Journal of the KSQC
Vol.21, No.2, Dec 1993

1993년 12월 韓國品質管理學會誌 제21권 제2호

The Designs for Prediction of Future Reliability Using the Stochastic Reliability.

Chung-Hwan, On*
Bok-Mahn, Kim**

ABSTRACT

The newly proposed model of the future reliability results in earlier fault-fixes having a greater effect than the fault which make the greatest contribution to the overall failure rate tend to show themselves earlier, and so are fixed earlier. The suggested model allows a variety of reliability measures to be calculated.

Predictions of total execution time(debugging time) is to achieve a target reliability. This model could also apply to computer-hardware reliability growth resulting from the elimination of design error and fault.

1. Introduction

First of all, consider the case of software reliability. When a program fails, it does so because it contains faults(errors, bugs). The removal of a fault will ensure that the class of

* The College of Suwon
** University of Ulsan

failures caused by the removed fault does not recur.

In the unlikly event that all faults were to be removed, the program would perfect and execute failure free forever. Contrast this with classical hardware reliability theory, where failures of the overall system are caused by component failures; here we can be sure that a system will always fail. In practice, clearly the failure process depends on the number of faults remaining, and reliability growth will be related to fault removal.

In early work [7, 8, 9], it was assumed that each fault contributed the same amount to the overall failure rate, so that debugging resulted in the failure rate's improvement in a series of equally sized steps. If the program starts with W faults, the execution time between the $(j-1)$th and j th failures, T_j, has an exponential distribution

$$f(t_j) = \lambda_j e^{-\lambda_j t_j} \tag{1}$$

with the failure rate

$$\lambda_j = (W - j + 1)U \tag{2}$$

Since at this state $(j-1)$ faults have been eliminated. Assuming (1) and (2) specify the model completely, and the unknown parameters W and U can be estimated by maximum likelihood estimation (MLE).

Even the basic model arising from (1) and (2) can occasionally give rise to difficulties. Sukert [6] seems to have been the first to report these, and they have been studied by Forman & Singpurwalla [6]. There seems to be evidence that the MLE estimate W may not be S-consistent.

The main criticism of these models, though, is that the assumption which results in equation (2) is unrealistic. The effect of debugging itself is treated as a deterministic process. It is more accurate to view the debugging process as the creation of a sequence of programs, $\{S_i\}$ than as series of operations carried out upon a single program. Even though programs S_i and S_{i+1} usually differ only slightly, their properties should be so different.

The fault-removal operation which changes program S_i into program S_{i+1} usually contains uncertainties about the relationship between λ_i (failure program) and λ_{i+1}, since the occurrence frequency of the fault removed at this stage will not be known and cannot be predicted. In fact, it is not usually possible to measure the occurrence rate of a fault even after it has been removed, although rough estimates can be made from the operational profile of the program.

Assuming a program begins life with W faults which have occurrence rates that vary according to some distribution and when a failure occurs the corresponding fault is removed

with certainty. A program might have less than half the failure rate after the removal of half the faults. Late in the debugging period, the program may achieve very high reliability which still containing many faults. These consequences of stochastic debugging seem to be in closer accord with experience than the implication of equation.

Hardware reliability theories have tended to concentrate behavior [1, 6]. Such component failures are essentially repetitious. If a component fails it is replaced by a similar, working component which will itself eventually fail. The contrasts with the software situation where in if a program fault is fixed it stays fixed and will never again precipitate a failure.

Improvements in component reliability, particularly in microelectronics, have resulted in a shift of interest towards other sources of failure:

What are loosely termed design-errors. The most common type is the genuine design error which can be eliminated by system redesign. However, other categories, such system as faulty maintenence procedures or incorrect specifications resulting from faulty understanding of the informal system requirements, can usefully be included. They all have in common the property that, in principle, they can be removed and the system will there by exhibit a permanent improvement in reliability.

2. Notation

f : probability density function
F : c. d. f
P : failure rate of program ($P = V_1 + V_2 + \cdots + V_{W-I}$)
λ : realization of random variable P
V_k : random variable contribution to P to from fault K
U_k : realization of V
T : time to next failure random variable
T_i : time between $(i-1)$ th and i th failures
t_i : realization of random variable T_i
W : initial total number of faults in program
η : total elapsed execution time
C : normalizing constant
$B(x)$: beta function of x
$B(y)$: beta function of y

Other, standard notation is given in Reader and Authors.

3. Current Reliability Modelling of the Program

First of all we obtain the probrability density function of T of (1), (2)

$$f(t) = \int_0^\infty f(t \mid P = \lambda) f(\lambda) d\lambda$$

$$= \int_0^\infty \lambda e^{-\lambda t} f(\lambda) d\lambda \tag{3}$$

$$= [(W - i)\alpha(\beta + \eta)^{(W-i)\alpha}] / [(\beta + \eta + t)^{(W-i)\alpha + 1}] \tag{4}$$

which is $(\beta + \eta)^{-1}$, pareto distribution $[t / (\beta + \eta); (N - i)^{\alpha}]$. It is this result which undelies the rest of the paper. From (4) we may obtain all the usual scalar measures used in reliability. Thus the reliability and failure rate functions are:

$$R(t) = [(\beta + \eta) / (\beta + \eta + t)]^{(W-i)\alpha} \tag{5}$$

$$\lambda(t) = (W-i)\alpha / (\beta + \eta + t) \tag{6}$$

The mean time to failure(MTTF) at the is the s-expected value of T :

$$E[T] = (\beta + \eta) / [(W-i)\alpha - 1] \tag{7}$$

from equation (4), This exists as long as

$$(W-i)\alpha > 1, \alpha \geq 1.$$

Consider the failure rate $(t = 0)$

$$\lambda(0) = (W-i)\alpha / (\beta + \eta) \tag{8}$$

$$= \lim_{h \to 0} \{ P_r \{ \text{failure in } (\eta, \eta + h) \mid i \text{ failures in } (0, \eta) \} / h \}.$$

We shall call this the conditional failure rate since it is defined in terms of a conditional probability from(6). This is the failure rate of a a program which has failure i times(and thus had i faults eliminated) in a total execution time η. And in the equation(6), this is decreasing in t. This decreasing failure rate(DFR) property is independent of the distribution chosen for V; it is a general consequence of the mixing operation(3). The choice of a Γ distribution to represent our uncertainty about the occurrence rates of the faults is dictated by mathematical tractability.

When a failure occurs and a fault is fixed, the failure rate duops by an amount $\alpha / (\beta + \eta)$

early fault fixes, at small η, cause greater reduction in the programs conditional failure rate than later ones. The model thus produces a plot of conditional failure rate vs time. Assuming exponentially distributed time to failure for each fault, we could describe the current reliability of the program completely if the "U" were known. The U, however, are not known; there will not even be any failure data available to estimate the U for remaining faults.

Hardware reliability will be noted the difference between the modelling of failure rate in this model and the Duane types of models seem to be variants of the nonhomogeneous Poisson process with continuously decreasing failure rate functions.

The advantage of the current model over this earlier work is that, in addition to the decreasing failure rate property between failures, the discrete improvements in the failure rate resulting from fault removal are modelled directly.

4. The Derivation for Predictable Model of Future Reliability

Models of this type are used for two purposes. At each stage of the debugging process it is important to be able to measure the current reliability of the program.

Predictions of future reliability can be approached in two ways. On the one hand we might want to knows how reliable the program will be after some specified execution time has elapsed. This approach might be more useful when execution time and calendar time are almost identical. Alternatively, we might be more interested in the reliability after some specified number of faults have been removed. This approach also would be more appropriate if execution time were small compared to repair times.

Now, let's take in how reliable the program will have become after a further debugging time η' has elapsed. Let random variable T represent time until next failure. Since there is a non-zero probability of all $W-i$ remaining faults being removed during $(\eta, \eta + \eta')$, it follows that there is a non-zero probability that T is infinite.

$$F(t) = \sum_{K=0}^{W-i} P_r\{T \leq t \mid K = k\} \cdot P_r\{K = k\} \tag{9}$$

where K represents the number of failures in $(\eta, \eta + \eta')$. η is i bugs fixed in $(0, \eta)$, η' is between η and T, B is the point just passed the end of η' and also just before starting T.

$$= \sum_{K=0}^{W-i} \{1-[(\beta+\eta+\eta')/(\beta+\eta+\eta'+t)]^{(W-i-K)\alpha} \cdot P_r(K=k)$$

$$= \text{Pareto p. d. f}\,[t/(\beta+\eta+\eta');(W-i-K)\alpha] \cdot P_r\{K=k\}$$

$$= \sum_{K=0}^{W-i} \{1 - [(\beta+\eta+\eta')/(\beta+\eta+\eta'+t)]^{(W-i-K)\alpha} \cdot \binom{W-i}{K} \cdot$$

$$[1 - \{(\beta+\eta)/(\beta+\eta+\eta')\}^{\alpha}]^{K} [(\beta+\eta)/(\beta+\eta+\eta')]^{\alpha(W-i-K)} \quad (10)$$

So,

$$1 - F(t) = R(t) = \{1 - [(\beta+\eta)/(\beta+\eta+\eta')]^{\alpha} + [(\beta+\eta)/(\beta+\eta+\eta')]^{\alpha(W-i-K)} \quad (11)$$

And let's consider the number of failures in a given time. The distribution of K, the number of failures in $(\eta, \eta+\eta')$ when i failures have occurred in $(0, \eta)$, has been given in

$$P_r\{K = k\} = \binom{W-i}{K} \cdot q^{K}(1-q)^{W-i-K} \quad (12)$$

where $q \equiv 1 - [(\beta+\eta)/(\beta+\eta+\eta')]^{\alpha}$.

This result would be of value in those situations were debugging continues in the use environment of the program. Also, considering the CDF of the total time from now until K failures have occurred. It can be shown that

$$P_r\{T_{i+1} + T_{i+2} + \cdots + T_{i+k} < t\}$$

$$= P_r\{\text{Beta}(W-i-k+1, k) > [(\beta+\eta)/(\beta+\eta+\eta')]^{\alpha}\} \quad (13)$$

4-1. Reliability of the Progrom after K More Failures

Total elapsed execution time is η, and in this time i failures have been observed, and so i faults fixed. We approach to the reliability of the program after k more failures have occurred.

$$f(t) = \int \cdots \int f(t \mid T_{i+1} = t_{i+1}, \cdots, T_{i+k} = t_{i+k}) \cdot \prod_{j=2}^{k} (\beta+\eta^*)^{-1} \cdot \text{pareto}$$

$$d[t_{i+j}/(\beta+\eta^*); (W-i-j+1)\alpha] \quad (14)$$

where $\eta^* = \eta + \sum_{m=i+1}^{i+j-1} t_m$

from equation (4), unfortunately, it seems that (14) may not be obtained analytically.

Reliability measures can, however, be derived from this distribution: notably the failure rate and mean time to failure.

Let's consider first the failure rate at the point of $(i+k)$ th failure (between T_{i+k} and T): denote this by

$$P = [(W - i - k)\alpha] / [\beta + \eta + \sum_{m=i+1}^{i+k} T_m] \tag{15}$$

In view of the intractable nature of (14), that we can find the exact distribution of P. In fact, if we write

$$\begin{aligned} Z &= \text{Beta}(W - i - k + 1, k) \\ &= \{[\{(W - i - k)\alpha\}(\beta + \eta)] / [\{(W - i - k)\alpha\}(\beta + \eta + \sum_{m=i+1}^{i+k} T_m]\}^{\alpha}, \\ &\quad 0 < Z < 1 \end{aligned} \tag{16}$$

Now, expectation of P follows that

$$\begin{aligned} E(P) = [(W - i - k)/(\beta + \eta)][(W - i - k + 1)/(W - i - k + 1)\alpha] \cdot \\ \cdots [(W - i)/(W - i + 1)\alpha + 1] < \lambda^* \end{aligned} \tag{17}$$

where $\lambda^* = (W - i)\alpha(\beta + \eta)^{\alpha} / (\beta + \eta + \eta')^{\alpha+1}$ and also

$$\eta^* = [(W - i)\alpha(\beta + \eta)^{\alpha} / \lambda^*]^{1/(\alpha+1)} - (\beta + \eta)$$

4-2. Expectation of the Debugging Time (T)

Denote the debugging time until removed of the last fault by T^*. Then

$$T^* = \max\{\text{pard}(X_j)\} \tag{18}$$

The cdf of T^* is

$$\begin{aligned} \text{cdf}(T^*) &= P_r\{X_1 \le t, \cdots, X_{W-i} \le t\} = [P_r\{X_j \le t\}]^{W-i} \\ &= \{1 - [(\beta + \eta)/(\beta + \eta + t)]^{\alpha}\}^{N-i} \end{aligned} \tag{19}$$

Since i faults have already been eliminated in $(0, \eta)$, there remain $(W-i)$ in the program. Then T^* is the time we should wait until the last fault is eliminated. We can find

that the equation(19) would be easily used to calculate the debugging time t(execution time) with a given probability that all faults will have been eliminated in given t.

Meanwhile we can consider the mean time to reach the faults-free state by denoting $E(T)$

$$E(T) = [\{ \text{Beta}(1 - 1/\alpha, W - i) / \text{Beta}(1, W - i) \} - 1] \quad (20)$$

5. Conclusion

The prediction model described here is a result of computer programs and hardware reliability design, involved in eliminating design errors in hardware. This recognition justifies the use of similar mathematical models of reliability growth in the two situations. The relationship between software and hardware reliability theories has always been contentious. Early work on software reliability modelling[8] tended to emphasize the lessons to be drawn from the earlier comprehensive hardware theory. More recently an awareness has grown that simple parallels might be misleading[4]. Most employees agree that measures of reliability which are compatible between hardware and software must be used, so that reliability studies of complex hardware and software systems become feasible.

Models for the removal of software faults might appropriately represent the depletion of hardware design errors. The models like the one presented here, and the earlier fault-counting models, only treat the failure behavior of the program or hardware design. Some failure will have more serious consequences than others and we should be modelling this cost process. This observation applies with equal force to classical reliability studies, but it has particular force when we study design errors. [The classical theory would then be used to model the nonremoval class, and the new reliability growth models, such as the one described here used for removal faults.] We need similar techniques in the software and design reliability field for the future reliability, but the paucity of data has so far prevented headway in modelling life cycle costs. Our ability to protect ourselves from the effect of failures in critical parts of a design is extremely limited. This contrasts with situation in conventional reliability studies, where it is usually possible to have redundancy among critical components.

REFERENCES

1. R. E. Barlow, F. Proshan, (1975) "*Statistical Theory of Reliability and Life Testing: Probability Models*," New York: Holt, Rinehart and Winston.
2. E. H. Forman, N. D. Singpurwalla, (1977) "*An Empirical Stopping Rule for Debugging and Testing Computer Software*," J. Amer, Statistic Association, Vol. 72, Dec., pp. 750-757.
3. B. Littlewood, J. L. Verral, (1981) "*On the Likelihood Function of a Debugging Model for Computer Software Reliability*," IEEE Trans Reliability, Vol. R-30, Jun., pp. 145-148.
4. B. Littlewood, (1979) "*How to Measure Software Reliability and How not to*," IEEE Trans. Reliability, Vol. R-28, Jun., pp. 103-110.
5. B. Littlewood, (1980) "*Theories of Software Reliability: How good are they and how can they be improved ?*," IEEE Trans. Software Engineering, Vol. SE-6, Sep., pp. 489-500.
6. N. R. Mann, R. E. Schafer, N. D. Singpurwalla, (1974) "*Methods for Statistical Analysis of Reliability and Life Data*," New york: John Wiley & Sons.
7. J. D. Musa, (1975) "*A Theory of Software Reliability and its Application*," IEEE Trans. Software Engineering, Vol. SE-1, Sep., pp. 312-327.
8. M. Shooman, (1972) "*Probabilistic Models for Software Reliability and Prediction*," In Statistical Computer Performance Evaluation (W. Freiberger, ed.), New York: Academic Press, pp. 485-502.
9. Z. Jelinski, P. B. Moranda, (1972) "*Software Reliability Research*," In Statistical Computer Performance Evaluation (W. Freiberger, ed.), New York: Academic Press, pp. 465-484.