A Binomial Software Reliability Model Based on Coverage of Structural Testing Criteria

Adalberto Nobiato Crespo
USF- Itatiba/CTI - Campinas - SP - Brazil
crespo@ic.cti.br

Mario Jino
FEEC/Unicamp - Campinas - SP - Brazil
jino@dca.fee.unicamp.br

Alberto Pasquini
ENEA - Roma - Italy
pasquini_a@casaccia.enea.it

José Carlos Maldonado
ICMC/USP - São Carlos - SP - Brazil
jcmaldon@icmc.sc.usp.br

Abstract: A new approach to software reliability modeling is discussed where variables indirectly related with software reliability are used to provide additional information for the modeling process. Previous studies, empirical evidences, and results from experiments indicate that there is a strong relationship between software reliability and coverage of program elements required to be exercised by structural testing criteria. In this vein, a Binomial software reliability Model Based on Coverage – BMBC – is proposed and discussed. The BMBC, the first software reliability model based on coverage, was assessed with test data from a real application, making use of the following structural testing criteria: all-nodes, all-edges, and potential-uses – a data-flow based family of criteria. Results from the experiments show a clear superiority of the BMBC over the traditional models and point to a very promising research direction in software reliability.

Keywords: software reliability models, structural testing criteria, test coverage.

I - Introduction

1.1 - Context

The quality-cost tradeoff is a designer's main concern in the development of software products, as it happens in every engineering discipline. Quality is a software characteristic difficult to measure and, according to [ISO9126], it includes the following attributes: *functionality*, *reliability*, *efficiency*, *portability*, *usability*, and *maintenance*. Reliability is defined as the probability that the software won't fail over an interval of time, in a given environment [Musa87]. It is a measure very relevant in making a decision on the release of software. Reliability has been extensively considered in the analysis of software quality and, as a relatively new research area, its study has attracted a significant attention from the scientific community. Measuring the reliability of a software is a very challenging task. Reliability represents quality from the user's point of view; it is known that it is practically impossible to achieve 100% reliability, even for programs which are not very complex. Software reliability is used as a reference by software development organizations; the level of the measured reliability of a product is a criterion for its approval. Tests are extensively performed not only to remove defects from a product but also to determine its reliability level. Concern about software reliability started with Hudson [Huds67] and, from the seventies on, based on concepts from hardware reliability, the first studies and models of software reliability were proposed ([Jeli72] and [Shoo72]). In the eighties, studies on software reliability became more intense and many other diverse models appeared.
The study of software reliability is an analytic approach based on the following concepts:

metrics, a set of precisely defined attributes of the software; *measurements*, objective and mechanical ways for determining values of metrics; and *models*, mathematical laws describing the influence of the metrics on software reliability.

1.2 - Motivation

A mathematical model is called a software reliability model if it is used to obtain a measure of software reliability [Xie93]. Mathematical models of software reliability have a probabilistic nature and attempt somehow to specify the probability of occurrence of software failures. The ultimate goal of the models is to quantify the software reliability as precisely as possible.
Models are used to measure reliability, to analyze failure data, to make inferences about the future behavior of the software, and for decision making during the testing and debugging processes. However, in spite of the great effort in modeling, the proposed models provide only a rough estimate of software reliability. Limitations of reliability models and reasons for their insufficient predictive ability have been pointed out by several authors ([Bish90], [Butl91], [Haml92], [Litt93], [Vara95], [Chen95], and [Chen96]). All models make use of failure data obtained from functional or "black box" testing, assuming that reliability grows automatically with the progress of the test [Haml92]. The concrete result is that a model may provide a good fitness for a given software but be completely inadequate for another one. The usual procedure for software reliability modeling does not make use of information on the coverage of required elements of none of the well-known "white box" testing criteria, despite the occurrence of failures being always connected to the exercise of a program's elements during its execution. Another restriction to reliability models is due to the testing criteria themselves. Every test criterion determines for each program a finite set of program elements that should be exercised by execution of the program on the test data. This set of required elements imposes a limit to the capacity of test data generation of the criterion because, once all the elements in the set are exercised, additional test data will exercise required elements already exercised and are, therefore, equivalent to data already used in the execution of the program [Bast86]. Another restriction to the use of time as a control variable is that there is no guarantee that the test effort is adequate if test data do not exercise at least the main functions of the software. Hence, testing software for reliability estimation merely as a function of time does not make much sense.
Thus, the models so far in use try just to simulate the mathematical function representing the growth of reliability; they do not address the reasons underlying the phenomenon represented by their functional form.
Based on the reasoning above, the approach has been reinforced of modeling software reliability by using information somehow connected to it, such as code coverage. Experiments and studies have been conducted to investigate the relationship between code coverage and software reliability, such as: Ramsey and Basili [Rams85], Adams [Adam80], Garg [Garg94], Veevers and Marshall [Veev94], Malaiya and others [Mala94], and [Vara95]. A recent experiment using control flow and data flow based testing was conducted by Frate, Mathur and Pasquini [Frat95]. Their results provide evidence of a relationship between software reliability and coverage of elements required by the testing criteria. A more recent experiment, by Crespo, Jino, Pasquini, and Maldonado [Cresp97a], investigates the relationships between software reliability and coverage of elements required by the following testing criteria: all-nodes, all-arcs, and the data flow based family of criteria - potential-uses, all-potential-uses, all-potential-uses/du, and all-potential-du-paths.
Previous studies, empirical evidences, and results from the experiments provide the motivation for a research effort aimed to determine and make use of the relationship between software reliability and code coverage; that is, to devise and adopt an alternative approach

which takes into account the structure of the software in the process of reliability modeling. There is no record of software reliability models using coverage information for reliability estimation. The development is described of such a reliability model where information on the coverage of testing criteria is used directly as a parameter of the model.

1.3 - The Approach

Reliability is expressed typically as a function of time units. On the other hand, coverage of elements required by a testing criteria does not involve time but the number of test data or input data. Nevertheless, the relationship between coverage and reliability can be easily derived from the frequency of execution of test data, that is, the number of test data executed by time unit.
In this approach, the execution of a test datum is assumed to correspond to a time unit of software execution. Thus, the information on the coverage can be used directly in the process of software reliability modeling.
Malaya [Mala94] makes the same assumption to create a model establishing a relationship between code coverage and number of test data. His model explains the coverage as a function of test data. Chen [Chen94] also makes that assumption when using information on coverage to define a compression factor to correct the overestimation of reliability by the traditional reliability models.

1.4 - Organization of the Paper

Section II presents basic concepts of testing criteria and of coverage of elements required by a testing criterion. Section III contains the theoretical background of software reliability, the development of the binomial model based on coverage, the measurements of reliability evaluation, and the procedure for estimation of the parameters of the model. Section IV describes the use of the model with data from testing a real application software; results from the proposed model are compared to those from traditional models. Section V presents our conclusions and suggestions for future work.

II – Testing Criteria and Code Coverage

2.1 - A General View of Testing Criteria

There are several ways of testing software. Functional test, structural test, regression test, unit test, integration test, and system test are terms used to characterize diverse features and points of view of software testing. Such terms usually characterize the testing criteria to be applied in testing or correspond to the testing phases/products of the software life cycle.
Functional test or "black box" test is a term used to encompass the testing criteria which address the functions of software; structural test or "white box" test is a term used for criteria addressing the structure of software. Both approaches can be applied to: testing of a module - unit test; testing the integration of modules - integration test; testing the system as a whole - system test; testing during the maintenance phase - regression test. A unit is the code of a small component of the software – a product of the coding phase; the integrated modules are the aggregation of all units as a whole, as defined by the design phase; and the system is the whole of which the software is a part. The regression test is done to evaluate the effect of modifications due to maintenance activities on a released product as well as to evaluate new versions of the product.

- *Functional Test*, or "black box" test - test data are selected to exercise a program's functions, that is, test requirements are derived from the functional requirements of the software regardless of how they are implemented.
- *Structural Test*, or "white box" test – test data are selected to exercise the elements of the

implementation of the software, that is, test requirements are derived from information associated to the control and data structures of the procedural design.
Structural testing methods or criteria may classified as:
Control Flow Based Testing – the test requirements are based on the constructs of the control flow of a program code. Examples of control flow testing criteria: *statement testing (or node testing), branch testing* (or *arc testing*), *condition testing*, and *multiple-condition testing*.
Data Flow Based Testing – the test requirements are based on the types of occurrence of variables in a program code, basically, definitions and uses of variables. Examples of data flow based testing criteria: i) data flow based family of criteria - *all-uses, all-c-uses and all-p-uses* [Rapp85]; ii) potential-uses family of criteria – *all-potential-uses* (PU), *all-potential-uses/du* (PUDU), *all-potential-du-paths* (PDU) ([Mald92a] and [Mald92b]).
Each of those testing criteria can be used either to evaluate or to select test data sets.

2.2 - Coverage of Elements Required by Testing Criteria

Test coverage is always associated to a testing criterion. A coverage value expresses the percentage of already exercised required elements with respect to the total number of elements required by the testing criterion. Coverage of 100% indicates that all the elements required by the criterion were exercised by execution of a test data set.
Functional testing is the most usual way of validation of a software product. A certain amount of input data is selected, the software is executed on these data, and a check is made whether the produced outputs match the expected results; the internal structure of the software is not considered in this process. The confidence level on the product reliability increases when a large number of input data executed by the software yields correct results.
However, the increasingly greater capacities of hardware and computational systems have made software systems and applications increasingly complex and diversified. Functional testing, although still necessary for validation, is not enough to assure that the product has been adequately tested, as it does not check the constructs of a software's implementation; it does not provide information on what is or is not being exercised in the code.
Code coverage is a measure of the extent a test data set exercises the several regions and elements of the code. That is, the coverage is a metrics used to assess the adequacy of a test data set concerning the appropriate degree of code exercising. Coverage information can also be used to evaluate the quality of the test data set; if all the elements required by a strong (more demanding) criterion are exercised we can say the test data set is of good quality. A criterion stronger than another one demands usually a larger number of test cases to exercise its required elements. If there is a growth in coverage, that is, a new test datum exercises an area or element of the code still not exercised, the chance increases of revealing new defects. Therefore, coverage information provides a more concrete basis for the use of the testing activity to measure appropriately software reliability.

III - A Binomial Model Based on Coverage - BMBC

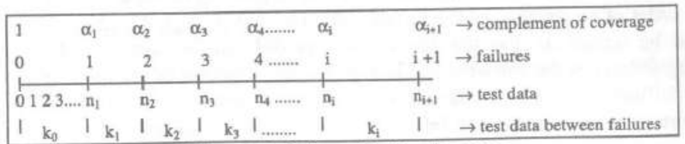
3.1 - Foundations

The failure process of the software for a binomial type of model is characterized by the behavior of the rate of occurrence of failures in the software. The number of failures occurring in the software follows a binomial distribution of probabilities, this being the reason for its characterization as a binomial type, according to Musa's classification [Musa87]. The functional form of the failure rate characterizes completely the software reliability model.
In the proposed model , *time* is replaced as the control variable by the variable *number of test data*, that is, the measurement unit of software testing is the test datum. This is necessary as

reliability is a characteristic defined as a function of time. Thus, the failure rate of a software is characterized by the number of failures by test datum or by test data set.
However, as discussed previously, a number of test data executed by the software results in a certain coverage of the software code. Moreover, the coverage value depends on the testing criterion used for evaluation of the test data. Hence, the failure rate of the software is also related to the coverage of the testing criterion achieved by execution of the test data.
It should also be remarked that the failure rate is high and the test coverage is low in the initial stages of testing as few test data have been executed; in the final phase of the testing, the failure rate is low and the test coverage is usually high. That is, the complement of test coverage can also be used as a variable directly related to the failure rate.
Hence, the basic assumption of the proposed binomial model is that the failure rate of the software is directly proportional to the complement of the coverage achieved by execution of the test data. Empirical evidence to support this assumption is given in [Cresp97a] and [Cresp98].
The normalized coverage instead of the observed coverage is used with the purpose of estimating the software reliability independently of testing criteria.

3.2 - Development of the Model

As previously mentioned, the binomial type of model is characterized by the functional form of the failure rate by defect, $Z_a(n)$, where n is the number of test data applied to reveal the defect "a".
Thus, in the proposed model, the failure rate $Z_a(n)$ for a defect "a" is proportional to the following measurements:
a) Number of test data executed by the software until the occurrence of the failure caused by the defect "a".
b) complement of the coverage reached with the execution of the test data up to the occurrence of the failure caused by the defect "a".
c) weight of the test criterion used as strategy for the generation of the test data.
With those assumptions, the following functional form is proposed for the failure rate of defect "a": $Z_a(n) = \alpha n^{\alpha - 1}$, where n is the number of test data executed for detection of the defect "a" and α is the complement of the coverage reached with the execution of n test data, $0 \leq \alpha \leq 1$. The value of α is obtained according to the strength of the criterion used for selection of the data used in the test. The strength of a criterion is associated to the level of difficulty in achieving the coverage of its required elements.
Since the coverage reached by the test data is associated to the criterion, its value is related to the strength of the selection criterion used in the test. When the maximum coverage is reached, $\alpha = 0$ (reminding that α is the complement of the coverage), the selection criterion used in the test reaches its saturation point; from this point on, it is not possible to assess the quality of a test datum, making the failure rate $Z_a(n) = 0$.
The binomial model proposed here is based on the following assumptions:


1 α1 α2 α3 α4....... αi αi+1 → complement of coverage
0 1 2 3 4 .... i i +1 → failures
0 1 2 3.... n1 n2 n3 n4 ...... ni ni+1 → test data
k0 k1 k2 k3 .... ki → test data between failures
Figure 3.1- Software Testing Process

1. The softwarpzpe is operated in a similar manner as the anticipated operational usage;
2. Each defect has the same probability of being detected within the same class of difficulty of detection;
3. Defects 1, 2, 3, ...k, detected , respectively, in each one of the intervals $(0 ; n_1)$, $(n_1 ; n_2)$, $(n_2 ; n_3)$,..... $(n_{k-1} ; n_k)$, are independent;
4. There are N defects in the software in the beginning of the test;
5. The test data are executed and the coverage of the elements required by the selection criterion used in test data evaluation is calculated for each failure occurrence; and
6. For a defect "a", the failure rate, Za(n), has the functional form: $Z_a(n)=\alpha n^{\alpha-1}$, where n is the number of test data executed for detection of the defect "a" and α is the complement of the coverage reached with the execution of n test data.

Assumption 1 guarantees that the model estimates obtained with data collected in the test environment are valid in the operation environment of the software. Assumption 2 guarantees that all the defects have the same properties within their distributions. By Assumption 3 the joint probability density function of the maximum likelihood method is obtained by multiplying the density functions of each of the random variables – number of test data between failures. If the hypotheses are valid the maximum likelihood method guarantees the desirable properties for the estimator [Mood74]. Assumption 4 indicates that the number of detected failures is finite and, therefore, the model belongs to the category of finite failures, according to Musa's classification [Musa87]. This assumption considers that the process of removal of defects is perfect; that is, no new defect is inserted into the software when removing an existent defect. Assumption 5 is a characteristic of the testing procedure and indicates that the coverage should be measured at each failure occurrence. Assumption 6 characterizes the functional form of the failure rate per defect in the software. It should be remarked that the random variable *test data* is discrete and the failure rate, as defined in Assumption 6, is the failure rate of the Weibull's model, with parameter $\beta = 1$ [Musa87]. Thus, it is implicit in Assumption 6 that the random variable *number of test data* has approximately a Weibull's distribution. This approximation is made possible by the assumption that the application of a test datum corresponds to a time unit of execution. The failure rate of the software is a measure that depends on the failure rate by defect and on the number of defects in the software [Musa87]. Figure 3.1 illustrates the software testing process.

Observe that: $n_0 = 0$; $n_1 = k_0$; $n_2 = n_1+k_1$; $n_3 = n_2+k_2$; $n_{i+1} = n_i+k_i$; and $\alpha_0 = 1$.

Notice that k_i is the number of additional test data to be applied for the occurrence of the (i+1)-th failure.

Therefore, according to Assumptions 4 and 6 the failure rate of the software, conditioned to the remaining defects in the software, is defined as:

$Z(k_i \mid n_i) = [N - i].Z_a(n_i+k_i)$, that is, $Z(k_i|n_i) = [N-i]\alpha_i(n_i+k_i)^{\alpha_i-1}$, where

α_i : is the coverage's complement, reached with the execution of the n_i test cases, $0 \le \alpha_i \le 1$;
N: is the number of defects in the software at the beginning of the test ;
i: is the order of occurrence of the failures, that is, i = 0, 1, 2, 3, 4,..., N.

It should be remarked that the failure rate, as defined, is conditioned to the number of remaining defects in the software and is a decreasing function of the number of test data.

Figure 3.2 illustrates the behavior of the failure rate function of the software.

The failure rate function has the following interpretation: Consider the weighting factors of a selection criterion considered weak and a selection criterion considered strong. The test data generated to satisfy a stronger selection criterion have a greater chance than those of a weaker one of revealing defects in the software as they exercise a larger number of distinct paths of the program. This means that, when a stronger criterion is adopted, the greater is the chance of revealing a larger number of defects. That is, the failure rate will decrease as the coverage increases probably more slowly for a stronger criterion than for a weaker criterion.

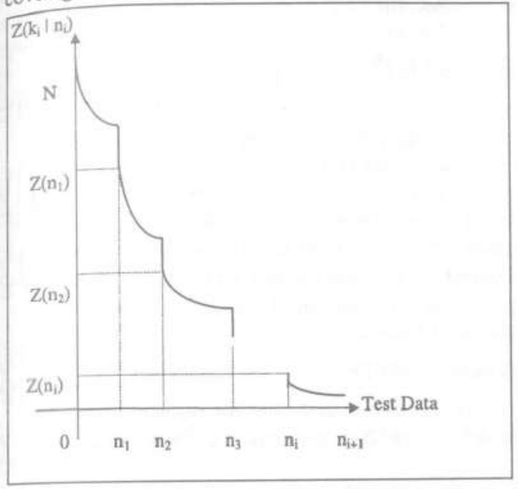


Figure 3.2 – **Behavior of the Failure Rate Fucntion**

Hence, the weighting factor of the criterion (the criterion strength) influences the scale of the failure rate of the software.

Therefore, the normalized coverage will be used instead of the observed coverage to standardize the estimates generated by the model, independently of any criterion. Thus, the parameter α will represent the normalized coverage, that is, $\alpha_i = g(c_i)$ where c_i represents the coverage observed in the test, after the detection of the i-th failure and g(.) is an increasing function of the coverage.

In this model, the normalized coverage is obtained by adopting the linear transformation $\alpha_i = \theta_0 + \theta_1 c_i$. Other transformations can be used according to the assumptions and/or to the data obtained from testing. The parameters θ_0 and θ_1 can be estimated by the maximum likelihood method, according to Assumption 3. Observe that the complement of the coverage as well as the complement of the normalized coverage are decreasing functions of test data. The constraint is that $0 \le \theta_0 + \theta_1 c_i \le 1$, that is, $0 \le \alpha_i \le 1$.

Using the linear transformation for the coverage normalization, the conditional function failure rate of the software has the following functional form:

$$Z(k_i|n_i) = [N-i]\alpha_i(n_i+k_i)^{\alpha_i-1} \tag{3.1}$$

where:

$\alpha_i = \theta_0 + \theta_1 c_i$ is the normalized complement of the coverage and c_i is the complement of the observed coverage reached with the execution of test data n_i, $0 \le c_i \le 1$;
N is the number of defects in the software at the beginning of the test; and
i is the order of occurring failures, that is, i = 0, 1, 2, 3, 4, ... N

3.3 - Measures of Reliability Evaluation

The quantitative measures for evaluation of software reliability based on a binomial model of reliability growth are:

a) Reliability Function of the Software

When reliability growth is of concern, a usual measure of software reliability is the conditional reliability ([Goel79] and [Musa87]). Given that i failures occurred, the conditional reliability is the survival function associated with the occurrence of the (i+1)-th failure.

Musa [Musa87] shows that, for binomial models of reliability growth, the conditional reliability R(ki I ni) can be obtained by the failure rate. Thus, the conditional reliability has the following form:

$$R(k_i \mid n_i) = \exp\left\{-[N-i][(n_i+k_i)^{\alpha_i} - (n_i)^{\alpha_i}]\right\} \qquad (3.2)$$

If $k_i = 0$ then $R(0 \mid n_i) = 1$. That is, given that i defects were removed with the execution of n_i test data, if no additional test datum is applied, the software won't be executed and, consequently, there won't be failures.

If $k_i \to \infty$ then $\lim\limits_{k_i\to\infty} R(k_i \mid n_i) = \lim\limits_{k_i\to\infty} \exp\left\{-[N-i][(n_i+k_i)^{\alpha_i} - (n_i)^{\alpha_i}]\right\} = 0$

if $[N - i]$, n_i and α_i are known values.

Therefore, the conditional reliability depends on the number of remaining defects in the software, $[N - i]$, and on the number of test data used in the removal of the i defects. That is, it is conditioned to the defects still remaining in the software and to the number of test data used up to the current period of test. It should be observed that the intervals of test data between failures, k_i, do not have the same size; their average size grows as testing continues. Thus, the conditional reliability decrease is less pronounced in the final stages than in the initial stages of testing. The variation with defect removal in the complement of the coverage, α_i, causes alterations in the form of the conditional reliability function.

b) Mean Number of Test Data Between Failures - MTTF

The probability density function, $f(k_i|n_i)$, of the number of test data for occurrence of the failure "i+1", given that "i" failures were detected with n_i test data, can be obtained as a function of the failure rate, as follows:

$$f(k_i \mid n_i) = \alpha_i (n_i+k_i)^{\alpha_i-1} \exp\left\{-(n_i+k_i)^{\alpha_i} + (n_i)^{\alpha_i}\right\} \qquad (3.3)$$

The mean number of test data for the occurrence of the failure "i+1" is given by:

$$MTTF = \exp(n_i^{\alpha_i})\left[\Gamma\left(\frac{1}{\alpha_i}+1\right) P[G\left(\frac{1}{\alpha_i}+1;1\right)] > n_i^{\alpha_i}\right] - n_i \qquad (3.4)$$

where $\Gamma(.)$ it is the Gamma function and G is a random variable with Gamma distribution with parameters $[(1/\alpha_i)+1;1]$.

Observe that the mean number of test data for the occurrence of the next failure depends on the number of test data executed n_i and on the coverage α_i reached in the testing process.

3.4 - Estimation of the Parameters of the Model

The parameters of the proposed model can be estimated by the maximum likelihood method. The relationship of the probability density function to the rate of failures is given by:

$$f(k_i \mid n_i) = Z(k_i \mid n_i) R(k_i \mid n_i)$$

Then, from (3.1) and (3.2), it follows that:

$$f(k_i|n_i) = [N-i]\alpha_i (n_i+k_i)^{\alpha_i-1} \exp\left\{-[N-i][(n_i+k_i)^{\alpha_i} - (n_i)^{\alpha_i}]\right\} \qquad (3.5)$$

From the assumption of the independence of failure occurrences and assuming that "r" failures were detected in the testing period, the likelihood function has the form:

$$L(K_0, K_2, \ldots, K_{r-1}; N, \alpha_i) = f(k_0) f(k_1 \mid n_1) f(k_2 \mid n_2) \ldots f(k_{r-1} \mid n_{r-1}) =$$

$$\prod_{i=0}^{r-1} [N-i]\alpha_i (n_i+k_i)^{\alpha_i-1} \exp\left\{-[N-i][(n_i+k_i)^{\alpha_i} - (n_i)^{\alpha_i}]\right\} \qquad (3.6)$$

The logarithm of the likelihood function above is taken, the normalized coverage is replaced by its expression ($\alpha_i = \theta_0 + \theta_1 c_i$), and (3.6) is differentiated with respect to N, θ_0, and θ_1, to obtain the system of equations (3.7).

The values of N , θ_0 , and θ_1, that satisfy simultaneously the system (3.7) are the estimates of the parameters. The solution of the system can be found through numerical procedures.

$$\begin{cases} \sum\limits_{i=0}^{r-1} \dfrac{1}{[N-i]} = \sum\limits_{i=0}^{r-1} \left\{[(n_i+k_i)^{(\theta_0+\theta_1 c_i)} - (n_i)^{(\theta_0+\theta_1 c_i)}]\right\} \\ \\ \sum\limits_{i=0}^{r-1} \left[\dfrac{1}{\theta_0+\theta_1 c_i} + \ln(n_i+k_i)\right] = \\ \sum\limits_{i=0}^{r-1} \left[-[N-i](n_i+k_i)^{(\theta_0+\theta_1 c_i)} \ln(n_i+k_i) - (n_i)^{(\theta_0+\theta_1 c_i)} \ln(n_i)\right] \\ \\ \sum\limits_{i=0}^{r-1} \left[\dfrac{c_i}{\theta_0+\theta_1 c_i} + c_i \ln(n_i+k_i)\right] = \\ \sum\limits_{i=0}^{r-1} \left[-[N-i](n_i+k_i)^{(\theta_0+\theta_1 c_i)} c_i \ln(n_i+k_i) - (n_i)^{(\theta_0+\theta_1 c_i)} c_i \ln(n_i)\right] \end{cases} \qquad (3.7)$$

An alternative way of obtaining the estimates of the parameters is to use optimizing routines to maximize the likelihood function (3.6), simultaneously as a function of N, θ_0 , and θ_1.

IV - An Example of Application of the BMBC

Application of the BMBC is shown using data from an experiment conducted by Crespo [Cresp97a] at the research center ENEA - "Ente per le Nuove tecnologia L'Energie e L'Ambiente" , Rome , Italy.

4.1 - Description of the Experiment

The experiment was carried out to investigate the relationships involving software reliability and code coverage of elements required by the testing criteria all-nodes, all-arcs, all-Potential-Uses (PU), all-Potential-Uses/DU (PUDU), and all-Potential-DU-paths (PDU).

The study of those relationships has provided a fairly detailed evaluation of the behavior of software reliability as a function of test data and code coverage. Details on the experiment can be found in [Cresp97a].

Data collected on code coverage and on software reliability, from the relationships investigated in the experiment, are used to evaluate the proposed model.

The application software used in the experiment was developed for the European Space Agency – ESA, in the language C; it consists of approximately 10,000 lines of code, 6,000 being lines of executable code. The system comprises a main program and 134 routines interconnected through parameters.

During the integration testing and the operational use of the software, 33 defects were revealed, corrected, and recorded; thus, the recorded failures in the software were caused by real defects detected in testing and in the operational use of the system.

In the experiment, 28 defects of the 33 recorded defects were detected after the execution of 1,240 test data; the 5 remaining defects were not detected, even after the execution of 20,000 test data. pp

The experiment was performed with the support of a testing tool called POKE-TOOL -

Potential uses Criteria TOOL for program testing, an integrated set of tools aimed to help its users in the tasks of software testing [Chaim91].
The values of reliability were estimated through Nelson's method [Nels78], that is, the "brute force" method, to establish the relationship of software reliability to other variables.
In this method, for each occurrence of a failure and after the removal of the corresponding defect, the software is executed with thousands of randomly generated test data. The ratio is calculated between "n_e", the number of executions with failure, and "n", the total number of executions of the software. This ratio is an estimate of the probability of occurrence of a failure in the software. The reliability of the software is then estimated as $R = 1 - \frac{n_e}{n}$.
For each defect removed from the software, the method is applied again to recalculate the reliability. Thus, an estimate of the behavior of the reliability growth of the software is obtained as a function of defect removal. Notice that reliability is always calculated after the removal of a defect.
Table 4.1 shows removed defects, test data accumulated up to removal of defects, and observed reliability. It also gives a general picture of the coverage achieved for each of the testing criteria.

Table 4.1 - **Coverage and Reliability as a Function of Test Data and Removed Defects**

Removed Defects	Test Data between Failures	Accumulated Test Data	COVERAGE OF TESTING CRITERIA					Observed Reliability
			NODES	ARCS	PU	PUDU	PDU	
0	0	0	0	0	0	0	0	0.028554
1	1	1	0.30684	0.21296	0.16709	0.15177	0.07406	0.046522
2	1	2	0.33769	0.22341	0.17012	0.15913	0.07812	0.046941
3	1	3	0.35792	0.23576	0.17975	0.16221	0.08145	0.047827
...	...	...	...	...	...	...	...	...
25	60	186	0.68356	0.51984	0.40992	0.36453	0.2106	0.990667
26	253	439	0.71575	0.58068	0.41163	0.41134	0.24917	0.999333
27	400	839	0.71575	0.58068	0.45418	0.41163	0.25011	0.997333
28	401	1240	0.71781	0.58465	0.46212	0.41560	0.25224	0.998667

Legend: PU – All- Potential-Uses; PUDU – All-Potential-Uses/DU; PDU – All-Potential-DU-paths

4.2 – Application of the Binomial Model Based on Coverage - BMBC

The results are shown of the application of reliability models on the data of Table 4.1. Some of the best known reliability models are used to compare their results to those of the BMBC, to evaluate the performance of the proposed model in comparison to traditional software reliability models.
Estimation is made with the available data to establish the parameters' values which characterize the reliability models for the software under test; from this, the estimates of reliability of the software can be obtained. The reliability estimates generated by the proposed model are graphically and statistically compared to those generated by traditional models, as well as to the observed reliability of Table 4.1.

4.2.1 - Results from Traditional Models

Several researchers say that the best model of software reliability does not exist. A model can be a very good fit to a data set and, at the same time, be a terrible fit to another one. The recommended procedure is to apply the largest possible number of models and to choose the one that best fits the data [Lyu96].
The following models were applied to the set of failure data: **GEO** - Geometric Model; **LV-Q** - Bayesian Model of Littlewood and Verral - Quadratic; **LV-L** - Bayesian Model of Littlewood and Verral - Linear; **MUS-B** - Basic Model of Time of Execution of Musa; **MUS-**

Table 4.2 - **Quality of Fitness of the Traditional Models**

Model	Distance of Kolmogorov	Statistics χ^2	Significant to level 0.05
GEO	0.09166	*	No
LV-Q	0.12478	*	No
MUS-B	0.43031	*	Yes
MUS-L	0.13543	*	No
BM-B	*	26.83	Yes
BM-P	*	1.827	No

Legend: * - Statistics not available for the model type

L - Model of Log Poisson Time of Execution of Musa; **BM-B** - Binomial Model of Brooks and Motley; **BM-P** - Poisson Model of Brooks and Motley; **GPM** – Generalized Poisson Model; **NHPP** - Non Homogeneous Poisson Process Model; **SCHN** - Schneidewind Model; and **JM** - Model of Jelinski and Moranda.

SMERFS - "Statistical Modeling and Estimation of Reliability Functions for Software", a software developed specifically for the purpose was used to apply those models. Some models require the grouping of failure data, others require the number of test data between failures. Whenever necessary, the conversion of the data from one type to the other was done.
The following of the eleven models were rejected by SMERFS in a first analysis: **LV-L, GPM, NHPP, SCHN,** and **JM.**
Table 4.2 illustrates, in a summary form, the results of the statistics obtained by SMERFS for the six remaining models. Some statistics are available for just a type of model.
The Distance of Kolmogorov and the χ^2 statistics are measures that evaluate the quality of fitness of the model to the data. The Distance of Kolmogorov is a measure of the discrepancy between the sampling distribution function and the model. It represents the maximum distance between the observed function distribution and the theoretical function distribution assumed for the model. If this distance is significantly large the model is inadequate and should be rejected. The χ^2 statistics is a measure to evaluate the hypothesis test that the data concur with the adjusted model. The χ^2 test rejects the null hypothesis that the data concur with the model if the value of the statistics is significantly larger than $\chi^2(1-\alpha,gl)$, a Chi-square distribution with gl degrees of freedom at a significance level α. This statistical test is applied for reliability models that deal with grouped data. In this way, the models' analysis is made based on the results of the quality of fitness.

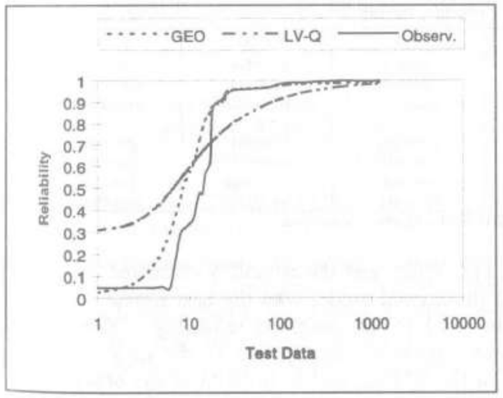


Figure 4.1 - **Models with Best Fitness**

From the results in Table 4.2, the **MUS-B** is rejected because it has the largest Distance of Kolmorogov and the distance is significant. The **BM-B** is rejected because the χ^2 Statistics is the largest and it is significant. Considering a classification based on the Distance of Kolmogorov and on the χ^2 Statistics, the models that are better fitted to the data are: **GEO, LV-Q, MUS-L,** and **BM-P.**
Of these four, **GEO** and **LV-Q** are the models with the best fitness. Figure 4.1 illustrates graphically the behavior of the reliability estimated by these two models compared with the observed reliability of the software. From Figure 4.1, we can see that **GEO** – the Geometric Model, is the one that best fits the data; it is the model chosen for comparison to the proposed model.

Table 4.3 - **Parameters of the Binomial Model Based on Coverage - BMBC**

Testing Criteria	Parameters of the BMBC		
	N	θ_0	θ_1
NÓDES	29.8757	0.1521	0.1354
ARCS	30.2713	0.1069	0.1685
PU	30.2553	0.0577	0.2142
PUDU	30.3434	0.0325	0.2417
PDU	29.7748	0.0156	0.2358

Legend: PU – All-Potential-Uses; PUDU – All-Potential-Uses/DU; PDU – All-Potential-DU-paths

4.2.2 - Results of the Binomial Model Based on Coverage - BMBC

The parameters of the model were estimated using an optimization routine of the software Matlab, applied to the likelihood function from (3.6).
Table 4.3 shows the results of the estimation of the parameters of the model by using the data from Table 4.1, for each of the testing criteria.
Table 4.4 shows partially the observed reliability and the reliability estimated by the model, for each of the testing criteria.
Figure 4.2 illustrates graphically the behavior of the observed reliability and of the reliability estimated by the BMBC, for each of the testing criteria. The lack of difference among the values of the reliability estimated by BMBC for the different testing criteria is due to the use of the normalized coverage instead of the observed coverage; the values of reliability estimated by the model will be approximately the same for all of the testing criteria. Visually, the values of the observed reliability and of the estimated reliability are very close indicating that the model is well fitted to the data.

Table 4.4 - **Observed Reliability and Reliability Estimated by the BMBC.**

Observed Reliability	ESTIMATED RELIABILITY				
	TESTING CRITERIA				
	NODES	ARC S	PU	PUDU	PDU
0.046522	1.0596E-13	7.1341E-14	7.2492E-14	6.6379E-14	1.1721E-13
0.046941	0.00466664	0.00506045	0.00559184	0.00546866	0.00631073
0.047827	0.03354123	0.03426364	0.03699339	0.03562533	0.03940258
.....	...	...	...	...	...
0.990667	0.98298697	0.98309167	0.98389291	0.98332042	0.98139217
0.999333	0.99292896	0.99294813	0.99328957	0.99385162	0.99253303
0.997333	0.99682726	0.99701581	0.99713337	0.99707067	0.9968329
0.998667	0.99828307	0.99834167	0.9984451	0.99838577	0.99830531

Legend: PU – All-Potential-Uses; PUDU – All-Potential-Uses/DU; PDU – All-Potential-DU-paths

Figure 4.3 illustrates graphically the observed reliability and the reliability estimated by the BMBC and by the Geometric Model - GEO (traditional model with the best fitness to the data). BMBC has an adjustment closer than GEO to the observed reliability. Table 4.5 consolidates the results from the Kolmogorov Smirnov's test applied to the data of the observed reliability and estimated reliability of the BMBC, and to the data of the observed reliability and estimated reliability of the GEO. In a sample of size 28 with a significance level of $\alpha = 0.01$, the bilateral hypothesis H_0 of equality between the observed reliability and the reliability estimated by the model is accepted for a value of statistical KD below 13 [Sieg77]. From Table 4.5, the hypothesis of equality is accepted for both models. However, for the maximum distances (Dmax), the smaller happens for the BMBC, showing that its results are closer to the observed reliability than those of the geometric model (GEO).
Expression (3.1) refers to a failure rate of the model of Weibull with the parameter $\beta = 1$ and the parameter α varying with the coverage [Musa87]. Thus, the BMBC is a model of *test data between failures* based in the model of Weibull. The BMBC is equivalent tpo a model of Weibull where $\beta = 1$ and α changes at each occurrence of a failure. Hence, by applying the model of Weibull to the data of Table 4.1 (without use of the coverage), the influence of the

Table 4.6 - **Estimated Parameters of the Models BMBC and Weibull**

	BMBC			Weibull		
	N	θ_0	θ_1	N	α	β
Parameters	29.8757	0.1521	0.1354	28.001	0.673066	0.024074

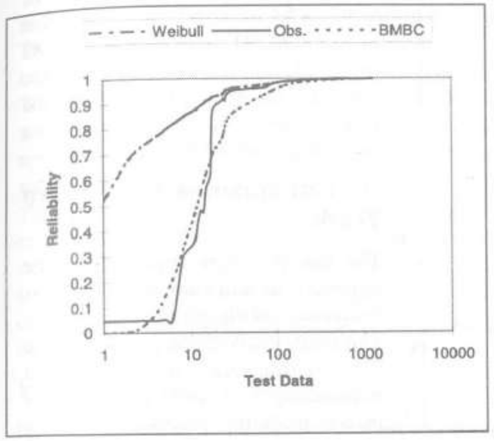


Figure 4.4 - **Reliability Estimated by the models BMBC and Weibull**

Table 4.7 **Reliability and MTTF, Estimated and Observed.**

	Estimated by BMBC	Observed
Reliabilitv	0.974967	0.990667
MTTF	82	253

Legend: MTTF- Mean Number of Test Data for the Ocurrence of the Next Failure

BMBC underestimates the value of software reliability, an importan property for a software reliabilit model. Most of the traditional model overestimate the reliability of th software, a much criticize characteristic. The mean number of test data for the occurrence of the nex failure – MTTF, was estimated as 8 by the model; according to Table 4.1 the 26th failure happened with th 253th test data, again a underestimation of the reliability.

V - Conclusions and Futur Work

The use of a new approach for th modeling of software reliability wa described and discussed.
The main focus of the approach is t use information which has relationship with software reliabilit in the modeling process. The mai experiments and studies o investigation of the relationship between code coverage and softwar reliability were discussed. Thei results provided evidence that th coverage of structural testing criteri is strongly related with softwar reliability. The approach where cod coverage is used to deriv information related to reliability is a alternative to the traditional "black box" testing approach.
The described approach considers the effect of saturation of the selection criterion used in th test. The saturation effect happens when a testing criterion reaches the limit of its capacity o generating data that are distinct, according to the criterion, from data already generated Information on the coverage of the elements required by the selection criterion provides a wa to control the saturation effect.
A new software reliability model was proposed: the Binomial Model Based on Coverage - BMBC, a model based on the coverage of elements required by structural testing criteria where information on the coverage is used as a parameter in the formulation of the model The BMBC is the first software reliability model based on coverage.
Data on software failures from a real application were applied in the model. Failure data wer obtained of the application of the testing criteria: all-nodes, all-arcs, and the data flow base criteria of the Potential-uses family of criteria. The same failure data were used to estimat software reliability with well-known traditional software reliability models. For the particula application software, the results showed a clear superiority of the model BMBC – Binomia Model Based on Coverage.

Table 4.6 - **Estimated Parameters of the Models BMBC and Weibull**

	BMBC			Weibull		
	N	θ_0	θ_1	N	α	β
Parameters	29.8757	0.1521	0.1354	28.001	0.673066	0.024074

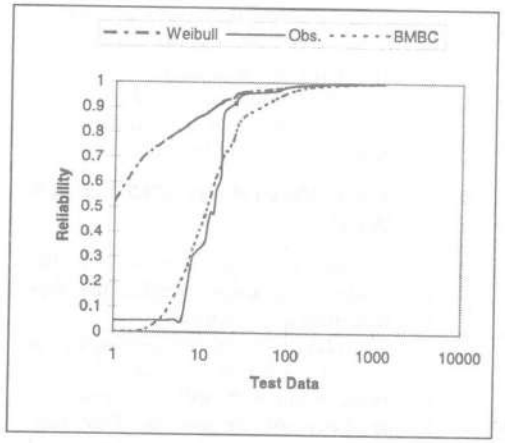


Figure 4.4 - **Reliability Estimated by the models BMBC and Weibull**

Table 4.7 **Reliability and MTTF, Estimated and Observed.**

	Estimated by BMBC	Observed
Reliabilitv	0.974967	0.990667
MTTF	82	253

Legend: MTTF- Mean Number of Test Data for the Ocurrence of the Next Failure

BMBC underestimates the value of software reliability, an important property for a software reliability model. Most of the traditional models overestimate the reliability of the software, a much criticized characteristic. The mean number of test data for the occurrence of the next failure – MTTF, was estimated as 82 by the model; according to Table 4.1, the 26th failure happened with the 253th test data, again an underestimation of the reliability.

V - Conclusions and Future Work

The use of a new approach for the modeling of software reliability was described and discussed.

The main focus of the approach is to use information which has a relationship with software reliability in the modeling process. The main experiments and studies of investigation of the relationships between code coverage and software reliability were discussed. Their results provided evidence that the coverage of structural testing criteria is strongly related with software reliability. The approach where code coverage is used to derive information related to reliability is an alternative to the traditional "black box" testing approach.

The described approach considers the effect of saturation of the selection criterion used in the test. The saturation effect happens when a testing criterion reaches the limit of its capacity of generating data that are distinct, according to the criterion, from data already generated. Information on the coverage of the elements required by the selection criterion provides a way to control the saturation effect.

A new software reliability model was proposed: the Binomial Model Based on Coverage – BMBC, a model based on the coverage of elements required by structural testing criteria, where information on the coverage is used as a parameter in the formulation of the model. The BMBC is the first software reliability model based on coverage.

Data on software failures from a real application were applied in the model. Failure data were obtained of the application of the testing criteria: all-nodes, all-arcs, and the data flow based criteria of the Potential-uses family of criteria. The same failure data were used to estimate software reliability with well-known traditional software reliability models. For the particular application software, the results showed a clear superiority of the model BMBC – Binomial Model Based on Coverage.

The results confirmed the empirical evidences from the literature and reinforce the results of previous studies and experiments. A major conclusion is the importance of using variables related to software reliability in the process of modeling software reliability; use of time only can lead to bad estimates of reliability. Moreover, coverage of elements required by structural testing criteria has a strong relationship to software reliability and, therefore, is an information that should be used in the modeling process.

The BMBC estimated a reliability level as a function of the code coverage reached in software testing; the overestimation effect of the traditional software reliability models did not happen. The normalized coverage resulted in getting the same reliability estimates, regardless of the testing criterion.

The results reported here are from a single experiment and, being of a preliminary nature, do not allow us to claim the superiority of the reliability models designed according to this approach. They provide evidence, however, that this is a very promising research direction in software reliability.

The existence of other information related to software reliability should be investigated and other reliability models should be proposed and tested. Additional experiments should be devised and performed to confirm the obtained results. Models should be evaluated with failure data from software with diverse characteristics; other testing criteria should be considered. Integration test criteria should also be considered to investigate the relationship between their coverage and software reliability.

VI - References

[Adam80] Adams, E. N., "Minimizing Cost Impact of Software Defects", *IBM Research Division*, Report RC 8228(35669), 1980.

[Bast86] Bastani, F. B. and Ramamoorthy, C. V., "Input-domain-based Models for Estimating the Correctness of Process Control Programs", *A. Serra and R. E. Barlow (eds), Reliability Theory*, North Holland, Amsterdam, pp. 321-378, 1986.

[Bish90] Bishop, P. (ed.), "Prediction and Measurement of Software Reliability", *Dependability of Critical Computer Systems 3*, Elsevier Science Publisher, London, 1990.

[Butl91] Butler, R. W. and Finelli, G. B., "The Unfeasibility of Experimental Quantification of Life-critical Software Reliability", *Proceedings of the ACM SIGSOFT '91 Conference on Software for Critical Systems, ACM Press*, December 1991, pp. 66-76.

[Chaim91] Chaim, M. L., "Uma Ferramenta para Suporte ao Teste Estrutural de Programas Baseado em Análise de Fluxo de Dados", Dissertação *de Mestrado*, DCA/FEE/UNICAMP - Campinas, SP, Abril 1991.

[Chen94] Chen, M., "Tools and Techniques for Testing Based Software Reliability Estimation," *Ph. D. Thesis*, Purdue University, West Lafayette – Indiana, August 1994.

[Chen95] Chen, M.; Mathur, A. P. and Rego, V. J., "Effect of Testing Technique on Software Reliability Estimates Obtained Using A Time-Domain Model," *IEEE Transactions on Reliability*, vol. 44, no. 1, March 1995, pp. 97-103.

[Chen96] Chen, M.; Lyu, M. R. and Wong, W. Eric "An Empirical Study of the Correlation between Code Coverage and Reliability Estimation", *Proceedings of the Third International Software Metrics Symposium*, Berlin, Germany, March 25-26, 1996.

[Cresp96] Crespo, A. N.; Matrella P. and Pasquini, A., "Sensitivity of Reliability Growth Models to Operational Profile Errors", *Proceedings, The Seventh International Symposium on Software Reliability Engineering*, White Plains, New York, November 1996, pp. 35-44.

[Cresp97a] Crespo, A. N.; Pasquini, A.; Jino, M. e Madonado, J. C., "Cobertura dos Critérios Potenciais-usos e Confiabilidade do Software", Anais do XI Simpósio Brasileiro de Engenharia de Software - Fortaleza, outubro de 1997.

[Cresp97b] Crespo, A. N., "Modelos de Confiabilidade de Software Baseados em Cobertura de Critérios Estruturais de Teste", Tese de Doutado, DCA/FEEC/UNICAMP – Campinas, SP, dezembro de 1997.

[Crespo98] Crespo, A. N.; Pasquini, A.; Jino, M. e Madonado, J. C., "Code Coverage of the Potential-Uses Criteria and Software Reliability", Proceedings of the Fourth ISSAT International Conference, Seattle, Washington, USA, August, 1998.

[Frat95] Frate, F. D.; Garg, P.; Mathur, A. P. and Pasquini, A., "Experiments to Investigate the Correlatio Between Code Coverage and Software Reliability", *SERC-TR-162-P, Software Engineering Researc Center*, Purdue University, West Lafayette, Indiana 47907, April 1995.

[Garg94] Garg, P., "Investigating Coverage - Reliability Relationship and Sensitivity of Reliability to Errors in Operational Profile", *Technical Report - Department of Computer Sciences*, Purdue University, Wes Lafayette, IN 47907, May 1994.

[Goel79] Goel, A. L. and Okumoto, K., "A Time Dependent Error Detection Rate Model for Softwar Reliability and Other Performance Measures", *IEEE Transactions on Reliability*, Vol. 28, 1979, pp. 206-211.

[Haml92] Hamlet, D., "Are We testing for True Reliability", *IEEE Software*, vol. 9, no. 4, July 1992.

[Huds67] Hudson, A., "Program Errors as a Birth and Death Process", *Technical Report SP - 3011*, Santa Monica, Cal.: Systems Development Corporation, 1967.

[ISO9126] – ISO/IEC 9126. *Information technology – Software product evaluation – Quality characteristics and guidelines for their use*, ISO/IEC – 1991.

[Jeli72] Jelinski Z. and Moranda P. B., "Software Reliability Research", *Proceedings of the Statistical Methods for the Evaluation of Computer System Performance, Academic Press, 1972, pp. 465-484.*

[Li94] Li, N. and Malaiya, Y. K., "On Input Profile Selection For Software Testing", *Computer Science Department*, Colorado State University, Fort Collins, CO 80523, 1994.

[Litt93] Littlewood, B. and String, L., "Validation of Ultrahigh Dependability for Software-based Systems", *Communication of the ACM*, vol. 36, no. 1, Jan. 1993

[Lyu96] Lyu, M. R., *"Handbook of Software Reliability Engineering"*, McGraw-Hill, 1996.

[Mala94] Malaiya, Y. K.; Li, N.; Bieman, J.; Karcick, R. and Skibe, B., "The Relationship Between Test Coverage and Reliability", *Proceedings of the Fifth International Symposium on Software Reliability Engineering, Monterey*, CA, November 6-9, 1994, pp. 186-195.

[Mald92a] Maldonado, J. C.; Chaim, M. L.; e Jino, M., "Bridging the Gap in the Presence of Infeasible Paths: Potential Uses Testing Criteria"; *XII International Conference of the SCCC, Sociedad Chilena de Ciencia de la Computacion*, Santiago, Chile, outubro de 1992, pp. 323-340.

[Mald92b] Maldonado, J. C.; Chaim, M. L.; e Jino, M., "Using the essencial Branch Concept to Support Data-Flow Based Testing Criteria Application"; *Toulouse'92, Fifth International Conference on Software Engineering and Its Applications*; Toulouse, França, dezembro de 1992, pp. 613-623.

[Mood74] Mood, A.; Graybill, F. and Boes, D., *"Introduction to the Theory of Statistics"*, 3d ed., McGraw-Hill, Inc., New York, 1974.

[Musa87] Musa, J. D.; Ianino, A., and Okumoto, K., *Software Reliability - Measurement, Prediction, Application*, McGraw-Hill, New York, 1987.

[Nels78] Nelson, F., "Estimating Software Reliability From Test Data", *Microelectronics and Reliability*, vol. 17, no. 1, 1978, pp. 67-73.

[Pasq96]. Pasquini, A.; Crespo, A. N. and Matrella, P., "Sensitivity of Reliability Growth Models to Operational Profile Errors vs Testing Accuracy", *IEEE Transactions on Reliability*, December 1996, vol. 45, no. 4, pp. 531-540.

[Rams85] Ramsey, J. and Basili, V. R., "Analyzing the Test Process Using Structural Coverage," *Proceedings ICSE'85*, pp. 306-312, 1985.

[Rapp85] Rapps, S. and Weyuker, E. J., "Selecting Software Test Data Using Data Flow Information", *IEEE Transactions on Software Engineering*, April 1985, Vol. SE-11, No. 4, pp. 367-375.

[Shoo72] Shooman, M. L., "Probabilistic Models for Software Reliability Prediction," *Statistical Compute Performance Evaluation*, W. Freiberg, Ed. New York: Academic Press, 1972, pp. 485-502.

[Sieg77] Siegel, Sidney, *"Estatística Não Paramétrica"*, McGraw Hill, 1977.

[Vara95] Varadan, G. S., "Trends in Reliability and Test Strategies," *IEEE Software*, May 1995, pp. 10.

[Veev94] Veevers, A. and Marshall, A. "A Relationship Between Software Coverage Metrics and Reliability", *Software Testing, Verification and Reliability*, vol. 4, 1994, pp. 3-8.

[Xie93] Xie, M., "Software Reliability Models - A Selected Annotated Bibliography", *Software Testing Verification and Reliability*, vol. 3, 1993, pp. 3-28.

p