NUREG/CR-2186

ANL-81-84

NUREG/CR-2186

ANL-81-84

QUANTITATIVE SOFTWARE RELIABILITY ANALYSIS OF COMPUTER CODES RELEVANT TO NUCLEAR SAFETY

ARGONNE NATIONAL LABORATORY

ARGONNE NATIONAL LABORATORY, ARGONNE, ILLINOIS

Prepared for the Office of Nuclear Regulatory Research
U. S. NUCLEAR REGULATORY COMMISSION
under Interagency Agreement DOE 40-550-75

The facilities of Argonne National Laboratory are owned by the United States Government. Under the terms of a contract (W-31-109-Eng-38) among the U. S. Department of Energy, Argonne Universities Association and The University of Chicago, the University employs the staff and operates the Laboratory in accordance with policies and programs formulated, approved and reviewed by the Association.

Available from

GPO Sales Program
Division of Technical Information and Document Control
U. S. Nuclear Regulatory Commission
Washington, D.C. 20555

and

National Technical Information Service
Springfield, Virginia 22161

NUREG/CR-2186
ANL-81-84

Distribution
Codes: RG
and XA

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

QUANTITATIVE SOFTWARE RELIABILITY ANALYSIS OF COMPUTER CODES RELEVANT TO NUCLEAR SAFETY

C. J. Mueller
Editor

Contributors

H. Komoriya
C. C. Meek
E. E. Morris
W. E. Vesely*

Reactor Analysis and Safety Division

December 1981

Prepared for the Office of Nuclear Reactor Regulation
U. S. Nuclear Regulatory Commission
Washington, D. C. 20555
Under Interagency Agreement DOE 40-550-75

NRC FIN No. A2226

*U. S. Nuclear Regulatory Commission

Abstract

This report presents the results of the first year of an ongoing research program to determine the probability of failure characteristics of computer codes relevant to nuclear safety. An introduction to both qualitative and quantitative aspects of nuclear software is given. A mathematical framework is presented which will enable the a priori prediction of the probability of failure characteristics of a code given the proper specification of its properties. The framework consists of four parts: 1) a classification system for software errors and code failures; 2) probabilistic modeling for selected reliability characteristics; 3) multivariate regression analyses to establish predictive relationships among reliability characteristics and generic code property and development parameters; and 4) the associated information base. Preliminary data of the type needed to support the modeling and the predictions of this program are described. Illustrations of the use of the modeling are given but the results so obtained, as well as all "results" of code failure probabilities presented herein, are based on data which at this point are preliminary, incomplete, and possibly non-representative of codes relevant to nuclear safety.

NRC FIN NO.	TITLE
A2226	Software Reliability

Table of Contents

List of Figures

List of Tables

List of Tables (cont'd)

Executive Summary

This report presents the results of the first year of an ongoing research program to determine the probability of failure characteristics of computer codes relevant to nuclear safety. An introduction to both qualitative and quantitative aspects of nuclear software is given. A mathematical framework is presented which will enable the a priori prediction of the probability of failure characteristics of a code given the proper specification of its properties. Preliminary data of the type needed to support the modeling and the predictions of this program are described. Illustrations of the use of the modeling are given but the results so obtained, as well as all "results" of code failure probabilities presented herein, are based on data which at this point are preliminary, incomplete, and possibly non-representative of codes relevant to nuclear safety.

1.0 Introduction and Summary

This report describes the results of the initial phase of a research program sponsored by the Nuclear Regulatory Commission (NRC) and designed to establish quantitative predictions of reliability for the computer software used in the nuclear industry. The overall objective of this program is to develop generically applicable analytic tools and an associated information base that can be used to estimate the probability of success and other reliability characteristics of any specific computer code relevant to nuclear reactor safety, given the characteristics of the specified code. Relevant applications in nuclear safety include both analysis computer codes used in design and safety assessment and data acquisition codes used in real and simulated on-line and emergency situations.

Illustrative examples of the type of question that this research program is designed to answer or to help to answer are:

1. What is the probability that a large safety code yields the correct (or at least a conservative estimate of the) containment pressure for a hypothetical loss of coolant accident? Although the phenomenological modeling uncertainties are evident and may dominate the "answer" a contributor to this probability is the uncertainty in whether the associated software is input and exercised properly. This latter contribution is to be assessed in this program.

2. What is the probability that a computer program monitoring and analyzing real time nuclear reactor sensor data correctly interprets, analyzes, and presents this data to a reactor operator in an emergency situation?

To not only answer questions such as these for existing specific computer programs, but to provide generically applicable answers for use in enhancing nuclear safety, the designated specific goals for the analytic tools and information developed as part of this program are to:

1. Allow NRC to make an a priori estimate of a computer code's probability of successful operation given the code's general properties and its intended application. These properties include not only physical characteristics such as size, complexity and programming language but development and quality assurance parameters such as testing history and operational history after testing. It is to be emphasized that the usage or application of a code can be a strong determinant of its probability of successful operation.

2. Provide information for NRC to determine quantitative criteria for the acceptability of software relevant to nuclear safety for eventual use in the licensing arena.

The motivation for this program is provided by the important role played by computers in reactor design and safety analysis and by the growing need for improved on-line information processing. Propagation of software errors in reactor design codes can lead to analytical predictions, resulting among other things, in unsafe plant hardware design, unsafe operational and safety parameters such as trip settings, and erroneous predictions of plant lifetime related parameters. Software errors in reactor safety assessment codes may result in either false "security" in which needed safety measures go unrecognized or in undue alarm leading to the incorporation of unnecessary, costly, and even potentially unsafe mitigation devices. As pointed out by Fabic [1.1], the accuracy of a safety code in predicting the response of a plant to a nuclear accident is a function of programming errors as well as a host of plant and modeling uncertainties. All these factors make it clear that software reliability must be integrated into quantitative assessments of nuclear safety.

The growing need for data acquisition and information processing codes in on-line situations has been highlighted by the President's Commission on the Three Mile Island Accident [1.2]. Long et al. [1.3] and Ramamoorthy et al. [1.4] have emphasized the necessity for a methodology to assess the development and validation of critical software for nuclear power plants. On-line safety protection system software such as the Core Protection Calculator (CPC) implemented in the Arkansas Nuclear One, Unit 2 [1.5] and a proposed disturbance analysis system (DAS) [1.6] are examples of critical software. The CPC is a software system which processes sensor information and if necessary initiates a High Local Power Density or Low DNBR trip. DAS is a computer network of hardware and software which allows plant personnel to access and act upon real-time information relating to possible causes and consequences of disturbances and corrective actions. Propagation of software errors through these systems can lead to delayed or erroneous control and operator actions, and unnecessary forced outages. For example, in a review of 14 events [1.7] in which reactor control was either lost or impaired at one Canadian power station, nearly half of the events were traced to deficiencies in control computer software. These deficiencies were about evenly split between changes introduced since software implementation and problems with code design.

Given the goals of this research, the software reliability program in FY 81 was broken into interrelated tasks:

1. Review the literature for and discuss with experts in the field known mathematical approaches and associated data bases for quantifying software reliability. Investigate this information as to its applicability for this program.

2. Develop a mathematical approach that will meet the specific goals outlined above. As will be described in Section 3, existing models have limited usefulness here. Further, to support the approach of this program, the existing data base will have to be greatly expanded. This is discussed in Section 4.

The work that was done to satisfy the first task, which included information exchanges and discussion with personnel in aerospace, defense, communications, and computer technologies, clearly pointed out the need for the quantitative analytic capability and associated information base that are the end products of this program.

The remainder of this report discusses the progress that has been made on the program to date. Section 2 of this report briefly discusses some of the design and development considerations that enter into the production of reliable software. Organizations instrumental in establishing software standards in the nuclear industry are cited. Finally, problems in the validation and verification of nuclear-safety-related software are discussed. It is intended to give the reader insight into qualitative reliability characteristics of computer software.

Section 3 discusses the functional requirements and desired results of this program in mathematical forms. The mathematical reliability models that have been published are described. Obtaining and analyzing failure data with these models to understand their underlying bases was a significant part of the FY81 research. The role of these models in assessing nuclear software is described in light of our studies with them. A mathematical approach that will meet the quantitative requirements of this program is outlined with descriptions of the necessary supporting mathematical and statistical models and information. The implementation of this approach is next described.

The most significant problem in developing mathematical models to predict software reliability characteristics is the lack of failure data upon which to base and subsequently test these models. As will be seen in Section 3, published models do not explicitly treat code characteristics such as complexity, testing history, and application, which are critical to a code's probability of successful operation. Further, published models were developed to evaluate the reliability of continuously running codes such as those used in continuous data acquisition and generally predict failure rates vs. time, a parameter which has little significance in evaluating the failure probability of an analysis code in a specific execution. The accumulation of non-nuclear and nuclear software failure data, both published and unpublished, as well as the initiation of documentation of related failure data within ANL is being pursued with the objective to categorize code failure data according to the general code properties. This categorization is necessary for research goal number 1 to be achieved. Section 4 reports progress to date and future efforts in accumulating and generating these data.

Section 5 discusses initial results of analyzing published data to obtain preliminary but illustrative reliability estimates. Far more extensive studies will be performed but the flavor of the studies is shown in Section 5. This section also describes some very brief results of preliminary surveys to obtain subjective estimates of software reliability among computer code developers and users. Because the data to date are preliminary, unverified, and possibly non-representative of related nuclear software, these results are presented only for illustration.

Section 6 outlines the directions that should be followed to best achieve the goals of this research program. Specific research paths and tasks are suggested with the end product of each described.

References

1.1 S. Fabic, "Code Assessment for Nuclear Reactor Accident Analysis Programs," Trans. Am. Nucl. Soc., 35, 254 (1980).

1.2 J. G. Kemeny (Chairman), "Report of the President's Commission on the Accident at Three Mile Island," U.S. Government Printino Office, Washington, D.C., October (1979).

1.3 A. B. Long, et al., "A Methodology for the Development and Validation of Critical Software for Nuclear Power Plants," 1st International Computer Software and Applications Conference, IEEE COMSAC 77, 620 (1977).

1.4 C. V. Ramamoorthy, et al., "A Systematic Approach to the Development and Validation of Critical Software for Nuclear Power Plants," Proceedings 4th International Conference on Software Engineering, 231 (1979).

1.5 Combustion Engineering, Inc., "Assessment of the Accuracy of PWR Safety System Actuation as Performed by the Core Protection Calculators," CENPD-170, July (1975).

1.6 A. B. Long, "Technical Assessment of Disturbance Analysis Systems," Nuclear Safety, Vol. 21-1, 38 (1980).

1.7 L. Magagna, "Control Computer Systems in Generating Stations in Ontario Hydro Design Approach and Experience," 1976 EEI Engineering Computer Forum, 155 (1976).

2.0 Considerations in the Development of Reliable Software

This section briefly discusses some of the design and development considerations that enter into the development of reliable software. Organizations instrumental in establishing software standards, especially in the nuclear industry, are cited. Finally, comments regarding the validation and verification of codes are offered. It is seen that safety codes in the nuclear industry pose special problems.

The initial phase of software development, as depicted in Figure 1, emphasizes the structure of software, with the assumption that well structured or modular software results in improved reliability. By simplifying logic and program control, program functions become easier to understand and consequently, testing of such software is facilitated [2.1,2.2,2.3].

In the fast reactor physics community, there is an ongoing, major effort to implement standardization procedures and guidelines to maximize the exchangeability of software, as proposed by the Committee on Computer Code Coordination (CCCC) [2.4] established by DOE. Recommendations of the CCCC also include standardization of utility subroutines, data storage, and input/output (I/O) files. This reduces the introduction of new errors (bugs) and the dependence of coding upon a specified computer, thereby facilitating exportability of software, exchangeability of information, and availability of documentation. The CCCC is strongly supported by the national laboratories and to a lesser extent by the nuclear industry.

Another approach to the development of reliable software is redundancy. This is a familiar concept in the hardware design of nuclear power plants. Application of a similar concept to fault-tolerant software development has been suggested by Randall [2.5]. Redundancy may be accomplished through code design rather than by simple replication of programs. For example, in the operation of nuclear power plants, redundancy in the on-line software can be provided by comparing computed signals with signals from the sensors located at the various parts of the plants. This approach provides independent validation of both the sensors and the software.

In the military, aerospace, and communication industries, where development of highly reliable software is critical, emphasis [2.6,2.7,2.8] has been focused on guidelines and standard procedures to achieve highly reliable software. In the nuclear field, efforts in software reliability have been also directed to developing or assuring reliable software as cited earlier.

In the nuclear energy field, standards committees and organizations have developed standards applicable to nuclear software as follows:

1. American National Standard Institute (ANSI)

ANSI X3.9-1966	American National Standard FORTRAN
ANSI/IEEE Std 730	IEEE Trial-Use Standard for Software Quality Assurance Plan

2. American Nuclear Society (ANS)

ANS-STD.3-1971	Recommended Programming Practices to Facilitate the Interchange of Digital Computer Programs
ANS-10.3/N413-1974	Guidelines for the Documentation of Digital Computer Programs (also approved by ANSI)

3. Nuclear Standards Management Center

Coordinates development of DOE nuclear program standards

4. Committee on Computer Code Coordination

LA-6941-MS	Standard Interface Files and Procedures for Reactor Physics, Version IV

5. Reactor Development Technology (RDT), now designated as Nuclear Energy (NE) Standards

RDT Std F1-4	Computer Coding, Documentation, and Distribution (Draft, 1975)

6. National Energy Software Center

7. Nuclear Energy Programs/Organizations

LA-7812-MS	Quality Assurance for TRAC Development
R0010-1001-SA-00	Argonne National Laboratory Quality Assurance Policy and Procedures Manual

Uniform application of these standards throughout the industry will help ensure reliable software. For the majority of nuclear codes specific approaches to software validation and verification (V&V) are adequate. Code validation and verification is normally accomplished by testing each of the several modules in a code and the code as a whole. Several possible modes of V&V are as follows:

1. Comparison of the results generated by the module to known analytic solutions to the equation sets solved in the module or appropriate simplification of these sets;
2. Comparison of the results generated by the module to the results of other computer codes that model the same phenomena;
3. Comparison of the results generated by the module to experimental results; and
4. Comparison of the results generated by the module to the "expected behavior" of the models it contains.

Although such comparisons may validate portions of a code they rarely test the whole system. It is important to note that testing can guarantee the presence of, but not the absence of, bugs.

There are codes, however, especially in the risk assessment area for which this procedure is not totally satisfactory because:

1. Either analytical solutions do not exist or an inability to place bounds on uncertainty associated with data makes the solutions inconclusive.

2. No other codes of this type exist.

3. Experimental results do not exist or are unavailable.

4. The driving phenomena are not understood well enough to compare the results with "expected behavior".

Clearly there is a need to develop additional procedures and methods to assess reliability of codes which fall in the category of the unverifiable.

SOFTWARE RELIABILITY

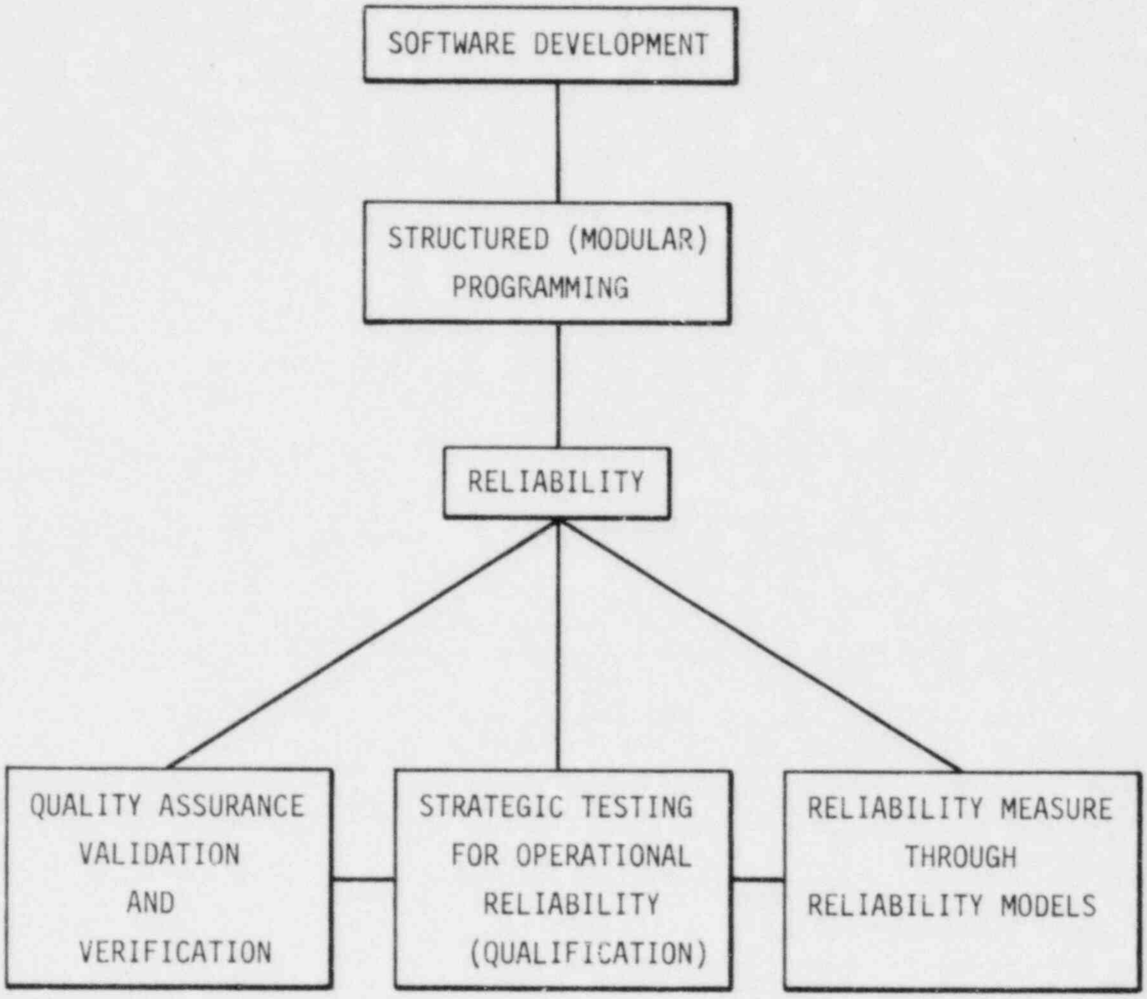

Figure 2.1 A Structure for Developing Reliable Software

References

2.1 E. C. Nelson, "Software Reliability," TRW-SS-75-05, (1975).

2.2 B. W. Boehm, "Software and Its Impact: A Quantitative Assessment," Datamation, May (1973).

2.3 H. D. Mills, "Software Development," IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, (1976).

2.4 R. D. O'Dell, "Standard Interface Files and Procedures for Reactor Physics, Version IV," LA-6941-MS, (1977).

2.5 B. Randall, "System Structure for Software Fault Tolerance," Proceedings of International Conference on Reliable Software (1975).

2.6 "Final Report of the Joint Logistics Commanders Software Workshop," Vol. 1, Army Material Development and Readiness Command, Naval Material Command, Air Force Logistics Command and Air Force Systems Command, October (1979).

2.7 A. D. Schuman, "Proposed Multi-Service Documentation Standards and Requirement," Panel Presentation at the IEEE Reliability and Maintainability Symposium, (1981).

2.8 "Software Quality Assurance Program Requirements," MIL-S-52779.

2.9 Record of the IEEE-NRC Working Conference on Advanced Electrotechnology Applications to Nuclear Power Plants, (1980).

2.10 J. S. Moore and L. Lamport, "Program Verification: An Approach to Reliable Hardware and Software," Trans. Am. Nuc. Soc., 35 (1980).

2.11 H. R. Downs, "Automated Tools for the Verification of Computer Programs," Trans. Am. Nucl. Soc., 35 (1980).

2.12 T. E. Dunn, "Development of a Standard for Computer Code Control and Verification," Trans. Am. Nuc. Soc., 35 (1980).

3.0 Quantitative Software Reliability Analysis

This section describes an analytic framework through which the goals of this research program, as described in Section 1, can be met. The mathematical approach that is formulated here is designed to be able to yield predictions of software probability or reliability characteristics for all types of computer codes relevant to nuclear safety. Codes of interest in nuclear safety applications fall into two broad categories: What might be called discrete task codes, exemplified by analysis codes such as RELAP or TRAC, are run to accomplish a specified task and then stopped for examination of computational results. The second category consists of continuous task codes exemplified by data acquisition codes such as might be used to provide a reactor operator with a continuous display of data from an operating reactor. Because these two types of codes have conceptually different measures of reliability, it is important to formulate an analytic framework which encompasses both.

Section 3.1 discusses the functional requirements of this analytic framework and the form of the desired results. It begins with some basic definitions and then describes mathematical parameters that can be used to measure the probabilities of successful or unsuccessful operation of computer codes in nuclear applications. These probabilities are affected by the way failures are categorized and counted, by the specific properties and usage of the computer code, and by the techniques used in developing and testing the code. The concept of classifying the failure, computer code property, and code development parameters that affect these probabilities is described and examples are shown. The potential use of regression analyses to assess the functional dependencies between the calculated software probability characteristics and these parameters is cited.

Section 3.2 describes mathematical modeling that can be used to fill out the analytical framework. The section begins with a brief review of failure rate models which have been used extensively for more than ten years in communications, aerospace, and defense applications. The applicability that these models have in the mathematical framework of this program is discussed in view of the special needs of the nuclear industry. It will be seen that the approach used in the models published to date have rather limited usefulness in meeting the goals of this program because they have been primarily designed for and applied to calculating the reliability characteristics of the so called continuous task codes. Modeling that comprises the analytic framework which is proposed here to encompass both the discrete and continuous task codes is then described. Possible model extensions that may prove useful in the longer term are proposed and the concepts and models needed to support or augment this framework are discussed.

Finally, Section 3.3 summarizes data requirements needed to implement the models described in Section 3.2. Some alternative approaches which may be attempted in the absence of the desired data are also outlined, and the use of standard statistical analysis packages to obtain correlations between probabilistic code parameters and other code characteristics is briefly discussed.

3.1 Functional Requirements and Desired Results

Before describing the functional requirements and desired results of the analytical framework in detail, a few comments will be made regarding definitions. In the discussions that follow, reference will be made to the detection of code errors. A code error will be regarded as synonymous with a software error and is here defined as any defect in a line of code or in the input data which can cause the computer code to fail. Code failure includes abnormal termination, normal termination with erroneous results, or any other unacceptable departure of program operation from the required operation.

It may sometimes be convenient to refer to code faults rather than errors. For example, a computer code may compute an incorrect temperature. It may be necessary to correct several code errors to correct the temperature calculation, but the incorrect temperature could be regarded as the result of a single fault. As envisioned here, a code failure may be the result of one or more faults and a fault may involve one or more code errors. It should be possible to apply the framework described below for code errors to code faults; however, the numerical values of pertinent parameters would change.

The first step in formulating the mathematical approach for this research program is to clearly delineate the form of the desired results. Probabilistic or reliability characteristics that can be used to describe the successful or unsuccessful operation of a code are described in Table 3.1. These characteristics depend on the generic type of code. Codes are categorized according to whether they have discrete tasks or continuous tasks. As seen in the table, the operational mode distinction means that for analysis codes the desired code probability characteristics are "per run or execution" oriented, i.e., an estimate of the probability of failure is desired for a given run or set of runs as would occur in a safety assessment; while for data acquisition codes, the desired probability characteristics are "execution time oriented", i.e., an estimate of the probability of failure is desired for a given length of time for a specified demand.

Perhaps the single, most important code property is its reliability, i.e., the probability that the code executes without failure on a given run (discrete task code) or over a specified time period (continuous task code). There does not appear to be a uniquely useful means to estimate the reliability of a discrete task code based on the reliability of a continuous task code or vice versa. The importance of these differences in code reliability characteristics takes two forms:

(1) The mathematical models used in estimating the desired characteristics and/or the techniques used in their implementation can be expected to be somewhat different depending on the type of code.

(2) The required (as well as available) failure data needed to verify these approaches are different.

Thus both the requirements, namely the mathematical framework and the supporting failure data, as well as the form of desired results, namely the predicted software failure probability characteristics, depend upon the type of computer code. It will be seen that all the published work to date on providing probability or reliability models has focused on one type of code, the continuous-running type.

A second quantity of considerable interest and one that is independent, from a mathematical standpoint, of the type of code is the expected number of code errors per line of code or more generally, per machine language instruction. While this quantity does not provide a direct indication of code reliability, it is a useful figure of merit by which to gauge the developmental progress of a computer code. In addition, it appears that estimates of this quantity would be similar for continuous and discrete task codes at similar stages of development.

As an aid to developing the correlations between probability-of-failure characteristics and code properties, it is desirable to classify errors according to error type using one or more classification systems. Table 3.2 shows one method of categorizing errors, failures, and code properties. In this table code execution failures are categorized according to their consequences and their software error causal mechanisms; the controlling code properties and software development characteristics that are assumed to affect the failure frequency are categorized as shown. The detail in the table is presented for completeness--it will be shown in this report that actual failure data are not detailed enough to support estimates of failure probabilities due to a specific faulty code unit, for example. However, by coarsely segregating the failure data according to potential consequence and probable cause, probability-of-failure characteristics may be categorized. For example, assuming sufficient failure data can be obtained, the probability of either a random (e.g., keypunch) or logical (e.g., incorrect implementation of an equation) error causing an execution failure of a specified type may be obtained. Although the current paucity of data precludes anything but gross estimates of such probabilities, even these can be used to make bounding estimates relevant to nuclear safety.

In general, two or three classification systems should probably be used simultaneously. In setting up error classification systems, the following general rules should be followed. First, error types should be mutually exclusive within a single classification system. Second, the number of errors of one type should be independent of the number of errors of other types within the same classification system. Third, ground rules should be clearly stated regarding how to count errors. Finally, when more than one classification system is in use, each system should result in the same total number of errors being counted.

Although, in general, it may be desirable to set up classifications that are not programming language specific, the following example illustrates how a single classification system for a program written in FORTRAN might be set up. Error types are defined according to whether they occur in

1. Input Data,
2. Arithmetic or Logical Assignment Statements,
3. Input or Output Statements,

4. Data Initialization Statements,
5. Specification Statements,
6. Subprogram Statements,
7. Control Statements.

A specific FORTRAN manual would be identified to resolve questions as to what FORTRAN statements fall into each of these types. In addition, one might stipulate that statements that are out of order or misnumbered would be counted as control statement errors and that more than one error in a single statement would be counted as a single error. The question as to whether the number of errors of one type is independent of the number of errors of other types with this classification system is not obvious. One may simply have to assume independence with the understanding that whenever possible, statistical tests will be performed to test the assumption.

Tabulating the controlling variables associated with each code and its failure history must be done to obtain the dependencies among the probability-of-failure characteristics and the code properties. A statistically meaningful number of codes must be investigated so that these characteristics can be correlated. Obtaining these failure data is considered the most difficult part of this program and is discussed in Section 4. Given that the appropriate failure data can be obtained and probability characteristics calculated, then regression analyses can be performed to identify the most important properties and controlling variables in software development. This knowledge could be used as a basis for defining development or quality assurance criteria for codes used or proposed for the licensing arena.

To summarize the above, the desired results include:

(1) Quantitative estimates of the key probability or reliability characteristics describing the successful or unsuccessful operation of a code. Table 3.1 illustrates some of these characteristics.

(2) Quantitative estimates of the dependence of these characteristics on the type of failure, on the type and generic properties of the code, and on the controlling variables in the development of the code so that probabilistic predictions in terms of these parameters may be made. Table 3.2 illustrates several classifications of failure, code, and development parameters that may be used to facilitate correlation with the probability characteristics to seek the functional dependencies.

The quantitative requirements include not only the mathematical models but the associated data base with which to accomplish these. The modeling is described in detail in Section 3.2. Failure data are discussed in Section 4.

3.2 Mathematical Modeling

The modeling to be described here is designed to fill out the analytic framework discussed in the previous section. This framework is designed to encompass predictions for the software probability or reliability characteristics for both discrete task codes and continuous task codes. The probability characteristics of interest in discrete task codes are the probability of failure parameters shown in Table 3.1 for a given run or set of runs; the analogous parameters of interest for continuous task codes are for a given duration of time for a specific application. An example of a parameter of interest in the former case is the probability that a run or set of runs using the RELAP code in a reactor safety analysis successfully (within the constraints of the modeling) predicts peak fuel temperatures for a postulated loss-of-coolant scenario. An example of a parameter of interest in the latter case is the probability that an on-line code successfully interprets sensor data and outputs proper analytically-derived temperatures, pressures, or other state parameters to a reactor operator during an emergency situation. In this latter case, clearly the time that the emergency situation required that the on-line software operate successfully would affect the probability of successful operation. As the TMI situation graphically illustrated, this time scale could be for many hours.

These examples imply that the required mathematical models and the associated failure data depend somewhat upon the type of code. However, once the appropriate probability characteristics have been obtained for each generic type of code, the analyses to be used for obtaining the dependencies on non-probabilistic code parameters such as those illustrated in Table 3.2 are the same.

The mathematical models currently used in software reliability analysis are based on failure rate models such as those used in hardware reliability applications. Given that an error of type k in classification j has not occurred prior to time t, the probability that such an error occurs between t and t + dt is expressed as

$$\lambda_{jk}(t)dt \tag{1}$$

where λ_{jk} is the detection rate for errors of type k in classification j and is, in general, a function of time. The probability that an error occurs between time t and t + dt, given no error prior to time t is expressed as

$$\lambda(t)dt \tag{2}$$

where

$$\lambda(t) = \sum_k \lambda_{jk}(t). \tag{3}$$

Here, $\lambda(t)$ is the detection rate for errors of all types. (The subscript j is not required after summation over all error types k since all classification systems are postulated to count the same total number of errors.) From the definition of $\lambda(t)$ a simple derivation shown in Appendix A yields the reliability

$$R(t) = \exp\left\{ -\int_0^t dt'\lambda(t') \right\}. \qquad (4)$$

Work to date has focused almost entirely on estimating $\lambda(t)$ for continuous task codes. Attempts to break $\lambda(t)$ into components of various types as in (3) are not apparent. Several models have been suggested to express λ as a function of parameters such as the number of errors present in the code, the number of errors discovered, and the probability that an error is detected given that it is present. These models are described in Appendix A. In most of the models, the detection rate is not regarded as a function of time; however, in the Littlewood-Verrall model, the parameter in the exponential distribution is regarded as a random variable having a gamma distribution. This leads to a time-dependent detection rate in (4).

A major fraction of the computer codes of interest in this research program are discrete task codes. So long as errors occur infrequently, it should be possible to apply failure rate models to tnese codes provided the time is the accumulated CPU time for all runs, and provided the codes are run on identical computers with identical compilers. However, detection rates obtained for a code operation on one computer are not necessarily the same as detection rates for operation on another computer. This is especially important in the case of nuclear safety analysis codes which are often developed on a single computing system but are then exported to different computing systems throughout the country. In addition, if errors are detected too frequently, as might be the case for a code undergoing a new phase of testing, the run time assigned to an individual task might impose a structure on the error detection time which has little to do with the actual error occurrence rate. For these reasons, it is necessary to develop more comprehensive modeling which is more naturally applied to discrete task codes and yet can also be applied to continuous task codes.

The modeling which will now be described can be used to meet these dual applications. Let X_{jk} be a random variable defined as the number of errors of type k in classification system j detected as the result of a given run. The expected value of X_{jk} will be denoted by Λ_{jk}. The probability distribution for X_{jk} is assumed to be Poisson, i.e.

$$P(X_{jk} = x) = \frac{\Lambda_{jk}^{x}}{x!} e^{-\Lambda_{jk}} \qquad (5)$$

where x is a non-negative integer. If the values for X_{jk} are independent for each type k and if the error types are mutually exclusive, then it is easy to show that the probability distribution for the total number of errors X, defined as

$$X = \sum_k X_{jk}, \qquad (6)$$

is also Poisson, i.e.

$$P(X = x) = \frac{\Lambda^x}{x!} e^{-\Lambda} \ , \tag{7}$$

where

$$\Lambda = \sum_k \Lambda_{jk} \ . \tag{8}$$

With this model, the reliability is

$$R = P(X = 0) = e^{-\Lambda} \ . \tag{9}$$

The use of the Poisson distribution has certain mathematical advantages. First, the Poisson distribution is often used to approximate the binomial distribution. There may be instances where it is useful to model the detection of errors of type k in terms of M_{jk}, the number of code units (lines of code, instructions, or some other unit) capable of producing type-k errors, and the probability q_{jk} that an error of type k is detected in a given unit during a given run. If q_{jk} is the same for each unit and if error detection in each unit is independent of error detection in other units, then the probability distribution for X_{jk} will be binomial with parameters M_{jk} and q_{jk}. In general, M_{jk} is likely to be large and q_{jk} small. The binomial distribution can then be approximated by the Poisson distribution in (5) with

$$\Lambda_{jk} = M_{jk} \, q_{jk} \ . \tag{10}$$

In (10) failure data may be used to estimate Λ_{jk} or q_{jk} directly; some other source such as expert opinion or the human error literature, using analogous error probabilities to estimate q_{jk}, may also be used. Of course, the value of M_{jk} is provided from the code properties or specifications.

A second advantage arises for codes that are conveniently described in terms of error detection rates $\lambda_{jk}(t)$ as in (1). For runs of duration t, the probability distribution for X_{jk} is easily seen to be given by the Poisson distribution in (5) with

$$\Lambda_{jk} = \int_0^t dt' \, \lambda_{jk}(t') \ . \tag{11}$$

The reliability estimates (4) and (9) become identical in this case.

Finally, for a given error type and classification system, only one parameter needs to be estimated for the Poisson distribution in (5). The accumulation of appropriate failure data is needed to verify whether values of Λ_{jk} obtained for a code operating on one computing system can be applied to the same code when it operates on a different computing system.

This discussion of reliability modeling will conclude with a dicussion of a few potentially useful extensions to the Poisson model described above. Reference will be made only to the total number of errors; however, there does not appear to be any a priori reason why the same developments could not be applied to individual error types.

Table 3.1 suggests that probability distributions for important parameters describing probabilistic code characteristics should be estimated if possible. In the case of the Poisson model, the important characteristic is the expected number of errors detected in a given run or equivalently the probability q_{jk} of error detection. Let $f(\Lambda)$ represent the probability density of the expected number of errors. The probability distribution of X, the number of errors occurring on a given run, taking into account the uncertainty in the value of Λ expressed by $f(\Lambda)$ can be computed as

$$P(X = x) = \int_0^\infty d\Lambda \; \frac{\Lambda^x e^{-\Lambda}}{x!} f(\Lambda) \tag{12}$$

where (12) becomes the replacement for (7). Generally $f(\Lambda)$ would be chosen from a parametric family of functions and (12) should be regarded as the conditional probability that $X = x$ given the appropriate parameter values. The dependence of $P(X = x)$ on such parameters should be understood but will not be explicitly noted.

It may or may not be possible to evaluate the integral in (12) analytically depending on the precise form of $f(\Lambda)$. In any event the integral can be evaluated numerically, perhaps using Monte Carlo techniques. Further, the use of Monte Carlo would allow the sampling of Λ from actual data without the need for determining an analytic form for $f(\Lambda)$ (or equivalently $f(q)$).

If an analytic form is desired, a particularly convenient choice for the function $f(\Lambda)$ would be to select it from the family of gamma distributions, i.e.

$$f(\Lambda) = \frac{\alpha^\beta}{\Gamma(\beta)} \Lambda^{\beta-1} e^{-\alpha\Lambda} \quad . \tag{13}$$

where $\Gamma(\beta)$ is the gamma function. With this choice, evaluation of the integral in (12) yields

$$P(X = x) = \frac{\alpha^\beta}{(\alpha + 1)^{\beta+x}} \frac{\Gamma(x+\beta)}{x! \, \Gamma(\beta)} \quad , \tag{14}$$

and the reliability becomes

$$R = \left(\frac{\alpha}{\alpha + 1}\right)^\beta . \tag{15}$$

The expected number of errors detected in a given run is found to be

$$E(X) = \frac{\beta}{\alpha} \tag{16}$$

and the variance of the number of errors detected is

$$\mathrm{Var}\ (X) = \frac{\beta}{\alpha}\ (1 + \frac{1}{\alpha})\ . \tag{17}$$

Section 1 states that one of the objectives of this program is to permit a priori estimates of a computer code's reliability given the code's general properties and its intended application. This implies that it will be possible to extrapolate data for codes of various types and for a variety of applications to a specific code and code application for which error detection data are not available. As one begins to apply the code, one may modify the reliability estimates based on experience in using the code. If the required data extrapolation is done in a way that provides estimates of the parameters α and β in (13), then the gamma distribution in (18) can be regarded as a prior distribution for a Bayesian analysis. If after running the code r times, one observes s errors, then a strightforward application of Bayes theorem leads to the following posterior distribution for Λ:

$$f(\Lambda|s,r) = \frac{(\alpha + r)^{\beta+s}}{\Gamma(\beta + s)}\quad \Lambda^{\beta+s-1}\quad e^{-(\alpha+r)\Lambda}\quad . \tag{18}$$

It then follows that updated estimates for the probability distribution of X, for the reliability, for the expected number of errors detected, and for the variance of the number of errors detected are given respectively by (14), (15), (16), and (17) with α replaced by α+r and β replaced by β+s.

The discussion up to this point has been concerned with methods to estimate the reliability of a computer code. However, as noted in Section 3.1, the expected number of code errors per instruction or per line of code is also a useful figure of merit for which estimates are desirable. Many of the failure rate models described in Appendix A provide estimates of this quantity as a result of attempting to model the way in which error detection and correction modifies the error detection rate. It is possible to devise analogous models to describe the influence of error detection and correction on the expected number of errors. A by-product of such models might be estimates of the expected number of code errors per instruction. Since such models have not been investigated, the following approach was developed:

An examination of failure data such as that of Musa as listed in Appendix B suggests a general picture in which, over a specified testing phase, the error discovery rate accelerates early in the testing phase, increases at an approximately linear rate during the central part of the testing phase, and then decelerates during the later stages. Qualitatively, such behavior might be anticipated. Early in the testing phase, the more obvious errors may result in relatively short code runs and exposure of relatively small portions of the code. As these errors are eliminated, the runs are likely to increase in length and a larger fraction of the code will exercised. Late in the testing phase, only the more subtle errors remain and the rate of error detection decreases. A convenient measure of progress through the testing period appears to be the time in working days.

To model this behavior, a relatively simple process may be envisioned. Let N be the total number of errors present in the code and $n(t)$ be the expected number of errors detected in t working days. (Introduction of new errors during error correction is assumed to be negligible.) If one defines

$$x = \frac{n - \frac{N}{2}}{\frac{N}{2}}, \tag{19}$$

then during the early part of the testing phase, x is negative; during the central part, x is nearly zero; and during the later part x approaches unity. Now let t_o be the time when $x = 0$ and define

$$y = k(t - t_o) \tag{20}$$

where k is a constant to be specified later. The error detection rate will be proportional to dx/dy. A relatively straightfoward way to model the qualitative behavior of the error detection rate, as described in the preceding paragraph, is to let

$$\frac{dx}{dy} = 1 - x^2 . \tag{21}$$

This equation clearly predicts an accelerating error detection rate early in the testing period and a decelerating rate late in the period.

If (21) is solved subject to the initial condition $x = 0$ when $y = 0$, one finds

$$x = \tanh y.$$

Substituting from (19) and (20), the following expression for $n(t)$ results.

$$n(t) = \frac{N}{2} [1 + \tanh k(t - t_o)]. \tag{22}$$

A least squares fit of (22) to failure data then provides estimates of the parameters N, k, and t_o.

Equation (21) assumes that the error detection rate is an even function of $t - t_o$. This assumption can be relaxed, e.g. by subtracting a term proportional to x on the right side of (21). The solution of the resulting equation leads to a result similar to (22), namely

$$n(t) = a + b \tanh k(t - t_o),$$

but there are now four constants to be determined rather than three. Further analysis is required to determine whether the additional flexibility provided by a fourth constant justifies the additional computational effort required to determine four constants instead of three.

The estimate of N, the total number of errors present in the code, as obtained through the use of (22) provides a means to estimate the number of errors in the code per line of instruction at any stage during the test phase. If this quantity is designated as η, one finds

$$\eta = \frac{N - n}{I} \tag{23}$$

where I is the number of lines of instructions.

In summary, modeling has been developed to calculate reliabilities and expected numbers of errors detected for both continuous task and discrete task codes. In addition, a means of estimating the expected number of errors per line of instruction has been developed. The implementation of these models to produce probability of code failure estimates useful in nuclear safety is illustrated in the next section.

3.3 Implementation

The implementation of the Poisson model described in Section 3.2 requires a procedure for estimating the expected number of errors per run* (discrete task codes) or during a specified period of operating time (continuous task codes). If one is interested only in the reliability, classification of errors detected is not essential; however, classifying errors promotes precision in the counting of errors. Further, the consequences of code failure as categorized, for example, in Table 3.2, obviously depend on the type of causal software error mechanism.

Data requirements are apparent from the discussion of the model. Ideally, one would like to have records of appropriately classified errors detected as a function of the number of the run (or time period in the case of continuous task codes) in which they were found. Data in this form would permit the direct investigation of the dependence of the expected number of errors on run number and error detection and correction. Such data should be recorded for codes following their release for general use; however, they will probably be more readily obtainable during the testing period just prior to release. It should be possible to make reasonable estimates of code reliability in the post-release period using data collected just prior to release, but, whenever possible, these estimates should be checked against post release data. Obtaining generic correlations between pre- and post-release data has been identified in this report as a means of expanding the usefulness of both data bases.

Data in the form just described have not been found for discrete task codes. Generally records of error corrections are recorded as a function of the date when the corrections were made. Sources of these data are discussed in Section 4. These data can be used to estimate the expected number of errors per run provided reasonable estimates can be made for the total number of runs involved in producing the data; however, meaningful analysis of the changes in the expected number of errors as errors are detected and corrected will be more difficult.

Another possible approach to the estimation of the expected number of errors per run is based on the observation that code errors are human errors. To use this approach, classification of errors would be essential. An estimate of the number of code units M_{jk} (see Eq. (10) in Section 3.2) capable of producing errors of a given type would be made. Then the human steps required to produce these units would be analyzed and human factor data would be used to estimate the probability that these steps were carried out incorrectly. Finally, the code structure would be examined to estimate the probability that one of the code units would be encountered during a run. The product of this probability and the probability that the code unit is produced incorrectly would then provide an estimate of the probability q_{jk} in Eq. (10) in Section 3.2.

---

*Since the expected number of errors per run is given by $\Lambda = qM$, where the notation in the previous section is used and M is specified, knowledge of Λ implies knowledge of q and vice-versa. Thus, the discussion relating estimation of Λ applies equally well to estimation of q.

Still another means of estimating the expected number of errors per run is through consultation with experts. For example, assume that for a certain set of conditions the probability that a code would incorrectly compute an important safety parameter is estimated to lie between 1/20 and 1/100 with 90% certainty. Further assume that this probability is estimated to be equally likely to be greater than 1/20 or less than 1/100. One way to use this information is to fix the parameters in a probability distribution for Λ, the expected number of errors per run. If the gamma distribution in Eq. (13) of Section 3.2 is used for convenience of illustration, $\alpha = 165$ and $\beta = 4.48$ and using Eq. (14), the probability of failure is estimated to be

$$P(X \geq 1) = 0.03 \pm 0.01,$$

where 0.01 represents one standard deviation. Although some type of combined subjective/statistical procedure may be necessary in the absence of data it is clearly preferable to estimate the expected number of errors Λ or the probability distribution for Λ directly from data.

Data requirements for estimating the number of errors per machine language instruction in a code are similar to those described above for the Poisson model. However, it is not necessary to have error detection records as a function of run number or to even know the number of runs. Records in terms of calendar days are adequate. For discrete task codes, the sigmoidal fault discovery model (Eq. (22) in Section 3.2) can be used to estimate the number of errors present or remaining in a code in terms of the number of errors detected. For continuous task codes, the sigmoidal model and the failure rate models of Appendix A appear to work about equally well. However, both the failure rate models and the sigmoidal model are shown in Section 5 to have a tendency to predict a total number of errors present in a code which is only slightly larger than the number of errors actually detected. Thus, if any of these models are used during code testing, the predicted number of errors remaining in the code cannot be used as an acceptability criterion for when code testing can be stopped. Some other quantity, such as the mean number of days between error detections, must be used instead.

Once estimates of the expected number of errors or the number of errors per machine language instruction have been made for several codes, correlation of these values with other code characteristics can be attempted. Some of the characteristics of potential interest in this correlation include code size, complexity, testing history, and other properties shown in Table 3.2. The nature of the relationships of the probabilistic code parameters to these characteristics can be explored using the multivariate regression analysis capabilities found in many standard statistical analysis code packages. Some examples of such analyses using the BMDP system described in Appendix C are shown in Section 5.

In summary, approaches or means to obtaining the relevant parameters in the analytic framework have been described. Sample calculations using the failure rate models and the sigmoidal model to predict error content and the probability of a line of code being in error are shown in Section 5. Calculations using the Poisson model will be performed using estimates of q_{jk} or Λ_{jk} obtained from these models as well as the other sources described above.

Table 3.1 Key Software Probability and Reliability Characteristics

Generic Type of Code	Applications	Key Probability and Reliability Characteristics[1,2]	Comments
Discrete task codes (Analysis codes)	Reactor design Safety assessment	Probability that a specified unit of code is in error[2]. Probability of an output error of a specified type in a given run. Probability of success (failure) in obtaining a key result in a given run(s). Probability that a specified unit of code produces an execution failure in a given run. Probability distributions or at least confidence or uncertainty bands for the above characteristics.	Key reliability characteristics are "per execution" oriented. Appropriate failure data include a history of failures vs. total runs.
Continuous task codes (Data acquisition codes)	Feedback to operator Input to simulator Operational control Emergency system activation	Probability that a specified unit of code is in error[2]. Probability of success or failure over a specified time (i.e., reliability or unreliability) during a specified demand. Failure rate. Probability distributions or at least confidence or uncertainty bands for the above characteristics.	Key reliability characteristics depend heavily on time. Appropriate failure data include a history of time to (between) failures.

[1]All probability and reliability characteristics may be categorized and normalized according to the type of failure they measure, the failure causes, and consequences of failure.

[2]Probabilities are defined both for actual code operation failures and for code component errors that have the potential to cause operation failures.

Table 3.2 Categories[1] of Operational Failures, Causal Mechanisms, and Controlling Properties and Development Variables in Software Reliability

Operational Failures	Causal Mechanism			Controlling Properties and Developmental Variables	
Type of Consequence	Faulty Code Unit	Faulty Modeling Unit	Type of Cause		
1. No output (i.e. a crash) 2. Absurd output (detectable) 3. Misleading output (may propagate failures in nuclear plant design or operation)	1. Symbol 2. Operand 3. Constant 4. Variable 5. Line (statement) 6. Storage byte or array (e.g. common block) 7. Logic block (subroutine) 8. Logical directive (e.g. IF statement)	1. Constant 2. Variable 3. Equation 4. Model (set of equations) 5. Table (e.g. property values)	In-Code[2] 1. Random (e.g. key punch) 2. Logical or decisional (incorrect or poor programming to implement modeling) Ex-Code 3. Documentation leading to misuse of code	Testing 1. Quality assurance methods 2. Size of QA effort 3. Capability for benchmarking or analytic verification Operational 1. Application (e.g. design, safety assessment, monitoring, control) 2. History (time, numbers of runs and failures since release)	Programming 1. Size 2. Complexity 3. Language 4. Structure and method of coding

[1]This table illustrates a plausible way of categorizing failures, their causes, and dominant variables. Other ways could be chosen.

[2]In-code causes could also be subdivided as follows: 1) "Pure" programming causes such as syntax errors or incorrect transfers of program control, logic, or data; 2) Program modeling causes such as incorrect algorithm approximations or improper treatment of singular points or critical parameters leading to overflow and roundoff errors.

4.0 Software Failure Data

This section describes the failure data accumulated to date and how they and future data have been or will be used to support the goals of this program. Sections 4.1 and 4.2 describe the non-nuclear and nuclear software failure data, respectively, that have been or are expected to be obtained in this program. Section 4.3 discusses conclusi ns on what should be done to satisfy prog am goals.

4.1 Non-nuclear Data

The collection of the non-nuclear data has been made for two reasons:

(1) To initiate mathematical modeling and investigation as early as possible. The available non-nuclear data has been in a much more readily useable form than the nuclear data. Tabulations of failure histories are available that could be directly used in investigating or formulating mathematical models. On the other hand, the nuclear data have been "buried" in logged records that require evaluation and conversion to a form adaptable to manipulation with mathematical models. Thus, the detailed investigation of these records has been deferred. However, it is a necessary component of this program.

(2) To form as wide a data base as possible. Although the applicability of non-nuclear software failure data to nuclear software remains to be shown, comparisons can be made. If the data are directly applicable the statistical data base for predicting software probability of failure characteristics is obviously enhanced. If the data are not applicable, comparing the data will be useful in evaluating the software development and application characteristics of nuclear codes to determine why nuclear and non-nuclear failure data are different.

Non-nuclear software failure data have been and are being compiled from the open literature, Rome Air Development Center (RADC) and private companies. RADC, at Griffiss Air Force Base, maintains an extensive literature and computer library data-base in software reliability based on software failure data from military, aerospace and communication industries. Most of the failure history data come from data acquisition systems rather than batch mode systems.

Failure history data are generally available in either of two forms: (1) individual failure tabulations showing successive execution times between failures or (2) blocked data in the form (n,t), n errors in time t where t may be in any unit of time including execution time. Although the former failure data form may be reduced to the form (n,t), the converse is not true. Most of the data to be presented in this section were obtained during system testing and integration and represent both faults, defined [4.1] as software defects that cause an operational failure, and failures, defined as any unacceptable departure of program operation from program requirements. In some cases, the failure data are referred to as "errors" or "changes" which are ill-defined but presumably represent faults or failures as just defined. Although the intent of this program is to predict probability of failure characteristics for codes in production status, not test status, it is assumed that extrapolation to production code failure probabilities can be made. Further, these data allow us to explore the mathematical approach of Section 3 while more relevant data are being acquired.

Compilations of failure data [4.2] in the first form from 16 systems ranging in their applications from military to real time command and control systems were obtained from Rome Air Development Center (RADC). These data are illustrated in Table 4.1 and comprise Appendix B. They are given in terms of successive execution times between failures in seconds. Descriptions of the systems from which the data were obtained are shown in Table 4.2. Clearly, these data represent a diversity of systems and system characteristics. Accordingly, as described elsewhere in this report, these data were used to investigate different models that predict probability of failure characteristics and to perform regression analyses to illustrate use of the statistical analysis systems.

Several software project studies, representing sets of failure data in either the above form or the (n,t) form, were obtained from the literature and studied to gain an insight into both qualitative and quantitative characteristics of software development and reliability. Data from these studies and a brief summary of each are as follows:

(1) The data shown in Table 4.3, taken from Brooks and Motley [4.3], were taken from projects involved in the development of a large scale command and control software system. These data represent a variety of system sizes; higher order programming languages JOVIAL and CENTRAN were used in addition to the assembly language ALC. The parameters N and q represent estimates of the total number of errors in the system at the initiation of the project and the probability that any one of these errors would be detected during a specified portion of the testing period. As described in the table, the lack of a quantitative specification for this test period implies that the estimate for q has no quantifiable significance. Another highlight of these data is illustrated in Projects 4 through 7, in which different error accounting methods were used with grossly different error probability characteristics estimated. The implication of course is that only crude error probabilities can be inferred from published failure data unless the definitions of errors, faults, failures and so forth are very clearly made.

(2) The data shown in Table 4.4, taken from Goel and Okumoto [4.4], were obtained from a real-time control system for a land-based radar system developed by Raytheon Co. Nearly all of the modules are written in JOVIAL/J3. One observation that can be made from these data is that the ratio of errors introduced in fixing the code to the total number of errors is very small ($\sim$ 2%). If data from other codes support this observation, reintroduction of errors does not constitute a significant contribution to code failure probability. Of course, this may not hold true for other codes. For large scale safety codes that routinely transfer large amounts of information among modules, correction of a code defect in one module may lead to the discovery of defects within modules with which it interfaces.

(3) The data shown in Table 4.5, taken from Goel [4.5], are from the real-time, multicomputer complex which forms the core of the Naval Tactical Data System (NTDS). The programming language(s) used is not specified. These data show the scatter in observed times to failure that is evident in much of the failure data in the literature. As observed in the last test phase, they also show that a new testing phase is likely to result in an enhanced detection rate even in a release version of a code.

(4) The data shown in Table 4.6, investigated by Shooman [4.6] but obtained by Dirkson, Hesse and Kientz [4.7], are from three operating systems subsequent to release and from four application systems during the system integration. These data yield estimates of the number of code defects corrected per line of instruction for the given programs. They imply that the given systems averaged roughly 1 (observable) defect per 100 instructions (as estimated using the Shooman model discussed in Appendix A) prior to the initiation of data collection. These data also show that this defect percentage is significantly reduced by the end of the observation period. However, the scatter in the data indicate that quantification of this reduction will have large uncertainties.

(5) The data shown in Table 4.7, taken from Akiyama [4.8], are from a system whose function is unspecified and written in FASP, the assembler language for FACOM 230-60. Data were available from all phases of software development from module testing to field use. These data represent a base upon which to explore relationships among error detection and the indicated code properties. Obviously nothing meaningful can be obtained with just these data; however, this table does represent an actual example of the type of data needed to draw correlations among probability of failure characteristics (here measured by number of errors detected) and code properties.

The software from which the above failure data were obtained were not written in FORTRAN, the principal programming language used in nuclear codes. Further, as mentioned above, the functions and usage of these software systems are sufficiently different that the applicability of the above data to nuclear systems needs to be questioned. However, they provide not only an insight into qualitative and quantitative characteristics of software systems but a part of the non-nuclear data base for comparison with future data from nuclear systems as well as a data base for testing the mathematical approach of Section 3.

4.2 Nuclear Data

This section discusses specific sources that will be tapped to provide a nuclear software failure data base. As related above, efforts in obtaining and appropriately categorizing nuclear software failure data have been deferred. Such data have only been found in logged records and are often in a form associating dates or time durations with code changes that may or may not have been made to correct a detected code bug which in turn may or may not have been revealed by an operational failure of the code. Nevertheless, the investigation and interpretation of such data is needed to provide a nuclear data base for this program.

One promising nuclear failure data base is that of the ARC (Argonne Reactor Computation) System maintained by the Applied Physics Division at ANL. Roughly 125 codes ranging in size from 2,000 to 50,000 FORTRAN statements comprise this data base which represents both pre-production and production codes. Production as used here refers to codes that are in unlimited use within as well as outside ANL. Pre-production refers to codes in varying stages of testing but prior to release to this unlimited user status. The ARC System was initiated in the mid 1960's and has been in continuous development since that time. Production use of the codes in the system has been under way since about 1970. Although the format of this base is as indicated in the preceding paragraph, implying that considerable time and effort will be required to properly evaluate it, the information gained from this evaluation as well as the data obtained to support the regression analyses of Section 3 justify these expeditures.

Actually the Applied Physics Division will provide data yielding two types of information. The first type is given by the data generated to support the ARC System requirements, namely that all changes to codes within the system and the reasons for these changes, be documented on a standard form. The second type is given by data from individual code developers who maintain their own records, generally in a more detailed form. These data have been generated for the more recent codes, some of which are not yet part of the ARC System, which have been written to standards set by the Committee on Computer Code Coordination (CCCC) established by DOE. By being more specific as to type of bug in some cases, this second type will facilitate categorizing; however, since all codes written by CCCC standards were not documented in this manner, this type comprises less of a code data base than the first type of information.

Two examples of codes for which these types of information have been obtained (but not yet investigated) are the GNIP4C and SYN3D codes. The first is a 20,000 word input processor used to arrange geometry and nuclear data for other codes within the ARC System. The data for this code is considered pre-production data since the GNIP4C code is not available throughout the industry. The second code is a 10,000 card, three dimensional, flux-synthesis, diffusion theory code used in eigenvalue calculations for criticals analyses and burnup calculations. Data for the SYN3D code include both pre-production and post release data.

Actual logging of errors has occurred throughout the nuclear industry. For example, extensive software error records have been kept at Los Alamos National Laboratory and EG&G on their large safety codes for many years. These records take two forms: 1) written records containing information relating the nature of bugs found and how and when they were corrected; 2) computer update files containing when and how changes due to software errors and other modifications and additions were implemented. For at least one of these codes, the latter is available on microfiche. These data will be explored to facilitate the categorization of both errors and failures.

Although it is planned to survey the information available throughout the industry, early detailed data evaluation and investigation will probably be limited to ANL and LANL sources to determine the needs to be filled by additional investigations. The data will be investigated for all possible nuclear applications to explore interdisciplinary differences, if any exist. For example, it is likely though not proven, that deterministic physics codes are more "reliable" than safety codes because of their greater ease of benchmarking. Structural analysis codes provide another nuclear application and should be similarly segregated. In summary, extrapolations of results from one type of code to another must be supported by actual failure data.

4.3 Conclusions on Data Needs and Future Acquisition Efforts

The above sections lead us to draw several conclusions relating to data needs and efforts to acquire these data:

(1) All data obtained to date comprise only a small part of the information base needed to achieve the goals of this program.

(2) The acquisition and appropriate categorization of nuclear software failure data will be a time-consuming but necessary component of this program. Categorizing the bugs according to the types of code defect consequence and cause categories is required as discussed in Section 3. To accomplish this will undoubtedly require some subjective categorization. Extended categorizations come to mind: for example, an important concern for nuclear safety is with what probability could an operational failure of a code or a documented code bug result in a misleading code result upon which an unsafe design decision or incorrect safety conclusion is made. No data to estimate this propagation probability have been found or indeed can be assumed to exist. Thus, expert opinions would have to be solicited and factored in to assign such a probability.

(3) The data obtained have been and will be largely from testing phases of codes. Post-release or production status data will be sought. The aforementioned nuclear data sources contain both preproduction and production status data as discussed. Clearly the goals of this program relate to production codes, not test versions. Drawing comparisons between the failure probabilities of pre- and post-release codes must be done to obtain the benefits of a large pre-release data base. However, the mathematical approach to treating the data is the same.

(4) It would be extremely fruitful to establish a computerized data bank and information center at the National Energy Software Center at ANL, to properly organize the failure data, integrate it with the mathematical and statistical analysis systems described herein, and to facilitate industry-wide participation (as well as access) in data collection. This is discussed further in Section 6.

Table 4.1

Illustration of Failure Data: Successive Execution Times Between Failures in Seconds

Fault Number	Execution Time	Day of Failure
1	3,	1
2	30,	2
3	113,	9
4	81,	10
5	115,	11
6	9,	11
7	2,	17
8	91,	20
9	112,	20
10	15,	20
11	138,	20
12	50,	20
13	77,	20
14	24,	20
15	108,	20
16	88,	20
17	670,	30
18	120,	30
19	26,	30
20	114,	30
21	325,	30
22	55,	30
23	242,	31
24	68,	31
25	422,	31
26	180,	32
27	10,	32
28	1146,	33
29	600,	34
30	15,	42
31	36,	42
32	4,	46
33	0,	46
34	8,	46
35	227,	46
36	65,	46
37	176,	46
38	58,	46
39	457,	47
40	300,	47
41	97,	47
42	263,	47
43	452,	53
44	255,	53
45	197,	54
46	193,	54
47	6,	54
48	79,	54
49	816,	56
50	1351,	56
51	148,	56
52	21,	57
53	233,	57
54	134,	57
55	357,	57
56	193,	59
57	236,	59
58	31,	59
59	369,	59
60	748,	59
61	0,	59
62	232,	59
63	330,	59
64	365,	61
65	1222,	62
66	543,	63
67	10,	63
68	16,	63
69	529,	64
70	379,	64
71	44,	64
72	129,	64
73	810,	64
74	290,	64
75	300,	64
76	529,	65
77	281,	65
78	160,	65
79	828,	66
80	1011,	66
81	445,	66
82	296,	66
83	1755,	67
84	1064,	67
85	1783,	68
86	860,	68
87	983,	68
88	707,	69
89	33,	69
90	868,	69
91	724,	69
92	2323,	70
93	2930,	71
94	1461,	72
95	843,	72
96	12,	72
97	261,	72
98	1800,	73
99	865,	73
100	1435,	74
101	30,	74
102	143,	74
103	108,	74
104	0,	74
105	3110,	75
106	1247,	76
107	943,	76
108	700,	76
109	875,	77
110	245,	77
111	729,	77
112	1897,	78
113	447,	79
114	386,	79
115	446,	79
116	122,	79
117	990,	79
118	948,	80
119	1082,	80
120	22,	80
121	75,	80
122	482,	80
123	5509,	81
124	100,	81
125	10,	81
126	1071,	82
127	371,	83
128	790,	83
129	6150,	83
130	3321,	83
131	1045,	84
132	648,	84
133	5485,	87
134	1160,	87
135	1864,	88
136	4116,	92

Table 4.2

Characteristics of Software Systems Studied by Musa

System No.	Phase for Which[3] Data Available	Total Instructions/ New or Modified Instructions	Programmers	Faults[4,5,6]
1	S,0	21700/ 19500	9	136 (2)[7]
2	S,0	27700/ 6600	5	54 (2)
3	S,0	23400/ 11600	6	38 (1)
4	S,0	33500/ 9000	7	53 (3)
5[1]	S*	244500	275	831
6	SS	5700	8	73
14C	0*	-	110	36
17	S	61900	8	38
27	S	126100	8	41
40	S	180000	8	101
SS1A[2]	0*	-	-	112
SS1B[2]	0*	-	-	375
SS1C[2]	0*	-	-	277
SS2	0*	-	-	192
SS3	0*	-	-	278
SS4	0*	-	-	196

[1]Design changes amounting to about 21% of the source lines of code were introduced after failure 288.

[2]SS1A, SS1B, and SS1C represent the same software system running in slightly different environments.

[3]Phases:
- SS = Subsystem Test, module testing
- S = System Test; after modules or subprograms are integrated into a larger system
- O = Operational; System Running in the Operational Environment.

Data is for a complete phase unless starred.

[4]A fault is a software defect that causes the failure.

[5]If a failure recurred before the fault that caused it was corrected, it was not counted.

[6]When a failure was spawned as the result of correcting an earlier failure, it was counted as a new failure.

[7]Data are for the system testing phase unless in parentheses; data for the operational testing phase are shown in parentheses.

Table 4.3

Characteristics of Software Systems Studied by Brooks and Motley

Project	Number of Instructions	Language	Phase[1]	Errors Detected	N[3]	q[3]
1	1317K	ALC, CENTRAN	T & I	2657	3771	1.17×10^{-4}
2	124K	JOVIAL, J3	T & I	1301	1438	1.27×10^{-3}
3	80K 40K	ALC JOVIAL, J3B	T & I	1239	3094	2.07×10^{-3}
4[2]	114K	JOVIAL, J4	T & I, post-integration testing	1138	1348	.1166
5[2]	-	-	-	1483	1824	.1060
6[2]	-	-	-	2707	3958	.0739
7[2]	-	-	-	2362	3446	.0742

[1]The test and integration phase refers to a period of testing during and after the integration of two or more programs into a single system. Data from Projects 4 through 7 encompassed this phase as well as an acceptance testing phase.

[2]Projects 4, 5, 6 and 7 constitute one set of error data counted four different ways and illustrate a need for standardized terminology. (Four attempts to remove different types of non-software related errors from the total set of SPRs (software problem reports) reported during testing by different people according to their own definition of a valid software error).

[3]N and q denote estimates, using the Brooks-Motley model discussed in Appendix A, of the total number of detectable code errors at the beginning of testing and the probability of any single error being detected during a unit of test effort. Since this unit of effort was not defined in the referenced literature, the physical significance of "q" and even the comparability of q estimates among projects in this table cannot be specified.

Table 4.4

Comparative Histories of Errors Introduced in Debugging and Total Errors Detected as Analyzed by Goel and Okumoto

Month	Errors Detected	Errors Caused by Imperfect Debugging
1	122	6
2	98	1
3	82	1
4	75	3
5	113	2
6	85	3
7	105	2
8	47	1
9	61	1
10	25	1
11	28	0
12	42	1
13	18	0
14	17	0
15	28	0
16	14	0
17	5	0
18	3	0
19	3	0
20	13	0
21	5	0
22	10	0
TOTAL	999	22

Table 4.5

NTDS Error Data

Error Number	Time Between Errors in Days
Checkout Phase	
1	9
2	12
3	11
4	4
5	7
6	2
7	5
8	8
9	5
10	7
11	1
12	6
13	1
14	9
15	4
16	1
17	3
18	3
19	6
20	1
21	11
22	33
23	7
24	91
25	2
26	1
Test Phase	
27	87
28	47
29	12
30	9
31	135
User Phase	
32	258
Test Phase	
33	16
34	35

Table 4.6

Change Data and Code Defect Estimates for Large Scale Programs Studied by Dickson, Hesse, Kientz and Shooman

	Application A 240,000 Inst.		Application B 240,000 Inst.		Application C 240,000 Inst.		Application D 240,000 Inst.	
Month	Changes	Changes/Inst.	Changes	Changes/Inst.	Changes	Changes/Inst.	Changes	Changes/Inst.
1	514	2.15×10^{-3}	905	3.76×10^{-3}	235	0.98×10^{-3}	331	1.38×10^{-3}
2	926	3.85	376	1.57	398	1.66	396	1.66
3	754	3.15	362	1.51	297	1.24	269	1.12
4	662	2.76	192	0.80	506	2.11	296	1.24
5	308	1.28	70	0.29	174	.72	314	1.31
6	108	0.45			55	.23	183	0.76
7					60	.25	158	0.66
8							368	1.54
9							337	1.41
10							249	1.02
11							166	0.69
12							108	0.45
13							31	0.13

	Supervisory A 210,000 Inst.		Supervisory B 240,000 Inst.		Supervisory C 230,000 Inst.	
Month	Changes	Changes/Inst.	Changes	Changes/Inst.	Changes	Changes/Inst.
1	110	0.52×10^{-3}	250	1.04×10^{-3}	225	0.98×10^{-3}
2	238	1.14	520	2.16	287	1.24
3	185	0.88	430	1.80	497	2.16
4	425	2.02	300	1.25	400	1.74
5	325	1.55	170	0.71	180	0.78
6	37	.18	120	0.50	50	0.22
7	5	.02	60	0.25		
8			40	0.17		

Table 4.6 (continued)

Code Defect Parameters Estimated by Shooman Model

Program	I_T	E_T/I_T	ρ_o
Supervisory A	210K	6.14×10^{-3}	0.875×10^{-3}
Supervisory B	240	7.97	0.996
Supervisory C	230	7.48	1.25
Application A	240	13.20	2.20
Application B	240	7.70	1.54
Application C	240	7.00	1.00
Application D	240	12.90	0.995
Average		8.92	1.26

E_T = Estimated total number of errors at initiation of data collection.

I_T = Number of machine language instructions in the system.

ρ_o = Number of errors/number of instructions/month, averaged over the entire duration of time in months.

Table 4.7

Relationship Between Errors and Program Properties
Program as Reported by F. Akiyama

Module	Size[1]	Program Properties[2] D	J	D+J	Total Detected Number of Errors	Errors Detected During Different Phases[3] DB 1	AT 2	BT(1) 1	BT(2) 1	F 2
MA	4,032	372	283	655	100	58	25	11	4	2
MB	1,329	215	44	259	18	9	4	4	0	1
MC	5,453	552	362	914	93	78	6	8	0	1
MD	1,674	111	130	241	26	21	0	4	1	0
ME	2,051	315	197	512	71	54	8	5	1	3
MF	2,513	217	186	403	37	21	9	6	1	0
MG	699	104	32	136	16					
MH	3,792	233	110	343	50					
MX	3,412	416	230	646	80					
TOTAL	24,955	2535	1574	4109	493					

[1]Number of machine language instructions.

[2]D: number of decision symbols.
J: number of subroutine call symbols.

[3]DB: testing individual modules.
AT: subsystem connection test.
BT: integration test before release.
F: field.

Duration of these phases are shown in month.

References

4.1 J. D. Musa, "Validity of Execution-Time Theory of Software Reliability," IEEE Transaction on Reliability, Vol. R-28, No. 3, August 1979.

4.2 J. D. Musa, "Software Reliability Data," Data and Analysis Center for Software, 312 (1980).

4.3 W. D. Brooks and R. W. Motley, "Analysis of Discrete Software Reliability Models," RADC-TR-80-84 (1980).

4.4 A. K. Goel and K. Okumoto, "Bayesian Software Prediction Models," Vol. V, RADC-TR-78-155, (1978).

4.5 A. L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," IEEE Transactions on Reliability, Vol. R-28, No. 3, August 1979.

4.6 M. L. Shooman, "Probabilistic Models for Software Reliability Prediction, Statistical Computer Performance Evaluation, W. Freiberger, Ed., Academic Press, New York, 1972.

4.7 J. D. Dickson, J. L. Hesse, A. C. Kientz, and M. L. Shooman, "Quantitative Analysis of Software Reliability," Proceedings 1972 Annual Reliability and Maintainability Symposium, IEEE, 1972.

4.8 F. Akiyama, "An Example of Software System Debugging," IFIP, 71, Ljubljana, Yugoslavia, 1971.

5.0 Summary of Results

This section presents illustrative results of software failure probability characteristics generated using the analytic tools described in Section 3. Section 5.1 reports results using models supporting the probabilistic framework described in Section 3.2. Section 5.2 reports illustrative regression analysis results on the correlations among failure probabilities and code variables. All these results are based on preliminary, incomplete, unverified, and possibly non-representative failure data and are presented only to illustrate the methodology. Section 5.3 describes estimates of software failure characteristics obtained from the literature and from informal surveys. It too is considered preliminary, incomplete, and possibly non-representative. However, it is used to illustrate that even in a very small sampling of observations and opinions, the estimated characteristics exhibit a wide uncertainty band and heavy dependence on code properties.

5.1 Illustrative Failure Probability Characteristics Results

This section describes results using the published failure rate models and the sigmoidal fault discovery model (Eq. (22) in Section 3.2) developed as part of this research program. Calculations were performed with these models to determine estimates of the number of software errors per line of instruction and the probability of code failure. As described above, the results here are presented to show the feasibility and usage of the methods, not to infer probabilistic results relevant to nuclear codes.

Studies performed using the published failure rate models described in Appendix A provided quantitative insight into both the published mathematical methods and the software failure data described in Section 4. Considerable effort was expended in understanding these models and the role they could play in the mathematical framework of this program. However, for brevity this section reports only a summary of the results obtained and representative illustrations of individual calculations.

First, a quantitative comparison of the above software reliability models was made to evaluate their relative merits. In particular, calculational results were obtained with the deterministic failure rate models of Jelinski-Moranda [5.1], Musa [5.2], Goel [5.3], and Brooks-Motley [5.4], and the stochastic model of Littlewood-Verrall [5.5], applied to the published data [5.6] shown in Table 4.1. An illustration of the results is presented in Figure 5.1 which shows the reliability as a function of code execution time. These reliability functions were obtained using the maximum likelihood estimates (MLE) of the individual model parameters based on the first 100 detected faults of Table 4.1. In all cases, the MLE equations were highly non-linear. Convergence to the solution of these non-linear equations was slow for many of the failure rate models, particularly for the stochastic model. A choice of initial values was critical in obtaining any solution, with a good set of initial values resulting in quickly converged solutions. However, as in any system of highly non-linear equations, it is not a trivial matter to determine whether the converged solution is unique.

For the results presented in Figure 5.1 and Tables 5.1 and 5.2, only the binominal distribution form of the Brooks-Motley model was treated; further, the error reduction factors, B in the Musa model and α in the Brooks-Motley model were assumed to be 1. The MLEs of the model parameters varied greatly depending upon over what range of failure data they were taken, as indicated in Table 5.1. However, for a given data set, Table 5.1 also shows that the analogous parameters for the different models agree reasonably well. For example, constants of proportionality ϕ in the Jelinski-Moranda model, $1/T_oN$ in the Musa model, b in the Goel model, and q in the Brooks-Motley model, are comparable for all cases. These reflect the probability of error detection and are seen to decrease with error detection as reliability improves.

Figure 5.1 and Table 5.1 reveal some of the generic traits of the failure rate models. Littlewood [5.8], by applying goodness-of-fit tests has shown that his stochastic model provides a better fit to the data than the deterministic models. In addition he observed that the stochastic model was more conservative in its estimate of the reliability. Figure 5.1, as well as other examples not reported here, substantiates Littlewood's observation regarding conservatism. A nonconservative feature of the deterministic models is illustrated in Table 5.1. The parameter $\hat{N}$ estimating the total number of errors in the system at the initiation of detection becomes larger as more fault data are factored into parameter estimation; i.e. $\hat{N}$ becomes larger as more errors are detected. Since the Littlewood-Verrall does not predict N per se, quantitative estimates of its conservatism are not readily made, although its conservatism relative to the deterministic models is shown in Figure 5.1.

Table 5.2 lists software failure probability characteristics of nine of the systems documented in Appendix B and studied with the Brooks-Motley binomial model. As was the case for the Table 4.1 data, $\hat{N}$, the estimated initial total number of faults in the system, is always just slightly larger than the observed number of faults. In one case, $\hat{N}$ was computed to be equal to the number of observations. Thus, it is clear that the underprediction of N is a characteristic of the failure rate models studied here, not simply a characteristic of a specific set of data.

To illustrate the use of the sigmoidal fault discovery model to determine the asymptotic number of faults for a given software system, the failure data of Appendix B were analyzed. As shown in Section 3, the model

$$n(t) = \frac{N}{2} \{1 + \tanh[k(t-t_o)]\} \qquad (1)$$

requires the estimation of the three parameters N, k, and t_o, with N being the measure of error content. Good initial estimates of the parameters must be supplied for successful regression to be performed. The PAR derivative free nonlinear regression module of the BMDP system was used in the present instance. This module is described in detail in Appendix C. Basically, PAR computes least square estimates of parameters using an iterative pseudo-Gauss-Newton algorithm. For the examples considered here, data near the end of the testing period were weighted more heavily than data near the beginning of the period.

Failure data for nine systems, namely Systems 1, 2, 3, 4, 5, 6, 7, 27, and 40 of Appendix B were analyzed. Figure 5.2 shows a comparison of the observed data and the values predicted with (1) for System 4. This system is a real time command and control software package of 33,500 lines extent, programmed by seven programmers with a total of 53 observed fault-producing failures over a 70-day period. Reasonable agreement is evident indicating that the assumptions underlying the model are supported by this example. An estimate of 55 total faults was obtained. This number is slightly larger than the number of faults actually found and is in reasonable agreement with failure rate model predictions.

Table 5.3 summarizes the values of N, k, and t_o for the nine systems to which the sigmoidal model was applied. The initial estimates used in the regression analysis were chosen as follows: The initial value for N was chosen to be the total number of faults observed during the testing period; the initial value of k was set equal to the reciprocal of the total number of observed faults; and t_o was set equal to the time when half the total observed faults were discovered. Comparison of the values obtained for N with the values listed in Table 5.2 for N shows that the sigmoidal model predicts a total number of faults somewhat larger than the total number of observed faults just as do the failure rate models. These results indicate that the sigmoidal model is a satisfactory alternative to the failure rate models for predicting N.

5.2 Illustrative Regression Analysis Results Correlating Software Failure Probabilities with Code Properties

To illustrate the use of regression analyses to correlate failure probability characteristics with code properties and development variables, the exercise described herein was performed. A log-linear form for the error content per line of instruction, or alternatively the probability per line of instruction that a software fault would cause code failure was assumed as follows:

$$\ln \eta = a_0 + a_1 t + a_2 I + a_3 P. \tag{2}$$

where

t = time in working days,
I = number of lines of instruction,
P = number of code programmers,

and

$$\eta = (N-n)/I, \tag{3}$$

where

N = the estimated number of total errors

and

n = the observed number of errors.

The choice of independent variables in (2) was determined by the information available about the systems on which the data were taken. The choice of time in working days, rather than time between current and previous failure was made because time in working days is thought to more correctly reflect debugging activity and measure of level of software exercise and fault detection. The choice of the number of lines of instructions as an independent variable was made because a nonlinear relationship between the number of faults and the number of lines of instructions was expected. The choice of the number of programmers was made because the information was available and the presence (or absence) of a multitude of programmers working on any software system can obviously affect the production, detection, and correlation of faults. Choice of the log-linear form for η was made to accommodate the several orders of magnitude change expected in η.

Fits of the log-linear expression were accomplished using BMDP's P1R multiple linear regression module, described in detail in Appendix C. Basically, P1R computes a multiple linear regression equation on all data and on groups or subsets of the data. The parameter a_0 in (2) can be set to zero prior to the analysis if desired. If a grouping variable is specified to form groups, homogeneity of regression coefficients across groups is tested. It is also possible to specify case weights.

Figure 5.3 shows a comparison between results obtained with (2) and the observed data for the same nine data sets to which the sigmoidal model was applied. Table 5.4 lists the estimated values of the parameters a_i. Rather extreme variations occur at small values of time between the predicted and observed values (the observed values show a lot of scatter in this range as well). As a consequence, the "F" ratio relating fit to the resulting residuals is relatively high (F = 40). Figure 5.4 shows a lognormal plot of the residuals. The straight line represents the expected behavior if the residuals are normally distributed. Significant departures from a normal distribution are evident. The serial correlation of the residuals is relatively high indicating that (2) does not successfully remove the inherent trends in the data.

Of the three independent variables considered in (2), increases in two of the variables act to reduce η while increases in the third cause η to increase. Increasing the number of working days causes η to decrease reflecting successful debugging efforts. The decrease in η with the number of lines of instruction suggests that the number of faults in the code may not be proportional to the number of lines of instruction. The tendency for η to increase with the number of programmers suggests inefficient coordination of the programming effort. It should be remembered that these results are only illustrative. The fact that N increases with the number of observed faults confuses the interpretation of the results. Therefore definitive conclusions cannot be drawn from this analysis.

Again, the above work was done for illustration of the methods and no conclusions are to be drawn. However, the effects of the variables on error content satisfied "reasonableness". Further, the log-linear function used to describe fault probability per line of instruction should also prove useful in future analyses.

5.3 Surveyed Estimates of Software Failure Probability Characteristics

This section describes observations from the literature as well as solicitations of experts to obtain estimates of software failure probability characteristics. In particular, observations for both code-specific and generic frequencies of software errors or bugs per line of instruction are discussed. In addition, conclusions of a small sample of experts regarding the failure probabilities of large nuclear safety codes are presented.

Consider, first, the frequencies of errors per line of instruction. Boehm reported in a summary paper [5.10] that "each new release of OS/360 contains roughly 1000 software errors." OS/360 is an IBM 360 operating system software package consisting of roughly 300,000 to 500,000 machine instructions. This translates to $\sim 2 - 3 \times 10^{-3}$ errors per instruction. Boehm also reported on code development by IBM utilizing a variety of on-line programming tools, programming systems, and innovative structuring of the software to achieve highly reliable software. With this concept, an 83,000 instruction system for the *New York Times* was developed. During the testing phase 21 errors were found; since release 25 additional errors have been found. For this system, the minimum estimated number of errors per instruction at the time of release is given by $\sim 3 \times 10^{-4}$. Additional error discoveries will increase this estimate. Other published values include a value of 5×10^{-3} errors per machine instruction observed by Rubey [5.11] and a "historical-rule-of-thumb" value by Shooman [5.12] of 10^{-2} for the testing and integration phase. In an informal survey of approximately 10 code developers at ANL, all responded with basically the same rough estimate of 10^{-3} errors/line of coding for codes that are in production status. An informal survey was also conducted with roughly 10 widely published "experts" in the software reliability field. This survey yielded estimates of 3-12 faults/1000 machine instructions during integration testing, roughly 10^{-2} faults per instruction during system testing, and $1 - 2 \times 10^{-2}$ faults per instruction during module testing. There was a concensus of opinion that further generalizations of various probabilities would be dangerously misleading without first examining the systems in question. It was further opined that the post-release failure data depend strongly on testing effectiveness.

The quick survey discussed above produced estimates of 10^{-2} to 10^{-4} software errors per line of instruction; however, the stage of development of the codes was not always clear. Further, it could not be always discerned whether line of instruction referred to machine instruction or line of coding. Advanced programming languages such as FORTRAN generally normalize to 5-10 machine instructions per line of code. This value depends on the language as well as on the compilers in the computer hardware. Thus, probably the only conclusion that can be drawn here is that the number of software errors in a code is highly dependent on both the specific code properties and on the computer hardware on which the code is executed. As implied above, this conclusion was supported in the informal discussions held with the surveyed personnel.

Concerning failure probabilities of large safety codes, the dicussion was initiated with the question, "In your opinion, what is the probability that a large hypothetical accident code properly executes to provide a qualitatively correct estimate of an important parameter?" "Important" here was loosely defined as one upon which a safety conclusion might be based. "Properly executes" was defined to mean that the software was functioning correctly independent of whether the physical or phenomenological modeling was correct. A "qualitatively correct" result was defined to imply that conclusions drawn would not change because of deviation from "perfect". The conclusions were as follows:

1. For a specific code, the application of the code determined the probability. For example, the reliability of an accident analysis code would depend on the type of transient analyzed.

2. A safety code is much more reliable after widespread use than when it is released. A newly released code is not reliable.

3. The reliability of the code is strongly dependent on the expertise of the user. An inexperienced analyst is unreliable.

For an application for which the code has been "verified" by experience and the input prepared by an expert, opinions of the probability of successful operation generally were above 0.95, although only qualitative statements about confidence were made.

The overall conclusion to be drawn from these discussions is that a subjective estimate of the "generic" probability of failure of a large safety code in calculating a hypothetical accident has little meaning. Rather, the estimate would depend upon the code itself, the use of the code, the analyst applying the code, and the experimental history of the code. The need for quantifying the effects of these kinds of dependencies is clearly shown.

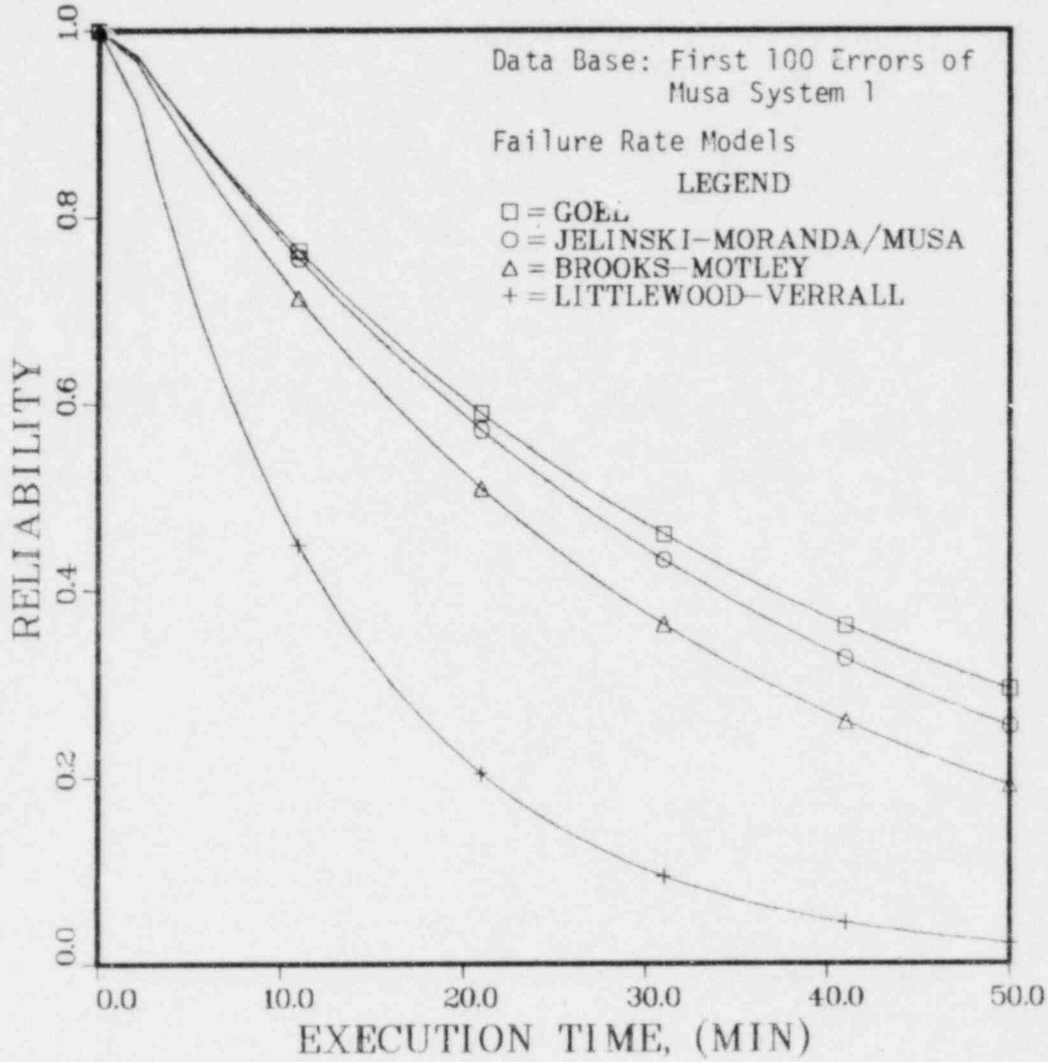

Figure 5.1 Illustrative Failure Rate Model Predictions of Software Reliability

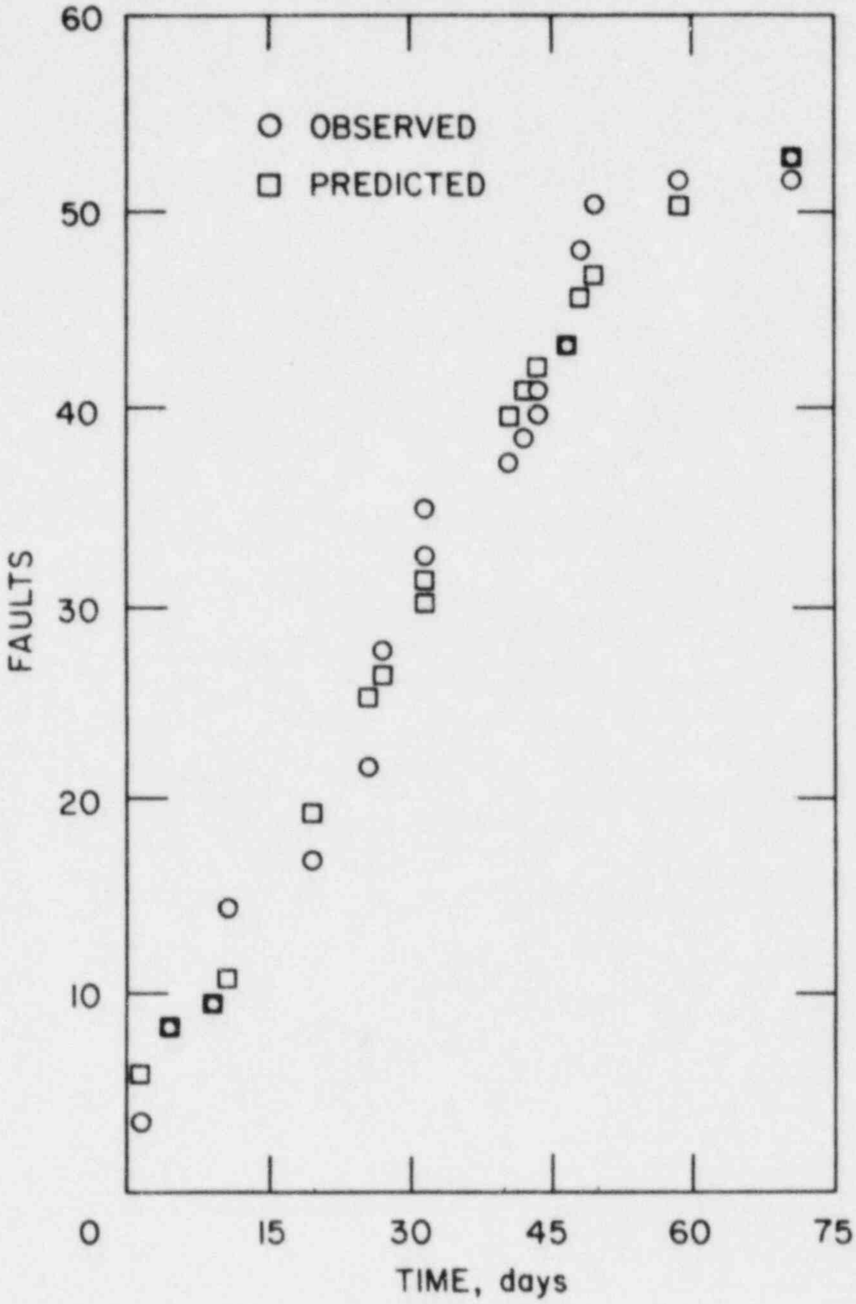

Figure 5.2 Sample Regression Results -- Detection of Faults vs. Time

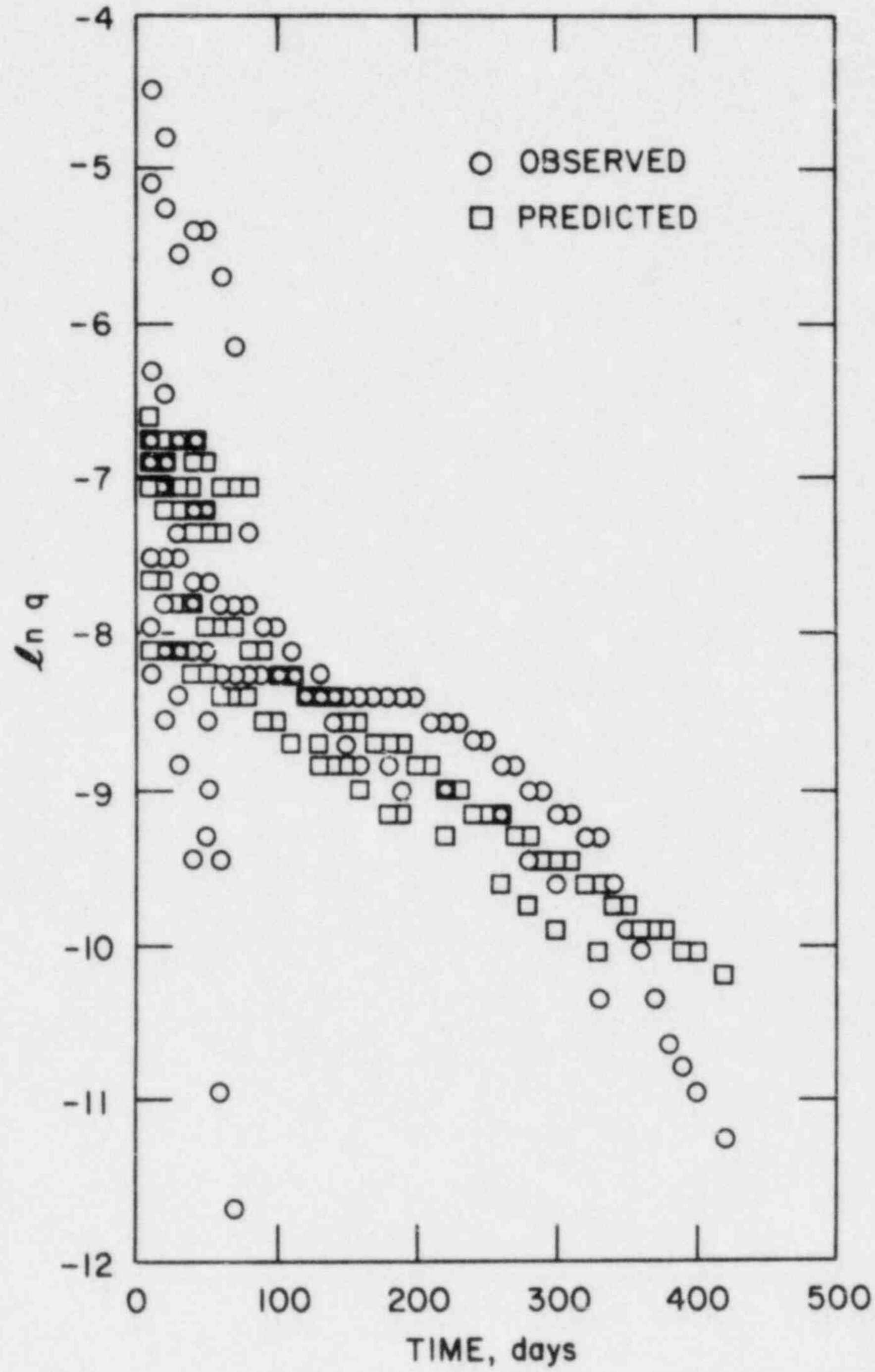

Figure 5.3 Sample Regression Results -- Time Behavior of Faults Per Line of Instruction

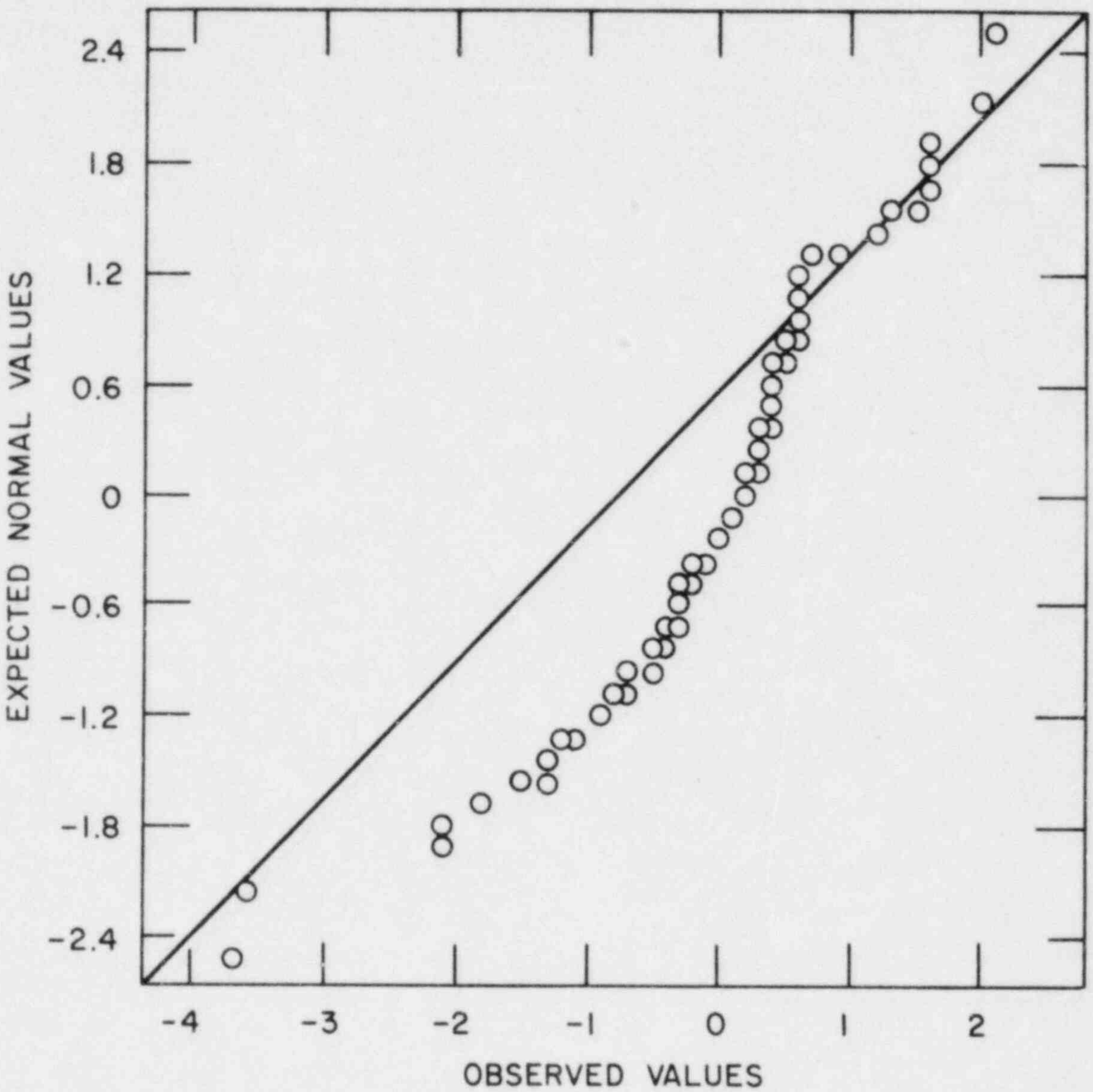

Figure 5.4 Sample Regression Results -- Log Normal Plot of Residuals

Table 5.1

Maximum Likelihood Estimates of Model Parameters[1] as a Function of Fault Detection Data

Fault Data Range	1-50		1-100		1-136	
	$\hat{N}$	$\hat{\phi},\hat{T}_o,\hat{b},\hat{q}$	$\hat{N}$	$\hat{\phi},\hat{T}_o,\hat{b},\hat{q}$	$\hat{N}$	$\hat{\phi},\hat{T}_o,\hat{b},\hat{q}$
Jelinski and Moranda	59	0.1807-3	106	0.6652-4	142	0.3482-4
Musa	59	93.787	106	141.812	142	202.270
		(0.1807-3)[2]		(0.6652-4)[2]		(0.3482-4)[2]
Goel	61	0.1667-3	107	0.6459-4	143	0.3409-4
Brooks and Motley	60	0.1784-3	109	0.6250-4	144	0.3382-4

[1] Definitions are as follows: $\hat{N}$ is an estimate of the number of errors in the code at the initiation of detection; $\hat{\phi}$, $\hat{T}_o$, $\hat{b}$, and $\hat{q}$ are parameters that measure or enter into the measure of error detection probability. Exact definitions are found in Appendix A.

[2] $\frac{1}{\hat{T}_o\hat{N}}$, which is comparable to $\hat{\phi}$, $\hat{b}$ and $\hat{q}$

Table 5.2 Estimated[1] Software Failure Probability Parameters[2] for Selected Data Systems

System	I/I*	N	N/I	$\hat{N}$	$\hat{N}$-N	$\hat{q}$	$\hat{q}(\hat{N}-N)$
1	21,000/19,500	136	6.5-3	142	6	3.5-5	2.1-4
2	27,700/6,600	54	1.9-3	56	2	2.9-5	5.8-5
3	23,400/11,600	38	1.6-3	38	0	6.8-5	0
4	33,500/9,000	53	1.6-3	61	8	6.2-5	5.0-4
5	244,500	831	3.4-3	900	69	8.9-8	6.1-6
6	5,700	73	1.3-2	91	18	3.2-2	5.8-1
17	61,900	38	6.1-4	40	2	1.2-5	2.4-5
27	126,100	41	3.3-4	43	2	7.5-7	1.5-6
40	180,000	101	5.6-4	103	2	2.1-7	4.2-7

[1]The Brooks-Motley failure rate model was used to provide the estimates.

[2]Parameter definitions:

- I: total number of instructions.
- I*: number of instructions that were modified during fault collection; this data was available only for Systems 1, 2, 3, and 4.
- N: actual number of faults detected when projection estimates were made.
- $\hat{N}$: estimated total number of faults at the initiation of fault data collection; obviously $\hat{N} \geq N$.
- $\hat{N}$-N: estimate of the number of faults remaining in the system that would be detected eventually; this is a measure of code failure potential.
- $\hat{q}$: the probability that, given a fault in the code, it will be detected in a unit of test effort (here 1 s);
- $\hat{q}(\hat{N}-N)$: the expected number of faults detected in a unit of test effort and is numerically equivalent to the probability of code failure in a unit of test effort (here 1 s) provided $\hat{q}(\hat{N}-N) << 1$.

Table 5.3

Parameters of Interest in the Fault Discovery Model

System	N	k	t_0
1	161 $\pm$ 4 faults	$3.76 \times 10^{-2} \pm 2 \times 10^{-3}$ day^{-1}	66 $\pm$ 1 days
2	53 $\pm$ 1	$5.00 \times 10^{-2} \pm 5 \times 10^{-3}$	35 $\pm$ 1
3	53 $\pm$ 4	$1.87 \times 10^{-2} \pm 4 \times 10^{-3}$	52 $\pm$ 14
4	55 $\pm$ 2	$3.83 \times 10^{-2} \pm 4 \times 10^{-3}$	29 $\pm$ 2
5	922 $\pm$ 64	$5.09 \times 10^{-3} \pm 4 \times 10^{-4}$	258 $\pm$ 14
6	77 $\pm$ 3	$4.44 \times 10^{-2} \pm 3 \times 10^{-3}$	23 $\pm$ 1
17	37 $\pm$ 1	$6.31 \times 10^{-2} \pm 4 \times 10^{-3}$	24 $\pm$ 1
27	42 $\pm$ 4	$2.39 \times 10^{-2} \pm 5 \times 10^{-3}$	29 $\pm$ 6
40	102 $\pm$ 3	$6.76 \times 10^{-3} \pm 1 \times 10^{-3}$	130 $\pm$ 10

Note: Quoted uncertainties represent one standard deviation.

Table 5.4

Parameters of Interest in the Log-Linear Probability Model

Coefficient	Value
a_0	-7.08329
a_1	$6.19 \times 10^{-3} \pm 1 \times 10^{-3}$ days^{-1}
a_2	$-8.10 \times 10^{-3} \pm 1 \times 10^{-3}$ (thousand lines)$^{-1}$
a_3	$6.98 \times 10^{-2} \pm 1.6 \times 10^{-2}$ (programmers)$^{-1}$

Note: Quoted uncertainties represent one standard deviation.

References

5.1 Z. Jelinski and P. Moranda, "Software Reliability Research," Statistical Computer Performance Evaluation, W. Freiberger, ed. Academic Press, New York, 465 (1972).

5.2 J. D. Musa, "A Theory of Software Reliability and its Application," IEEE Trans. on Software Engineering, Vol. SE-1, No. 3, 312 (1975).

5.3 A. L. Goel, "Software Error Detection Model with Applications," Journal of Systems and Software 1, 243 (1980).

5.4 W. D. Brooks and R. W. Motley, "Analysis of Discrete Software Reliability Models," RADC-TR-80-84 (1973).

5.5 B. Littlewood and J. L. Verrall, "A Bayesian Reliability Growth Model for Computer Software," 1973 IEEE Symposium on Computer Software Reliability, 70 (1973).

5.6 J. D. Musa, "Software Reliability Data," Data and Analysis Center for Software, (1980).

5.7 J. J. More, B. S. Garhom, and K. E. Hillstrom, "Users Guide for Minpack-1," ANL-80-74, 1980.

5.8 B. Littlewood, "A Critique of the Jelinski-Moranda Model for Software Reliability," 1981 Proceedings Annual Reliability and Maintainability Symposium, 357 (1981).

5.9 Private Communication, W. Vesely, January 15, 1981.

5.10 B. W. Boehm, "Software and Its Impact: A Quantitative Assessment," Datamation, 48, May (1973).

5.11 R. J. Rubey, "Quantitative Aspects of Software Validation," International Conference on Reliable Software, 246 (1975).

5.12 M. L. Shooman, "Software Reliability Data Analysis and Model Fitting," Workshop on Quantitative Software Models, 182 (1979).

6.0 Summary, Conclusions, and Future Directions

This section will summarize the preceding sections and outline the status and future directions of the tasks involved in this research program.

Section 2 discussed validation and verification of software. Although this was not a direct part of the FY 81 research program there is a clear need to compile a recommended list of procedures for validation and verification of nuclear safety software to provide the best assurance that code results that cannot be checked by conventional means are correct (within the framework of the modeling). Two examples of "unverifiable" results in the area of nuclear safety are presented for illustration: (1) An energy release from a postulated steam explosion in an LWR accident; and (2) A probability of a certain mode of failure in an LWR plant. In the first example, the phenomenology of large scale fuel coolant interactions is not known well enough to allow reasonable prediction even if the boundary and initial conditions for such an event could be accurately established (they cannot be). Thus, analytic and experimental benchmarks are not available and bounding considerations provide the only clue to reasonableness of results. In the second example, consider the specific case of a very complicated and uncertain system evaluated using a fault tree network and for which the predicted probability of a certain mode of failure is 10^{-15}. Here, not only do the system behavior uncertainties dominate, but the magnitude of the answer is beyond the analysts "intuition" for ascribing reasonableness to a result. Thus, an outline of ways to provide assurance that such "unverifiable" results are correct is needed.

Section 3 presented the mathematical approach of this program and Section 5 presented results generated using some of the supporting models. As stated throughout this report, these results were generated only to illustrate the methods and were based on an analysis of failure data that are preliminary, unverified, and possibly non-representative of software relevant to nuclear safety. Further, what has been done to date has been to estimate parameters that support the framework of this program. Actual implementation of the Poisson model to predict probabilities of failure and associated uncertainty bonds for nuclear codes remains to be done.

Specific areas within the approach that must be dealt with include the proper classification of software errors, faults, and failures both for data organization and subsequent regression analysis. With respect to the regression, the optimum choice of dependent variables needs to be established. For example, in the Poisson model, the characteristic probability q_{jk} is an obvious candidate. However, expected numbers of errors, i.e. $M_{jk}\, q_{jk}$ may in certain circumstances be amendable to regression. Finally, regression analysis may be performed directly on the reliability. Investigation of the number of errors per instruction in a code is useful since this quantity is a convenient figure of merit with which to measure the developmental progress of a code. A Bayesian approach to updating reliability estimates was also introduced.

Although all these concepts bear investigation, an overriding factor in this program is that the data that will become available in the near term, say several years, simply do not justify the development of sophisticated mathematical modeling. Thus, a pragmatic approach dictates that after the initial computer establishment of the framework, data acquisition efforts will dominate model development work.

Section 4 described the data acquired to date. Short term data collection efforts will be directed to nuclear codes. Other sources of relevant data include the human error literature and solicitation of "experts". However, this is a minor part of the overall data acquisition effort. A development effort that would unify these and subsequent data acquisition efforts is the creation of a readily accessible computerized information base and resource center. The components of this information base would include the necessary mathematical framework and associated models with which to make reliability predictions as well as the software failure data with which to verify and upgrade this framework. To facilitate access to and use of this information base with its concomitant analytic capability, as well as to provide a continuous updating capability that could accept failure data from throughout the industry, a Software Reliability Information Center and Computerized Data Bank and Analysis System is proposed. This would be established at the National Energy Software Center and could be made accessible to the nuclear industry. The need for this centralized data and analysis center has been emphatically demonstrated to our staff in our attempts to obtain documented code failure data. Such data have been logged by very few code developers in any consistent fashion in the nuclear industry and have been difficult to obtain. A failure data center should provide an industry-wide focus on software reliability, as well as providing a comprehensive, centrally located data base.

Section 5 described the results to date. As cited they were strictly used to illustrate the models. In the following year, gross estimates of reliability characteristics and their dependencies, similar to those provided, but based on an improved nuclear data base will be provided using the mathematical approach of this program. Tie-ins between the non-nuclear and nuclear data base will also be investigated.

In summary, a general mathematical framework for predicting reliability characteristics has been established. At the core of this framework is a Poisson model for the number of code errors of a specified type in a given computer application. Methods of implementing this framework have been developed and include both mathematical modeling and the proper specification and handling of data. The product of this framework includes predictions of reliability characteristics such as expected number of errors and probability of failure in a given computer run. By accumulating failure data and specifying the code characteristics associated with each data set, multivariate regression techniques can be used to identify the important code parameters and controlling developmental parameters. Actual computer implementation of this framework was started. Efforts in the near term have been described above.

Appendix A

Summary of Published Failure Rate Models

This appendix provides a review and summary of predictive mathematical software reliability models developed in the military, aerospace and communication industries. As a background for this review, the mathematical basis of software reliability is first provided. Summaries of the individual models are then provided with highlights discussed in a "comments" paragraph. Results of the application of a selected group of these models to failure data are presented in the text of this report.

In general the models discussed here assume functional forms for the code failure rate or the number of errors remaining in the code. These models are then fit to failure data to estimate, generally using maximum likelihood, values for the model parameters. The models are subdivided into:

(1) Deterministic models using error detection vs. time histories to evaluate the parameters of deterministic functions assumed for the failure rate;

(2) Deterministic models geared to similar parameter estimation using histories that measure numbers of errors in time intervals;

(3) A stochastic model that uses error detection vs. time histories to estimate the parameters of probability density functions assumed for the failure rate and time to failure.

Prior to the presentation of the models, a short introduction to the mathematics of reliability is provided to facilitate the discussion of the models. The reliability is the probability of successful operation over time t and is given as a function of time by

$$R(t) = P\ (\underset{\sim}{t} > t)$$

where $\underset{\sim}{t}$ is the time of failure. The probability of failure is given as the complement of reliability, namely

$$F(t) = 1 - R(t) = P\ (\underset{\sim}{t} < t) = \int_0^t f(x)dx,$$

where the failure density function f is given by

$$f(t) = \frac{dF(t)}{dt} = -\frac{dR(t)}{dt} \quad .$$

The hazard function or failure rate is defined by

$$Z(t) = \frac{P(t \leq \underset{\sim}{t} \leq t + dt)}{P(\underset{\sim}{t} > t)dt} = \frac{f(t)}{R(t)} = -\frac{1}{R(t)}\frac{dR(t)}{dt}$$

and is the conditional probability of failure in the interval $t < \underline{t} \leq t + dt$ given survival up to time t. From the definition of the hazard rate, the reliability function can be shown to be

$$R(t) = e^{-\int_0^t Z(x)dx}$$

The mean time to failure (MTTF) is given by

$$MTTF = \int_0^\infty tf(t)dt,$$

from which integrating by parts leads to

$$MTTF = \int_0^\infty R(t)dt.$$

In general, because software bugs are corrected as they are detected, the failure rate Z(t) is a decreasing function of code lifetime. This characteristic gives rise to a reliability function that increases with the life of the code.

Reliability models developed by Jelinski-Moranda (J-M) [A.1], Musa [A.2], Goel [A.3], Shooman [A.4], Schick-Wolverton [A.5], Brooks-Motley (B-M) [A.6], and Littlewood-Verrall (L-V) [A.7] are summarized in the following paragraphs. These models represent a group of the most frequently cited models in the literature. The first six of these models are considered deterministic because the failure rate or number of remaining errors is expressed as a deterministic function of a small number of parameters. As cited above, the deterministic models use either time-to-failure data or number-of-failures-in-a-time-interval data to estimate parameters. The Littlewood-Verrall model is considered stochastic because the failure rate is treated as a random variable with a probability density function dependent upon a small number of parameters. The Jelinski-Moranda model served as a basis for the development of other deterministic models. In fact, Musa, Shooman, Goel, Schick-Wolverton and Brooks-Motley models are often referred to as Jelinski-Moranda type models because of the similarity in the formulation of the hazard function. In all of these models, the hazard function is assumed to be proportional to the number of errors remaining. To facilitate reference to the literature the nomenclature used in the original papers is preserved in this report.

A.1 Deterministic Models Using Time-to-Failure Data

A.1.1 Jelinski-Moranda Model

$$Z(t_i) = \phi[N - (i - 1)]$$

where N = total number or errors initially in the system

ϕ = constant of proportionality

i = number of errors found in debugging time interval t_i

Comments

The Jelinski-Moranda (J-M) model assumes that each software bug has an equal probability of being detected, an unrealistic assumption since more easily-detectable errors are eliminated first. A potentially more serious drawback with this model is that the MLE (maximum likelihood estimate) may predict infinite values for model parameters.

A.1.2 M. L. Shooman Model

$$Z(t) = K[E_T/I_T - \varepsilon_c(\tau)] = K\,\varepsilon_r(\tau);$$

thus

$$R(t) = e^{-K\varepsilon_r(\tau)\,t}$$

and

$$\mathrm{MTTF}\,(\tau) = \frac{1}{K\left[\frac{E_T}{I_T} - \varepsilon_c(\tau)\right]}$$

where

E_T = total initial # of errors (unknown)

I_T = # of instructions

ε_c = # of errors corrected (normalized to I_T)

ε_r = # of errors remaining (unknown) normalized to I_T

τ = debugging time measured from the beginning of the system testing period

t = code operating time measured from the beginning of the system testing phase (may consist of many testing periods). In some cases $\tau = t$

K = constant of proportionality (unknown)

Comments

1) The Shooman model is essentially a J-M model where all parameters are normalized to the number of instructions.

2) The reliability function R(t) is defined for all $t \geq 0$. The time variable t is a measure of the operating time since the initial activation of the system and τ is a measure of the calendar time since the beginning of the system integration.

3) There are two unknown parameters, E_T and K, in R(t). To solve for these two parameters, the MTTF function, is evaluated at two different debugging times, τ_1 and τ_2, where $\tau_1 < \tau_2$ and $\varepsilon_c(\tau_1) < \varepsilon_c(\tau_2)$. With the computed E_T and K, R(t) can be evaluated for τ_3 for which ε_c is known; if ε_c is not known, an estimate must be made based on the previous data. The adequacy of this technique depends on the knowledge of the failure detection rate; the extrapolated value of R(t) based on τ_i for τ_n for large n may be very poor unless the failure detection rate is almost constant.

A.1.3 J. D. Musa Model

$$Z(\tau) = KfE_r = \frac{1}{T_o}\left[1 - \frac{i - 1}{M_o}\right]$$

τ = accumulative execution time

E_r = # of errors remaining

f = linear execution frequency,
$= \dfrac{\text{average instruction execution rate}}{\text{number of instructions in the program}}$

K = a constant of proportionality, the "error exposure ratio", which relates the error exposure frequency to f

i = number of errors detected

T_o = initial MTTF

M_o = N_o/B

where

N_o = number of initial errors

B = error reduction factor

= average ratio of the error reduction rate to the failure occurrence rate

Comments

1) The Musa model appears to be widely accepted in the field [A.8]. $Z(\tau)$ is proportional to the number of errors remaining and the linear execution frequency. The time variable τ in Musa's model is the accumulative execution time or the actual CPU time utilized in executing the software.

2) The parameter B can be adjusted to treat incorrect debugging (additional errors introduced by attempts to correct errors), as well as corrections for the learning process that the

programmers experience. The parameter B is postulated by Musa to be nearly constant for large projects due to averaging effects. This factor is usually positive and less than one; however, it can be greater than one when correcting one error leads to other errors. In practice, B is difficult to quantitatively determine. For B = 1, the Musa model reduces to the J-M model.

A.1.4 Schick and Wolverton Models

$$Z = \phi \left[N - (i - 1)\right] t_i; \qquad \text{S-W model}$$

$$Z = \phi \left[N - (i - 1)\right] (-at_i^2 + bt_i + c) \qquad \text{modified S-W model}$$

where

t_i = debugging time

ϕ = proportionality constant

N = initial error content

i = number of errors detected

a,b,c = positive constants

Comments

1) The Schick and Wolverton model is a modified J-M model to better account for the variation of the failure rate with time.

2) The modified Schick and Wolverton model expresses the time variable as a quadratic function, which implies that the failure rate initially increases to a maximum before becoming a decreasing function.

A.2 Deterministic Models Using Failures-in-Fixed-Time-Intervals Data

The models described above are used with error data characterized by recording the time between failures. The next two models express the hazard function in terms of the number of remaining errors as do the J-M type models, but are designed to use failure data in the form of number of failures in time t, (n,t) denoting the number of errors detected in time t. The model by Goel utilizes a nonhomogeneous Poisson process for software failure detection. The model by Brooks and Motley features the use of binomial and Poisson distributions (only the binomial distribution is presented here) to represent the number of errors in a test.

A.2.1 A. L. Goel Model

This model describes a nonhomogeneous Poisson process (NHPP) whose mean value function is derived by a deterministic analysis of the software failure process.

Let $n(t)$ be the cumulative number of software errors detected by time t and a be the total number of initial errors or the total number of errors to be detected eventually; then $n(t)$ has following properties

$$n(t) = \begin{cases} o & \text{when } t = 0 \\ a & \text{when } t = \infty \end{cases}$$

The number of errors detected in $(t, t + \Delta t)$ is assumed to be proportional to the number of remaining errors;

$$n(t + \Delta t) - n(t) = b(a - n(t))\Delta t$$

where b is a proportionality constant. Then

$$n(t) = a(1 - e^{-bt}) \quad .$$

Let $\{N(t), t \geq 0\}$ be a nonhomogeneous Poisson process (NHPP) describing the number of errors counted in $(0,t)$ and having the following properties:

1. $N(0) = 0$
2. $\{N(t), t \geq 0\}$ has independent increments
3. $P\{2 \text{ or more events in } (t, t+\Delta t)\} = o\ (\Delta t)$
4. $P\{\text{exactly 1 event in } (t, t+\Delta t)\} = \lambda(t)\Delta t + o(\Delta t);$

where $\lambda(t)$ is the hazard function and is called the intensity function. Further, let

$$m(t) = \int_o^t \lambda(s)ds.$$

Then for $t \geq 0$, $N(t)$ has the Poisson distribution

$$P[N(t) = y] = \frac{\{m(t)\}^y}{y!} e^{-m(t)}, \ y \geq 0$$

with the expected value of $N(t)$ given by

$$E[N(t)] = m(t);$$

$m(t)$ is called the mean value function of the NHPP. For the homogeneous case where λ is a constant, $m(t) = \lambda t$ and

$$P[N(t) = y] = \frac{\mu^y}{y!} e^{-\mu}$$

where the mean value $\mu = \lambda t$. Choosing the mean value function $m(t)$ to be equal to the cumulative number of errors $n(t)$ yields

$$m(t) = a(1 - e^{-bt}),$$

implying that

$$\lambda(t) = abe^{-bt}$$

and

$$R(t) = \exp\left[-\int_{s}^{s+t} \lambda(x)dx\right] = \exp[-a\{e^{-bs}-e^{-b(s+t)}\}],$$

where

s = time the last failure occurs

This conditional reliability function describes reliability since the last failure.

Comments

1. A deterministic model for $n(t)$ is required to obtain the functional representation of $m(t)$ which satisfies the NHPP conditions, rather than the converse.

2. This model analyzes data sequences of the form (ε_i, t_i) where ε_i is the number of failures in calendar time t_i.

A.2.2 Brooks and Motley Model

This model treats each program as a set of modules. The expected number of errors in the portion of the total system which is under test on occasion i can be found by taking the summation over all modules which are being tested. The estimate of the number of errors prior to test i for the portion of the system under test is given by

$$\bar{N}_i = \sum_{j\varepsilon J_i} (W_j N - \alpha N_{i-1,j})$$

$$= N \sum_{j\varepsilon J_i} W_j - \alpha \sum_{j\varepsilon J_i} N_{i-1,j}$$

where

J_i = the set of modules tested on occasion i

$\sum_{j\varepsilon J_i}$ ≡ summation over all modules that are elements of the set J_i

$\sum_{j\varepsilon J_i} W_j$ = the fraction of the system which is under test

N = number of errors in system before the first testing occasion

α = probability of correcting errors without reinserting additional errors

$N_{i-1,j} = \sum_{m=1}^{i-1} n_{mj}$ and is the number of errors actually detected in that portion of the system prior to the ith occasion.

If one defines

q = probability that any given error is detected during a unit of test effort

and

t_i = the system test effort on occasion i; it can be given as the execution (CPU) time, calendar time, number of tests, etc.,

then the probability of detecting any given error during the i-th test occasion will be

$$q_i = 1 - (1 - q)^{t_i} .$$

Since $\bar{N}_i$ errors are exposed during the i-th test occasion, the probability of detecting x_i errors will be given by the binomial distribution

$$p(x_i) = \binom{\bar{N}_i}{x_i} q_i^{x_i} (1 - q_i)^{\bar{N}_i - x_i} .$$

The probability of detecting no more than M errors during the i-th test occasion is given by

$$\sum_{x=0}^{M} p(x) = \sum_{x=0}^{M} \binom{\bar{N}_i}{x} q_i^{x} (1 - q_i)^{\bar{N}_i - x} .$$

The relationship between the Brooks-Motley model and other J-M type models becomes more apparent if t_i is regarded as execution or calendar time. Then the reliability as a function time t_i after the end of the (i-1)st testing period is obtained by setting M=0 in the preceding equation. Then

$$R = (1-q_i)^{N_i} = (1 - q)^{t_i \bar{N}_i}$$

and for the failure rate,

$$\lambda_i = -\frac{1}{R}\frac{dR}{dt_i} = -\bar{N}_i \ln(1-q) \approx \frac{\bar{N}_i q_i}{t_i}$$

where

$$n_i' = \bar{N}_i q_i$$

is the expected number of failures in time t_i. The failure rate is proportional to the number of undetected errors remaining in the system.

Comments

1. Basic assumptions are very similar to the J-M model, where the proportionality factor ϕ in the J-M model and q in the B-M model are the measure or probability of error detection.

2. The B-M model has an advantage over the others in that it treats modular level and system level reliabilities. This is desirable in analyzing complex reactor design and safety codes. The B-M model examines the modular level reliability by assigning "probability-of-usage" weights, W_i, the determination of which may be non-trivial. This treatment accommodates the fact that programs may be used for years without error if program use is restricted to a set of well validated modules.

3. α is similar to B in the Musa model. All three parameters, N, q, and α, are estimated by the maximum likelihood method.

4. This model can make estimates of the following items based on the data of form n failures in period j.

   - current and future reliability,
   - time to achieve specified reliability,
   - probability of passing a reliability requirement test.

A.3 A Stochastic Model

The final model is a stochastic model, developed by Littlewood and Verrall, which treats the execution time and the failure rate as random variables.

Littlewood and Verrall Model

This model attempts to account for the random process of input selection and the stochastic nature of the times to failure and the associated failure rates with a mathematically tractable model. The function

$$pdf(t_i | r_i = \lambda_i) = \lambda_i e^{-\lambda_i t_i}, \ t_i > 0$$

represents the probability per unit time that the i-th failure occurs at time t_i given that the failure rate $r_i = \lambda_i$, and

$$\mathrm{pdf}(\lambda_i|i,\alpha) = \frac{\psi(i)^{\alpha}\lambda_i^{\alpha-1}\ e^{-\psi(i)\lambda_i}}{\Gamma(\alpha)} \quad , \quad \lambda_i > 0$$

gives the probability per unit λ that the failure rate between the i-1st and i-th failure is λ_i. $\psi(i)$ is a monotonically increasing function of i, e.g., $\psi(i) = \beta_1 + \beta_2 i$. The exponential distribution for time between failures is chosen because failures are considered as constituting a random process (Poisson process) and the choice of a family of Gamma distributions for the failure rates is justified by its flexibility (having two parameters, $\psi(i)$ and α), correct range $(0,\infty)$ and mathematical tractability.

Combining these two,

$$\mathrm{pdf}(t_i|\psi(i),\alpha) = \alpha\left[\frac{\psi(i)}{t_i + \psi(i)}\right]^{\alpha}\frac{1}{t_i + \psi(i)} \quad .$$

The failure rate function is given by

$$Z(t_i) = \frac{\alpha}{t_i + \psi(i)} \quad ,$$

from which the reliability function is

$$R(t_i) = \left(\frac{\psi(i)}{t_i + \psi(i)}\right)^{\alpha} \quad .$$

Comments

1. The stochastic nature of time and failure rate are considered.

2. $\psi(i)$ is a scaling factor and aids in estimating the qualitative behavior of the reliability without computing $R(t)$. If $\psi(i)$ increases more rapidly than i, a faster than linear reliability growth is implied.

References

A.1 Z. Jelinski and P. Moranda, "Software Reliability Research," Statistical Computer Performance Evaluation, W. Freiberger, ed. Academic Press, New York, 465 (1972).

A.2 J. D. Musa, "A Theory of Software Reliability and its Application," IEEE Trans. on Software Engineering, Vol. SE-1, No. 3, 312 (1975).

A.3 A. L. Goel, "Software Error Detection Model with Applications," Journal of Systems and Software 1, 253 (1980).

A.4 M. L. Shooman, "Software Reliability Data Analysis and Model Fitting," Workshop on Quantitative Software Models, 182 (1979).

A.5 G. J. Schick and R. W. Wolverton, "An Analysis of Competing Software Reliability Models," IEEE Transactions on Software Engineering, Vol. SE-4, No. 2, 104 (1978).

A.6 W. D. Brooks and R. W. Motley, "Analysis of Discrete Software Reliability Models," RADC-TR-80-84 (1980).

A.7 B. Littlewood and J. L. Verrall, "A Bayesian Reliability Growth Model for Computer Software," 1973 IEEE Symposium on Computer Software Reliability, 70 (1973).

A.8 H. B. Chenoweth, "Modified Musa Theoretic Software Reliability," 1981 Proceedings Annual Reliability and Maintainability Symposium, 353 (1981).

Appendix B

Software Failure Data Sets

This appendix contains 16 data sets generated by John Musa [B.1]. These data sets were obtained from the Rome Air Development Center and are stored on tape.

Reference

B.1 J. D. Musa, "Software Reliability Data," Data and Analysis Center for Software, 312 (1980).

Table B.1 Failure Data Set 1 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	3.	1	51	148.	56	101	30.	74
2	30.	2	52	21.	57	102	143.	74
3	113.	4	53	233.	57	103	108.	74
4	81.	10	54	134.	57	104	0.	74
5	115.	11	55	357.	57	105	3110.	75
6	9.	11	56	193.	59	106	1247.	76
7	2.	17	57	236.	59	107	943.	76
8	91.	20	58	31.	59	108	700.	76
9	112.	20	59	369.	59	109	875.	77
10	15.	20	60	748.	59	110	245.	77
11	138.	20	61	0.	59	111	729.	77
12	50.	20	62	232.	59	112	1897.	78
13	77.	20	63	330.	59	113	447.	79
14	24.	20	64	365.	61	114	386.	79
15	108.	20	65	1222.	62	115	446.	79
16	88.	20	66	543.	63	116	122.	79
17	670.	30	67	10.	63	117	990.	79
18	120.	30	68	16.	63	118	948.	80
19	26.	30	69	529.	64	119	1082.	80
20	114.	30	70	379.	64	120	22.	80
21	325.	30	71	44.	64	121	75.	80
22	55.	30	72	129.	64	122	482.	80
23	242.	31	73	810.	64	123	5509.	81
24	68.	31	74	290.	64	124	100.	81
25	422.	31	75	300.	64	125	10.	81
26	180.	32	76	529.	65	126	1071.	82
27	10.	32	77	281.	65	127	371.	83
28	1146.	33	78	160.	65	128	790.	83
29	600.	34	79	828.	66	129	6150.	83
30	15.	42	80	1011.	66	130	3321.	83
31	36.	42	81	445.	66	131	1045.	84
32	4.	46	82	296.	66	132	648.	84
33	0.	46	83	1755.	67	133	5485.	87
34	8.	46	84	1064.	67	134	1160.	87
35	227.	46	85	1783.	68	135	1864.	88
36	65.	46	86	860.	68	136	4116.	92
37	176.	46	87	983.	68			
38	58.	46	88	707.	69			
39	457.	47	89	33.	69			
40	300.	47	90	868.	69			
41	97.	47	91	724.	69			
42	263.	47	92	2323.	70			
43	452.	53	93	2930.	71			
44	255.	53	94	1461.	72			
45	197.	54	95	843.	72			
46	193.	54	96	12.	72			
47	6.	54	97	261.	72			
48	79.	54	98	1800.	73			
49	816.	56	99	865.	73			
50	1351.	56	100	1435.	74			

i= fault number, t_i= execution time, d_i= day of failure

Table B.2 Failure Data Set 2 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	191.	1	21	638.	32	41	180.	46
2	222.	2	22	293.	32	42	4225.	46
3	280.	11	23	1212.	33	43	15600.	53
4	290.	11	24	612.	33	44	0.	53
5	290.	14	25	675.	33	45	0.	53
6	385.	23	26	1215.	33	46	300.	53
7	570.	23	27	2715.	37	47	9021.	57
8	610.	23	28	3551.	37	48	2519.	64
9	365.	23	29	800.	38	49	6850.	64
10	390.	23	30	3910.	38	50	3348.	67
11	275.	23	31	6900.	38	51	2750.	69
12	360.	27	32	3300.	38	52	6675.	71
13	800.	27	33	1510.	41	53	6945.	71
14	1210.	28	34	195.	42	54	7859.	72
15	407.	29	35	1956.	42			
16	50.	29	36	135.	43			
17	660.	29	37	661.	43			
18	1507.	31	38	50.	43			
19	625.	31	39	729.	43			
20	912.	32	40	900.	45			

i= fault number, t_i= execution time, d_i= day of failure

Table B.3 Failure Data Set 3 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	115.	1	16	788.	18	31	10571.	47
2	0.	1	17	222.	18	32	563.	47
3	83.	3	18	72.	18	33	2770.	47
4	178.	3	19	615.	18	34	652.	48
5	194.	3	20	589.	26	35	5593.	50
6	136.	3	21	15.	26	36	11696.	54
7	1077.	3	22	390.	26	37	6724.	54
8	15.	3	23	1863.	27	38	2546.	55
9	15.	3	24	1337.	30			
10	92.	3	25	4508.	36			
11	50.	3	26	834.	38			
12	71.	3	27	3400.	40			
13	606.	6	28	6.	40			
14	1189.	8	29	4561.	42			
15	40.	8	30	3186.	44			

i= fault number, t_i= execution time, d_i= day of failure

Table B.4 Failure Data Set 4 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	5.	1	21	183.	26	41	0.	44
2	73.	1	22	10.	26	42	643.	46
3	191.	1	23	115.	27	43	887.	46
4	491.	5	24	17.	27	44	109.	48
5	5.	5	25	288.	27	45	469.	4
6	5.	5	26	296.	27	46	716.	46
7	28.	5	27	215.	27	47	604.	46
8	138.	5	28	116.	27	48	0.	48
9	478.	9	29	283.	31	49	774.	50
10	325.	9	30	50.	31	50	256.	50
11	147.	10	31	308.	31	51	14637.	58
12	198.	10	32	279.	31	52	18740.	70
13	22.	10	33	140.	32	53	1526.	71
14	56.	10	34	678.	32			
15	424.	20	35	183.	32			
16	92.	20	36	2462.	41			
17	520.	20	37	104.	41			
18	1424.	26	38	2178.	42			
19	0.	26	39	285.	43			
20	92.	26	40	171.	44			

i = fault number, t_i = execution time, d_i = day of failure

Table B.5 Failure Data Set 5 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	37320.	2	57	5461.	20	113	21180.	40	169	21900.	73	225	7847.	99
2	31140.	3	58	780.	20	114	11740.	41	170	26700.	73	226	7455.	99
3	3480.	3	59	840.	20	115	33720.	42	171	49220.	74	227	25512.	100
4	14260.	3	60	32580.	22	116	11220.	42	172	9000.	75	228	5554.	100
5	21480.	4	61	15180.	22	117	10860.	43	173	175080.	79	229	5380.	100
6	600.	4	62	600.	22	118	46380.	44	174	2968.	79	230	1500.	100
7	2280.	4	63	10850.	22	119	2220.	44	175	134497.	84	231	77280.	102
8	18840.	4	64	5340.	23	120	22800.	45	176	3455.	84	232	8580.	102
9	77400.	6	65	5400.	23	121	4500.	45	177	23462.	84	233	24240.	102
10	12600.	7	66	5400.	23	122	6300.	45	178	3038.	84	234	100380.	105
11	42340.	8	67	10800.	23	123	5400.	45	179	0.	84	235	24220.	105
12	2740.	8	68	300.	23	124	26400.	46	180	0.	84	236	133931.	109
13	16020.	9	69	360.	23	125	13500.	47	181	146800.	88	237	21627.	110
14	7200.	9	70	300.	23	126	29200.	49	182	1200.	88	238	77042.	111
15	120.	9	71	37200.	24	127	16500.	49	183	2160.	88	239	5880.	111
16	31020.	10	72	38400.	26	128	35700.	50	184	136440.	90	240	12120.	111
17	26640.	10	73	7200.	26	129	3840.	50	185	2940.	90	241	1215.	111
18	300.	10	74	300.	26	130	180.	50	186	1440.	90	242	42945.	112
19	8640.	11	75	17100.	26	131	4800.	50	187	8160.	90	243	13151.	113
20	780.	11	76	58740.	29	132	7980.	51	188	0.	90	244	18164.	113
21	7680.	11	77	6720.	29	133	3600.	51	189	178.	90	245	18122.	113
22	600.	11	78	2580.	29	134	2100.	51	190	17286.	90	246	61553.	115
23	6420.	11	79	16800.	30	135	13860.	51	191	12000.	91	247	88200.	117
24	12900.	12	80	61860.	32	136	9840.	51	192	35436.	91	248	3268.	118
25	360.	12	81	38040.	33	137	13980.	52	193	35225.	92	249	12612.	118
26	25800.	12	82	25200.	33	138	12120.	52	194	5577.	92	250	24540.	118
27	600.	12	83	60.	33	139	6660.	52	195	5795.	92	251	100260.	121
28	15720.	13	84	58740.	35	140	17040.	53	196	3524.	92	252	66600.	122
29	19380.	13	85	7200.	35	141	15240.	53	197	616.	92	253	22500.	123
30	8280.	14	86	24180.	35	142	960.	53	198	2535.	92	254	127845.	125
31	15000.	14	87	22620.	36	143	600.	53	199	4217.	92	255	82105.	129
32	10860.	15	88	4800.	36	144	4500.	54	200	2331.	92	256	61798.	131
33	600.	15	89	480.	36	145	5400.	54	201	360.	92	257	44089.	132
34	2280.	15	90	10200.	36	146	5400.	54	202	4560.	92	258	55313.	134
35	420.	15	91	900.	36	147	300.	54	203	10068.	92	259	17040.	134
36	10680.	15	92	4020.	36	148	300.	54	204	18852.	93	260	8160.	134
37	3720.	16	93	5700.	36	149	39840.	55	205	6240.	93	261	4660.	134
38	9000.	16	94	5490.	37	150	2640.	55	206	15000.	93	262	500.	134
39	7800.	16	95	17100.	37	151	19200.	56	207	5640.	93	263	12084.	135
40	8940.	16	96	1260.	37	152	5900.	56	208	37860.	94	264	152155.	139
41	20460.	17	97	13380.	37	153	9000.	56	209	13500.	94	265	66600.	140
42	9000.	17	98	4260.	37	154	110700.	59	210	0.	94	266	19800.	141
43	12300.	18	99	180.	37	155	43200.	60	211	0.	94	267	4800.	141
44	24000.	18	100	6060.	38	156	27900.	62	212	0.	94	268	40200.	142
45	60.	18	101	21500.	38	157	10800.	62	213	0.	94	269	1800.	142
46	1740.	18	102	360.	38	158	27060.	63	214	2880.	94	270	100800.	145
47	60.	18	103	3720.	38	159	35280.	63	215	1969.	94	271	295080.	153
48	2520.	18	104	13380.	39	160	27720.	64	216	3840.	94	272	36840.	154
49	2280.	18	105	11280.	39	161	64540.	67	217	37800.	95	273	86468.	156
50	16980.	19	106	4020.	39	162	19200.	68	218	0.	95	274	10612.	156
51	1380.	19	107	2460.	39	163	660.	68	219	31620.	96	275	24554.	157
52	16620.	19	108	14640.	39	164	20640.	69	220	50115.	97	276	86446.	159
53	7380.	20	109	600.	39	165	23700.	69	221	5373.	97	277	100509.	163
54	3600.	20	110	600.	39	166	14100.	69	222	24532.	99	278	37200.	164
55	11580.	20	111	600.	39	167	26940.	70	223	2280.	99	279	13500.	164
56	11520.	20	112	600.	39	168	61320.	72	224	17742.	99	280	11466.	164

Table B.5 (Cont'd)

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
281	120578.	169	337	31523.	208	393	5720.	238	449	3600.	254	505	67850.	276
282	23450.	169	338	393.	208	394	6974.	238	450	61200.	255	506	41342.	277
283	22020.	169	339	20582.	209	395	1211.	238	451	1800.	255	507	540.	277
284	28380.	169	340	5400.	209	396	16950.	239	452	71700.	256	508	40360.	278
285	4523.	169	341	27100.	211	397	4954.	239	453	23772.	257	509	26800.	278
286	72277.	171	342	21700.	211	398	5337.	239	454	1149.	257	510	10716.	278
287	29700.	173	343	22800.	212	399	315.	239	455	24379.	257	511	10884.	279
288	281.	173	344	600.	212	400	25841.	240	456	22500.	257	512	25236.	280
289	299419.	183	345	7200.	212	401	6391.	240	457	7200.	257	513	656.	280
290	74580.	185	346	59400.	213	402	29.	240	458	11820.	257	514	19842.	280
291	6240.	186	347	40800.	214	403	3128.	240	459	3400.	257	515	3260.	280
292	10680.	186	348	3600.	214	404	2868.	240	460	48666.	258	516	7106.	280
293	37800.	186	349	1500.	214	405	4668.	240	461	29534.	258	517	62709.	281
294	32400.	187	350	7500.	214	406	4920.	240	462	24675.	258	518	34140.	281
295	14400.	188	351	17880.	214	407	32475.	241	463	5712.	258	519	72360.	282
296	8400.	188	352	5400.	214	408	3560.	241	464	43813.	260	520	40500.	282
297	7868.	188	353	7352.	215	409	27905.	241	465	22400.	260	521	22500.	282
298	1880.	188	354	13800.	215	410	1646.	241	466	6000.	261	522	13125.	283
299	2525.	188	355	1200.	215	411	2201.	241	467	12000.	261	523	3210.	283
300	685.	188	356	37884.	217	412	17265.	242	468	5633.	261	524	58085.	283
301	1825.	188	357	4755.	217	413	4191.	242	469	112927.	262	525	46480.	284
302	21617.	189	358	3185.	217	414	6368.	242	470	29085.	263	526	172440.	286
303	34200.	189	359	2758.	217	415	4269.	242	471	92406.	263	527	12000.	286
304	25200.	190	360	30078.	218	416	34300.	243	472	155991.	265	528	2400.	286
305	25985.	191	361	37440.	218	417	8880.	243	473	1560.	265	529	600.	286
306	80815.	193	362	17100.	219	418	60.	243	474	4326.	265	530	11400.	286
307	48264.	195	363	4649.	219	419	120.	243	475	106140.	266	531	3480.	286
308	1891.	195	364	2734.	219	420	1440.	243	476	4080.	266	532	360.	286
309	29.	195	365	9297.	219	421	47940.	244	477	8880.	266	533	35040.	287
310	432.	195	366	110700.	222	422	18660.	244	478	360.	266	534	8535.	287
311	1785.	195	367	46620.	224	423	7200.	244	479	34651.	267	535	56681.	288
312	141499.	199	368	20640.	225	424	7200.	244	480	12767.	267	536	2400.	288
313	0.	199	369	97804.	228	425	3660.	244	481	6900.	267	537	3360.	288
314	1800.	199	370	47523.	229	426	720.	244	482	7500.	267	538	4200.	288
315	91800.	201	371	585.	229	427	2820.	244	483	11820.	267	539	6600.	288
316	22500.	201	372	32412.	231	428	4179.	245	484	8984.	267	540	9564.	288
317	120.	201	373	22524.	231	429	3708.	245	485	14400.	267	541	25200.	289
318	42180.	203	374	5897.	231	430	9160.	245	486	29056.	268	542	10800.	289
319	80700.	205	375	6841.	231	431	8013.	245	487	9601.	268	543	34080.	289
320	0.	205	376	23474.	232	432	7740.	245	488	3060.	268	544	18400.	290
321	5316.	205	377	5400.	232	433	29160.	246	489	75819.	269	545	6451.	290
322	344.	205	378	9009.	232	434	13140.	246	490	0.	269	546	36409.	291
323	418.	205	379	38460.	233	435	43827.	247	491	16648.	270	547	53432.	292
324	2993.	205	380	13568.	234	436	74019.	249	492	5074.	270	548	2471.	292
325	24458.	206	381	7365.	234	437	10854.	249	493	6055.	270	549	38457.	292
326	754.	206	382	4987.	234	438	46920.	250	494	3600.	270	550	41760.	293
327	2294.	206	383	1320.	234	439	44340.	250	495	89175.	272	551	39512.	293
328	11823.	206	384	10200.	234	440	55170.	252	496	1990.	272	552	44288.	294
329	15780.	206	385	6300.	234	441	38216.	252	497	65006.	273	553	3600.	294
330	18420.	207	386	16941.	235	442	25619.	253	498	49869.	274	554	37800.	294
331	5400.	207	387	6459.	236	443	24618.	253	499	10800.	274	555	2940.	294
332	3600.	207	388	4500.	236	444	4200.	253	500	32400.	274	556	7860.	294
333	3600.	207	389	9180.	236	445	37448.	254	501	7200.	274	557	4800.	294
334	7020.	208	390	38880.	238	446	18976.	254	502	24098.	275	558	3300.	294
335	1080.	208	391	281.	238	447	13462.	254	503	45608.	275	559	120.	294
336	2700.	208	392	760.	238	448	19614.	254	504	2994.	275	560	2590.	294

Table B.5 (Cont'd)

i	t_i	d_i
561	57600.	295
562	32400.	295
563	27180.	296
564	21120.	295
565	15600.	296
566	46500.	297
567	35494.	297
568	144389.	299
569	46917.	300
570	39720.	300
571	65119.	301
572	29462.	302
573	10800.	302
574	298332.	305
575	10618.	305
576	15581.	305
577	5936.	305
578	40773.	306
579	76060.	307
580	37200.	307
581	4200.	307
582	175080.	309
583	25020.	310
584	163660.	311
585	600.	311
586	840.	311
587	61180.	312
588	11690.	312
589	1026.	312
590	1244.	312
591	2040.	312
592	52200.	313
593	106760.	314
594	63660.	315
595	2160.	315
596	115920.	317
597	19560.	317
598	192600.	319
599	1186.	319
600	45614.	319
601	8848.	320
602	32792.	320
603	40797.	321
604	62.	321
605	37381.	321
606	40320.	321
607	90600.	322
608	39110.	322
609	53890.	323
610	39000.	324
611	15000.	324
612	28800.	324
613	35520.	325
614	14400.	325
615	52048.	326
616	138752.	327
617	84600.	328
618	119700.	330
619	74880.	331
620	0.	331
621	7200.	331
622	52200.	332
623	84600.	334
624	5400.	334
625	5400.	334
626	7200.	334
627	86400.	335
628	21600.	336
629	1800.	336
630	0.	336
631	3600.	336
632	3600.	336
633	14400.	336
634	68400.	337
635	30600.	337
636	7200.	337
637	3600.	337
638	37800.	338
639	12300.	338
640	2100.	338
641	7200.	338
642	5100.	338
643	30600.	338
644	600.	338
645	3000.	338
646	720.	338
647	2880.	338
648	1080.	338
649	66120.	339
650	0.	339
651	36000.	339
652	75600.	340
653	7200.	340
654	15300.	340
655	900.	340
656	3600.	340
657	0.	340
658	1800.	340
659	88200.	341
660	3600.	341
661	1800.	341
662	3600.	341
663	1800.	341
664	30600.	342
665	7337.	342
666	7061.	342
667	3600.	342
668	75600.	343
669	0.	343
670	3032.	343
671	783.	343
672	982.	343
673	464.	343
674	1710.	343
675	6960.	343
676	165720.	346
677	45000.	347
678	21600.	347
679	1800.	347
680	3600.	347
681	0.	347
682	0.	347
683	61320.	348
684	6120.	348
685	26160.	348
686	0.	348
687	7200.	348
688	35400.	349
689	960.	349
690	1140.	349
691	3180.	349
692	31300.	349
693	9020.	349
694	154700.	353
695	124080.	353
696	1500.	353
697	6220.	353
698	26400.	353
699	4540.	353
700	258.	[illegible]
701	3959.	[illegible]
702	3190.	[illegible]
703	4405.	353
704	73591.	354
705	3608.	354
706	4078.	354
707	398.	354
708	4962.	354
709	2255.	354
710	1800.	354
711	4945.	354
712	2551.	354
713	48609.	355
714	36001.	355
715	6322.	355
716	2070.	355
717	147320.	359
718	2688.	359
719	210.	359
720	2440.	359
721	1244.	359
722	102.	359
723	2100.	359
724	3600.	359
725	10694.	359
726	2911.	359
727	2298.	359
728	3815.	359
729	14286.	359
730	1511.	359
731	2520.	359
732	128894.	361
733	15796.	361
734	23455.	362
735	635.	362
736	901.	362
737	6934.	362
738	1155.	362
739	67008.	363
740	347.	363
741	2408.	363
742	7475.	363
743	37197.	364
744	17742.	365
745	21738.	366
746	15057.	366
747	26528.	366
748	35593.	367
749	15710.	368
750	10634.	368
751	83343.	370
752	11164.	370
753	19763.	371
754	119272.	373
755	31234.	374
756	54557.	375
757	7417.	375
758	3273.	375
759	20762.	376
760	9690.	376
761	6514.	376
762	6853.	376
763	152344.	379
764	57868.	380
765	2642.	380
766	7440.	380
767	280.	380
768	4128.	380
769	416.	380
770	15417.	381
771	7662.	381
772	2266.	381
773	1333.	381
774	20553.	381
775	24491.	382
776	36034.	383
777	8059.	383
778	21232.	384
779	56462.	385
780	5076.	385
781	125221.	387
782	129023.	390
783	52957.	391
784	1942.	391
785	36013.	392
786	65400.	393
787	1090.	393
788	36459.	394
789	63001.	395
790	33604.	395
791	179232.	399
792	54496.	400
793	70558.	401
794	10900.	402
795	111502.	405
796	157275.	409
797	14700.	409
798	3528.	409
799	150432.	412
800	75400.	414
801	277380.	420
802	58550.	421
803	148020.	424
804	1320.	424
805	6975.	424
806	8967.	424
807	456.	424
808	60.	424
809	0.	424
810	1752.	424
811	32640.	424
812	46924.	426
813	1608.	426
814	32060.	427
815	12.	427
816	3488.	427
817	2828.	427
818	0.	427
819	336.	427
820	30332.	428
821	5216.	428
822	1472.	428
823	1168.	428
824	3648.	428
825	44924.	429
826	3424.	429
827	47275.	430
828	5805.	431
829	5720.	431
830	55664.	432
831	13088.	432

i = fault number, t_i = execution time, d_i = day of failure

Table B.6 Failure Data Set 6 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	3.	1	31	23.	16	61	5.	37
2	14.	1	32	1.	16	62	38.	37
3	59.	1	33	672.	24	63	74.	38
4	32.	2	34	189.	24	64	40.	39
5	8.	2	35	83.	26	65	2.	39
6	52.	2	36	52.	26	66	60.	40
7	2.	2	37	8.	26	67	221.	42
8	25.	2	38	1.	26	68	6.	42
9	2.	2	39	41.	27	69	891.	52
10	3.	5	40	7.	27	70	23.	53
11	4.	6	41	43.	28	71	4.	53
12	1.	6	42	1.	28	72	437.	58
13	30.	6	43	4.	28	73	66.	58
14	21.	7	44	5.	28			
15	196.	12	45	1.	28			
16	265.	12	46	16.	28			
17	6.	12	47	70.	29			
18	3.	12	48	60.	30			
19	8.	12	49	2.	30			
20	1.	12	50	2.	30			
21	12.	12	51	3.	30			
22	36.	13	52	169.	31			
23	38.	13	53	29.	32			
24	1.	13	54	88.	33			
25	74.	14	55	55.	35			
26	43.	14	56	27.	35			
27	236.	14	57	24.	35			
28	121.	15	58	27.	35			
29	18.	16	59	140.	37			
30	9.	16	60	33.	37			

i= fault number, t_i= execution time, d_i= day of failure

Table B.7 Failure Data Set 14 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	191520.	3	16	65173.	63	31	1563300.	129
2	2078820.	27	17	2370.	63	32	513000.	135
3	514560.	33	18	1581.	63	33	177660.	137
4	1140.	33	19	228315.	66	34	2469000.	165
5	3120.	33	20	51480.	67	35	1678260.	185
6	327480.	37	21	44820.	67	36	170760.	187
7	15420.	37	22	850080.	77			
8	60000.	38	23	361860.	81			
9	140160.	39	24	34390.	82			
10	937620.	50	25	545280.	88			
11	72240.	51	26	256980.	91			
12	737700.	60	27	396780.	96			
13	250680.	63	28	91260.	97			
14	2965.	63	29	1225620.	111			
15	196.	63	30	120.	111			

i= fault number, t_i= execution time, d_i= day of failure

Table B.8 Failure Data Set 17 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	8100.	3	16	600.	21	31	15600.	41
2	4800.	9	17	15060.	23	32	1800.	41
3	900.	9	18	120.	23	33	5400.	41
4	450.	9	19	360.	23	34	3900.	42
5	450.	9	20	1200.	24	35	2400.	42
6	6000.	11	21	300.	25	36	1200.	43
7	2400.	14	22	1200.	26	37	79510.	55
8	2100.	15	23	1200.	26	38	9000.	56
9	2100.	15	24	3300.	27			
10	1260.	17	25	14800.	30			
11	1200.	18	26	3000.	32			
12	300.	18	27	600.	32			
13	9000.	19	28	9600.	35			
14	600.	20	29	8400.	37			
15	2400.	21	30	8100.	38			

i=fault number, t_i= execution time, d_i= day of failure

Table B.9 Failure Data Set 27 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	20336.	1	16	38005.	19	31	336600.	60
2	11776.	1	17	16200.	23	32	268140.	63
3	40933.	1	18	6000.	23	33	74880.	65
4	34796.	1	19	1000.	24	34	288200.	65
5	17136.	2	20	10000.	24	35	23320.	65
6	148446.	2	21	240.	24	36	7080.	65
7	7995.	2	22	35580.	24	37	59820.	66
8	1636.	2	23	81000.	25	38	87900.	67
9	15830.	2	24	643095.	33	39	76200.	68
10	21937.	10	25	47857.	34	40	89280.	69
11	2485.	12	26	154800.	35	41	1209600.	79
12	11000.	15	27	170400.	37			
13	2880.	15	28	108540.	55			
14	61182.	16	29	73800.	57			
15	4800.	18	30	1800.	60			

i= fault number, t_i= execution time, d_i= day of failure

Table B.10 Failure Data Set 40 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	320.	1	41	640.	100	81	167200.	256
2	14390.	2	42	640.	100	82	10800.	258
3	9000.	9	43	2880.	102	83	170200.	260
4	2880.	21	44	110.	104	84	487800.	266
5	5700.	24	45	22080.	104	85	634200.	275
6	21800.	27	46	60654.	106	86	334560.	279
7	26800.	27	47	52163.	112	87	1458800.	296
8	113540.	37	48	12546.	114	88	66720.	297
9	112137.	41	49	734.	114	89	144200.	300
10	660.	42	50	10193.	114	90	215200.	303
11	2700.	43	51	7641.	114	91	86400.	304
12	28793.	43	52	31365.	114	92	88640.	305
13	2173.	44	53	24313.	115	93	1814400.	320
14	7263.	44	54	298890.	125	94	4160.	326
15	10865.	44	55	1280.	125	95	3200.	326
16	4230.	45	56	22099.	127	96	199200.	329
17	8460.	48	57	14150.	127	97	356160.	333
18	14805.	49	58	2611.	127	98	518400.	337
19	11449.	50	59	39170.	127	99	345600.	343
20	5361.	51	60	55794.	129	100	31360.	343
21	6553.	51	61	42632.	133	101	265600.	347
22	6479.	56	62	267600.	135			
23	3124.	56	63	87074.	136			
24	51323.	58	64	149606.	141			
25	17010.	58	65	14400.	141			
26	1890.	64	66	34560.	141			
27	5400.	69	67	39600.	141			
28	62312.	76	68	334395.	148			
29	24826.	76	69	296015.	148			
30	26355.	79	70	177399.	149			
31	363.	79	71	214622.	150			
32	13989.	85	72	156400.	154			
33	15058.	85	73	166800.	156			
34	32377.	85	74	10800.	156			
35	41632.	85	75	267000.	159			
36	4160.	87	76	2098833.	183			
37	82040.	91	77	614080.	190			
38	13189.	91	78	7680.	190			
39	3426.	97	79	2629667.	220			
40	5833.	99	80	2948700.	254			

i = fault number, t_i = execution time, d_i = day of failure

Table B.11 Failure Data Set SS1A of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	82530.	2	41	75420.	50	81	95580.	101
2	713700.	11	42	900.	50	82	338310.	105
3	671220.	17	43	43290.	50	83	35550.	105
4	61470.	18	44	380610.	53	84	251820.	107
5	77130.	18	45	500850.	57	85	199710.	109
6	205650.	20	46	69930.	58	86	469430.	113
7	105660.	21	47	3600.	58	87	169180.	114
8	33390.	21	48	990.	58	88	99900.	115
9	16560.	21	49	14400.	58	89	270.	115
10	9360.	22	50	2430.	58	90	90.	115
11	67500.	22	51	17910.	58	91	50670.	116
12	4590.	22	52	5670.	59	92	162180.	117
13	123210.	23	53	210060.	60	93	228330.	119
14	5400.	23	54	82620.	61	94	[illegible]620.	119
15	678960.	24	55	151790.	62	95	28440.	119
16	57600.	27	56	57130.	63	96	150480.	121
17	85680.	30	57	3510.	63	97	36000.	121
18	26460.	30	58	188820.	64	98	180.	121
19	100620.	31	59	88560.	65	99	110160.	122
20	12150.	31	60	176130.	67	100	513450.	126
21	154350.	32	61	5130.	67	101	93420.	127
22	164610.	34	62	298080.	70	102	5400.	127
23	3150.	34	63	141480.	71	103	10710.	127
24	10350.	34	64	13680.	71	104	331830.	130
25	157500.	35	65	900.	71	105	406350.	134
26	66870.	36	66	92430.	72	106	306900.	137
27	245700.	38	67	213480.	74	107	580500.	142
28	167310.	40	68	31950.	74	108	1260.	142
29	27720.	40	69	373500.	77	109	178200.	143
30	224860.	42	70	232020.	79	110	124380.	144
31	21420.	42	71	117900.	80	111	313470.	147
32	3780.	42	72	64260.	80	112	158940.	148
33	105570.	43	73	4500.	80			
34	2700.	43	74	253710.	84			
35	69390.	44	75	302940.	86			
36	62730.	45	76	71460.	87			
37	20340.	45	77	515970.	92			
38	413550.	49	78	260010.	94			
39	55170.	49	79	97920.	95			
40	41940.	49	80	673470.	100			

i= fault number, t_i= execution time, d_i= day of failure

Table B.12 Failure Data Set SS1B of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	5400.	1	51	208020.	51	101	841020.	143	151	1560.	234	201	46740.	325
2	10200.	1	52	62340.	52	102	33480.	143	152	600.	234	202	596820.	333
3	24060.	1	53	40350.	53	103	158640.	145	153	22620.	234	203	474760.	339
4	8460.	1	54	29820.	53	104	548460.	152	154	600.	234	204	64480.	340
5	33900.	2	55	257334.	56	105	130560.	153	155	600.	234	205	8760.	340
6	16500.	2	56	480.	56	106	556380.	160	156	21090.	235	206	5100.	340
7	6300.	2	57	108360.	58	107	16740.	160	157	35880.	235	207	73140.	341
8	87540.	3	58	133920.	59	108	539820.	167	158	59940.	236	208	1380.	341
9	5400.	3	59	137160.	61	109	94980.	168	159	141540.	238	209	12300.	341
10	51780.	4	60	18420.	61	110	13920.	168	160	291720.	242	210	540.	341
11	1140.	4	61	170040.	64	111	84780.	169	161	512880.	249	211	55260.	342
12	6300.	4	62	32640.	65	112	3780.	169	162	171360.	253	212	14760.	342
13	2400.	4	63	10200.	65	113	233640.	173	163	229740.	253	213	4200.	342
14	16260.	4	64	24000.	65	114	13860.	173	164	170700.	255	214	13020.	342
15	169140.	6	65	120480.	66	115	420.	173	165	175620.	257	215	18600.	343
16	94380.	7	66	352260.	71	116	40920.	174	166	81660.	258	216	28020.	343
17	5400.	7	67	75000.	72	117	4560.	174	167	23340.	258	217	360420.	347
18	15600.	7	68	1380.	72	118	507840.	180	168	63420.	259	218	116400.	349
19	14400.	7	69	163020.	75	119	55740.	180	169	58020.	260	219	120780.	350
20	30660.	8	70	13980.	75	120	82680.	181	170	2100.	260	220	1109520.	365
21	131460.	9	71	9120.	75	121	409620.	186	171	396240.	266	221	119520.	367
22	38700.	10	72	29400.	76	122	1167840.	201	172	10020.	266	222	49660.	369
23	273480.	14	73	9360.	76	123	69120.	202	173	132700.	269	223	23760.	369
24	64540.	15	74	1200.	76	124	668700.	211	174	70620.	270	224	17800.	369
25	23160.	15	75	3600.	76	125	160320.	213	175	163080.	273	225	155880.	371
26	175860.	17	76	1500.	76	126	11220.	213	176	348000.	277	226	165220.	374
27	310020.	21	77	79800.	77	127	22380.	214	177	66960.	278	227	104880.	375
28	148920.	23	78	75600.	78	128	219300.	216	178	190260.	280	228	47940.	376
29	79680.	24	79	159600.	80	129	266340.	220	179	123540.	282	229	406840.	381
30	12720.	24	80	600.	80	130	2100.	220	180	5520.	282	230	24180.	381
31	4920.	24	81	292080.	84	131	157740.	222	181	960.	282	231	34560.	382
32	70320.	25	82	190040.	85	132	144480.	224	182	180900.	284	232	612600.	389
33	71340.	26	83	207660.	88	133	14040.	224	183	423900.	289	233	30420.	389
34	1200.	26	84	14640.	88	134	34200.	224	184	335460.	293	234	52420.	390
35	219660.	28	85	1200.	88	135	346652.	229	185	5220.	293	235	1200.	390
36	36540.	29	86	136320.	90	136	1080.	229	186	559140.	300	236	637860.	398
37	1680.	29	87	3000.	90	137	9000.	229	187	95280.	301	237	35760.	398
38	44160.	29	88	39240.	90	138	6600.	229	188	247360.	305	238	691600.	409
39	20160.	30	89	249960.	93	139	120.	229	189	23220.	305	239	253140.	412
40	48300.	30	90	446460.	101	140	240.	229	190	3480.	305	240	288360.	416
41	210060.	34	91	97600.	102	141	240.	229	191	186240.	307	241	1800.	416
42	30180.	34	92	780.	102	142	68700.	230	192	101940.	308	242	101280.	417
43	168000.	36	93	72900.	103	143	1620.	230	193	245160.	311	243	105540.	418
44	173580.	38	94	69960.	104	144	77340.	231	194	298020.	315	244	863400.	430
45	56820.	39	95	104120.	105	145	234540.	234	195	405920.	320	245	172440.	432
46	271140.	42	96	249420.	108	146	900.	234	196	186600.	322	246	70320.	433
47	139920.	44	97	730560.	117	147	960.	234	197	25440.	322	247	554940.	440
48	166240.	46	98	506880.	123	148	1260.	234	198	35220.	324	248	11700.	440
49	15900.	47	99	46920.	123	149	540.	234	199	15360.	324	249	432480.	446
50	185640.	49	100	790260.	133	150	1260.	234	200	48600.	325	250	26700.	446

Table B.12 (Cont'd)

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
251	7560.	447	276	78120.	501	301	97740.	546	326	1320.	610	351	142020.	629
252	28200.	447	277	304060.	505	302	1560.	548	327	156540.	612	352	74440.	631
253	39060.	449	278	26640.	505	303	61380.	549	328	3900.	612	353	113640.	633
254	511380.	454	279	209700.	509	304	41520.	550	329	43860.	612	354	59400.	634
255	315960.	458	280	18000.	508	305	70080.	551	330	68340.	613	355	78300.	635
256	131460.	460	281	195000.	512	306	4800.	551	331	13440.	614	356	146380.	638
257	37260.	460	282	2700.	512	307	68040.	552	332	31360.	614	357	25980.	638
258	750480.	470	283	79560.	513	308	15660.	552	333	123120.	617	358	5520.	638
259	23940.	470	284	86460.	514	309	130860.	554	334	39180.	617	359	50460.	639
260	400080.	475	285	60100.	515	310	213660.	557	335	130620.	619	360	1800.	639
261	180.	475	286	77160.	516	311	73260.	558	336	4960.	619	361	2880.	639
262	240.	475	287	9000.	516	312	242760.	561	337	9360.	619	362	23520.	639
263	480.	475	288	5640.	516	313	722940.	571	338	4320.	619	363	41520.	640
264	262200.	479	289	676720.	526	314	423360.	576	339	48780.	620	364	71040.	641
265	279160.	482	290	57120.	527	315	420.	576	340	4980.	620	365	143140.	645
266	3960.	482	291	18120.	527	316	1440.	576	341	7560.	620	366	162180.	647
267	43320.	483	292	780.	527	317	340860.	582	342	55500.	621	367	245700.	651
268	364020.	485	293	263820.	531	318	128700.	584	343	30900.	622	368	64060.	652
269	11820.	486	294	210600.	534	319	940500.	597	344	140940.	624	369	123900.	654
270	447660.	492	295	2640.	534	320	446980.	604	345	29160.	625	370	57000.	655
271	249460.	495	296	600.	534	321	21900.	604	346	21480.	625	371	21360.	655
272	146040.	498	297	523500.	541	322	157200.	607	347	33420.	626	372	22140.	656
273	26940.	498	298	147420.	543	323	360.	607	348	20700.	626	373	26700.	656
274	142920.	500	299	199740.	546	324	225600.	610	349	50040.	627	374	12180.	656
275	32100.	500	300	25800.	546	325	[illegible]9860.	610	350	10740.	627	375	268260.	660

i= fault number, t_i= execution time, d_i= day of failure

Table B.13 Failure Data Set SS1C of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	5400.	1	57	240.	50	113	1800.	130	169	219328.	194	225	247160.	361
2	81060.	2	58	600.	50	114	1680.	130	170	38940.	195	226	720.	361
3	2100.	2	59	13080.	51	115	7980.	130	171	19200.	195	227	660.	363
4	13620.	2	60	58980.	52	116	420.	130	172	6700.	195	228	2460.	361
5	22800.	2	61	1140.	52	117	11460.	130	173	161280.	196	229	2760.	361
6	13380.	3	62	62160.	53	118	25440.	131	174	174180.	199	230	6880.	361
7	4620.	3	63	34620.	54	119	99360.	133	175	80040.	200	231	420.	361
8	5100.	3	64	118140.	57	120	45360.	133	176	186280.	203	232	840.	361
9	26640.	3	65	262980.	60	121	62040.	134	177	2880.	203	233	480.	361
10	45960.	4	66	47400.	61	122	24360.	134	178	43260.	203	234	87960.	362
11	5100.	4	67	1320.	61	123	3420.	134	179	494640.	210	235	228360.	366
12	28800.	4	68	41880.	62	124	2220.	134	180	38790.	211	236	100020.	368
13	59620.	5	69	34500.	62	125	41460.	136	181	53520.	212	237	163260.	371
14	1920.	5	70	115860.	64	126	128040.	137	182	100560.	213	238	60420.	372
15	99540.	6	71	66960.	65	127	134340.	139	183	207840.	216	239	255840.	378
16	60000.	7	72	178740.	69	128	16800.	139	184	810840.	227	240	13080.	378
17	3000.	7	73	2340.	69	129	5760.	139	185	47720.	227	241	4740.	378
18	34500.	7	74	6780.	69	130	41760.	140	186	130920.	229	242	1980.	378
19	52500.	8	75	780.	69	131	540.	140	187	280260.	234	243	430000.	385
20	2280.	8	76	42840.	69	132	540.	140	188	23160.	234	244	12660.	385
21	42100.	8	77	55500.	70	133	600.	140	189	25020.	234	245	40020.	386
22	167940.	11	78	890340.	84	134	420.	140	190	531060.	242	246	16680.	386
23	42840.	11	79	1620.	84	135	540.	140	191	175476.	246	247	1080.	386
24	145920.	14	80	73560.	85	136	305040.	144	192	659392.	255	248	2100.	386
25	43380.	15	81	284007.	91	137	102480.	145	193	790440.	268	249	1020.	386
26	43380.	15	82	85140.	93	138	60840.	146	194	107400.	270	250	13080.	386
27	26820.	15	83	177360.	95	139	86880.	147	195	625140.	282	251	842220.	400
28	286500.	20	84	45360.	96	140	130980.	148	196	507300.	290	252	805176.	412
29	27000.	20	85	44400.	96	141	172140.	150	197	1016040.	309	253	18780.	412
30	44400.	21	86	270472.	101	142	144660.	152	198	1080.	309	254	47100.	413
31	110220.	22	87	172140.	103	143	12180.	152	199	30540.	310	255	110520.	415
32	269100.	26	88	960.	103	144	66960.	153	200	56160.	312	256	840.	415
33	60840.	27	89	23700.	103	145	155820.	155	201	224520.	316	257	266760.	418
34	85800.	28	90	190020.	107	146	5100.	155	202	12120.	316	258	33460.	419
35	98820.	30	91	55260.	107	147	242820.	158	203	13080.	316	259	38520.	420
36	61260.	32	92	48660.	108	148	33840.	158	204	142140.	320	260	70800.	421
37	156300.	34	93	264900.	112	149	34140.	159	205	510780.	330	261	119100.	423
38	52740.	34	94	1080.	112	150	479968.	165	206	181500.	332	262	462600.	431
39	54960.	35	95	86700.	113	151	31260.	165	207	131580.	334	263	17400.	431
40	32100.	35	96	60.	113	152	23520.	165	208	6180.	334	264	543060.	441
41	129660.	38	97	1.	113	153	57720.	166	209	120.	334	265	191040.	445
42	116760.	39	98	337920.	118	154	454200.	173	210	228900.	338	266	3720.	445
43	75000.	40	99	31320.	119	155	36060.	173	211	4320.	338	267	103080.	446
44	37260.	41	100	245760.	122	156	3780.	173	212	9300.	338	268	117360.	449
45	24180.	41	101	36480.	123	157	226440.	177	213	75720.	339	269	63960.	450
46	146640.	44	102	187260.	125	158	73560.	178	214	13080.	340	270	3960.	450
47	35760.	45	103	28980.	126	159	180.	178	215	118020.	342	271	442860.	457
48	16080.	45	104	120.	126	160	20760.	178	216	33900.	343	272	22260.	458
49	900.	45	105	119340.	127	161	10340.	178	217	29940.	344	273	241500.	461
50	10320.	45	106	1740.	127	162	45660.	179	218	86400.	345	274	49920.	462
51	2280.	45	107	840.	127	163	117720.	180	219	116980.	347	275	144240.	465
52	28800.	45	108	1500.	127	164	10860.	180	220	8400.	349	276	37260.	467
53	75960.	46	109	1200.	127	165	71280.	181	221	33000.	350	277	122580.	470
54	52140.	47	110	2820.	127	166	83460.	183	222	43020.	350			
55	28920.	48	111	51960.	130	167	79560.	184	223	118620.	353			
56	52920.	50	112	8940.	130	168	319260.	190	224	137940.	356			

i = fault number, t_i = execution time, d_i = day of failure

Table B.14 Failure Data Set SS2 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	26100.	1	41	960.	199	81	525360.	347	121	3360.	447	161	555600.	542
2	139680.	3	42	140940.	201	82	253620.	350	122	59680.	447	162	56340.	543
3	300.	3	43	678500.	210	83	70800.	351	123	119220.	449	163	476100.	547
4	409980.	8	44	13140.	210	84	936720.	363	124	7560.	449	164	47430.	547
5	751920.	17	45	1200.	210	85	132480.	365	125	192840.	452	165	1337220.	559
6	1579800.	37	46	302640.	214	86	74200.	366	126	20580.	452	166	844290.	566
7	374280.	42	47	780.	214	87	330720.	370	127	755040.	461	167	1578780.	578
8	111600.	43	48	1354276.	231	88	161580.	372	128	5940.	461	168	588720.	584
9	113460.	45	49	91500.	232	89	13560.	372	129	393240.	466	169	175930.	586
10	625300.	52	50	2760.	232	90	3700.	372	130	157140.	468	170	2790.	586
11	321920.	56	51	231000.	235	91	289620.	376	131	1134660.	482	171	2520.	586
12	240120.	59	52	227820.	238	92	13500.	376	132	78660.	483	172	1080.	586
13	74160.	60	53	80700.	239	93	1980.	376	133	6360.	483	173	248850.	588
14	274560.	63	54	1380.	239	94	59940.	377	134	358140.	487	174	21690.	588
15	225000.	66	55	501840.	245	95	7920.	377	135	41700.	488	175	347130.	591
16	2460.	66	56	15660.	245	96	2040.	377	136	131700.	490	176	168370.	594
17	539340.	73	57	1106100.	258	97	18300.	377	137	100440.	491	177	13680.	594
18	243000.	77	58	1200.	258	98	1032360.	390	138	7920.	491	178	424890.	599
19	900.	77	59	363780.	264	99	183660.	392	139	238620.	495	179	119340.	600
20	557220.	85	60	66480.	265	100	1484160.	410	140	142740.	497	180	219690.	602
21	667440.	92	61	4380.	265	101	19140.	410	141	18180.	497	181	12870.	602
22	319800.	99	62	16020.	265	102	554100.	417	142	315780.	502	182	1524330.	615
23	124560.	100	63	169440.	267	103	1680.	417	143	73380.	503	183	361800.	619
24	389520.	105	64	138560.	270	104	23400.	417	144	4980.	503	184	114930.	620
25	1680180.	125	65	720.	270	105	517740.	425	145	430560.	508	185	95760.	621
26	773220.	134	66	2700.	270	106	29520.	425	146	48540.	509	186	240570.	623
27	1056420.	147	67	53880.	271	107	54660.	426	147	90240.	510	187	675360.	628
28	193740.	149	68	10980.	271	108	37800.	427	148	216840.	514	188	70440.	629
29	521340.	155	69	1806980.	293	109	38340.	427	149	72660.	515	189	1411350.	641
30	4980.	155	70	84900.	294	110	78540.	428	150	55980.	516	190	15210.	641
31	2065920.	180	71	361200.	298	111	534660.	434	151	110400.	516	191	1566270.	655
32	143340.	182	72	420.	298	112	54960.	435	152	504720.	522	192	8010.	655
33	522660.	188	73	1020.	298	113	56580.	436	153	3060.	522			
34	572160.	196	74	1156140.	312	114	195180.	438	154	32520.	522			
35	139920.	198	75	733500.	321	115	68940.	439	155	59940.	523			
36	720.	198	76	435540.	327	116	70380.	440	156	91560.	524			
37	9900.	198	77	507000.	333	117	401700.	445	157	360.	524			
38	1980.	198	78	166080.	335	118	87240.	446	158	550380.	531			
39	3060.	198	79	204300.	337	119	32520.	446	159	637410.	536			
40	81900.	199	80	232140.	340	120	40920.	447	160	184050.	538			

i= fault number, t_i= execution time, d_i= day of failure

Table B.15 Failure Data Set SS3 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	107400.	2	57	351901.	107	113	540.	222	169	840.	320	225	522240.	500
2	17220.	2	58	369480.	111	114	900.	222	170	540.	320	226	66000.	501
3	180.	2	59	380220.	116	115	521252.	229	171	589980.	328	227	43500.	502
4	32880.	3	60	848640.	126	116	420.	229	172	332280.	332	228	2040.	502
5	960.	3	61	120.	126	117	518640.	236	173	94140.	333	229	600.	502
6	26100.	3	62	3416.	126	118	1020.	236	174	240060.	336	230	226320.	505
7	44160.	4	63	74160.	127	119	4140.	236	175	2700.	336	231	327600.	509
8	333720.	8	64	262500.	130	120	480.	236	176	900.	336	232	201300.	513
9	17820.	8	65	879300.	141	121	180.	236	177	1080.	336	233	226980.	516
10	40860.	9	66	360.	141	122	600.	236	178	11580.	336	234	553440.	523
11	18780.	9	67	8160.	141	123	53760.	237	179	2160.	336	235	1020.	523
12	960.	9	68	180.	141	124	82440.	238	180	192720.	339	236	960.	523
13	960.	9	69	237920.	144	125	180.	238	181	87840.	340	237	512760.	530
14	79860.	10	70	120.	144	126	273000.	241	182	84360.	341	238	819240.	537
15	240.	10	71	70800.	145	127	59880.	242	183	378120.	346	239	801660.	544
16	120.	10	72	12960.	145	128	840.	242	184	58500.	347	240	160380.	545
17	1800.	10	73	300.	145	129	7140.	242	185	83880.	348	241	71640.	546
18	480.	10	74	120.	145	130	76320.	243	186	158640.	350	242	363990.	549
19	780.	10	75	558540.	152	131	148680.	245	187	660.	350	243	9090.	549
20	37260.	11	76	188640.	154	132	237840.	248	188	3180.	350	244	227970.	551
21	2100.	11	77	56280.	155	133	4560.	248	189	1560.	350	245	17190.	551
22	72060.	12	78	420.	155	134	1920.	248	190	3180.	350	246	597900.	557
23	258704.	15	79	414464.	160	135	16860.	248	191	5700.	350	247	689400.	563
24	480.	15	80	240780.	163	136	77040.	249	192	226560.	354	248	11520.	563
25	21900.	15	81	206640.	166	137	74760.	250	193	9840.	354	249	23850.	564
26	478620.	21	82	4740.	166	138	738180.	259	194	69060.	355	250	15870.	564
27	80760.	22	83	10140.	166	139	147000.	262	195	68880.	356	251	123030.	566
28	1200.	22	84	300.	166	140	76680.	263	196	65460.	357	252	26010.	566
29	80700.	23	85	4140.	166	141	70800.	264	197	402900.	362	253	75240.	567
30	688860.	32	86	472080.	172	142	66180.	265	198	75480.	363	254	68130.	567
31	2220.	32	87	300.	172	143	27540.	265	199	380220.	368	255	811050.	574
32	758880.	42	88	87600.	173	144	55020.	266	200	704968.	377	256	498360.	578
33	166620.	44	89	48240.	174	145	120.	266	201	505680.	381	257	623280.	585
34	8280.	44	90	41940.	174	146	296796.	271	202	54420.	384	258	3330.	585
35	951354.	56	91	576612.	181	147	90180.	272	203	319020.	388	259	7290.	585
36	1320.	56	92	71820.	182	148	724560.	281	204	95220.	389	260	47160.	585
37	14700.	56	93	83100.	183	149	167100.	283	205	5100.	389	261	1328400.	598
38	3420.	56	94	900.	183	150	106200.	284	206	6240.	389	262	109800.	599
39	2520.	56	95	240300.	186	151	480.	285	207	494400.	390	263	393890.	602
40	162480.	58	96	73740.	187	152	117360.	286	208	420.	390	264	1615860.	616
41	520320.	65	97	169800.	189	153	6480.	286	209	667320.	398	265	14940.	616
42	96720.	66	98	1.	189	154	60.	286	210	120.	398	266	680760.	623
43	418200.	71	99	302280.	194	155	97460.	287	211	7200.	398	267	26220.	623
44	434760.	73	100	3360.	194	156	398580.	292	212	684400.	399	268	376110.	627
45	543780.	85	101	2340.	194	157	391380.	297	213	26820.	399	269	181890.	628
46	8820.	85	102	82260.	195	158	180.	297	214	448620.	405	270	64320.	629
47	488280.	91	103	559920.	202	159	180.	297	215	359420.	409	271	468180.	633
48	480.	91	104	780.	202	160	240.	297	216	480.	409	272	1568580.	648
49	540.	91	105	10740.	202	161	540.	297	217	1042680.	423	273	333720.	651
50	2220.	91	106	180.	202	162	336900.	301	218	779580.	433	274	180.	651
51	1080.	91	107	430860.	208	163	264480.	304	219	8040.	433	275	810.	651
52	137340.	93	108	166740.	210	164	847080.	314	220	1158240.	448	276	322110.	654
53	91860.	96	109	600.	210	165	26460.	315	221	907140.	459	277	21960.	654
54	22800.	96	110	376110.	215	166	349320.	319	222	58500.	460	278	363600.	657
55	22920.	96	111	5100.	215	167	4080.	319	223	383940.	465			
56	473340.	102	112	549540.	222	168	64680.	320	224	2039460.	492			

i= fault number, t_i= execution time, d_i= day of failure

Table B.16 Failure Data Set SS4 of J. D. Musa

i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i	i	t_i	d_i
1	242460.	4	41	1920.	127	81	8400.	235	121	2220.	350	161	348180.	482
2	6300.	4	42	902340.	138	82	40260.	236	122	1237800.	366	162	5100.	482
3	253680.	7	43	2100.	138	83	2940.	236	123	222960.	369	163	477900.	488
4	24120.	7	44	4200.	138	84	8700.	236	124	108300.	370	164	636420.	497
5	37620.	8	45	480.	138	85	7080.	236	125	58980.	371	165	575340.	505
6	928380.	19	46	177060.	140	86	64020.	237	126	20100.	371	166	222360.	508
7	534180.	25	47	42360.	141	87	360.	237	127	82320.	372	167	86580.	509
8	151700.	28	48	37020.	141	88	84840.	238	128	362160.	377	168	2760.	509
9	11760.	28	49	54540.	142	89	180.	238	129	600.	377	169	2820.	509
10	159240.	30	50	335520.	146	90	908340.	249	130	9000.	377	170	1560.	509
11	19140.	30	51	300.	146	91	7680.	249	131	1245660.	393	171	420.	509
12	595980.	38	52	152040.	148	92	58860.	250	132	16800.	393	172	1200.	509
13	476220.	44	53	111360.	149	93	22560.	250	133	1001040.	405	173	1440.	509
14	58980.	45	54	491760.	155	94	8280.	250	134	108.	405	174	107040.	510
15	5820.	45	55	20340.	156	95	569096.	257	135	116.	405	175	31440.	511
16	100620.	46	56	318060.	160	96	69660.	258	136	240.	405	176	304260.	516
17	6300.	46	57	3120.	160	97	293880.	263	137	480.	405	177	59100.	517
18	328140.	50	58	5400.	160	98	6900.	263	138	840.	405	178	267960.	520
19	4560.	50	59	184740.	162	99	259260.	266	139	419340.	410	179	21720.	520
20	6780.	51	60	15420.	162	100	3780.	266	140	120.	410	180	68760.	521
21	69420.	51	61	100860.	163	101	311776.	271	141	60060.	411	181	262380.	524
22	579120.	54	62	235136.	166	102	111036.	273	142	86880.	412	182	4560.	524
23	577080.	66	63	1620.	166	103	413940.	278	143	6180.	412	183	331020.	528
24	631920.	75	64	621660.	174	104	271620.	281	144	180.	412	184	1722300.	550
25	120.	75	65	560228.	181	105	64560.	282	145	98100.	414	185	47520.	551
26	148200.	77	66	504420.	187	106	299400.	286	146	3300.	414	186	301170.	555
27	928140.	89	67	6780.	187	107	32880.	286	147	342240.	418	187	1020.	555
28	43860.	89	68	245820.	190	108	957960.	298	148	1111320.	433	188	168000.	557
29	3600.	89	69	482288.	197	109	597780.	306	149	8400.	433	189	113310.	558
30	300.	89	70	429660.	202	110	60.	306	150	91680.	434	190	1613550.	578
31	660627.	100	71	995536.	215	111	113340.	307	151	621780.	441	191	92190.	579
32	167880.	102	72	75120.	216	112	466980.	313	152	875700.	453	192	446300.	585
33	186840.	104	73	460020.	222	113	511200.	319	153	634620.	461	193	65250.	586
34	279044.	108	74	32160.	222	114	276000.	322	154	420.	461	194	1900620.	610
35	420120.	113	75	202500.	225	115	185160.	325	155	379380.	466	195	420060.	615
36	67800.	114	76	317820.	229	116	323940.	329	156	18000.	466	196	245280.	619
37	323460.	118	77	265260.	232	117	318420.	333	157	62100.	467			
38	112020.	119	78	36180.	233	118	777960.	342	158	8820.	467			
39	5280.	119	79	174180.	235	119	4200.	343	159	503580.	474			
40	634261.	127	80	19740.	235	120	573140.	350	160	275820.	477			

i= fault number, t_i= execution time, d_i= day of failure

Appendix C
Statistical Analysis Systems

The BMDP biomedical computer program is a system developed at UCLA to provide an easily used set of statistical algorithms [C.1]. Its various modules deal with such topics as:

1. Data screening and description;
2. analysis of variance;
3. regression; and
4. multivariate analysis.

Two modules from the BMDP system were utilized to fit the fault discovery and fault probability expressions introduced in Section 3.

PAR Derivative-free Nonlinear Regression: PAR computes least square estimates of parameters in nonlinear regression. The program is used with regression functions for which analytical expressions for the derivatives are not provided. An iterative pseudo-Gauss-Newton algorithm is used to compute the parameter estimates. Case weights and inequality constraints on arbitrary linear combinations of parameters can be specified and parameters can be held fixed at initial values. Parameters can also be estimated by maximum likelihood.

In general, this program minimizes the weighted residual sum of squares

$$RSS = \sum_{i=1}^{n} w_i[y_i - f(s_i, p)]^2 \tag{1}$$

subject to linear constraints

$$c_\ell(p) = \tilde{b}_{\ell 1}p_1 + \ldots + \tilde{b}_{\ell m}p_m - \tilde{b}_\ell = 0 \tag{2}$$

where

s_i = set of t independent variables $(x_{i1}, x_{i2}, \ldots, x_{it})$ for the ith case,

u_i = dependent variable for the ith case,

w_i = weight for the ith case,

n = set of nonzero weighted cases used,

p = set of m parameters $(p_1, \ldots, p_m)$,

f = function to fit.

Step 1

The constraints (if any) are solved in terms of $m_1 (\leq k, \leq m)$ of the parameters. For simplicity of notation we assume that these m_1 parameters are $p_1, \ldots, p_{m_1}$.

$$p_\ell = \sum_{j=1}^{m_1} b_{\ell j}p_j + b_\ell \ . \tag{3}$$

Redundant constraints are ignored.

Step 2

The solution to the least squares problem is obtained through iterations, described in Steps 2-7. At each iteration the model is linearized at the previous value of $p = (p_1,\ldots,p_m)$ to

$$\tilde{y}_i = \sum_{j=1}^{m} z_{ij}(\tilde{p}_j) + \tilde{e}_i \tag{4}$$

where the p_j's are the parameter values to be estimated.

$$\tilde{y}_i = y_i - f(x_i,p) \tag{5}$$

and

$$z_{ij} = \frac{\partial f(x_i,p)}{\partial p_j} \tag{6}$$

$$j = 1,\ldots,m \quad .$$

If there are any linear equality constraints,

$$\tilde{y}_i = y_i - f(x_i,p) + \sum_{\ell=m_1+1}^{m} \frac{\partial f(x_i,p)}{\partial p_\ell} \; [p_\ell - \sum_{j=1}^{m_1} b_{\ell j}p_j - b_\ell] \tag{7}$$

and

$$z_{ij} = \frac{\partial f(x_i,p)}{\partial p_j} + \sum_{\ell=m_1+1}^{m} b_{\ell j} \frac{\partial f(x_i,p)}{\partial p_\ell} \tag{8}$$

$$j = 1,\ldots,m_1 \quad .$$

Step 3

The program forms the matrix

$$A = \begin{bmatrix} A_{11} & A_{12} \\ & \\ A_{12} & A_{22} \end{bmatrix} \tag{9}$$

(stored in lower triangular form) where A_{11} is m x m, $A_{12} = A_{21}$ is m x 1, and A_{22} is 1 x 1, with elements

$$a_{\ell j} = \sum_{i=1}^{n} w_i z_{i\ell} z_{ij} \tag{10}$$

where

$$Z_{i,m+1} = \tilde{y}_i \quad .$$

If there are linear constraints, m becomes m_1.

Step 4

Using the Gauss-Jordan algorithm for matrix inversion, the diagonal elements of A_{11} are pivoted on in a stepwise manner. Let $A = (a_{ij})$ be the pivoted matrix A. At each step the index of the pivot element is the one that maximizes $\tilde{a}_{rr}/a_{rr}$ among all r such that

$1 \le r \le m$ and r has not been used previously as a pivot index

$\tilde{a}_{rr}/a_{rr} > T$ where T is the tolerance.

Note that $1 - \tilde{a}_{rr}/a_{rr}$ is the squared multiple correlation of "variable" Z_r with previously entered Z "variables."

Step 5

Let $\tilde{p}_r$ be the provisionally updated value of p_r, for $r = 1,\ldots,m$. (If there are linear constraints,

$$\tilde{p}_\ell = \sum_{j=1}^{m_1} b_{\ell j}\tilde{p}_j + b_\ell \tag{11}$$

for $\ell = m_1 + 1,\ldots,m$.) The updated values are then defined by

$$\hat{p}_j = p_j + d(\tilde{p}_j - p_j) \tag{12}$$

$$j = 1,\ldots,m \quad ,$$

where $d(o<d\le 1)$ is the largest value such that maximums and minimums for the parameter values are satisfied. If there are linear constraints then

$$\hat{p}_\ell = \sum_{j=1}^{m_1} b_{\ell j}\hat{p}_j + b_\ell \tag{13}$$

$$\ell = m_1 + 1,\ldots,m \quad .$$

Maximum and minimum values for these $\hat{p}_\ell$ are not necessarily satisfied.

Step 6

A new value for the residual sum of squares RSS is computed. If the new RSS is larger than the previous RSS, the step size d is halved and the p_j are recomputed. This step halving is continued until the value of RSS is not greater than its value on the previous iteration or until the maximum number of increment halvings is reached.

Step 7

The program returns to Step 2 and iterations are continued until either a maximum number of iterations specified is reached or until

$$|RSS^{(N+1)} - RSS^{(N)}|/RSS^{(N+1)} < C \ . \tag{14}$$

After each iteration the parameter values and residual sum of squares are printed. After the last iteration the program reports the asymptotic correlations and standard deviations for the estimated parameters. For each case after the last iteration, PAR lists the predicted and observed values versus selected variable values, residuals versus selected variables, and normal and detrended normal probability plots of residuals can be requested.

P1R Multiple Linear Regression: P1R computes a multiple linear regression equation on all data and on groups or subsets of the data; equations with or without an intercept can be chosen. If a grouping variable is specified to form groups, homogeneity of regression coefficients across groups is tested. It is also possible to specify case weights.

In general the following steps are followed in P1R:

Step 1

The weighted means and covariance matrix (c) are computed. Actual minimums and maximums for each variable are also determined.

$$\bar{x}_i = \frac{\sum_\ell w_\ell x_{i\ell}}{\sum_\ell w_\ell} \qquad c_{ij} = \frac{\sum_\ell w_\ell (x_{i\ell}-\bar{x}_i)(x_{j\ell}-\bar{x}_j)}{\frac{n-1}{n}\sum_\ell w_\ell} \tag{15}$$

where

n = number of nonzero weighted cases to be used,

w_ℓ = weight for case ℓ

$x_{i\ell}$ = value of the ith independent variable for case ℓ.

Step 2

After all data have been read, the standard deviations are computed,

$$s_i = \sqrt{c_{ii}} \ . \tag{16}$$

The means, standard deviations, minimums and maximums, and (if requested) the correlation and/or covariance matrices are printed.

Step 3

The regression intercept and coefficients are estimated. The general form of the equation is

$$y_\ell = a + b_1 x_{1\ell} + b_2 x_{2\ell} + \ldots + b_q x_{q\ell} \tag{17}$$

where

y = the dependent variable = x_d,

a = the intercept,

x = independent variable,

b = regression coefficient,

q = the number of independent variables used.

The regression coefficients are determined by stepwise pivoting the covariance matrix. No variable is pivoted whose squared multiple correlation with previously pivoted variables exceeds 1 - tolerance.

$$c_{ij} = \frac{\sum_{\ell} w_{\ell} x_{i\ell} x_{j\ell}}{\sum_{\ell} w_{\ell}} \quad . \tag{18}$$

Step 4

The intercept (a) and each regression coeifficient (b_i) are computed

$$a = \bar{y} - \Sigma b_i \bar{x}_i = \bar{x}_d - \Sigma b_i \bar{x}_i \tag{19}$$

$$b_i = c_{id} \quad . \tag{20}$$

For all data and all requested groups, the output includes mean, standard deviation, minimums and maximums, multiple R, and standard error of estimates for each variable; an analysis of variance table consisting of regression and residual sum of squares, degrees of freedom and mean squares; F statistic and probability for the regression equation; and the regression coefficients, their standard errors, and t statistics and probabilities. The covariance or correlation matrix; scatter plots, normal and detrended normal probability plots of residuals, and partial residual plots; residuals, predicted values and data can be requested for each case.

Reference

C.1 W. J. Dixon and M. B. Brown, ed., BMDP-79 Bio-medical Computer Programs P-Series, University of California Press, Berkeley, 1979.

Distribution for NUREG/CR-2186 (ANL-81-84)

Internal:

E. S. Beckjord
C. E. Till
R. Avery
J. F. Marchaterre
A. J. Goldman
L. W. Deitrich
D. Rose
D. P. Weber
H. Komoriya
E. E. Morris
C. J. Mueller (30)
G. Ridges
ANL Patent Dept.
ANL Contract File
ANL Libraries
TIS Files (6)

External:

NRC Washington, for distribution per RG and XA (300)
Manager, Chicago Operations Office, DOE
DOE-TIC (2)
President, Argonne Universities Association
Reactor Analysis and Safety Division Review Committee:
- W. P. Chernock, Combustion Engineering, Inc., 1000 Prospect Hill Road, Windsor, Conn. 06095
- L. C. Hebel, Xerox Corp., 3333 Coyote Hill Road, Palo Alto, Calif. 94304
- W. Kerr, U. of Michigan, Ann Arbor, Mich. 48105
- S. Levine, NUS Corp., 4 Research Place, Rockville, Md. 20850
- S. Levy, S. Levy, Inc., 1901 S. Bascom Ave., Campbell, Calif. 95008
- T. H. Pigford, U. of California, Berkeley, Calif. 94720
- J. J. Taylor, Electric Power Research Inst., P. O. Box 10412, Palo Alto, Calif. 94303

120555078377 1 ANRC
US NRC
ADM DIV OF TIDC
POLICY & PUBLICATIONS MGT BR
PDR NUREG COPY
LA 212
WASHINGTON DC 20555