IN-61 CR 84121 P.15

DEPARTMENT OF MATHEMATICAL SCIENCES
COLLEGE OF SCIENCES
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23508

THE PREDICTIVE INFORMATION OBTAINED BY TESTING MULTIPLE SOFTWARE VERSIONS

By

Larry D. Lee, Principal Investigator

Final Report
For the period ending September 3, 1987

Prepared for the
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665

Under
Research Grant NAG-1-744
Dr. Dave E. Eckhardt, Jr., Technical Monitor
ISD-Systems Architecture Branch

Submitted by the
Old Dominion University Research Foundation
P. O. Box 6369
Norfolk, Virginia 23508

August 1987

THE PREDICTIVE INFORMATION OBTAINED BY TESTING MULTIPLE SOFTWARE VERSIONS

L. D. Lee*
Old Dominion University
Norfolk, Virginia 23508

Dave E. Eckhardt, Jr.
NASA Langley Research Center
Hampton, Virginia 23665

ABSTRACT

Multiversion programming is a redundancy approach to developing highly reliable software. In applications of this method, two or more versions of a program are developed independently by different programmers and the versions then combined to form a redundant system. One variation of this approach consists of developing a set of n program versions and testing the versions to predict the failure probability of a particular program or a system formed from a subset of the programs. In this paper we examine the precision that might be obtained, and also the effect or programmer variability if predictions are made over repetitions of the process of generating different program versions.

Key Words: N-version programming, multiversion software, binomial mixture sampling model, failure intensity, intensity distribution, estimation.

*L. D. Lee was supported by the National Aeronautics and Space Administration under Grant NAG-1-744.

1. INTRODUCTION

N-version or multiversion programming, originally proposed by Avizienis [1], is a redundancy method of structuring software components to cope with residual software design faults. Ideally, the use of multiple versions will greatly decrease the probability that a majority of the versions fail on the same input, thus providing a system having greater reliability than a single version. The N-version method involves independently generating $N \geq 2$ versions of a program and running them concurrently, all versions receiving the same inputs and producing their own outputs. The outputs of the programs are compared by a voter to determine, for each input, a majority decision output.

To model the effect of using multiple versions, several assumptions are required concerning the process by which programs are created and are run in an operational setting. The model we proposed in an earlier paper (Eckhardt and Lee, [2]) assumes that if program versions are generated by physically separated programmers or programming teams and according to a common set of requirements, they are, in some sense, independent and identically distributed objects. Littlewood and Miller [3] argue in favor of dropping the identically distributed assumption to model the effect of diverse methodologies (i.e., the use of different languages, development environments, etc.). In the absence of any systematic differences between versions (i.e., no forced diversity), the present paper is concerned with analyzing the predictive information obtained when testing multiple versions of a program.

2. DESCRIPTION

The purpose here is to describe modeling considerations for failure probability estimation. Predictive information may be measured as the precision with which one can estimate the failure probability of a given program or a given system of programs. It may also be measured as the precision obtained when estimating probabilities across a population of programs. Since it is unlikely that small probabilities can be estimated for a population of programmers with adequate precision, our concern is the precision obtained when estimating the failure probability of a particular program or system.

Uncertainity associated with the process of testing refers to whether a program fails on an input condition and whether the input condition ever occurs during testing. Failure is an event realized if a program produces incorrect output. However, a failure is recorded during testing only if there is a mechanism by which an error is detected; with multiple versions the testing process can be automated since different versions provide a basis for error detection. An important assumption is that the occurrence of identical incorrect ouput among the available programs has a very small probability, in comparison to other types of errors, so that errors are detected if they occur and the true failure probabilities are as low as indicated by the data obtained by testing.

Uncertainty concerning the failure of a program is conceptually inseparable from uncertainty about the effect of the programmer. Modeling the occurrence of failures is most easily motivated in terms of a sampling model

in which one imagines that programmers are picked at random. Such a model may not be the correct model, and this may have some effect when estimating small probabilities. Our purpose, rather than explore modeling possibilities, is to give a basis for the proposition that small probabilities may be estimated with adequate precision.

Since a model is a necessity in this context, we relate a sampling model under which failure data may be obtained to a theoretical model for the failure probability of a system having N component versions. When a larger set of n versions is run on a random input series $X_1, X_2, \ldots, X_K$, summary information is provided by counts $Y_1, Y_2, \ldots, Y_K$ of the numbers of versions which fail simultaneously (i.e., on the same input condition). This information may be summarized and conveyed through a cumulative distribution function (cdf). This cdf indicates the tendency for program versions to fail together as percentages of inputs on which various percentages of the versions fail.

Section 4 describes a sampling model and a theoretical model for a system of N versions. In section 5 we consider a statistic $\tilde{P}_N$ and describe a framework within which $\tilde{P}_N$ has the optimal property of minimum variance within a restricted class of statistics. In section 6 we obtain the variance of $\tilde{P}_N$. There and in section 7 we consider an interpretation of variability with the emphasis in section 7 being on the precision of estimates as obtained from the Knight and Leveson [4] failure data.

3. NOTATION

Ω	Input space for programs designed to a common set of requirements
$\theta(x)$	Probability a program fails on input x

$V(x)$	The binary random variable defined by $V(x) = 1$ if a program fails on input x and $V(x) = 0$, otherwise
Q	$Q(A)$ is the probability an input occurs in the set A
X_i	A randomly selected input condition
$G(z)$	A cdf giving the theoretical probability an input occurs in some subset of the input space for which $\theta(x) \leq z$
k,n,N	The number of input test cases, program versions, and component versions of a system, respectively
$Y_1, Y_2, \ldots, Y_k$	Counts of the numbers of versions that fail simultaneously on successive random inputs
$G_n(z)$	The cdf of $n^{-1} Y_i$
π_j	$P(Y_i = j)$, $j = 0,1,2,\ldots,n$
P_N	Failure probability of a system having N component versions
p	Failure probability of a single version
S_{jn}	Input frequency of j failures among n versions
S_n	$(S_{1n},S_{2n},\ldots,S_{nn})$
$\tilde{P}_N,\tilde{p}$	An unbiased statistic for estimating P_N,p, respectively
$k^{-1/2}\tau$	The standard deviation of $\tilde{P}_N$

4. THE SAMPLING MODEL AND RELATED PARAMETERS.

We first describe the assumed sampling model. Let Ω be the common input space of the software versions and let $\theta(x)$ be the probability a version fails on inputs x in Ω; $\theta(x)$ is called the failure intensity. Define a collection of binary random variables $V(x)$, $x\epsilon\Omega$, by $V(x) = 1$ if a version fails on input x and $V(x) = 0$, otherwise. For a set of n versions, similarly define $V_1(x)$, $V_2(x),\ldots, V_n(x)$, $x\epsilon\Omega$. The assumptions concerning the process of developing the programs and the input process are the following:

A1 $\{V_1(x), x\epsilon\Omega\}$, $\{V_2(x), x\epsilon\Omega\}$, ..., $\{V_n(x), x\epsilon\Omega\}$ are independent collections of random variables and for each $x\epsilon\Omega$, $V_1(x)$, $V_2(x)$,..., $V_n(x)$ are identically distributed.

A2 An input series $X_1, X_2, \ldots, X_k$ is stationary and independent; the probabilities $Q(A) = P(X_i \epsilon A)$ are given by a usage distribution Q.

A3 Failure counts $Y_i = \sum_{j=1}^{n} V_j(X_i)$, $i = 1,2,\ldots,k$ on successive random inputs are independent random variables.

A1 and A2 are the assumptions of the model we described in [2].

Although failures of the programs can be observed individually, it suffices for much of our discussion to consider the implications of A1-A3 for the series of failure counts $Y_1, Y_2, \ldots, Y_k$. From A1 and A2 each Y_i is conditionally, given $X_i = x$, binomial with parameter $(n,\theta(x))$. Unconditionally, $Y_1, Y_2, \ldots, Y_k$ are identically distributed and the distribution function of $n^{-1} Y_i$ is

$$G_n(z) = \int \sum_{j \leq nz} \binom{n}{j} u^j (1 - u)^{n-j} \, dG(u) \tag{1}$$

where

$$G(z) = \int_{\{x:\theta(x) \leq z\}} dQ \tag{2}$$

For fixed z, G(z) is the probability that random inputs occur in subsets of the input space for which $\theta(x) \leq z$. (The notation Gn(z) in (1) is rather weakly justified by the fact that G_n converges to G as n increases (Renyi, [5], p. 318)). By A3, Y_1, Y_2, ..., Y_k are then independent and identically distributed with the distribution function in (1), and we refer to this distribution as a binomial mixture sampling model.

Independence in A3 is a strong modeling assumption which can be checked if information is available concerning the failure counts Y_1, Y_2, ..., Y_k. Published failure data (Knight and Leveson, [4]) gives summary information only in the form of grouped frequency counts so we proceed as if A3 is a reasonable assumption.

To motivate the statistic considered in a later section, we now describe a theoretical model for a system of N versions. A system having N component versions (N = 1,3,5,...) fails if an input happens to fall in the subset of the input space where a majority m = (N + 1)/2 of the component versions produce incorrect output. The probability of system failure is

$$P_N = \int \sum_{j=m}^{N} \binom{N}{j} [\theta(x)]^j [1 - \theta(x)]^{N-j} \, dQ \qquad (3)$$

In the case N = 1, (3) is the failure probability p of a single version. Integrating (3) by substitution gives the reparmeterization

$$P_N = \int \sum_{j=m}^{N} \binom{N}{j} z^j (1 - z)^{N-j} \, dG(z) \qquad (4)$$

where the integrand does not depend on any unknown parameters.

Dependent failures of the component versions are modeled if $\theta(x)$ varies with different inputs x. If $\theta(x)$ is constant, i.e., $\theta(x) = p$ except on a set A with $Q(A) = 0$, then G is a degenerate distribution and (4) reduces to a model of independent failures. However, estimates of P_N obtained on the basis of the independence model differ substancially from estimates obtained without this restriction so (4) with a general form of G provides a more robust model.

5. A STATISTIC FOR ESTIMATING P_N.

For analyzing failure data obtained by testing a set of n versions, we consider the following statistic:

$$\tilde{P}_N = k^{-1} \binom{n}{N}^{-1} \sum_{j=0}^{n} \sum_{\ell=m}^{N} \binom{j}{\ell}\binom{n-j}{N-\ell} S_{jn} \qquad (5)$$

where $\binom{a}{b} = 0$ if $b > a$ and $S_{jn} = \sum_i I(Y_i = j)$ is the input frequency of j failures among n versions (I(E) is the indicator function of the set E). $\tilde{P}_N$ was derived (Eckhardt and Lee, [6]) by considering an average of the estimated failure probabilities of N version systems formed by selecting subsets of size N out of the total of n available versions.

The summary frequencies $S_{jn} = \sum_i (I(Y_i = j)$, $j = 0, 1, \ldots, n$ are obtained by grouping $Y_1, Y_2, \ldots, Y_k$ on the integers $j = 0, 1, \ldots, n$. The distribution function of $G_n(z)$ of $n^{-1} Y_i$ has mass

$$\pi_j = \int \binom{n}{j} u^j (1 - u)^{n-j} \, dG(u) \qquad (6)$$

at j/n, $j = 0, 1, \ldots, n$. Note that (6) defines a mapping $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$, or $\pi = \pi(G)$, from a family of distributions on the unit interval. If G is a member of the class of continuous distributions on [0,1], then the range of $\pi(G)$ is limited only by requiring that the probabilities π_j sum to one.

Let $J = \{i(1), i(2), \ldots, i(N)\}$ be a subset of the indexing set $\{1,2,\ldots,n\}$ for a set of n software versions. Define

$$\tilde{P}_N = k^{-1} \binom{n}{N}^{-1} \sum_i \sum_J \sum_{j=m}^{N} I\left(\sum_{\ell=1}^{N} V_{i(\ell)}(X_i) = j\right) \qquad (7)$$

which, by changing the order of summing, simplifies to (5). As a consequence of (7), $\tilde{P}_N$ is unbiased for P_N and is a U-statistic (Serfling, [7], p. 172) which has the desirable property of minimum variance among unbiased statistics that depend on $S_n = (S_{1n}, S_{2n}, \ldots, S_{nn})$. (A U-statistic is a statistic obtained from a function of the observable random variables having an expected value equal to the parameter to be estimated; averaging such a function over all subsets gives a U-statistic as in (7)).

Other unbiased statistics exist, however, which are not a function S_n. One example is the statistic defined by grouping the software versions into N sets and averaging over all selections, one version from each set.

The minimum variance property of $\tilde{P}_N$ is a consequence of S_n being a complete sufficient statistic for π in relation to $Y_1, Y_2, \ldots, Y_k$; S_n, however, is only a summary statistic in relation to the larger set

$\{V_j(X_i),\ j = 1,2,\ldots,n,\ i = 1,2,\ldots,k\}$. In staying with our purpose, the remainder of the discussion is limited to $\tilde{P}_N$.

6. THE VARIANCE OF $\tilde{P}_N$.

To express $\tilde{P}_N$ in a more convenient form, write

$$\tilde{P}_N = k^{-1} \sum_i \sum_{j=o}^{n} a_{nj}\, I(Y_i = j) \tag{8}$$

where

$$a_{nj} = \binom{n}{N}^{-1} \sum_{\ell=m}^{N} \binom{j}{\ell} \binom{n-j}{N-\ell}. \tag{9}$$

Except for the constant k^{-1}, (8) is the sum of the quantities

$$W_{ni} = \sum_{j=o}^{n} a_{nj}\, I(Y_i = j),\ i = 1,2,\ldots,k.$$

Since W_{ni} is a function only of Y_i, it follows from A3 that W_{ni}, $i = 1,2,\ldots,k$ are independent and identically distributed random variables. Therefore, for fixed n and k tending to infinity, $\tilde{P}_N$ has an asymptotic normal distribution with mean P_N and standard deviation $k^{-1/2}\tau$ where

$$\tau^2 = \sum_{j=o}^{n} [a_{nj}]^2\, \pi_j - \left(\sum_{j=o}^{n} a_{nj}\, \pi_j\right)^2 \tag{10}$$

$k^{-1/2}\,\tau$ measures the precision obtained when testing a given set of programs but it does not give a true reflection of variability over a population of programmers.

To clarify the interpretation of τ, consider the special case N = 1 of (8). If N = 1, then (8) reduces to

$$\tilde{p} = k^{-1} \sum_{i} \sum_{j=o}^{n} (j/n)\, I(Y_i = j) \qquad (11)$$

which estimates the average failure probability p of a single version. Suppose G is degenerate at the constant value of $\theta(x) = p$. In this case the parameters defined in (6) are binomial probabilities and (10) becomes $\tau^2 = p(1-p)/n$. This being the variance of a binomial random variable scaled by n^{-1}, the quantity τ seems appropriately described as a measure of the effect of variability over repetitions of the process of generating different programs.

7. ESTIMATES OF PRECISION.

The summary data in Table 1 was obtained (Knight and Leveson [4]) by testing n=27 programs on $k=10^6$ randomly selected input conditions. There were 2 input cases on which 8 of the versions failed together, 12 cases where 7 versions failed together, and so on.

Estimates of P_N, N = 1,3 and of the standard deviation, $K^{-1/2}\tau$ and τ, are given in Table 2. On average, for the given set of programs, a system of 3 versions has a much smaller (by a factor of 19) failure probability than a single version. When considering only the uncertainty associated with

testing the 27 versions, the standard deviation of these estimates, as given in the second column of Table 2, indicates that these probabilities may be estimated with high precision. The quantities in the third column suggest considerable variation in the estimates if the experiment were repeated for a different set of programmers.

8. CONCLUSIONS.

The primary motivation for this paper is to give a basis for the contention that, with multiple program versions, small failure probabilities might be estimated with a reasonable degree of precision. The calculations of the previous section suggest high precision. However, we emphasize that the precision obtained refers to predicting an average failure probability when testing a given set of programs (i.e., the precision does not apply to making predictions across a population of programmers) and that our modeling assumptions may have considerable effect on estimates of precision. Because of our indifference as to the choice of one program or set of programs, a statistic was used which estimates an average failure probability over the given set of programs.

REFERENCES

[1] A. Avizienis, "Fault Tolerance and Fault Intolerance: Complementary Approaches to Reliable Computing," in Proc. 1975 Int. Conf. Reliable Software, pp. 458-464.

[2] D. E. Eckhardt, Jr. and L. D. Lee, "A Theoretical Basis for the Analysis of Multiversion Software Subject to Coincident Errors," IEEE Trans. Software Eng., vol. SE-11, No. 12, pp. 1511-1517, 1985.

[3] B. Littlewood and D. R. Miller, "A Conceptual Model of Multi-version Software," in FTCS-17, Pittsburgh, July 1987.

[4] J. C. Knight and N. G. Leveson, "An Experimental Evaluation of the Assumption of Independence in Multiversions Programming," IEEE Trans. Software Eng., vol. SE-12, No. 1, pp. 96-109, 1986.

[5] A. Renyi, Foundations of Probability, Holden-Day, Inc., San Francisco, 1970.

[6] D. E. Eckhardt, Jr. and L. D. Lee, "An Analysis of the Effects of Coincident Errors on Multi-Version Software," in Proc. AIAA/ACM/NASA/IEEE Computers in Aerospace V Conference, pp. 370-373, 1985.

[7] R. J. Serfling, Approximation Theorems of Mathematical Statistics, John Wiley & Sons, Inc., New York, 1980.

Table 1. Failure proportions for 27 programs on 10^6 random input cases.

Number of failed versions	estimates of π_j
0	0.983607
1	0.015138
2	0.000551
3	0.000343
4	0.000242
5	0.000073
6	0.000032
7	0.000012
8	0.000002

Source: Knight and Leveson [4]

Table 2. Estimates of P_N, $k^{-1/2}\tau$, and τ.

N	P_N	$k^{-1/2}\tau$	τ
1	0.00069978	0.00000616	0.00616
3	0.00003669	0.00000144	0.00144