0418

ML MODEL ESTIMATES AFTER 13 ITERATIONS ARE:
     ALPHA    :  1.2975502
     BETA(0)  :  2514.0003
     BETA(1)  :  .042743287

   THE EXPECTED TIME (To Failure) IS (6.6 days)
```

Figure 2.1-1 presents a graph of the time-to-failure prediction (the points labeled "P") with the test data (the points labeled "*"), using the maximum likelihood estimator with linear $\pi(i)$.

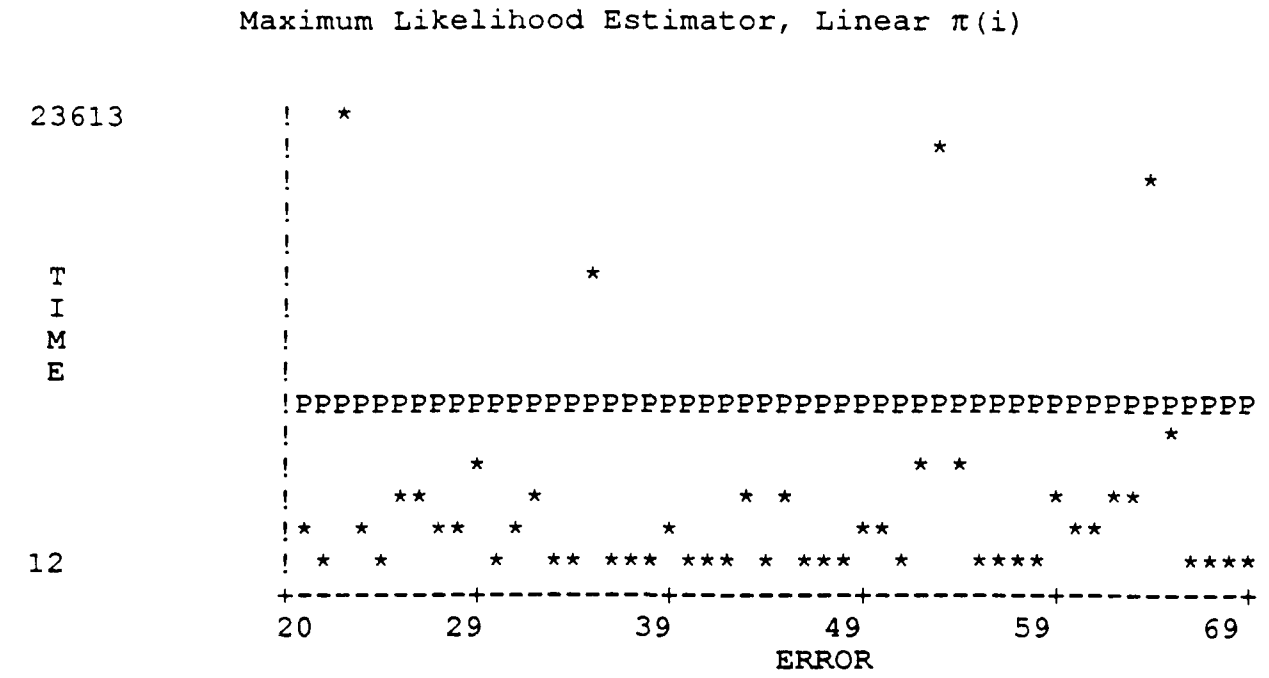


Figure 2.1-1. Bayesian Model Maximum Likelihood Time-To-Failure Prediction Versus Test Data.

Least Squares Estimator For π(i) and α, Linear π(i)

INITIAL ESTIMATES FOR BETA(0) AND BETA(1).
1.0 0.5

LS MODEL ESTIMATES AFTER 75 ITERATIONS ARE:
ALPHA : 1.0008941
BETA(0) : 4.5549322
BETA(1) : .01530675

THE EXPECTED TIME (To Failure) IS (4.6 days)

Using initial estimates closer to the derived values for the coefficients of π(i),

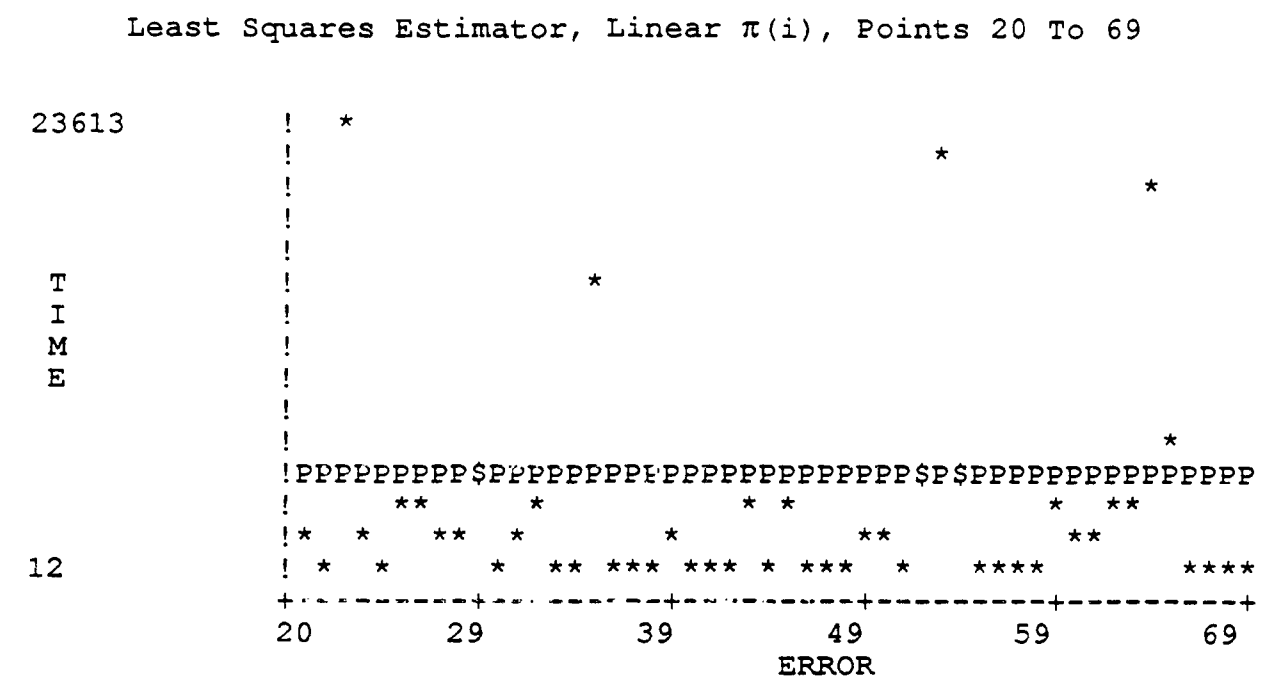
INITIAL ESTIMATES FOR BETA(0) AND BETA(1).
4555.0 0.0153

LS MODEL ESTIMATES AFTER 56 ITERATIONS ARE:
ALPHA : 1.8953041
BETA(0) : 4560.8694
BETA(1) : 15.326705

THE EXPECTED TIME (To Failure) IS (4.6 days)

Figure 2.1-2 presents plots of the predicted times between failure along with the test data points. Test data points are denoted by "*", predicted points are denoted by "P", and points at which predicted and actual values coincide are denoted by "$". The figure includes a plot of the residuals.

The graphs show the least squares estimator is the better fit to the data. The results of the LS estimator are close to the check value of 4.2 days, derived by (total time)/(total failures).



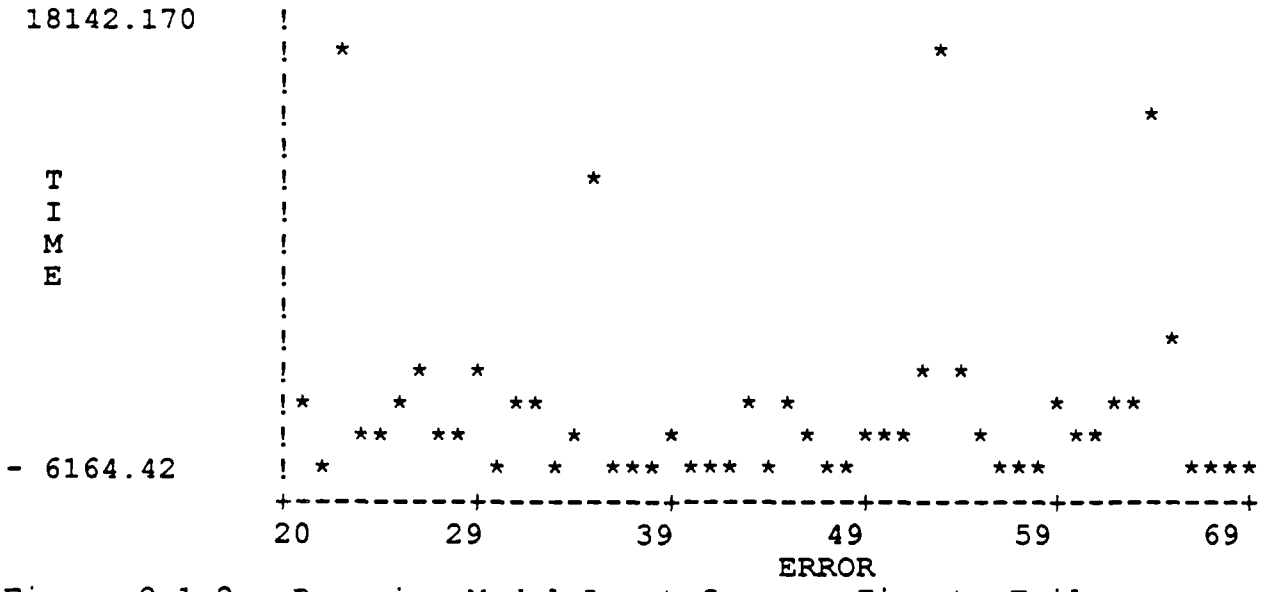


Figure 2.1-2. Bayesian Model Least Squares Time-to-Failure Prediction Versus Test Data.

2.2 MORANDA'S GEOMETRIC MODEL RESULTS

The GEOMOD assumes the software hazard rate is reduced by the same amount at the time of each error detection. The software hazard rate is defined as the conditional probability that a software failure occurs in an interval of time, given that the software has not failed up to the beginning of that time interval. The hazard rate function decreases in a geometric progression as the detection of errors occurs. The test data are used in this model to determine the proportionality constant of the hazard rate.

The results of this model, based on the data in Table 2-I, are as follows:

Maximum Likelihood Estimate of Hazard Rate Proportionality Constant

INITIAL ESTIMATE FOR THE PROPORTIONALITY CONSTANT.
0.99

ML MODEL ESTIMATES AFTER 3 ITERATIONS ARE:

PROPORTIONALITY CONSTANT OF THE MODEL IS .99782789
WITH 95% CONFIDENCE INTERVAL OF (.96803084 , 1.0)

THE INITIAL HAZARD RATE IS .00018827987
WITH 95% CONFIDENCE INTERVAL OF (0.0, 0.0059418343)

THE MEAN TIME BEFORE THE NEXT FAILURE IS (4.4 days)
WITH 95% CONFIDENCE INTERVAL OF (2.54, 6.33 days)

THE CURRENT "PURIFICATION LEVEL" IS .16875474
WITH 95% CONFIDENCE INTERVAL OF (0.0, 0.7778357)

The purification level indicates the error-freeness of the software. On a scale of 0 to 1, 1 is defined as an error-free program.

Figure 2.2-1 presents a plot of the actual and predicted values of the time between failures. The figure includes a plot of the residuals for the GEOMOD predictions.

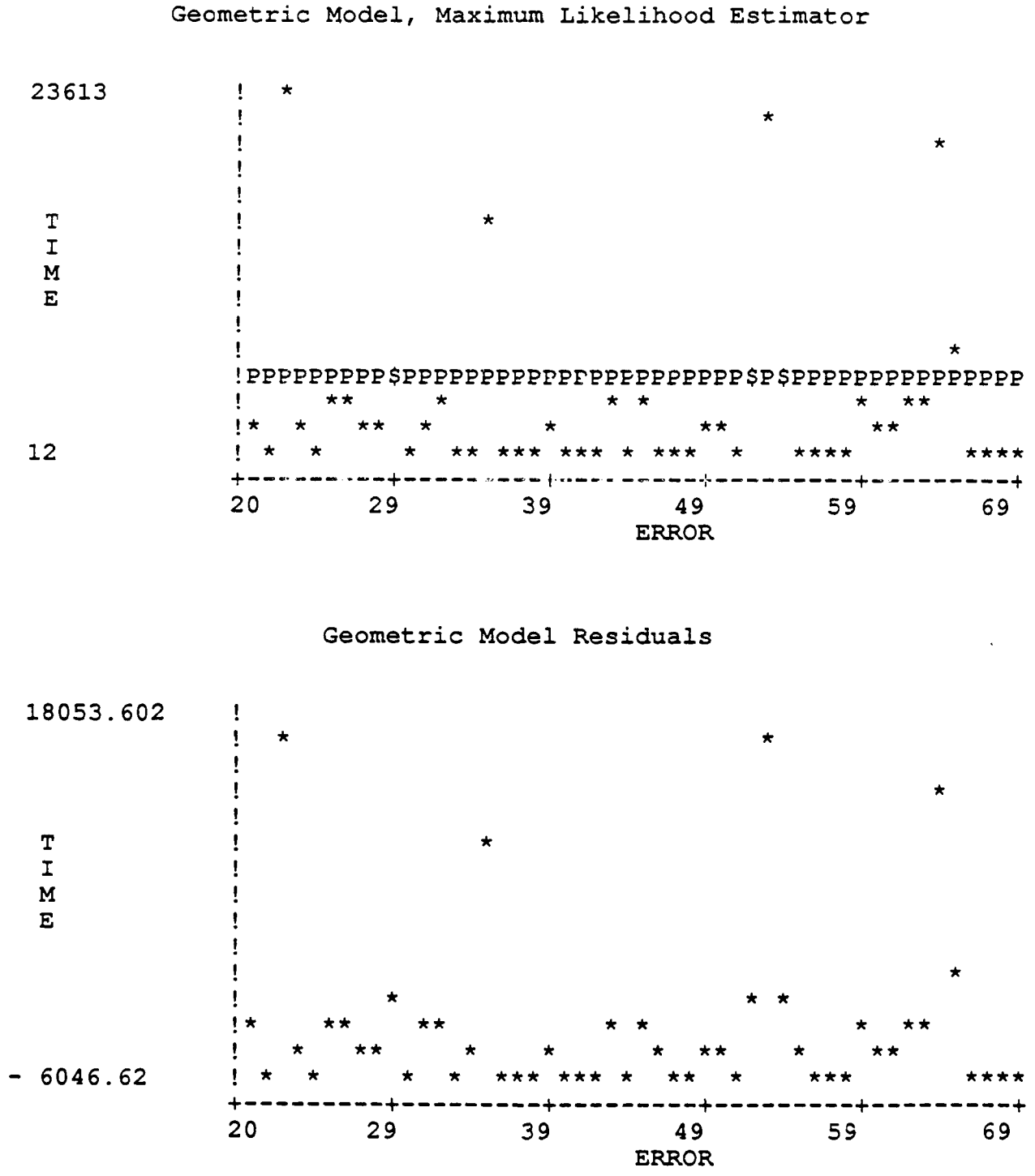


Figure 2.2-1. Geometric Model Maximum Likelihood Time-To-Failure Prediction Versus Test Data.

Least Squares Estimate of Hazard Rate Proportionality Constant

INITIAL ESTIMATE FOR THE PROPORTIONALITY CONSTANT.
0.99

LS MODEL ESTIMATES AFTER 5 ITERATIONS ARE:

PROPORTIONALITY CONSTANT OF THE MODEL IS .99532964

THE INITIAL HAZARD RATE IS .00021067985

THE MEAN TIME BEFORE THE NEXT FAILURE IS (4.9 days)

THE CURRENT "PURIFICATION LEVEL" IS .32827792

The MTBF is slightly higher for the LS estimate than for the ML estimate. The LS estimate of 4.9 days falls well within the 95% confidence interval for the ML estimate (2.54 to 6.33 days). Both estimates seem reasonable in comparison with the rough check estimate of 4.2 days. The LS estimate for the MTBF for the LAVMOD falls between the GEOMOD estimates.

Figure 2.2-2 presents a plot of the predicted versus actual values for the GEOMOD LS estimator, along with a plot of the residuals.

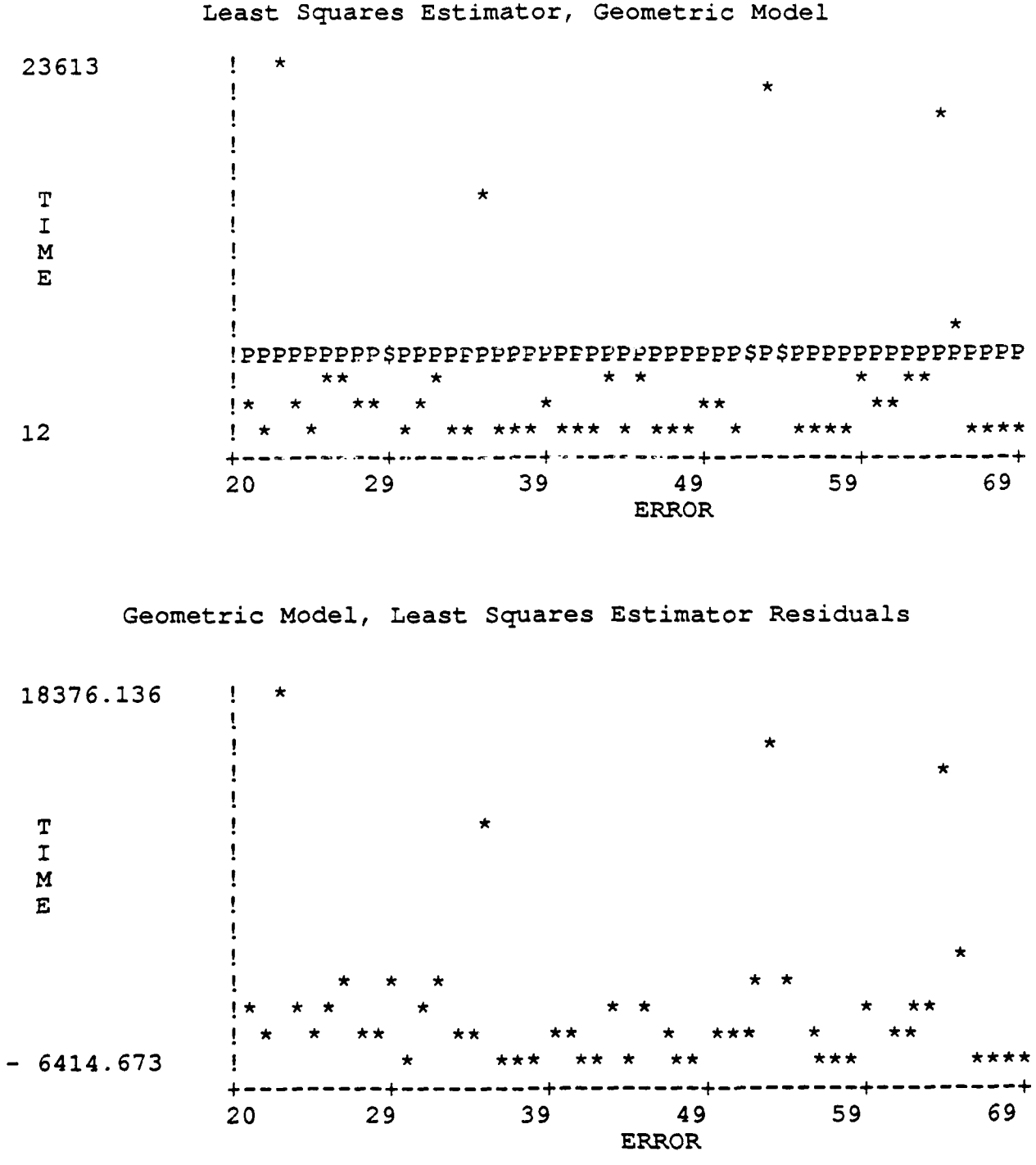


Figure 2.2-2. Geometric Model Least Squares Time-To-Failure Prediction Versus Test Data.

2.3 GENERALIZED POISSON MODEL RESULTS

Figure 2.3-1 provides the total test time of 355 days broken up into 71 5-day intervals. Software failures reported during each interval are counted and the counts during each interval are used in the statistical analysis. The GPOMOD model and the SDWMOD use the test data in this form. Figure 2.3-2 presents summary statistics on the raw test data.

The GPOMOD assumes the expected number of software errors discovered in any time interval is proportional to the product of the total number of existing software errors and some function of the amount of time spent in testing for software errors.

This model estimates the constant of proportionality, total errors, and errors remaining. Given a time interval, this model computes the expected number of failures encountered in that interval.

The estimates produced by the GPOMOD model are as follows:

Maximum Likelihood Estimate With Schick-Wolverton Weighting

Note that Schick-Wolverton weighting is defined as

Weight = $(x(i)^2)/2$, i = 1,2,...,n, where each x(i) is a sample point.

```
    INITIAL ESTIMATE OF THE TOTAL NUMBER OF ERRORS.
                        90.0

    ML MODEL ESTIMATES, USING SCHICK-WOLVERTON WEIGHTING, AFTER
   11 ITERATIONS ARE:

     PROPORTIONALITY CONSTANT OF THE MODEL IS  .00044340933
      WITH 95% CONFIDENCE INTERVAL (-.00025764047, .0011444591)

     THE TOTAL NUMBER OF ERRORS IS              259.6439
      WITH 95% CONFIDENCE INTERVAL (252.51749, 266.77031)

     THE REMAINING NUMBER OF ERRORS IS          175.6439
      WITH 95% CONFIDENCE INTERVAL (168.51749, 182.77031)

     PROJECTED LENGTH OF THE TESTING PERIOD.
                     1.0
     THE EXPECTED NUMBER OF ERRORS IS  .038719368
      WITH 95% CONFIDENCE INTERVAL (.022184252, .055254483)

     PROJECTED LENGTH OF THE TESTING PERIOD.
                    10.0
     THE EXPECTED NUMBER OF ERRORS IS  3.8719368
      WITH 95% CONFIDENCE INTERVAL (2.2184252, 5.5254483)
```

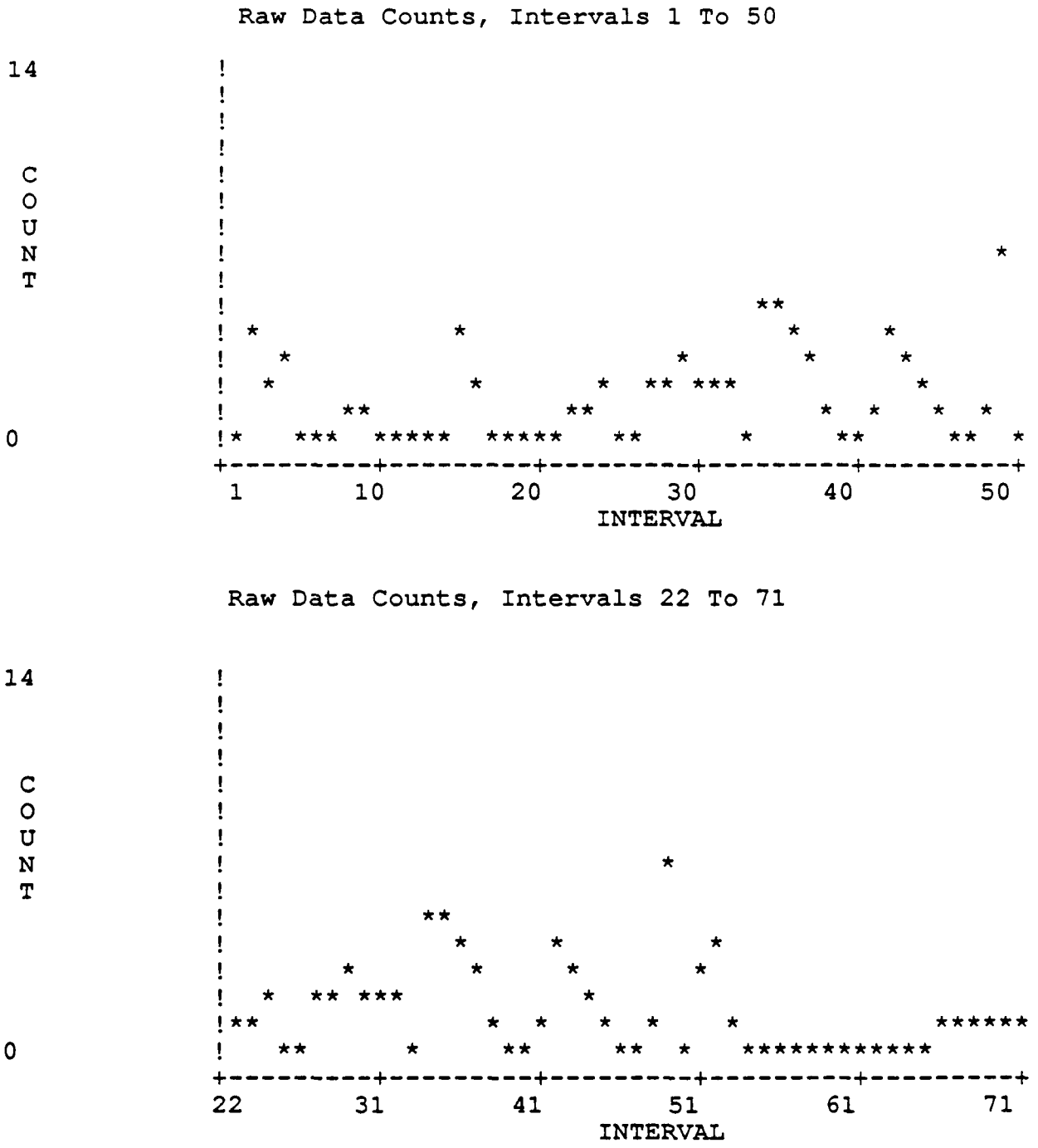


Figure 2.3-1. Interval Count Raw Data.

INTERVAL DATA WITH EQUAL LENGTHS
WITH ERROR COUNTS TOTALING TO 85

MEDIAN OF THE DATA	1	
LOWER & UPPER HINGES	0	2
MINIMUM AND MAXIMUM	0	7
NUMBER OF ENTRIES	71	
AVERAGE OF THE DATA	1.1971831	
STD. DEV. & VARIANCE	1.5731653	2.4748491
SKEWNESS & KURTOSIS	1.4448046	1.7009447

Figure 2.3-2. Interval Count Summary Statistics.

PROJECTED LENGTH OF THE TESTING PERIOD.
20.0

THE EXPECTED NUMBER OF ERRORS IS 15.487747
WITH 95% CONFIDENCE INTERVAL (8.8737008, 22.101793)

PROJECTED LENGTH OF THE TESTING PERIOD.
71.0
THE EXPECTED NUMBER OF ERRORS IS 195.18433
WITH 95% CONFIDENCE INTERVAL (111.83081, 278.53785)

The results show the number of expected errors grow more than geometrically as testing progresses. Longer testing intervals are weighted so that more errors are uncovered. No Chi-Square goodness of fit test was run for this estimator.

Maximum Likelihood Estimate With Selected Weighting

Selected weighting function is given by

Weight = $(x(i))\alpha$, i = 1,2,...,n, x(i) are sample points.

DESIRED ALPHA. 2.0

INITIAL ESTIMATE OF THE TOTAL NUMBER OF ERRORS.
90.0

ML MODEL ESTIMATES, USING THE SELECTED WEIGHTING FUNCTION, AFTER 11 ITERATIONS ARE:

PROPORTIONALITY CONSTANT OF THE MODEL IS .00022170466
WITH 95% CONFIDENCE INTERVAL (-.00012882023, .00057222956)

THE TOTAL NUMBER OF ERRORS IS .259.6439
WITH 95% CONFIDENCE INTERVAL (254.60477, 264.68304)

THE REMAINING NUMBER OF ERRORS IS 175.6439
WITH 95% CONFIDENCE INTERVAL (170.60477E+03, 180.68304)

PROJECTED LENGTH OF THE TESTING PERIOD.
1.0
THE EXPECTED NUMBER OF ERRORS IS .038719368
WITH 95% CONFIDENCE INTERVAL (.022184252, .055254483)

PROJECTED LENGTH OF THE TESTING PERIOD.
10.0
THE EXPECTED NUMBER OF ERRORS IS 3.8719368
WITH 95% CONFIDENCE INTERVAL (2.2184252, 5.5254483)

PROJECTED LENGTH OF THE TESTING PERIOD.
20.0

```
THE EXPECTED NUMBER OF ERRORS IS  15.487747
 WITH 95% CONFIDENCE INTERVAL (8.8737008, 22.101793)

PROJECTED LENGTH OF THE TESTING PERIOD.
                71.0
THE EXPECTED NUMBER OF ERRORS IS  195.18433
 WITH 95% CONFIDENCE INTERVAL (111.83081, 278.53785)
```

Figure 2.3-3 presents plots of the GPOMOD ML estimator, using the weighting function $(x(i))\alpha$, with the test data. Figure 2.3-4 shows the residual plots for this model. Each interval in the figures represents five days of test data. No Chi-Square goodness of fit test was run for this case.

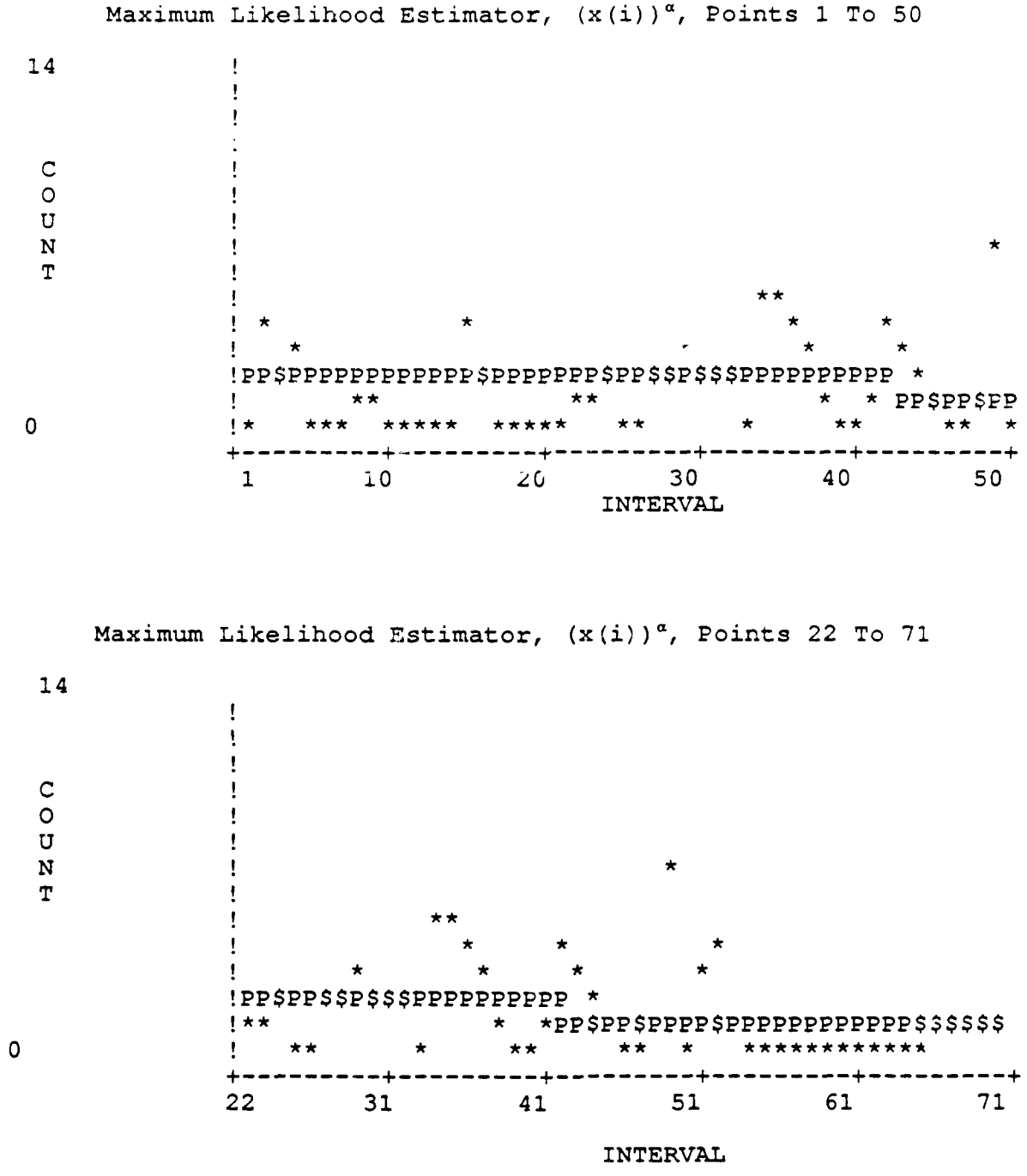
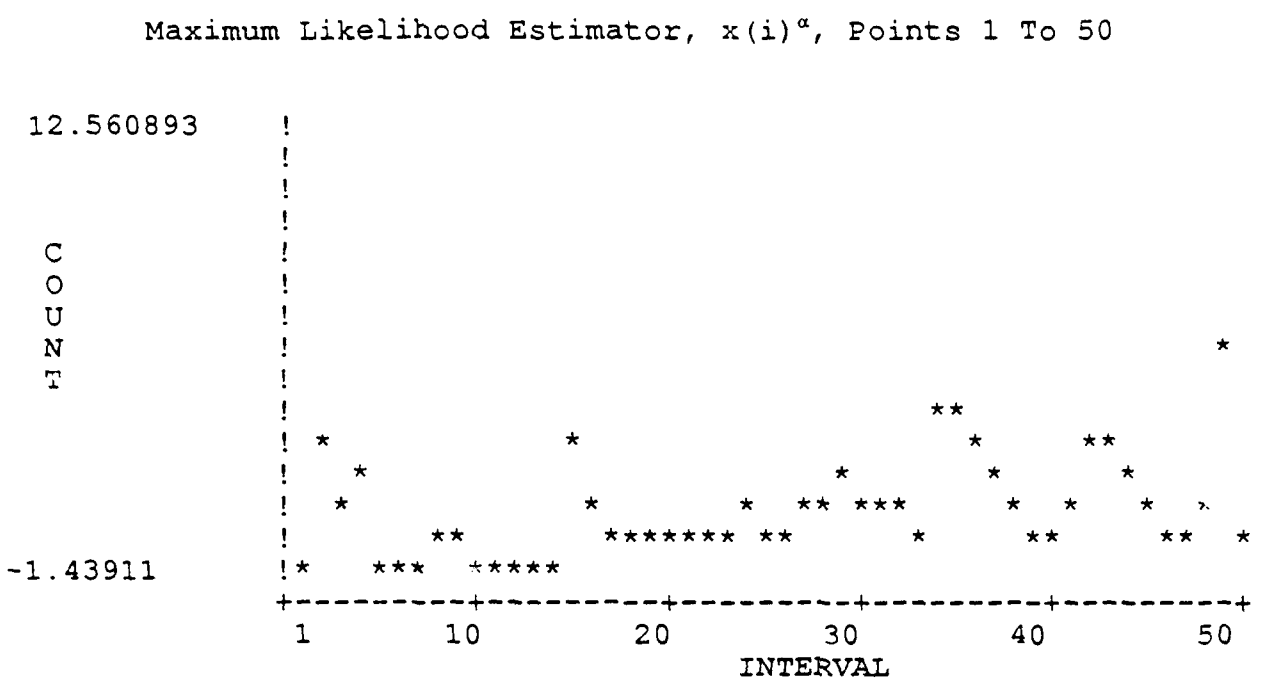


Figure 2.3-3. Poisson Model Maximum Likelihood Interval Count Prediction, with Selected Alpha, Versus Test Data.



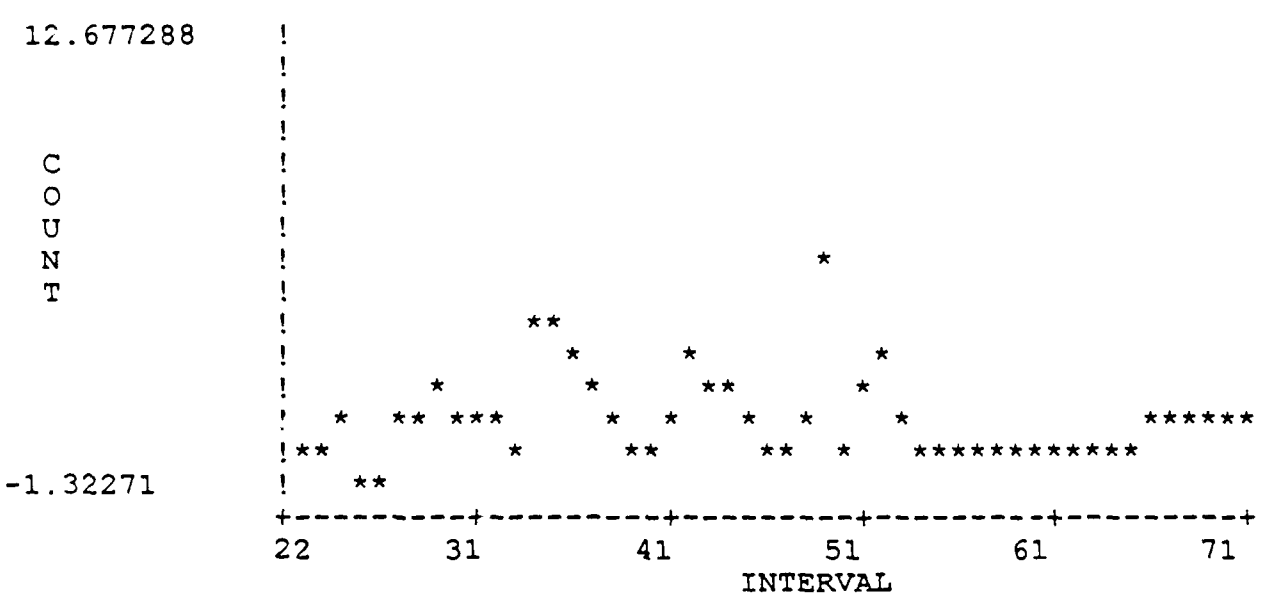


Figure 2.3-4. Poisson Model Residuals Maximum Likelihood with Selected Alpha.

Maximum Likelihood Estimate With Estimated Weighting

```
INITIAL ESTIMATE FOR ALPHA.
          1.5
INITIAL ESTIMATE OF THE TOTAL NUMBER OF ERRORS.
         90.0
```

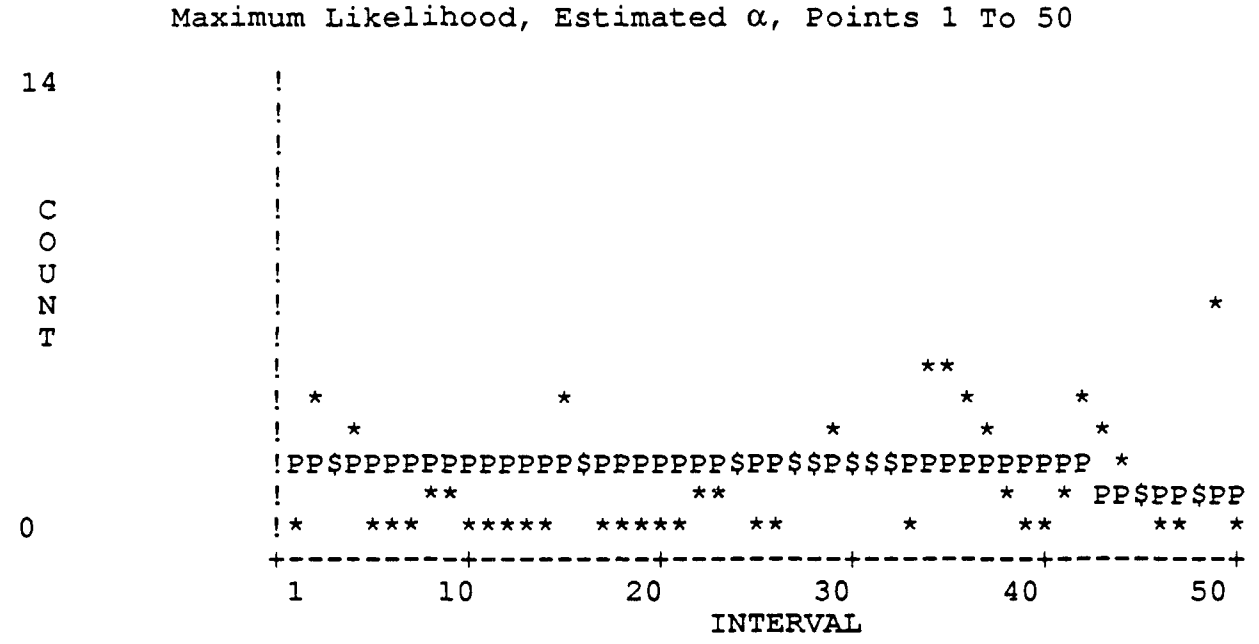
ML MODEL ESTIMATES, USING ESTIMATED WEIGHTING, AFTER 36 ITERATIONS ARE:

```
PROPORTIONALITY CONSTANT OF THE MODEL IS  5017.0868

THE TOTAL NUMBER OF ERRORS IS             259.6439

THE REMAINING NUMBER OF ERRORS IS         175.6439

AND ALPHA IS                              2.9232488

PROJECTED LENGTH OF THE TESTING PERIOD.
            1.0
THE EXPECTED NUMBER OF ERRORS IS  .0087620361

PROJECTED LENGTH OF THE TESTING PERIOD.
           10.0
THE EXPECTED NUMBER OF ERRORS IS  7.3426671

PROJECTED LENGTH OF THE TESTING PERIOD.
           20.0
THE EXPECTED NUMBER OF ERRORS IS  55.697976

PROJECTED LENGTH OF THE TESTING PERIOD.
           71.0
THE EXPECTED NUMBER OF ERRORS IS  2260.9647
```

THE CHI-SQUARE STATISTIC IS 65.764831 WITH 67 DEGREES OF FREEDOM.

The hypothesis that the estimator fits the data would be accepted with $\alpha = 0.05$ (critical region > 88).

Figure 2.3-5 presents the plots of the estimator along with the actual data. Figure 2.3-6 shows the residuals for this model.



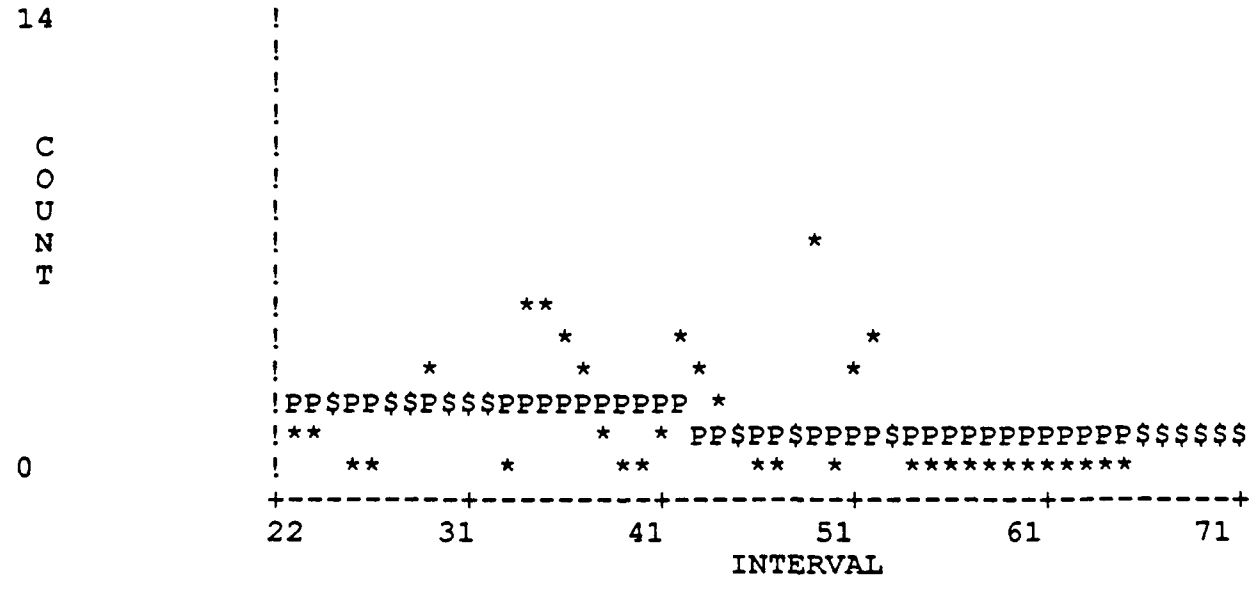


Figure 2.3-5. Poisson Model Maximum Likelihood Interval Count Prediction, with Estimated Alpha, Versus Test Data.

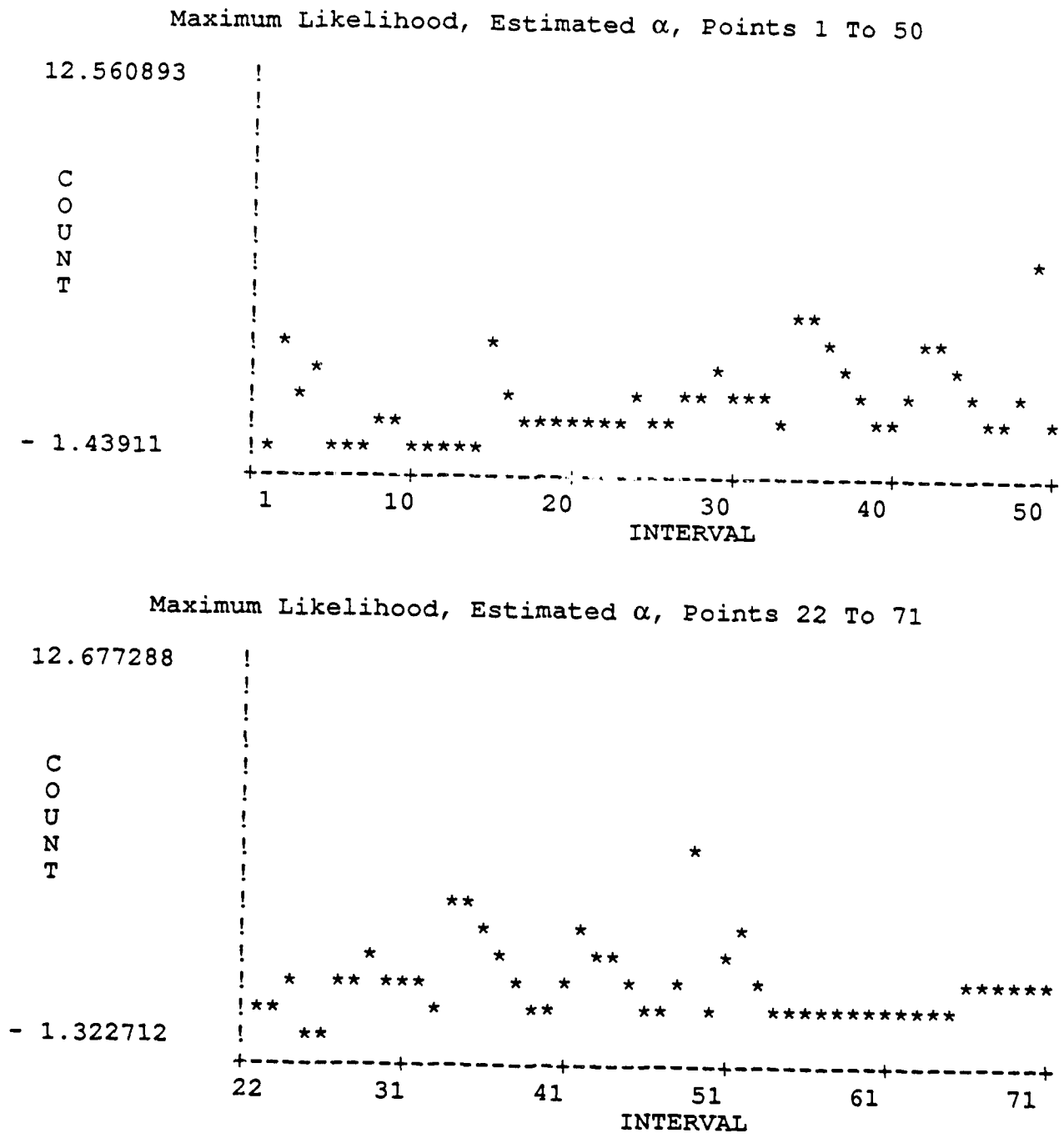


Figure 2.3-6. Poisson Model Residuals, Maximum Likelihood, with Estimated Alpha.

Least Squares Estimate, Schick-Wolverton Weighting

INITIAL ESTIMATE OF THE TOTAL NUMBER OF ERRORS.
90.0

LS MODEL ESTIMATES, USING SCHICK-WOLVERTON WEIGHTING, AFTER 2 ITERATIONS ARE:

PROPORTIONALITY CONSTANT OF THE MODEL IS .00032167169

THE TOTAL NUMBER OF ERRORS IS 341.38826

THE REMAINING NUMBER OF ERRORS IS 257.38826

PROJECTED LENGTH OF THE TESTING PERIOD.
1.0
THE EXPECTED NUMBER OF ERRORS IS .041236422

PROJECTED LENGTH OF THE TESTING PERIOD.
10.0
THE EXPECTED NUMBER OF ERRORS IS 4.1236422

PROJECTED LENGTH OF THE TESTING PERIOD.
20.0
THE EXPECTED NUMBER OF ERRORS IS 16.494569

PROJECTED LENGTH OF THE TESTING PERIOD.
71.0
THE EXPECTED NUMBER OF ERRORS IS 207.87280

THE CHI-SQUARE STATISTIC IS 65.379035 WITH 68 DEGREES OF FREEDOM.

The hypothesis that the estimator fits the data would be accepted with $\alpha = 0.05$. Figure 2.3-7 presents the plots of the Generalized Poisson model estimator, with Schick-Wolverton weighting, along with the actual data. Figure 2.3-8 shows the residual plots for this estimator.

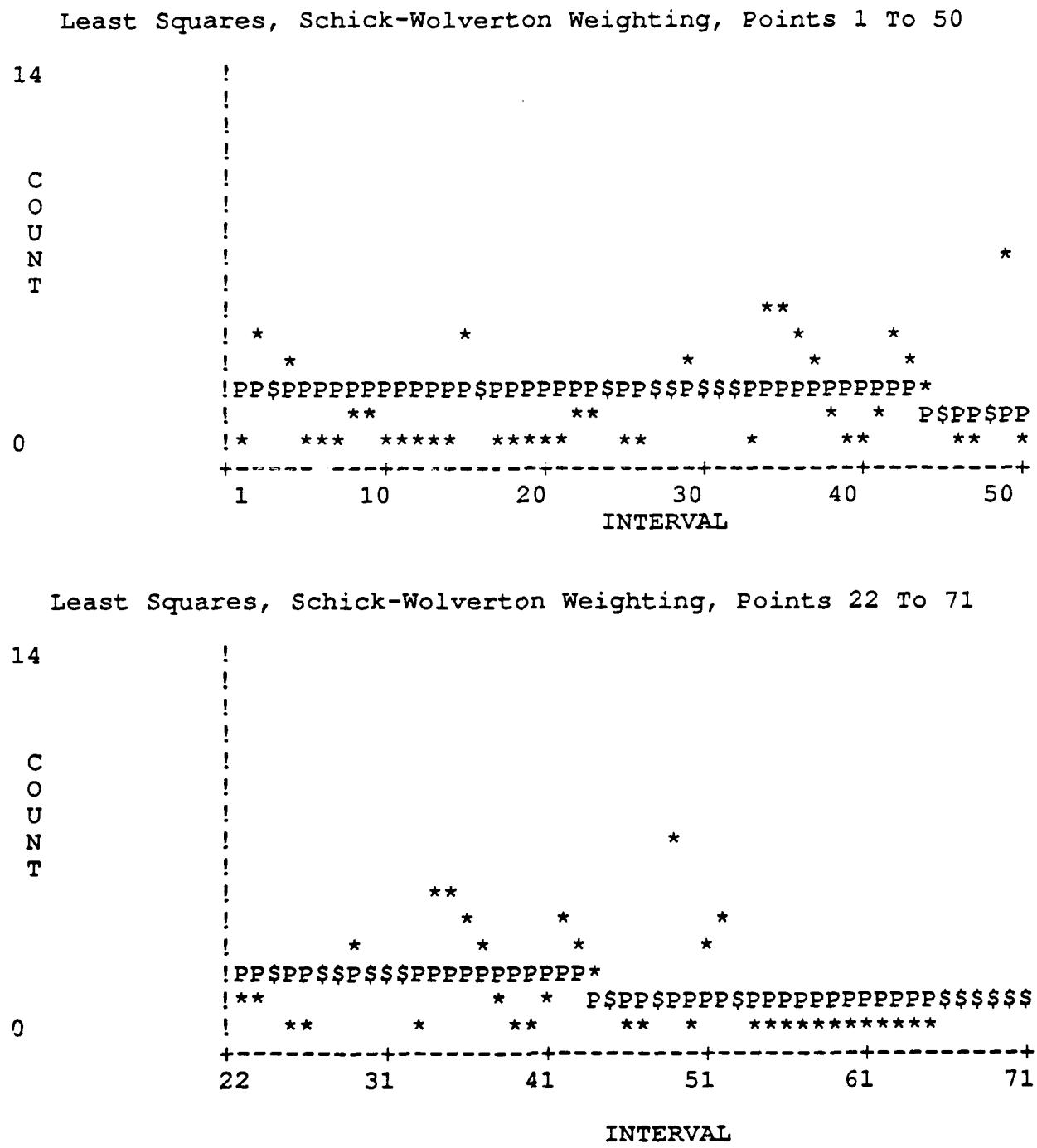


Figure 2.3-7. Poisson Model Least Squares Interval Count Prediction, with Schick-Wolverton Weighting, Versus Test Data.

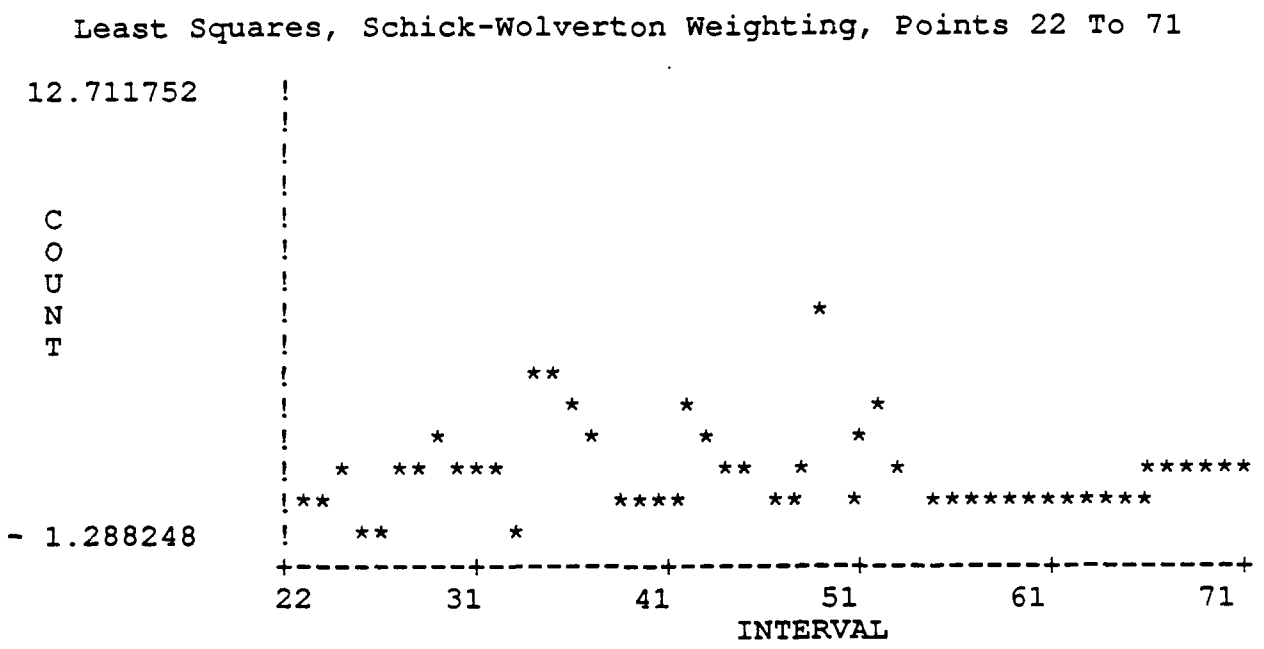


Figure 2.3-8. Poisson Model Residuals, Least Squares, with Schick-Wolverton Weighting.

Least Squares Estimate With Selected Alpha Weighting

DESIRED ALPHA. 2.0

INITIAL ESTIMATE OF THE TOTAL NUMBER OF ERRORS.
90.0

LS MODEL ESTIMATES, USING SELECTED WEIGHTING, AFTER 2 ITERATIONS ARE:

PROPORTIONALITY CONSTANT OF THE MODEL IS 160.83585

THE TOTAL NUMBER OF ERRORS IS 341.38826

THE REMAINING NUMBER OF ERRORS IS 257.38826

PROJECTED LENGTH OF THE TESTING PERIOD.
1.0
THE EXPECTED NUMBER OF ERRORS IS .041236422

PROJECTED LENGTH OF THE TESTING PERIOD.
10.0
THE EXPECTED NUMBER OF ERRORS IS 4.1236422

PROJECTED LENGTH OF THE TESTING PERIOD.
20.0
THE EXPECTED NUMBER OF ERRORS IS 16.494569

PROJECTED LENGTH OF THE TESTING PERIOD.
71.0
THE EXPECTED NUMBER OF ERRORS IS 207.87280

THE CHI-SQUARE STATISTIC IS 65.379035 WITH 68 DEGREES OF FREEDOM.

The hypothesis that the estimator fits the data would be accepted, with $\alpha = 0.05$. No plots of this estimator were made, but the plots do not differ greatly from previous figures.

The least squares estimate with estimated α did not run successfully.

The plots of the data versus the estimators derived through the use of the GPOMOD clearly show the model fits the data. This conclusion is supported by the Chi-Square goodness of fit tests, which confirm that the models fit the data. The maximum likelihood estimators and the least squares estimators do not concur on the number of errors remaining in the software, nor on total errors in the software. The number of errors expected in specified time intervals is fairly close.

Comparison of the results from the model runs with the rough estimator developed in Section 1 (4.2 days/failure or 0.239 failures/day) highlights the assumptions under which this model runs. The rough estimate and the GPOMOD results are fairly close for small test intervals. They differ significantly for large test intervals. The model assumption that more software errors are detected as the test time interval increases is discernable through this comparison. The validity of that assumption, however, is questionable. Regardless, under the conditions of this MSE test case (fixed five daytime intervals), the GPOMOD provides good estimators of the test data.

To use the Chi-Square test for goodness of fit with this model, it was necessary to combine cells, because every expected cell count was less than five.

2.4 SCHNEIDEWIND MODEL RESULTS

The SDWMOD uses interval count data to estimate reliability. The model assumes the process of error detection follows a non-homogeneous Poisson process (NHPP) where the error detection rate decreases exponentially.

The SDWMOD also assumes that recent error counts are more useful than earlier ones. To implement this, there are three possible "treatments" under this model:

a. Use all error counts from all testing intervals.

b. Use only error counts from recent intervals.

c. Use a cumulative error count for earlier test intervals and individual error counts for the most recent test intervals.

The model assumes the mean number of errors for the ith test interval is given as:

$$\mu(i) = \alpha\ (\exp(-\beta(i-1)) - \exp(-\beta(i)))/\beta.$$

The model estimates the parameters α and β from the data.

Treatment 1

INITIAL ESTIMATE FOR THE PARAMETER BETA
1.0

TREATMENT 1 MODEL ESTIMATES AFTER 85 ITERATIONS ARE:

BETA .0034769164

ALPHA 1.3510256

AND THE WEIGHTED SUMS-OF-SQUARES BETWEEN THE PREDICTED AND OBSERVED ERROR COUNTS IS 195.87153

ESTIMATE OF THE NUMBER OF ERRORS EXPECTED IN THE NEXT TESTING PERIOD IS 1.0536549

ESTIMATE OF THE NUMBER OF TESTING PERIODS NEEDED TO DISCOVER THE NEXT 5.0 ERRORS IS 4.7765922

ESTIMATE OF THE NUMBER OF TESTING PERIODS NEEDED TO DISCOVER THE NEXT 10.0 ERRORS IS 9.6338549

ESTIMATE OF THE NUMBER OF TESTING PERIODS NEEDED TO DISCOVER THE NEXT 20.0 ERRORS IS 19.601623

THE CHI-SQUARE STATISTIC IS 44.406500 WITH 68 DEGREES OF FREEDOM.

The hypothesis that the model fits the data is accepted, with $\alpha = 0.005$. Figure 2.4-1 presents the plots of the model along with the actual data. Figure 2.4-2 shows the residual plot.

INITIAL ESTIMATE FOR THE PARAMETER BETA
3.650000000000000E-007

TREATMENT 1 MODEL ESTIMATES AFTER 1 ITERATIONS ARE:

BETA .00000043903637

ALPHA 1.1972018

AND THE WEIGHTED SUMS-OF-SQUARES BETWEEN THE PREDICTED AND OBSERVED ERROR COUNTS IS 173.24204

ESTIMATE OF THE NUMBER OF ERRORS EXPECTED IN THE NEXT TESTING PERIOD IS 1.1971642

ESTIMATE OF THE NUMBER OF TESTING PERIODS NEEDED TO DISCOVER THE NEXT 5.0 ERRORS IS 4.1765395

ESTIMATE OF THE NUMBER OF TESTING PERIODS NEEDED TO DISCOVER THE NEXT 10.0 ERRORS IS 8.3530867

ESTIMATE OF THE NUMBER OF TESTING PERIODS NEEDED TO DISCOVER THE NEXT 20.0 ERRORS IS 16.706204

THE CHI-SQUARE STATISTIC IS 38.790525 WITH 68 DEGREES OF FREEDOM.

The hypothesis that the model fits the data is accepted, with $\alpha = 0.005$. Figure 2.4-3 presents plots of the estimator along with the actual data. Figure 2.4-4 shows the residual.

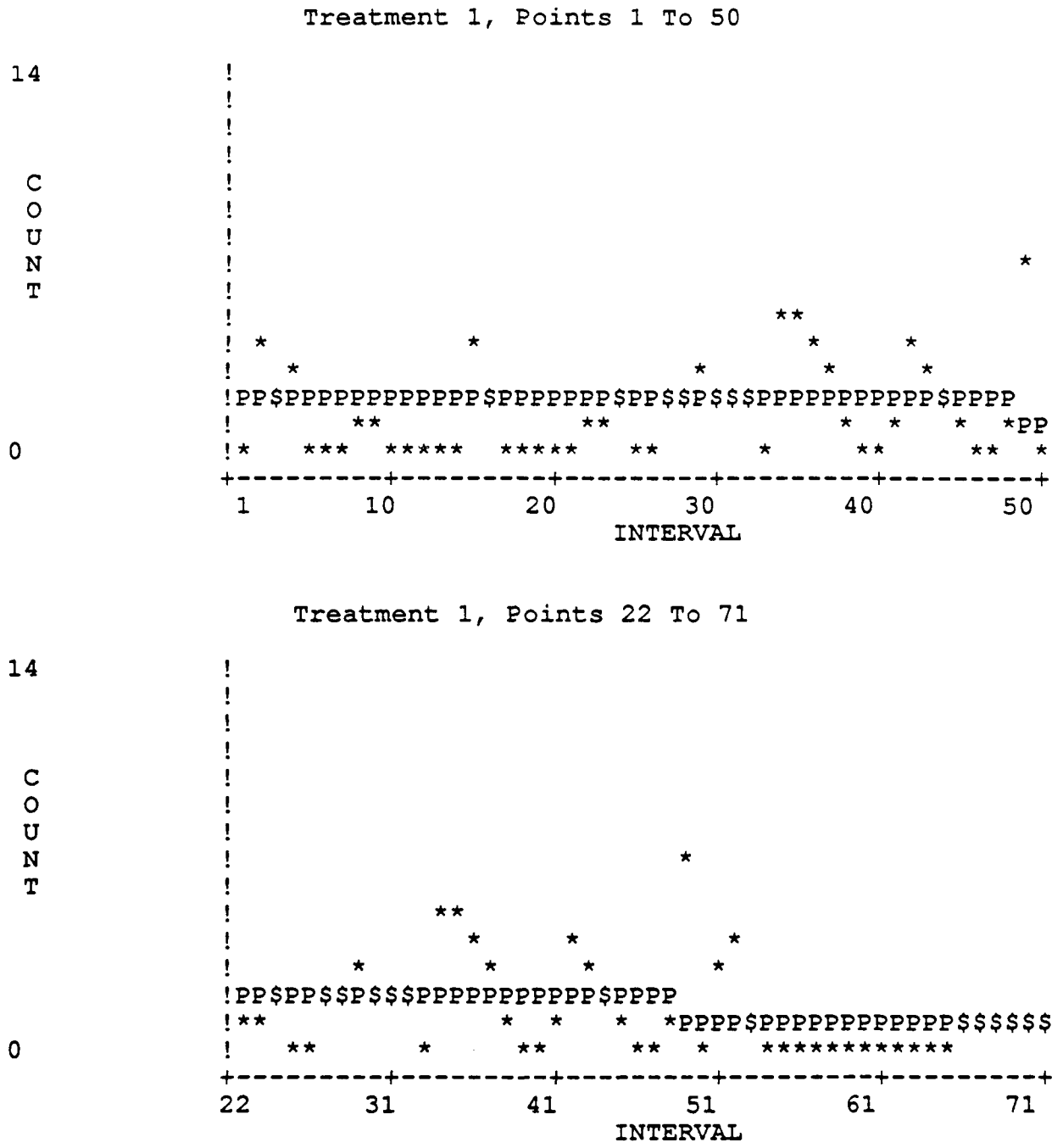
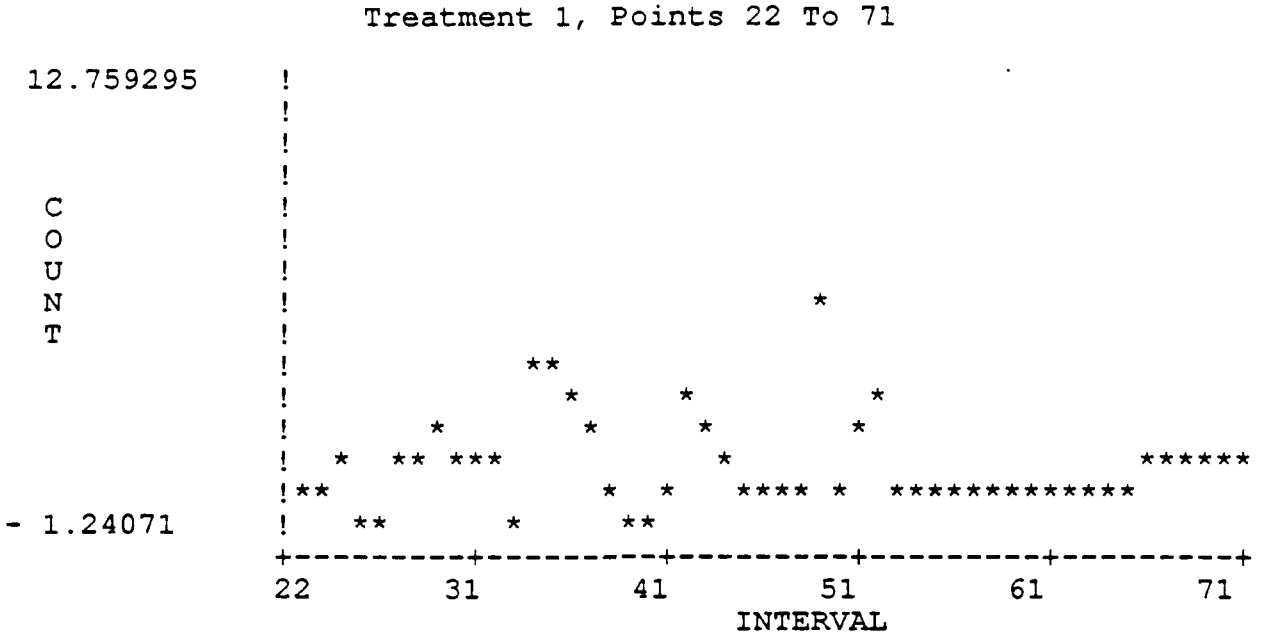


Figure 2.4-1. Schneidewind Treatment 1 Estimator Versus Actual Data.



A second run was made using Treatment 1, with a closer initial estimate of the parameter ß.

Figure 2.4-2. Schneidewind Treatment 1 Residual.

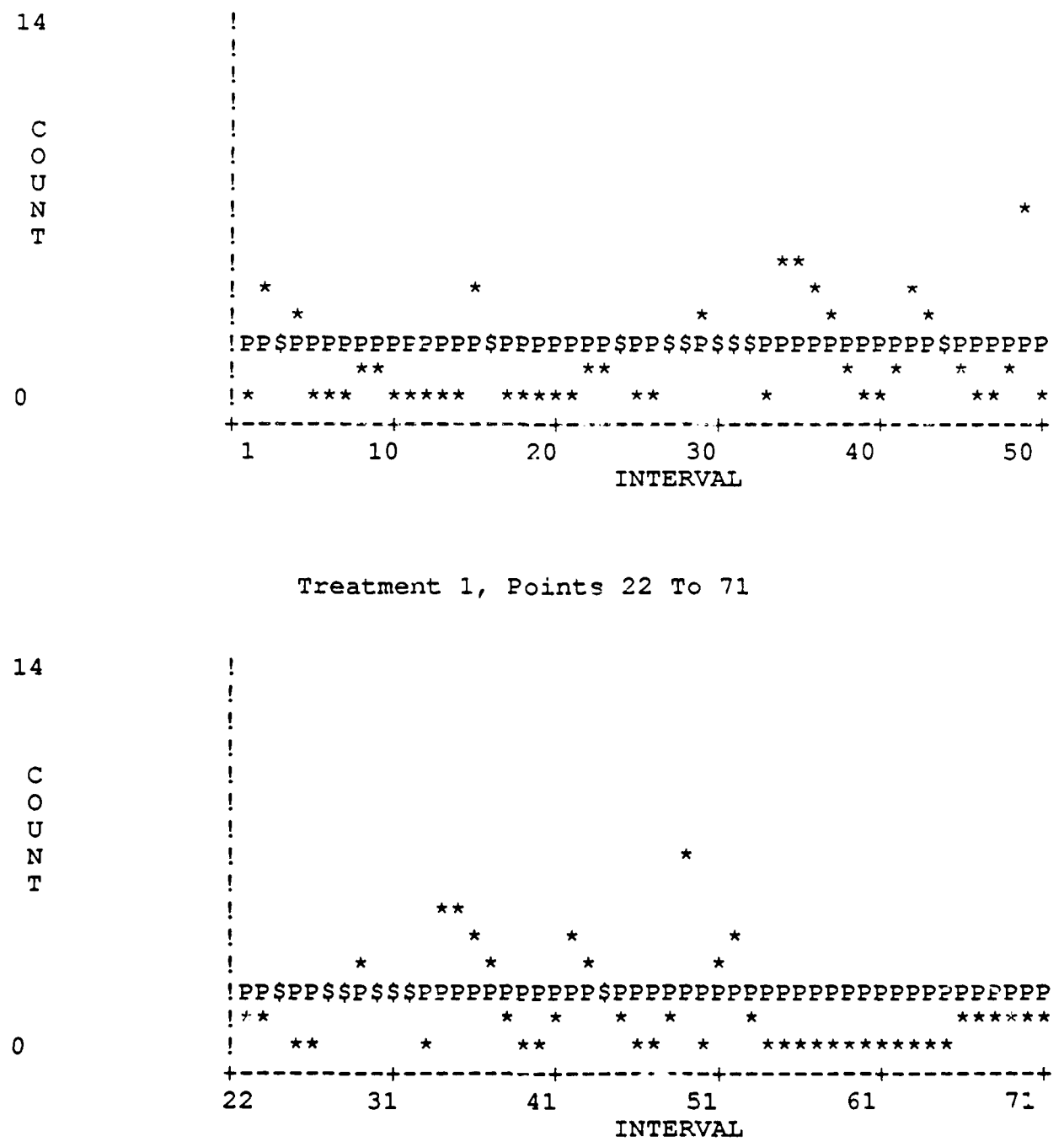


Figure 2.4-3. Schneidewind Treatment 1 Estimator, with Improved Initial Estimate for Beta, Versus Test Data.

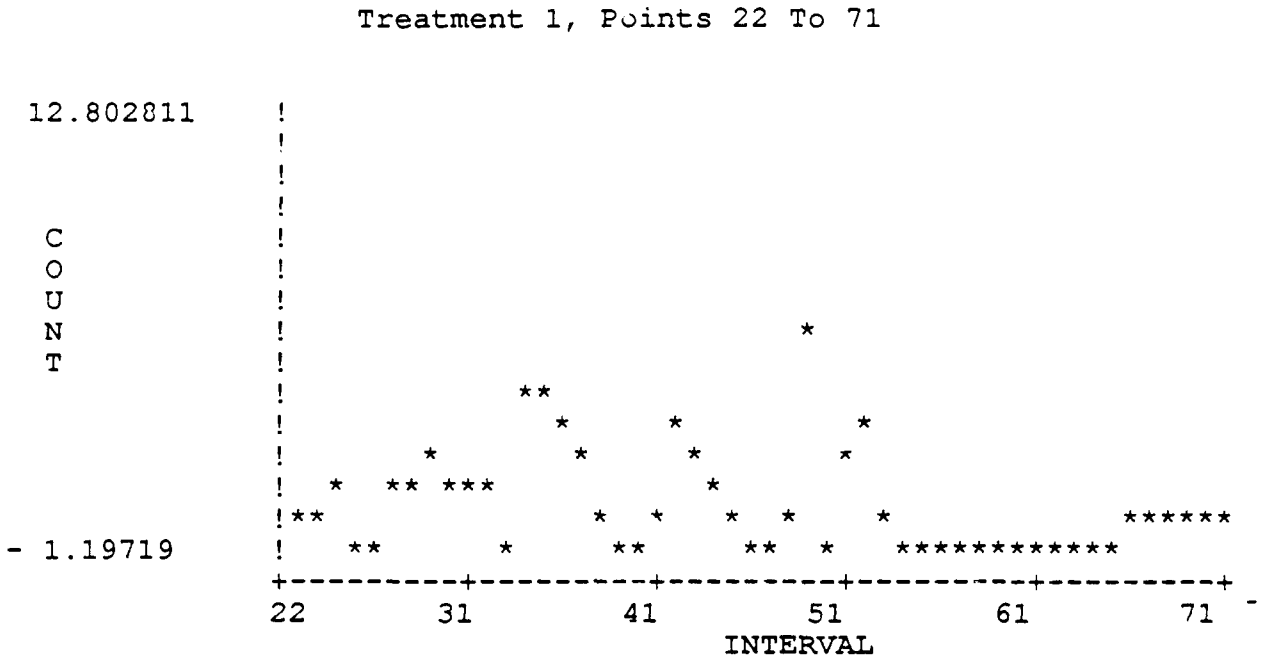


Figure 2.4-4. Schneidewind Treatment 1 Residual Improved Beta Estimate.

Treatments 2 and 3 did not provide as close a fit as Treatment 1, according to the Chi-Square goodness of fit test and the sum of squares indicator. The second run of treatment 1 of this model provided the smallest Chi-Square value (by half) of any derived for this report.

The second run estimate of 1.2 failures per 5-day period agrees very closely with the rough estimate of 1.195 failures per 5-day period. It agrees with the Bayesian and Geometric model results.

2.5 ANALYSIS AND CONCLUSIONS

Each of the four models tested are applicable and fit the data. All model runs presented fit the data acceptably, according to Chi-Square tests and examination of plots of predicted versus actual values. The SDWMOD (Treatment 1) fit the data better than any other model. Of the remaining models in the SMERFS package, NHPP model and the BAMMOD would not converge to an estimate for this data. The other models use only CPU time between failure data. Addition of procedures to collect CPU time data would greatly enhance the use of SMERFS.

The results presented above show that useful quantities, such as the number of software errors remaining, or the number of testing intervals required to eliminate software errors, can be calculated using the SMERFS package. Most important, reliability values can be determined using the package. MTBF data is calculated by all of the above models, either directly or indirectly. The MTBF estimate provided by this package can be used to determine software component reliability, given knowledge about the type and quality of software maintenance which the software will undergo.

Using this reliability value, with hardware reliability values and system configuration information, software and system reliability figures can be included in test reports. This satisfies the requirement of DoDD 5000.3-M-3 (reference 10).

SECTION 3. APPENDICES

This Page Is Intentionally Left Blank

APPENDIX A

METHODOLOGY INVESTIGATION PROPOSAL

A-1. TITLE. Software Maturity Model Validation.

A-2. CATEGORY. All Department of the Army (DA) mission areas for systems containing embedded computer resources (ECR) are supported.

A-3. INSTALLATION OR FIELD OPERATING ACTIVITY. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613-7110.

A-4. PRINCIPAL INVESTIGATOR. Mr. K. Van Karsen, Software and Interoperability Division, STEEP-ET-DS, AUTOVON 879-02090/2092.

A-5. STATEMENT OF THE PROBLEM. Essentially all systems being developed employ some use of computers and software. Unlike hardware, the metrics for "software RAM" are ill-defined. Hereafter, "maturity" will be used instead of RAM. DoD Directive 5000.3 requires a quantitative measure of software maturity; DT evaluators and testers are required to develop methods for determining software maturity. TECOM cannot quantitatively measure the maturity of the software embedded in computer driven systems.

A-6. BACKGROUND. A number of models have been developed to predict software maturity. However, none have been validated. Under TECOM project number 7-CO-RD9-EP1-004, USAEPG derived the SPPA, a mathematical model which provided estimates of software maturity. Unlike previous models, the SPPA took into consideration the repair process, wherein repairs need not be made directly after encountering a fault (bug). The SPPA model was used on data from Lipow and from the Position Location Reporting System (PLRS) project. A final report was submitted to Headquarters (HQ) TECOM and subsequently approved for distribution. Subsequent to the development of SPPA, new models have appeared, but have not been evaluated for applicability to the developmental testing environment. Also, some researchers have developed an integrated package of various models with the intent that one or more of the models would be appropriate for a given situation. USAEPG has acquired a government-owned package, courtesy of NSWC, containing eight different models. However, the suitability of these models with respect to the availability of required data has not been determined.

A-7. GOAL. To establish an accepted method for assessing the maturity of the software in ECR.

A-8. DESCRIPTION OF INVESTIGATION.

a. Summary. USAEPG will evaluate currently available software maturity models and propose the best for use in TECOM software testing.

b. Detailed Approach. USEPG will:

(1) Phase I - First Year's Effort:

(a) Identify software maturity (reliability) models available from industry, academia, and government agencies.

(b) Examine the data requirements of the various models with respect to the data available during DT.

(2) Phase II - Second Year's Effort:

(a) Select a set of available models which meet the constraints imposed by data availability.

(b) Consult with cognizant individuals on the applicability of the candidate models to TECOM's test and evaluation mission, and select a final set of models for use during DT.

(c) Demonstrate the recommended methods by applying data from a selected tactical system, and evaluate the results.

c. Final Product(s).

(1) Phase I:

(a) A set of initial candidate software maturity models.

(b) Evaluation of the data requirements for application to DT.

(2) Phase II:

Recommendations for a set of models to determine software maturity during DT.

d. Coordination. Coordination with TECOM activities will be accomplished through the TECOM Software Technical Committee

(TSOTEC). Coordination with other organizations will be performed directly.

e. Environmental Impact Statement. Execution of this task will not have an adverse impact on the quality of the environment.

f. Health Hazard Statement. Execution of this task will not involve health hazards to personnel.

A-9. JUSTIFICATION.

a. Association with Mission. One of TECOM's missions is to perform developmental tests on ECR. The investigation is needed to advance the concept of software maturity. The Army Science Board report on testing of electronic systems, with emphasis on software intensive systems, advocates RAM (maturity) programs as essential.

b. Association with Methodology/Instrumentation Program. This project supports thrusts of the TECOM Methodology Program to improve the quality of testing as well as the test process. Instrumentation developed or acquired previously would be used to form the basis of instrumentation required by the methodology.

c. Present Capability, Limitations, Improvement and Impact on Testing if Not Approved.

(1) Present Capability. The current test capability provides information in the form of TIRs, for assessing software maturity.

(2) Limitations. Appropriate maturity models have not been identified and validated for application to DT, even though some raw data (TIRs) are available for analysis. Most prior attempts to assess maturity have avoided the lack of a validated model by using rather crude methods. For example, maturity per DoD-STD-1679A is determined on the basis of the number and severity of unresolved software errors at the time of acceptance. The number of latent faults which may surface after deployment is not estimated.

(3) Improvement. USAEPG and other organizations have developed software maturity models which may be suitable for DT use. Identification and validation of a model which will work within the DT environment will greatly improve estimated maturity quality.

(4) Impact on Testing if Not Approved. The intent of DoDD 5000.3 is not met unless a quantitative means of evaluating maturity is provided. Reporting maturity as the amount of discovered faults, while ignoring latent faults, results in a

distorted view of actual maturity, given the current test techniques.

(5) Dollar Savings. No dollar savings can be assessed at this time. The potential of this project is that a quantitative measure of software maturity can be attained; provide insight to Program Manager's (PM's) and evaluator's as to the maturity of a given software system to prevent fielding of an immature system and the inherent high cost to fix once fielded.

(6) Workload. Over the past 5 years, USAEPG has experienced 17 tests requiring an evaluation of software maturity.

Examples of items anticipated for testing include:

Test Item	Fiscal Year (FY)	88	89	90
MSE	(Mobile Subscriber Equipment)	X	X	X
JTIDS	(Joint Tactical Information Distribution System)	X		
MCS	(Maneuver Control System)	X	X	X
VISTA	(Very Intelligent Surveillance and Target Acquisition)	X	X	X
FADDC^{2}I		X	X	X
JINTACCS	(Joint Interoperability of Tactical Command and Control Systems	X	X	X
EPLRS	(formerly PJH) (Enhanced Position Location Reporting System)	X	X	X
GPS	(Global Positioning System)	X		
ASAS	(All Source Analysis System)	X		
AFATDS	(Advanced Field Artillery Tactical Data System)	X	X	X

(7) Association with Requirements Documents. DoD Directive 5000.3 requires a quantitative measure of software maturity for each development phase. To date, there are no accepted measuring schemes.

(8) Others. N/A.

A-10. RESOURCES

a. Financial.

Dollars (Thousands)

	FY88		FY89	
	In-House	Out-of-House	In-House	Out-of-House
Personnel Compensation	10.0		12.0	
Travel	2.0		3.0	
Contractual Support		52.0		45.0
Consultants & Other Svcs				
Materials & Supplies		1.0		5.0
Equipment				
General & Admin costs				
Subtotals	12.0	53.0	15.0	50.0
FY Totals	65.0		65.0	

b. Explanation of Cost Categories.

(1) Personnel Compensation. This cost represents compensation chargeable to the investigation for using technical or other civilian personnel assigned to the investigation.

(2) Travel. This represents cost incurred while visiting government and industry facilities.

(3) Contractual Support. Performance of the investigation will be accomplished with resources provided under an existing support contract.

(4) Consultants and Other Services. N/A.

(5) Material and Supplies. N/A.

(6) Equipment. N/A.

(7) General and Administrative Costs. N/A.

c. Obligation Plan.

					FY88
Fiscal Quarter (FQ)	1	2	3	4	TOTAL
Obligation Rate (Thousands)	50.0	5.0	5.0	5.0	65.0

d. In-House Personnel.

(1) In-House Personnel Requirements by Specialty.

	Number	Man-hours FY88 Only Required	Available	Total Required
Elect Engr, GS-0855	1	450	450	450

(2) Resolution of Non-Available Personnel. N/A

A-11. INVESTIGATION SCHEDULE

	FY88												FY89											
	O	N	D	J	F	M	A	M	J	J	A	S	O	N	D	J	F	M	A	M	J	J	A	S
In-House	-	.	-	.	-	.	-	.	-	.	-	I	-	.	-	.	-	.	-	.	-	.	-	R
Contracts Consultants	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Symbols: ---- Active investigation work (all categories)

.... Contract monitoring (in-house only)

I Interim Report

R Final report due at HQ, TECOM

A-12. ASSOCIATION WITH TOP PROGRAM. TECOM Test Operations Procedure (TOP) 1-1-056, Software Testing, requires the assessment of software maturity. The results of this investigation may provide recommended changes to TOP 1-1-056 with regards to software maturity.

FOR THE COMMANDER:

ROBERT E. REINER
Chief, Modernization and
Advanced Concepts Division

This Page Is Intentionally Left Blank

APPENDIX B

REFERENCES

1. Software Reliability, Leone, A. M., Westinghouse Electric Corporation, Glen Burnie, Maryland, November 1988.

2. Software Reliability: Measurement, Prediction, Application, Musa, J. D., et al., McGraw-Hill, Inc., 1987.

3. RADC-TR-87-171, Methodology for Software Reliability Prediction, Volumes I-II, McCall, J., et al., November 1987.

4. RADC-TR-85-37, Volumes I-III, Specification of Software Quality Attributes, Bowen, T.P., et al., February 1985.

5. An Interactive Program for Software Reliability Modeling, Farr, W. H., Smith, O. D., Naval Surface Warfare Center. Proceedings of the 9th Annual Software Engineering Workshop, NASA Goddard, SEL-84-0004, Maryland, 1984.

6. NSWC TR 82171, A Survey of Software Reliability Modeling and Estimation, Farr, W. H., September 1983.

7. NASA Contractor Report 4187, Quality Measures and Assurance for AI Software, Rushby, J., October 1988.

8. DoDD 5000.3, "Test and Evaluation," 1986.

9. DoD 5000.3-M-1, "Software Test and Evaluation Manual", 1987.

This Page Is Intentionally Left Blank

APPENDIX C

ACRONYMS AND ABBREVIATIONS

ADMIN	Administrative
AFATDS.....	Advanced Field Artillery Tactical Data System
AFOTECP.....	Air Force Operational Test and Evaluation Center Pamphlet
AMC-P.....	Army Material Command - Pamphlet
ASAS.....	All Source Analysis System
AUTOVON.....	Automatic Voice Network
BAMMOD.....	Brooks and Motley's Model
C.....	Compression Factor
C^3I.....	Command, Control, Communications, and Intelligence
COMRAM.....	Communications Reliability and Maintainability
CPU.....	Central Processing Unit
CTC.....	Calendar Time Component
CSOLOP.....	Circuit Switching On-Line Operational Program
DA.....	Department of the Army
DATINP.....	Data Input
DEV.....	Deviation
DoD.....	Department of Defense
DoDD.....	Department of Defense Directive
DSVT.....	Digital Subscriber Voice Terminal
DT.....	Developmental Testing
ECR.....	Embedded Computer Resources
ELECT.....	Electrical
ENGR.....	Engineer
EPLRS.....	Enhanced Position Location Reporting System
EXP.....	Exponential
FQ.....	Fiscal Quarter
GEOMOD.....	Geometric Model
GPOMOD.....	Generalized Poisson Model
FY.....	Fiscal Year
GPS.....	Global Positioning System
HQ.....	Headquarters
IBM.....	International Business Machines Corporation
JINTACCS.....	Joint Interoperability of Tactical Command and Control Systems
JTIDS.....	Joint Tactical Information Distribution System
LAVMOD.....	Littlewood and Verrall Model
LEN.....	Large Extension Node
LS.....	Least Squares
M.....	Maximum Number of Iterations
MCS.....	Maneuver Control System
ML.....	Maximum Likelihood
MSE.....	Mobile Subscriber Equipment
MSRT.....	Mobile Subscriber Radio Terminal

MTBF	Mean Time Between Failure
MTTF	Mean Time To Failure
MUSMOD	Musa Model
NCS	Node Center Switch
NHPP	Non-Homogeneous Poisson Process
NSWC	Naval Surface Warfare Center
NPIMOD	NHPP Interval Count Model
NPTMOD	NHPP Time Model
OCC	Occurrence
OTEA	Operational Test and Evaluation Agency
PDL	Program Design Language
PDSS	Post Deployment Software Support
PJH	PLRS Joint Hybrid
PLRS	Position Location Reporting System
PM	Program Manager
RADC	Rome Air Development Center
RAM	Reliability, Availability, and Maintainability
RAU	Radio Access Unit
SCC	System Control Center
SDWMOD	Norman Schneidewind's Model
SEN	Small Extension Node
SMERFS	Statistical Modeling and Estimation of Reliability Functions for Software
SPPA	Software Performance Parameter Assessment
STD	Standard
SVCS	Services
TECOM	Test and Evaluation Command
TEMP	Test and Evaluation Master Plan
TFCS	Trident-I Fire Control System
TIR	Test Incident Report
TOP	Test Operations Procedure
TR	Technical Report
TSOTEC	TECOM Software Technical Committee
USA	United States Army
USAEPG	United States Army Electronic Proving Ground
VISTA	Very Intelligent Surveillance and Target Acquisition

APPENDIX D

SMERFS MODELS

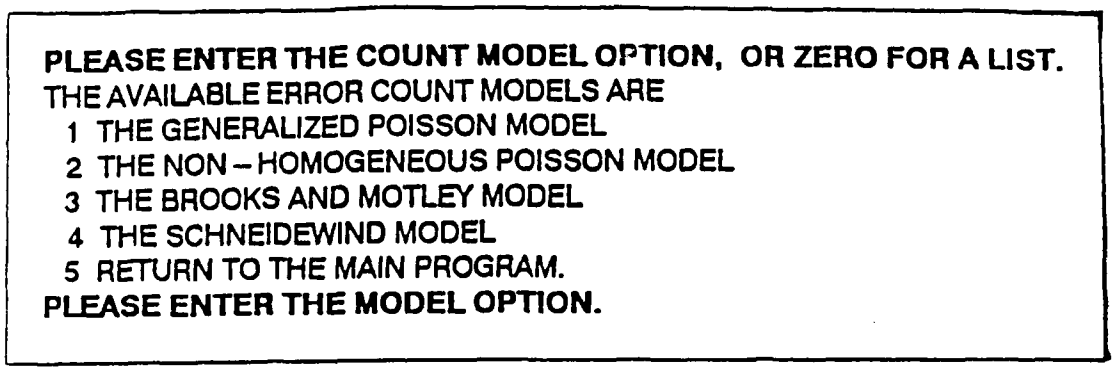
D-1. GENERAL DISCUSSION.

A set of software reliability models for use in estimating software maturity is described below. The eight models are contained in the SMERFS interactive software reliability estimation package. Four of these models are time between failure models and four are error count models. The following information is provided for each model: model description, model assumptions, model inputs, model outputs. For more detailed information on the various prompts and options provided by these models, consult the SMERFS User's Guide and Farr's Survey of Software Reliability Modeling and Estimation. The time between and error count models are invoked by an execution time data model menu and an interval data model menu, respectively (Figures D-1.1 and D-1.2).

PLEASE ENTER THE TIME MODEL OPTION, OR ZERO FOR A LIST.
THE AVAILABLE WALL CLOCK OR CPU TIME MODELS ARE
1 THE LITTLEWOOD AND VERRALL BAYESIAN MODEL
2 THE MUSA EXECUTION TIME MODEL
3 THE GEOMETRIC MODEL
4 THE NHPP MODEL FOR TIME – BETWEEN – ERROR OCC.
5 RETURN TO THE MAIN PROGRAM
PLEASE ENTER THE MODEL OPTION.
IF WALL CLOCK AND CPU TBE DATA, THEN:
PLEASE ENTER ONE FOR WC TBE OR TWO FOR CPU TBE.
*** DATA TYPE ERROR; PLEASE TRY AGAIN (AFTER THE NEXT PROMPT).
END IF

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-1.1. Menu for Execution Time Models.



Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-1.2. Menu for Interval Data Models.

D-2. THE LITTLEWOOD AND VERRALL BAYESIAN RELIABILITY GROWTH MODEL.

D-2.1. **Model Description**. Proposed by Littlewood and Verrall, this execution time data model tries to take into account the fact that the software correction process can introduce errors.

D-2.2. **Model Assumptions**. This model makes the following assumptions:

a. The software is operated in a manner similar to its expected operational usage.

b. Successive times between software failures are independent, exponentially distributed random variables $x(i)$, $i= 1,2,...,n$ with parameter $\mu(i)$.

c. The $\mu(i)$ are independent, Γ distributed variables with parameters α and $\pi(i)$. α is a Γ function parameter. $\pi(i)$ is a function which describes a programmer's quality and the programming task's difficulty. Littlewood and Verrall recommend a simple linear or quadratic function for the form of π. This recommendation is implemented in SMERFS.

D-2.3. **Model Inputs**. The model inputs include data entered via the SMERFS data input module, DATINP, and responses to prompts from the SMERFS LAVMOD module.

D-2.3.1. **DATINP Inputs**. The data input via the DATINP consists of the times between the error occurrences (i.e., the $x(i)$'s) measured in CPU or wall clock time. This is the raw data needed to run the model (i.e., the model data requirements).

D-2.3.2. **LAVMOD Prompts**. LAVMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-2.3.2.1. **LAVMOD Description and List Prompts**. SMERFS prompts the user through LAVMOD to see if the user wishes to see a list of the model's assumptions and data requirements. The model assumptions are those discussed above. The model data requirements are the inputs to DATINP.

D-2.3.2.2. **LAVMOD Input and Prediction Vector Creation Prompts**. The LAVMOD input prompts are exhibited in the menu in Figure D-2.1. The first prompt in that menu lets the user specify the desired method of estimating α, the Γ function parameter, and the linear or quadratic coefficients of the π function. The two methods of estimation allowed are maximum likelihood and least squares. The second prompt allows the user to specify whether he or she wants a linear or quadratic π function. The third prompt lets the user enter initial estimates for the linear or quadratic

coefficients known as the ß parameters. The final prompt lets the user enter the number of iterations to perform to obtain the maximum likelihood or least squares estimates of the α and ß parameters.

D-2.4. Model Outputs. If successful convergence is achieved, LAVMOD outputs the expected mean time before the next error; otherwise, it lets the user try a larger number of iterations. In either case, estimates for the α and ß parameters for the π function discussed above are output. The LAVMOD successful convergence output menu is seen in Figure D-2.2.

D-3. JOHN MUSA'S EXECUTION TIME MODEL.

D-3.1. Model Description. This execution time data model is based upon the amount of CPU time used in testing rather than upon the amount of wall clock or calendar time. In addition to modeling software reliability, this model can be used to model allocation of resources for testing segments and relate CPU time to wall clock time. The model is important for this reason.

D-3.2. Model Assumptions. The following assumptions are those needed only for reliability modeling. The assumptions for modeling resource allocation are documented in the SMERFS User's Guide.

a. The software is operated in a way similar to its expected operational usage.

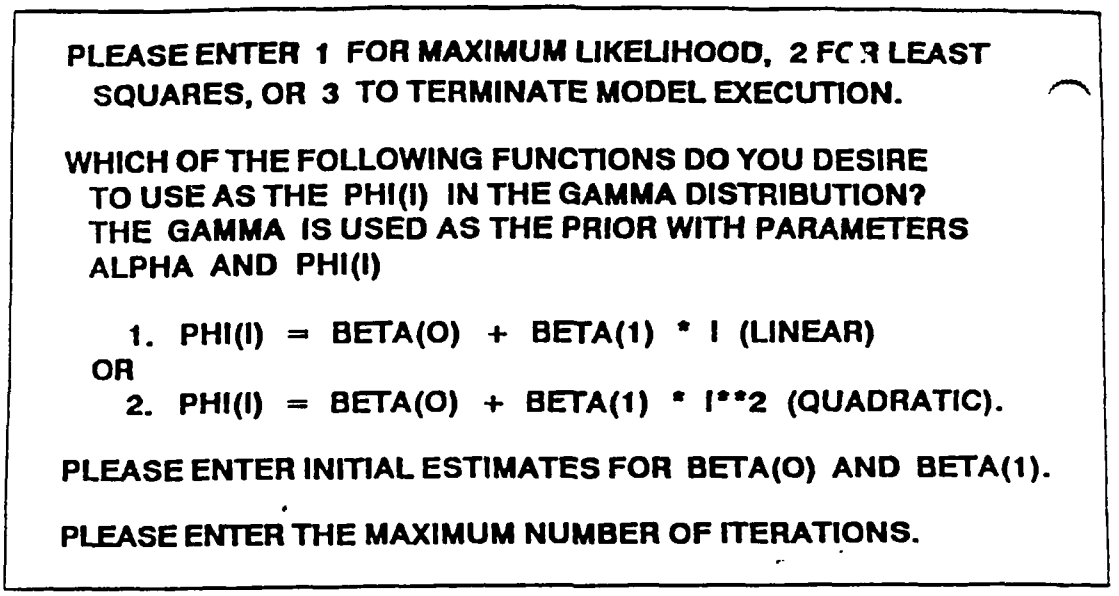
b. The probability of detecting any given error is in no way affected by the occurrence of detecting another error (i.e., error detections are independent).

c. Every failure of software is observed.

d. The execution times between software failures are piecewise exponentially distributed. That is, the hazard rate function is a constant which changes whenever an error is corrected.

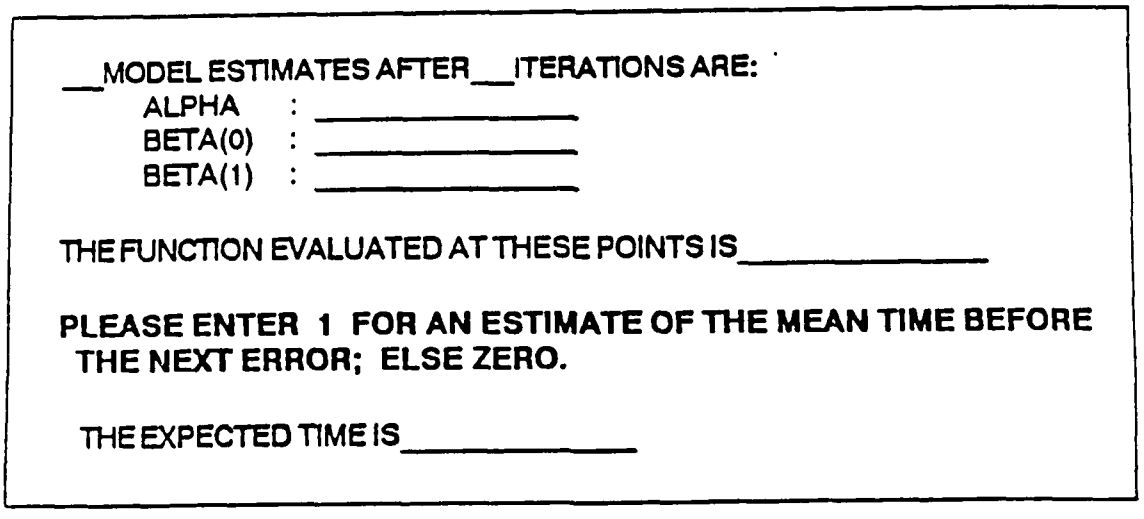
e. The ratio of the hazard rate to the number of errors remaining in the program is a constant.

f. The ratio of the rate of fault correction to the rate of failure occurrence is a constant.



Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-2.1. LAVMOD Input Prompts.



Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-2.2. LAVMOD Successful Convergence Output.

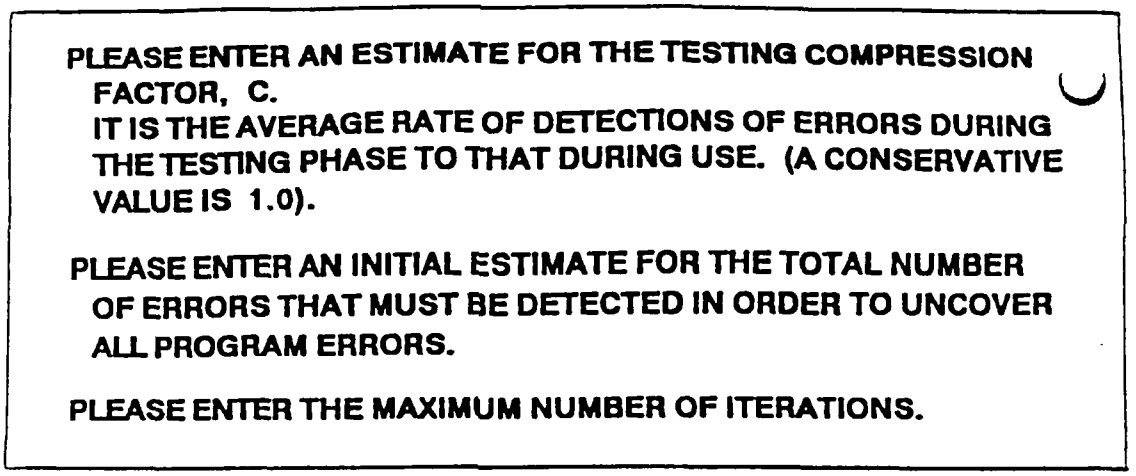
D-3.3. **Model Inputs**. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS MUSMOD module.

D-3.3.1. **DATINP Inputs**. The data input via the DATINP module consists of the times between software failure occurrences measured in CPU time.

D-3.3.2. **MUSMOD Prompts**. The Musa Model (MUSMOD) prompts consist of description and list, input, and prediction vector creation prompts.

D-3.3.2.1. **MUSMOD Description and List Prompts**. SMERFS prompts the user through MUSMOD to see if the user wishes to see a list of the model's assumptions and data requirements. The model assumptions are those discussed above. The model data requirements are the inputs to DATINP and the testing compression factor, C. This factor is the average ratio of the error detection rate during testing to that during operational use. This factor allows for changes in the operational environment.

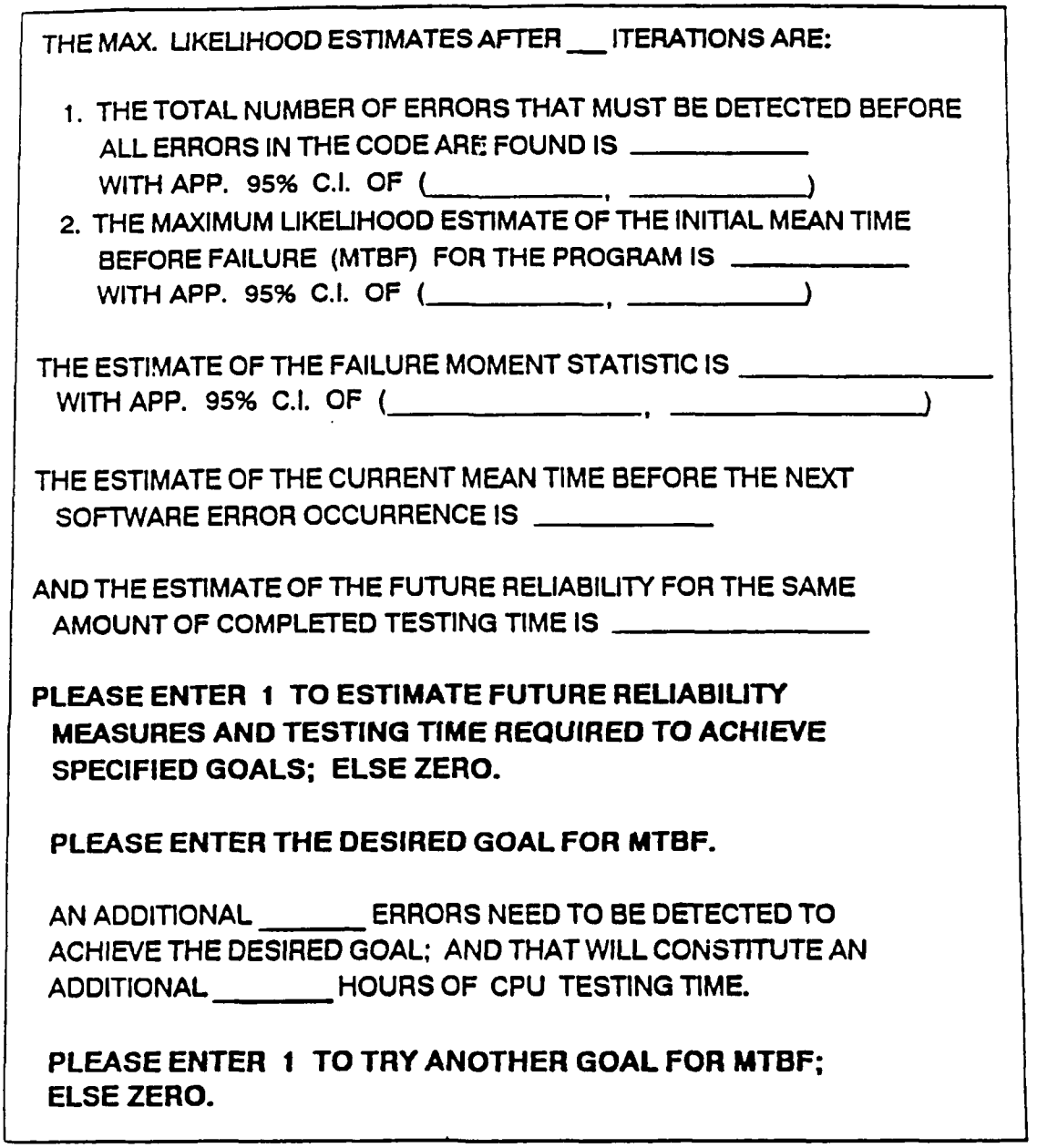
D-3.3.2.2. **MUSMOD Input and Prediction Vector Creation Prompts**. The MUSMOD input prompts are exhibited in Figure D-3.1. The first prompt lets the user specify the testing compression factor. If there is no basis for estimation of this factor, a conservative approach would be to let C equal one. The second prompt allows the user to enter an initial estimate of the required number of software failures that must be experienced to uncover all software faults within the program. The final prompt lets the user enter the maximum number of iterations to compute M, which is the required number of failures one needs to experience to uncover all faults within the program.



Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-3.1. MUSMOD Input Prompts.

D-3.4. Model Outputs. If a solution is found before the maximum number of iterations is reached, successful convergence output occurs (Figure D-3.2). Otherwise, the user is allowed to repeat execution of the MUSMOD. After successful output, the user is prompted to see whether he or she wishes to run the Calendar Time Component (CTC). This component computes resource allocation for the testing segments. For further information on the description and outputs of the Musa CTC, see the SMERFS User's Guide.



Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-3.2. MUSMOD Successful Convergence Output.

D-4. MORANDA'S GEOMETRIC MODEL.

D-4.1. Model Description. This execution time data model is a variation of the Jelinski-Moranda De-Eutrophication Model. The process of de-eutrophication presumes that the software hazard rate is reduced by the same amount at the time of each error detection. The software hazard rate is defined as the conditional probability that a software failure occurs in an interval of time given that the software has not failed up to the beginning of that time interval. The de-eutrophication process is geometric for this model because the hazard rate function decreases in a geometric progression as the detection of errors occurs.

D-4.2. Model Assumptions. The model presumes the following:

a. The software is operated in a way similar to its expected operational usage.

b. The program will never be error free.

c. The probability of detecting a given error may not equal the probability of detecting another given error.

d. The probability of detecting a given error is not affected by the probability of detecting another given error (i.e., the detection of errors is independent).

e. The rate at which errors are detected follows a geometric progression which is constant between error occurrences. This implies that errors become harder to detect as debugging progresses.

D-4.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS GEOMOD module.

D-4.3.1. DATINP Inputs. The data input via the DATINP module consists of the time between software failure occurrences measured in either CPU time or calendar (wall clock) time.

D-4.3.2. GEOMOD Prompts. GEOMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-4.3.2.1. GEOMOD Description and List Prompt. GEOMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP.

D-4.3.2.2. GEOMOD Input and Prediction Vector Creation Prompts. The GEOMOD input prompts are exhibited in Figure D-4.1. The

first prompt lets the user terminate model execution or indicate the least squares or maximum likelihood method to estimate the proportionality constant for the software hazard function. The second prompt enables the user to enter an initial estimate for this constant. The user should choose a number between 0 and 1 to guarantee convergence of the solution. The final prompt allows the user to enter the maximum number of convergence iterations for the estimation technique chosen.

PLEASE ENTER 1 FOR MAXIMUM LIKELIHOOD, 2 FOR LEAST SQUARES, OR 3 TO TERMINATE MODEL EXECUTION.

PLEASE ENTER AN INITIAL ESTIMATE FOR THE PROPORTIONALITY CONSTANT (A NUMBER BETWEEN ZERO AND ONE).

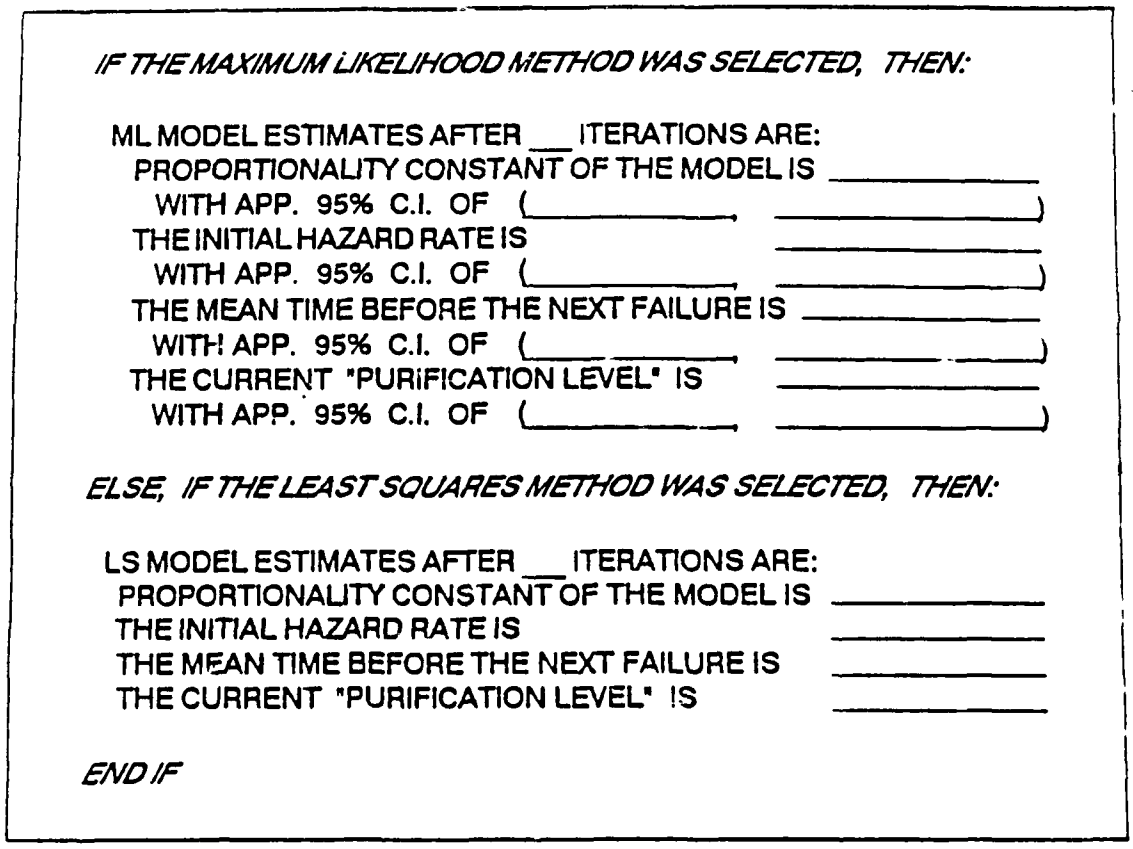
PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-4.1. GEOMOD Input Prompts.

D-4.4. Model Outputs. If a solution is found before the maximum number of iterations is reached, successful convergence output occurs (Figure D-4.2). Otherwise, the user is allowed to repeat execution of the model. As can be seen from Figure D-4.2, outputs include estimates for the proportionality constant, initial hazard rate, mean time before the next failure, and current purification level regardless of the estimation technique chosen (i.e., ML or LS). If ML is chosen, it also provides 95 per cent confidence intervals for these estimates.

Since the model assumes infinite errors, it cannot compute the total number of errors in the program. Instead, it estimates the degree of "purification" for the program.



Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-4.2. GEOMOD Successful Convergence Output.

D-5. ADAPTATION OF GOEL'S NON-HOMOGENEOUS POISSON PROCESS MODEL.

D-5.1. Model Description. This execution time data model is an adaptation of Amrit Goel's NHPP interval count model.

D-5.2. Model Assumptions. This model's assumptions include the following:

a. The software is operated in a way similar its expected to operational usage.

b. The probability of detecting any given software error is the same as the probability of detecting any other given error.

c. The cumulative number of software errors detected up to a point in time are Poisson distributed. The expected number of software errors in any small interval of time $(t,t+\delta t)$ is proportional to the number of undetected software errors at time t.

d. The mean of the Poisson distribution, M(t), is a bounded non-decreasing function. As the length of testing tends to infinity, M(t) approaches the expected total number of eventually detected software errors.

D-5.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP module and response to prompts from the SMERFS NPTMOD module.

D-5.3.1. DATINP Inputs. The data input via the SMERFS DATINP consists of the time between software failure occurrences measured in either CPU time or calendar (wall clock) time.

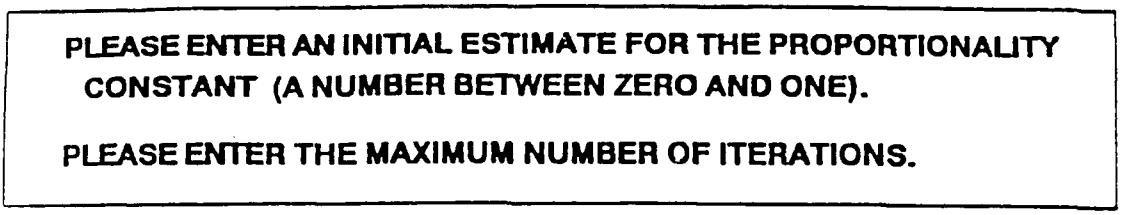
D-5.3.2. NPTMOD Prompts. NPTMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-5.3.2.1. NPTMOD Description and List Prompts. NPTMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP.

D-5.3.2.2. NPTMOD Input and Prediction Vector Creation Prompts. The NPTMOD input prompts are exhibited in Figure D-5.1. The first prompt lets the user enter an initial estimate for the proportionality constant. The user should choose a number between 0 and 1. The final prompt allows the user to enter the maximum number of iterations.

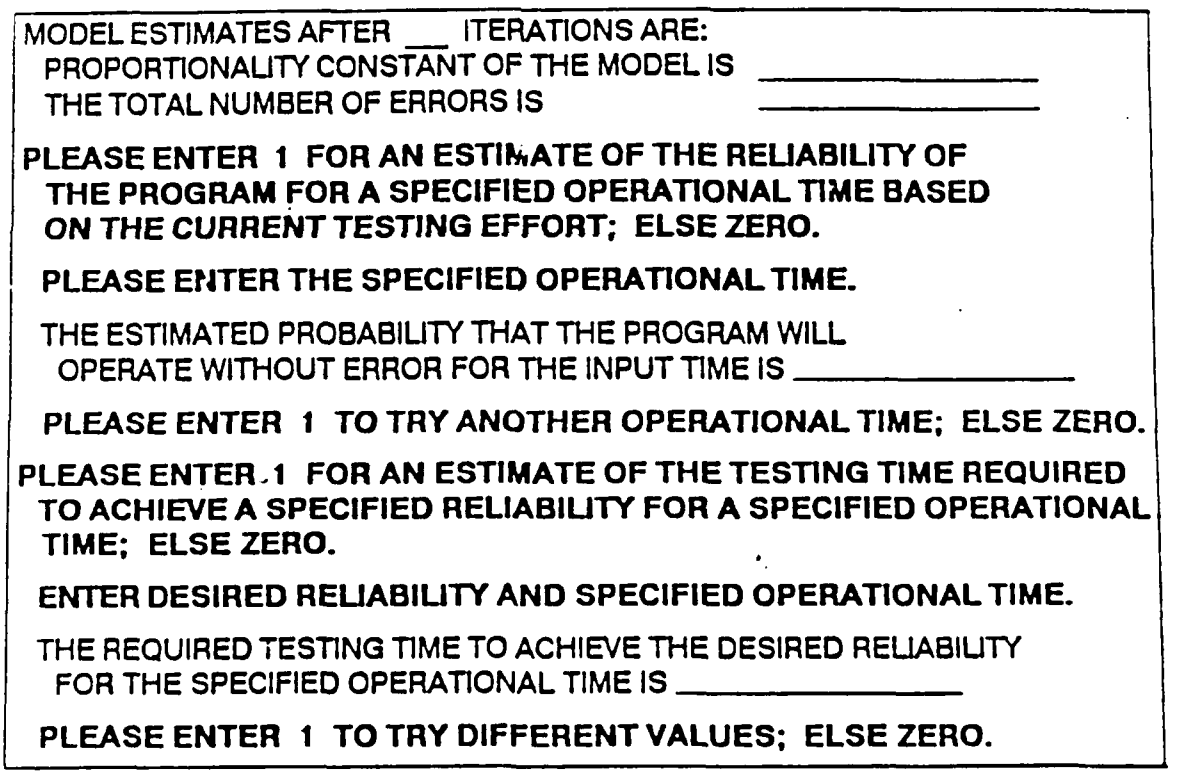
D-5.4. Model Outputs. If a solution is found before the maximum number of iterations is reached, successful convergence output

occurs (Figure D-5.2). Otherwise, the user is allowed to repeat execution of the model.



Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-5.1. NPTMOD Input Prompts.



Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-5.2. NPTMOD Successful Convergence Output.21

D-6. THE GENERALIZED POISSON MODEL.

D-6.1. Model Description. This model, which is one of the four models within SMERFS that obtains reliability estimates and predictions for interval data, is analogous in form to other models such as the Jelinski-Moranda, Lipow, and Schick-Wolverton models. It is documented in a report by Schafer, Alter, Angus, and Emoto written under contract to the Rome Air Development Center (RADC).

D-6.2. Model Assumptions. The GPOMOD makes the following five assumptions:

a. The software is operated in a way similar to its expected operational usage.

b. In any time interval, the expected number of discovered software errors is proportional to the product of the total number of existing software errors and to some function of the amount of time spent in testing for software errors. The function is expressed as an exponential function; however, the function could be a linear or parabolic function to allow for a broader class of adaptability.

c. All errors occur with the same probability, and the chance of any given error occurring in no way affects the occurrence or lack of occurrence of any other error (i.e., the errors are independent of each other).

d. Each error has equal severity.

e. At the end of the testing intervals, errors are corrected without introducing new errors.

D-6.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS GPOMOD module.

D-6.3.1. DATINP Inputs. The data input through the SMERFS module consists of the lengths of the various testing intervals and the number of software faults discovered in each testing interval.

D-6.3.2. GPOMOD Prompts. GPOMOD prompts consist of description and list, correction vector creation, input, and prediction vector creation prompts.

D-6.3.2.1. GPOMOD Description and List Prompt. GPOMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP.

D-6.3.2.2. **GPOMOD Correction Vector Creation Prompts**. SMERFS prompts for a flag which indicates whether or not software fault corrections were performed in the same interval in which they were detected. An error correction vector is created if all error detections and corrections happened during the same intervals; otherwise, the user must enter the number corrected at the end of each period of testing.

D-6.3.2.3. **GPOMOD Input and Prediction Vector Creation Prompts**. The GPOMOD input prompts are exhibited in Figure D-6.1. The first prompt lets the user terminate model execution or specify the method of estimating (i.e., maximum likelihood or least squares) the model's proportionality constant and the initial total number of errors in the software. After the user enters the desired method, GPOMOD prompts the user for the weighting function or a list of the available functions. If the user desires a list, two weighting functions are listed if least squares was chosen as the method of model parameter estimation; otherwise, one weighting function is listed. The third prompt lets the user specify the weighting function which is either a simple parabolic function or some other polynomial function of order α. If the latter choice is made, GPOMOD will additionally prompt for the order, α, of the polynomial. If the maximum likelihood method was selected earlier, the GPOMOD will prompt the user for an initial estimate of α. In either case, the user is prompted for an initial estimate of the total number of software errors and finally for the maximum number of iterations to be used for the model parameter estimation method.

The GPOMOD prediction vector creation prompts occur later upon successful convergence of the model parameter estimation method.

D-6.4. **Model Outputs**. If the maximum number of iterations is reached before a solution is found, SMERFS outputs attempted estimates of the model's parameters and the number of remaining software errors. If processing errors occur, then appropriate error messages are output. In either case, the user is allowed to try again. If the model successfully converges to a solution, the output seen in Figure D-6.2 occurs.

PLEASE ENTER 1 FOR MAXIMUM LIKELIHOOD, 2 FOR LEAST SQUARES, OR 3 TO TERMINATE MODEL EXECUTION.

PLEASE ENTER THE WEIGHTING FUNCTION NUMBER, OR ZERO FOR A LIST.

THE AVAILABLE WEIGHTING FUNCTIONS ARE
1 X(I) ** 2/2 (SCHICK–WOLVERTON MODEL)
2 X(I) ** ALPHA (WHERE ALPHA IS INPUT)

IF THE MAXIMUM LIKELIHOOD METHOD WAS SELECTED, THEN:

3 X(I) ** ALPHA (WHERE ALPHA IS ESTIMATED)

END IF

PLEASE ENTER THE WEIGHTING FUNCTION NUMBER.

IF AN ALPHA INPUT FUNCTION WAS SELECTED, THEN:

PLEASE ENTER THE DESIRED ALPHA.

ELSE, IF THE ALPHA ESTIMATION FUNCTION WAS SELECTED, THEN:

PLEASE ENTER AN INITIAL ESTIMATE FOR ALPHA.

END IF

PLEASE ENTER AN INITIAL ESTIMATE OF THE TOTAL NUMBER OF ERRORS.

PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-6.1. GPOMOD Input Prompts.

IF THE MAXIMUM LIKELIHOOD METHOD (OTHER THAN ALPHA ESTIMATED) WAS SELECTED, THEN:

ML MODEL ESTIMATES, USING THE WEIGHTING FUNCTION TYPE __, AF_ _
___ITERATIONS ARE:
PROPORTIONALITY CONSTANT OF THE MODEL IS _____
WITH APP. 95% C.I. OF (_____, _____)
THE TOTAL NUMBER OF ERRORS IS _____
WITH APP. 95% C.I. OF (_____, _____)
THE REMAINING NUMBER OF ERRORS IS _____
WITH APP. 95% C.I. OF (_____, _____)

ELSE, IF THE LEAST SQUARES METHOD WAS SELECTED, THEN:

LS MODEL ESTIMATES, USING THE WEIGHTING FUNCTION TYPE __, AFTER
___ITERATIONS ARE:
PROPORTIONALITY CONSTANT OF THE MODEL IS _____
THE TOTAL NUMBER OF ERRORS IS _____
THE REMAINING NUMBER OF ERRORS IS _____

ELSE, IF THE MAXIMUM LIKELIHOOD METHOD (WITH ALPHA ESTIMATED) WAS SELECTED, THEN:

ML MODEL ESTIMATES, USING THE WEIGHTING FUNCTION TYPE 3, AFTE_
___ITERATIONS ARE:
PROPORTIONALITY CONSTANT OF THE MODEL IS _____
THE TOTAL NUMBER OF ERRORS IS _____
THE REMAINING NUMBER OF ERRORS IS _____
AND ALPHA IS _____

END IF

PLEASE ENTER 1 FOR AN ESTIMATE OF THE NUMBER OF ERRORS EXPECTED IN THE NEXT TESTING PERIOD; ELSE ZERO.

ENTER THE PROJECTED LENGTH OF THE TESTING PERIOD.

THE EXPECTED NUMBER OF ERRORS IS _____
WITH APP. 95% C.I. OF (_____, _____)

PLEASE ENTER 1 TO TRY ANOTHER TESTING LENGTH; ELSE ZERO.

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-6.2. GPOMOD Successful Convergence Output.

D-7. GOEL'S NHPP MODEL.

D-7.1. Model Description. This model is one of the four models within SMERFS that obtains reliability estimates and predictions for interval data. It was developed by Amrit Goel and Kazu Okumoto. Followi ıg other models, it assumes that counts of software failures over time intervals that don't overlap follow a Poisson distribution. A difference between this model and other Poisson models is that this model treats a program's initial error content as a random variable, and not as a fixed constant.

D-7.2. Model Assumptions. Goel's NHPP model (NPIMOD) makes the following assumptions:

a. The software is operated in a way similar to its expected operational usage.

b. The number of software errors detected in successive time intervals are independent.

c. The probability of detecting any given error is the same as the probability of detecting any other given error. In addition, each error is assumed to have equal severity.

d. At any time t, the cumulative number of errors detected follows a Poisson distribution with mean m(t). m(t) satisfies a first order non-homogeneous linear differential equation.

e. m(t) is a bounded, nondecreasing function of t which approaches the expected total number of errors to be detected as t tends to ∞.

D-7.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS NPIMOD module.

D-7.3.1. DATINP Inputs. The data input through the SMERFS DATINP module consists of the lengths of the various testing intervals and the number of software errors discovered in each testing interval.

D-7.3.2. NPIMOD Prompts. NPIMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-7.3.2.1. NPIMOD Description and List Prompt. NPIMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP.

D-7.3.2.2. **NPIMOD Input and Prediction Vector Creation Prompts**. The NPIMOD input prompts are exhibited in Figure D-7.1. The first prompt lets the user terminate model execution or specify the method of estimating (i.e., maximum likelihood or least squares) the model's proportionality constant and the total number of errors in the software. The second prompt allows the user to enter an initial estimate for the model's proportionality constant. A number between 0 and 1 must be chosen to guarantee convergence of the solution. It is recommended that the user choose a small number first, say 0.05 or 0.1, and then gradually increase it. The last prompt in the first menu lets the user enter the maximum number of iterations to use for the estimation method selected.

The NPIMOD prediction vector creation prompts occur later upon successful convergence output of the model. Through these prompts, NPIMOD lets the user compute predicted interval error counts.

D-7.4. **Model Outputs**. If the maximum number of iterations is reached before a solution is found, maximum iteration output occurs unless a processing error happens. If the model successfully converges to a solution, the output seen in Figure D-7.2 occurs. If maximum likelihood is chosen, then ML estimates are shown; otherwise, least squares estimates are output. In either event, the user is allowed to estimate the number of expected errors in the next testing period. Figure D-7.2 shows ensuing output if the user does want an estimate of the number of expected errors in the next testing period.

PLEASE ENTER 1 FOR MAXIMUM LIKELIHOOD, 2 FOR LEAST SQUARES, OR 3 TO TERMINATE MODEL EXECUTION.

PLEASE ENTER AN INITIAL ESTIMATE FOR THE PROPORTIONALITY CONSTANT (A NUMBER BETWEEN ZERO AND ONE).

PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-7.1. NPIMOD Input Prompts.

IF THE MAXIMUM LIKELIHOOD METHOD WAS SELECTED, THEN:

ML MODEL ESTIMATES AFTER __ ITERATIONS ARE:
PROPORTIONALITY CONSTANT OF THE MODEL IS _____
WITH APP. 95% C.I. OF (_____, _____
THE TOTAL NUMBER OF ERRORS IS _____
WITH APP. 95% C.I. OF (_____, _____

ELSE, IF THE LEAST SQUARES METHOD WAS SELECTED, THEN:

LS MODEL ESTIMATES AFTER __ ITERATIONS ARE:
PROPORTIONALITY CONSTANT OF THE MODEL IS _____
THE TOTAL NUMBER OF ERRORS IS _____

END IF

PLEASE ENTER 1 FOR AN ESTIMATE OF THE NUMBER OF ERRORS EXPECTED IN THE NEXT TESTING PERIOD; ELSE ZERO.

ENTER THE PROJECTED LENGTH OF THE TESTING PERIOD.

THE EXPECTED NUMBER OF ERRORS IS _____

PLEASE ENTER 1 TO TRY ANOTHER TESTING LENGTH; ELSE ZERO.

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-7.2. NPIMOD Successful Convergence Output.

D-8. BROOKS AND MOTLEY'S MODEL.

D-8.1. Model Description. This model actually consists of four models each of which obtains reliability estimates and predictions for interval data. They were developed by Brooks and Motley of IBM and include the following: Binomial and Poisson Models for a component of a program and Binomial and Poisson Models for a program. Each of these models accounts for unequal testing of programs in a given testing period.

D-8.2. Model Assumptions. Each model makes the following assumptions:

a. The software is operated in a way similar to its expected operational usage.

b. The ratio of the number of errors reintroduced during the software correction process to the number of errors that are detected is constant.

c. The probability of detecting any error during a given unit interval of testing is constant for any occasion and independent of error detections. The constant is denoted as q in the case of the binomial model, and Φ for the Poisson model.

D-8.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS BAMMOD module.

D-8.3.1. DATINP Inputs. The data input through the SMERFS DATINP module consists of the lengths of the various testing intervals and the number of software errors discovered in each testing interval.

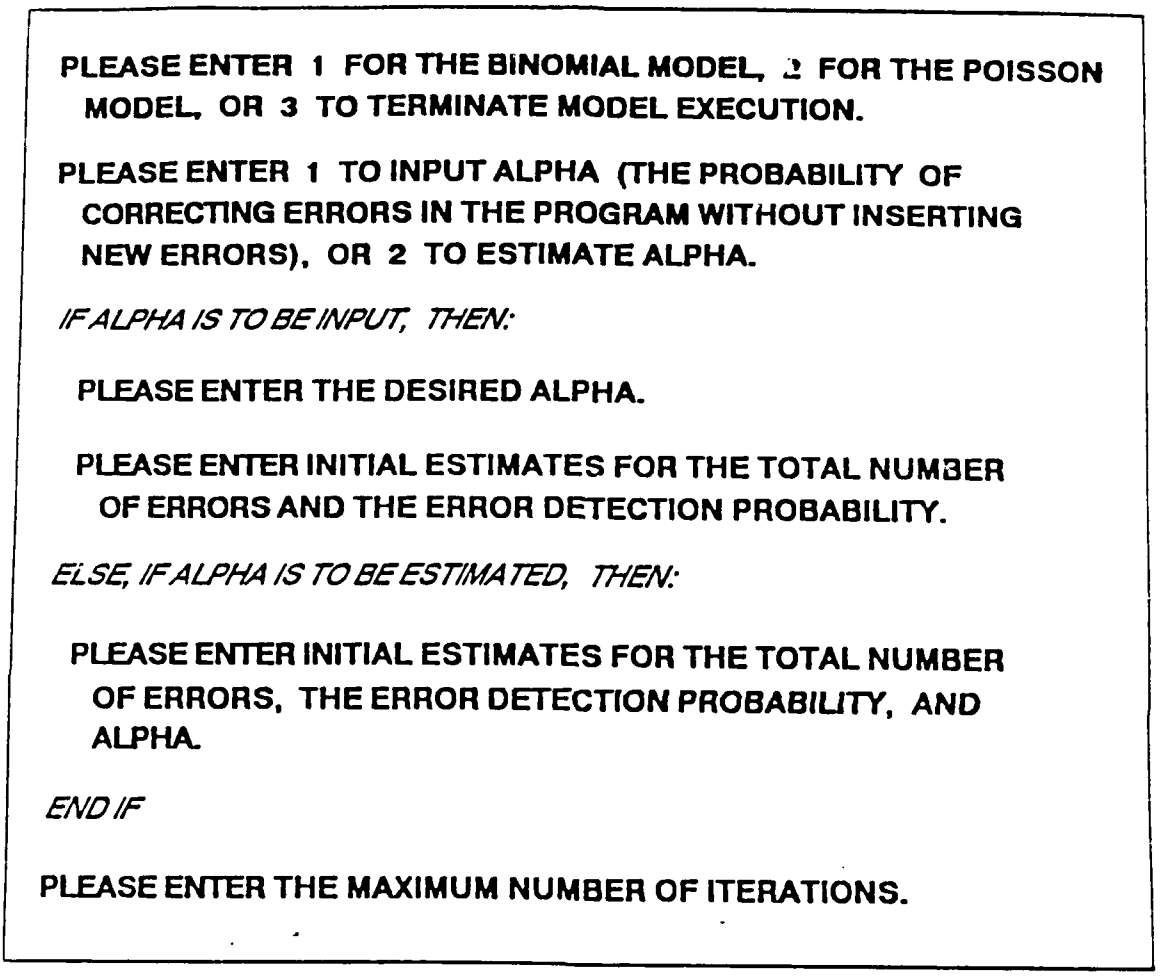
D-8.3.2. BAMMOD Prompts. BAMMOD prompts consist of description and list, fraction of code under test, extended description and list, input, and prediction vector creation prompts.

D-8.3.2.1. BAMMOD Description and List Prompts and Fraction of Code Under Test Prompt. BAMMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP. In addition, BAMMOD has extended description and list prompts which provide extended descriptions of the Binomial and Poisson Models. The fraction of code under test prompt lets the user compensate for partial software testing.

D-8.3.2.2. BAMMOD Input and Prediction Vector Creation Prompts. The BAMMOD input prompts are exhibited in Figure D-8.1. The first prompt lets the user select the appropriate model of interest (Binomial or Poisson) or terminate model execution. The

second prompt allows the user to either input or select an initial estimate for α, the probability of correcting errors without inserting new ones. If a decision is made to input α, a suggested range is 0.85-0.95 if no prior knowledge is available. In either event, the total number of errors and the error detection probability are then estimated. The behavior of the estimation process can be observed by trying both low values, such as 0.05-0.1, and high values, such as 0.85-0.90 for the error detection probability. The last prompt lets the user enter the maximum number of convergence iterations to use for the maximum likelihood estimation method of computing the model parameters.

The BAMMOD prediction vector creation prompts occur later upon successful convergence output of the model. Through these prompts, BAMMOD lets the user compute predicted interval error counts.



Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-8.1. BAMMOD Input Prompts.

D-8.4. **Model Outputs**. If the maximum number of iterations is reached before a solution is found, maximum iteration output occurs unless a processing error happens. If the model successfully converges to a solution, the output seen in Figure D-8.2 occurs. BAMMOD solutions are based upon maximum likelihood estimates. The last estimate in the figure will be listed only if the user selected alpha estimation. Observing the lower portion of the figure, one may see that SMERFS allows for the optional prediction of errors in the next testing period.

THE _____ MODEL WITH _____ ESTIMATES, AFTER ___ INTERATIONS ARE:
PROBABILITY OF DETECTING ERRORS _____
THE TOTAL NUMBER OF ERRORS IS _____

IF ALPHA WAS ESTIMATED, THEN:

PROB. OF CORRECTING ERRORS WITHOUT ERROR _____

END IF

PLEASE ENTER 1 FOR AN ESTIMATE OF THE NUMBER OF ERRORS EXPECTED IN THE NEXT TESTING PERIOD; ELSE ZERO.

ENTER THE PROJECTED LENGTH OF THE TESTING PERIOD.

ENTER THE FRACTION OF THE PROGRAM TO BE TESTED (FOR FULL PROGRAM, ENTER A 1).

HOW MANY ERRORS HAVE BEEN FOUND TO DATE IN THE SECTION OF THE CODE TO BE TESTED.

THE EXPECTED NUMBER OF ERRORS IS _____

PLEASE ENTER 1 TO TRY ANOTHER TESTING LENGTH; ELSE ZERO.

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-8.2. BAMMOD Successful Convergence Output.

D-9. NORMAN SCHNEIDEWIND'S MODEL.

D-9.1. Model Description. This model is another one of the four models within SMERFS that obtains reliability estimates and predictions for interval data. It was developed by Norman Schneidewind. The model theorizes that recent error counts are generally more useful than earlier ones when predicting future error counts because the error detection process changes as testing progresses over time. The model employs three approaches in utilizing error count data:

a. Use all error counts for all m intervals of testing.

b. Completely ignore error counts from the first s - 1 intervals of testing where $2 \leq s \leq m$. Only data from intervals s through m are considered.

c. For intervals 1 through s - 1 use the cumulative error count. For interval s through m, use the individual error counts.

D-9.2 Model Assumptions. The SDWMOD makes the following assumptions:

a. The software is operated in a way similar to the way it is expected to be used.

b. All errors are independent and occur with equal probability.

c. The ratio of the error correction rate to the number of errors to be corrected is constant.

d. As testing progresses, the mean number of errors that are detected decreases from one interval to the next.

e. The length of each testing period is of the same duration.

f. At the time of the test, the ratio of the rate of error detection to the number of errors within the program is constant. The process of error detection follows a non-homogeneous Poisson process where the error detection rate decreases exponentially.

D-9.3. Model Inputs. The model inputs include data entered through the SMERFS DATINP and responses to prompts from the SMERFS SDWMOD module.

D-9.3.1. **DATINP Inputs**. The data input through the SMERFS DATINP consists of the number of software errors discovered in each testing interval.

D-9.3.2. **SDWMOD Prompts**. SDWMOD prompts consist of description and list, input, and prediction vector creation prompts.

D-9.3.2.1. **SDWMOD Description and List Prompts**. SDWMOD prompts the user to see whether he or she wants a list of the model's assumptions and data requirements. The assumptions listed are those discussed above. The data requirements of the model are previously input to DATINP. The description prompt also includes a description of the three approaches for utilizing the error count data. SMERFS refers to these three approaches as the three treatment types.

D-9.3.2.2. **SDWMOD Input and Prediction Vector Creation Prompts**. The SDWMOD input prompts are exhibited in Figure D-9.1. The first prompt lets the user terminate model execution or specify one of the three treatment types. If treatment type is 2 or 3, then the user must also enter the associated value of s. The user is then prompted for an initial estimate of the ß parameter in the formula for the mean number of errors for the i-th period of testing. Finally, the user is prompted for the maximum number of iterations to use for the maximum likelihood method.

PLEASE ENTER THE DESIRED MODEL TREATMENT NUMBER, OR A 4 TO TERMINATE MODEL EXECUTION.

IF THE TREATMENT TYPE IS 2 OR 3, THEN:

PLEASE ENTER THE ASSOCIATED VALUE OF S.

END IF

PLEASE ENTER AN INITIAL ESTIMATE FOR THE PARAMETER BETA, WHERE THE MEAN NUMBER OF ERRORS FOR THE I–TH PERIOD IS TAKEN AS:
MEAN(I) = ALPHA*(EXP(–BETA(I–1))–EXP(–BETA(I)))/BETA.

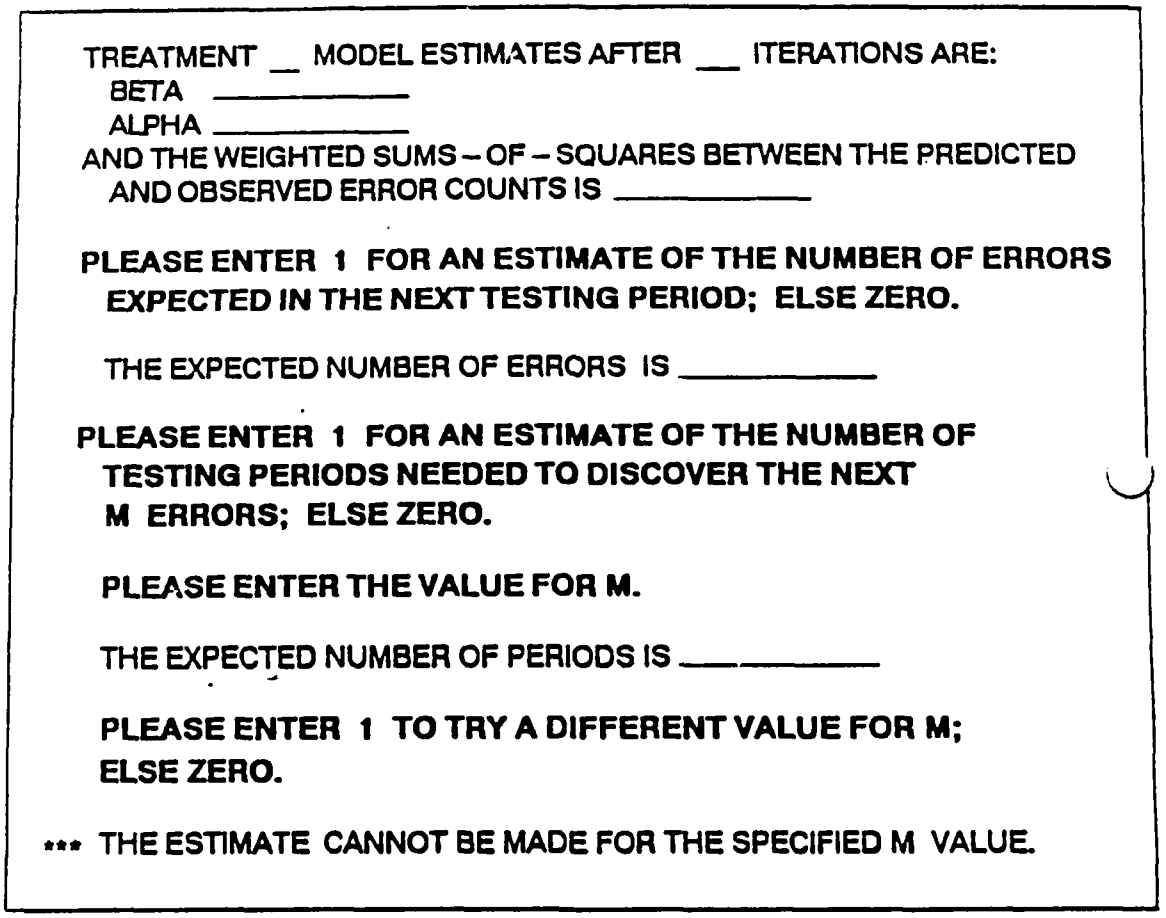
PLEASE ENTER THE MAXIMUM NUMBER OF ITERATIONS.

Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-9.1. SDWMOD Input Prompts.

The SDWMOD prediction vector creation prompts occur later upon successful convergence output of the model. Through these prompts, SDWMOD lets the user compute predicted interval error counts.

D-9.4. **Model Outputs**. If the maximum number of iterations is reached before a solution is found, maximum iteration output occurs unless a processing error happens. If the model successfully converges to a solution, the output seen in Figure D-9.2 occurs. The α and ß parameters of the error detection rate formula and the weighted sum of squares between the predicted and observed error counts are output. This latter quantity helps decide which treatment type is best. Output also includes an estimate of the number of errors expected in the next testing period and the number of testing periods needed to discover the next M errors, where M is specified by the user.



Source: NSWC TR 84-373 Revision 1, Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) USER's Guide, Farr, W.H., Smith, O.D., December 1988

Figure D-9.2. SDWMOD Successful Convergence Output.

APPENDIX E

EXAMPLE SOFTWARE RELIABILITY TEST PLAN

E-1. SOFTWARE RELIABILITY SUBTEST

This subtest is concerned with determining software reliability metrics for the software configuration of the MSE System during Phase 1 testing. The NCS and LEN elements of that configuration, which was for a U.S. Army division, were chosen because of the functionality provided by these components. Were it not for time constraints, other components could have been examined in a similar fashion. Alternative analytical procedures for examining other components (e.g., overall system reliability) are included for those cases where the procedures would differ from those used in the example analysis. It should also be noted that quantitative reliability criteria was not available for this example, but would be required for an actual test.

E-2. OBJECTIVE

The objective of this software reliability portion of MSE testing is to determine the extent to which the MSE software can be expected to perform its intended capabilities. This test plan is limited to that portion of MSE consisting of the NCSs and LEN. Reliability, availability, and maintainability (RAM) data was not collected for the System Control Center (SCC), the Digital Subscriber Voice Terminals (DSVTs), and the individual Mobile Subscriber Radio Terminals (MSRTs) of the MSE system during Phase 1 testing. Data from Radio Access Unit (RAU) and Small Extension Node (SEN) components was not used during this example software reliability estimation exercise.

E-3. CRITERIA

a. The mean time to failure (MTTF) of the NCS and LEN subsystem software configuration of the MSE for an Army division shall not be less than [the number of hours specified in the appropriate requirements document].

b. The number of software faults remaining in the NCS and LEN subsystems shall not exceed the value specified in the appropriate requirements document.

E-4. DATA REQUIRED

a. The data required to calculate MTTF are:

(1) The severity classification of each MSE software failure.

(2) The time, measured in wall clock and/or CPU time, at which each MSE software failure occurred.

(3) The time, measured in wall clock and/or CPU time, at which each period of software testing began for each MSE software component within each MSE assemblage.

(4) The time, measured in wall clock and/or CPU time, at which each period of software testing ended for each MSE software component within each MSE assemblage.

(5) The software component in which the software failure occurred (e.g., CSOLOP software).

(6) The version of each MSE software component which failed.

(7) The number of installations of each software component (e.g., the number of CSOLOP components running during the Phase 1 test).

b. The data required to estimate the number of software faults remaining are:

[Same as a. above]

E-5. DATA ACQUISITION PROCEDURE.

a. For each software failure which occurs during the test, a problem report or test log identifying each software failure detected will be prepared. The report shall contain, at a minimum, the following information:

(1) The time, in wall clock or CPU time, at which the MSE software failure occurred.

(2) The start time, in wall clock or CPU time, of the testing period during which the MSE software failure occurred.

(3) The stop time, in wall clock or CPU time, of the testing period during which the MSE software failure occurred.

(4) The version of the MSE software which failed.

(5) Initial assessment of the cause of the MSE software failure.

(6) The severity classification of the MSE software failure. To obtain the severity classification, classify each software failure according to the following severity classification scheme:

(a) Class A - A Class A software failure causes service to be interrupted (for example, system not mission capable due to crash).

(b) Class B - A Class B software failure causes service to be degraded, but not brought to a complete standstill (for example, partially mission capable, no mission essential functions lost or aborted). The system still operates, but the correction of the failure cannot be deferred.

(c) Class C - A Class C software failure has minor tolerable effects. The correction of the failure can be deferred.

(7) The MSE hardware system component in which the installed software failed.

Acquire data items 2.1.3.a.1, 2.1.3.a.2, 2.1.3.a.3, 2.1.3.a.4, 2.1.3.a.5, and 2.1.3.a.6, from the software problem reports.

Data listed in item 2.1.3.a.7, above will be obtained from an examination of a functional block diagram for each MSE NCS or LEN assemblage and will result from multiplying the number of installations in the diagram by the number of MSE assemblages in which the installation occurs.

b. The data required to estimate the number of software faults remaining are:

[Same as a. above]

E-6. ANALYTICAL PROCEDURES

These procedures outline the way in which the software reliability metrics for the MSE software will be computed. The approach is to compute these metrics for each software component which performs a unique capability, or for groups of software components which perform a unique capability. (Analysis for the overall system software reliability would be performed in a similar fashion, if that had been a criterion also). Application of these procedures requires the use of software reliability models.

a. Procedures to assess MTTF are:

(1) It is assumed that the version of each software component for the Phase 1 test was constant. However, the version of each software component should be checked to ensure the validity of this assumption.

(2) The software failure data input to the model consists of time between failure data. Because there were multiple installations of MSE software configurations (i.e., the same software ran on more than one computer), the associated software failure data has to be interleaved to determine the MTTF of a given component.

(a) To compute the MTTF for the CSOLOP software only, for example, the following is done. Time between failure data is based on the cumulative execution time of all CSOLOP software. Suppose, for example, one component runs in one NCS while a second identical component runs in another NCS. The start time of testing of the software is the time the first component begins executing. The stop time of testing is the time the last component stops executing. Software failure times are adjusted to allow for the running of the same software at the two NCSs. (For example, if a software failure occurred at 9:00 A.M. in the CSOLOP on one NCS which began executing at 8:00 A.M., and the same version of the CSOLOP had been running on another computer without failure since 8:30 A.M. the same day, then 8:00 A.M. would be regarded as the start time of testing for the CSOLOP software and the first software time between failure would be 90 minutes to adjust for the cumulative execution time of the two installations of CSOLOP software). These adjusted times between failures should be input to the model.

(b) Alternative procedure for the case where the criterion applies to the overall system:

To compute the MTTF of the overall software system itself, apply Musa's simplified approach to distributed systems. Using this approach, all of the software components can be lumped together. That is, the software components can be regarded as one system. A software failure due to an activation of a software fault in any component is considered to be a system failure. Software failure data is interleaved for each component. Then, the resulting sets of interleaved software failure data are merged to come up with time between failure data for the system.

(c) Alternative procedure for the case where the criteria are structured by the severity of failures.

(1) Once the MTTF of the MSE software system is computed, MTTFs for each software failure class of the MSE software system will be determined by dividing the overall MTTF by the proportion of failures occurring in each software failure class. The validity of this procedure depends on the assumption that the proportion of software failures in each software failure class remains constant with time. If the assumption is not valid, then the procedure cannot be applied.)

(2) The Littlewood and Verrall Bayesian Model, the Musa Execution Time Model, the Geometric Model, or Amrit Goel's Non-Homogeneous Poisson Process Model will be used to determine the MTTF for the software configurations. Compare the MTTF of each configuration below against its criterion in section E-3 above. If the computed MTTF is less than the number specified in the criterion, then that shall constitute a failure.

b. Procedures to assess the number of faults remaining are:

(1) Follow steps in Section E-6, paragraph a, except for the models used for the analysis.

(2) The Generalized Poisson Model, the Non-Homogeneous Poisson Model, the Brooks and Motley Model, or the Schneidewind Model will be used to determine the number of software faults remaining in the MSE software components. Compare the number of faults remaining in each MSE software component against its criterion above. For each component, if the number of faults exceeds the value specified in the criterion, then that shall constitute a failure.

This Page Intentionally Blank

APPENDIX F

GLOSSARY

F-1. SCOPE

The following terms are identified and defined as they are used throughout the Software Maturity Model Investigation.

Hazard rate

The conditional probability that a software failure occurs in an interval of time given that the software has not failed up to the beginning of that time interval.

Mean Time Between Failure

The expected time between one error occurrence and another.

Mean Time To Repair

The expected time between the occurrence of an error and its repair.

Remaining Number of Errors

The number of software errors remaining in a program.

Software Error

A human error which introduces a fault in software.

Software Failure

A deviation of the operation of software from its requirements. It is caused by a software fault.

Software Fault

A defect in software which causes a software failure to occur when that software is executed.

Software Maturity

This is defined in AFOTECP 800-2 Volume 1, 1 August 1986, as "a measure of the software's evolution toward satisfying all documented user requirements."

Software Performance Parameter

An objectively quantifiable measure of an aspect of software behavior.

Software Quality Indicator

According to AMC-P 70-14, 30 April 1987, software quality indicators are "quality indicators designed for and specifically allied to software projects." AMC-P 70-14 further defines quality indicators as "process guidelines in the form of detailed data, derived from scheduled surveys, inspections, evaluations, and tests, that provide insight into the condition of a product or process."