Modelling the Failure Rate Function in Coverage and Software Reliability Growth

Joong-Yang Park and Young Soon Kim

Department of Statistics and Information, Gyeongsang National University, Jinju, KOREA,
<joongyang@hotmail.com>

Jae Heung Park

Department of Computer Science, Gyeongsang National University, Jinju, KOREA,
<pjh@nongae.gsnu.ac.kr>

Abstract

There is a new trend of incorporating software coverage metrics into software reliability modelling. This paper proposes a coverage-based software reliability growth model. Firstly, the failure rate function in coverage is analytically derived. Then it is shown that the number of detected faults follows a Nonhomogeneous Poisson distribution of which intensity function is the failure rate function in coverage. Practical applicability of the proposed models is examined by illustrative numerical examples.

Key Words : coverage metric, failure rate, software reliability, software reliability growth model

1. Introduction

In recent years there is growing use of computer systems. Software is an integral part of most critical computer systems. Since failures of a software system can cause severe consequences in terms of human life, environment impact, or economical losses, an important quality attribute of a software system is the degree to which it can be relied upon to perform its intended function. One of quantifiable measures for software quality attribute is software reliability, which is formally defined as the probability that no failure occurs in a specified environment during a specified exposure period.

Software reliability has attracted considerable attention for the last 2 decades. Many software reliability models have been developed to estimate software reliability measures such as the initial fault content, the time to next failure, the number of remaining faults and

the software reliability function. Software reliability growth models (SRGMs) are software reliability models concerned with the relationship between the cumulative number of faults detected during software testing or the time interval between software failures and the time span of the software testing. Generally the time is measured by the execution time or the number of test cases executed. Refer to Goel (1985), Gokhale, Marinos and Trivedi (1996), Horgan et al. (1995), Lyu (1996), Malaiya and Srimani (1990), Musa et al. (1987), Ramamoorthy and Bastani (1982), Shantikumar (1983) and Wood (1997) for general discussion on software reliability models.

As mentioned in Varadan (1995) and Wood (1997), a new approach for modeling software reliability is to utilize test coverage measures. This approach is based on the following observations:

1) The more a software system is covered, the more likely reliable is the software system;
2) Reliability modelling can be improved by taking into account the structural coverage, even though structural testing may or may not be used.

The recent research results show a strong relation between coverage and software reliability. Chen, Mathur and Rego (1995) have investigated the correlation between coverage and software reliability using experimentation with randomly generated flow graphs. They present an empirical result that there is a need for making use of coverage measures in reliability estimation. They also show that in the absence of coverage measures one is likely to overestimate reliability. Frate et al. (1995) perform a similar investigation in empirical setting. Vouk (1992) and Dalal, Horgan and Kettenring (1993) investigated the relation between coverage and fault detection rate. Dalal, Horgan and Kettenring (1993) give a scatter plot of the number of faults detected during system testing versus the block coverage achieved during unit testing for 28 software systems. The plot shows that modules covered more thoroughly during unit testing are much less likely to contain faults. These studies indicate that there exists a strong relationship between reliability and coverage and provide a basis for the coverage-based SRGMs.

Several coverage-based SRGMs have been developed by Chen et al. (1992), Chen, Lyu and Wong (1996, 1997), Gokhale et al. (1996a, 1996b), Grottke (1999, 2000), Malaiya et al. (1994, 1996), Piwowarski et al. (1993), and Veevers and Marshall (1994). Similarly to the traditional time domain SRGMs, most coverage-based SRGMs are usually characterized by their mean value functions in coverage which relate the expected number of detected faults to coverage. The coverage-based SRGM appeared in Veevers (1990) and Veevers and

Marshall (1994) is the only coverage-based SRGM directly relating reliability to coverage. In this paper we propose a coverage-based NHPP SRGM. The failure rate function in coverage, the relationship between failure rate and coverage, is first derived in Section 2. Then Section 3 presents a NHPP SRGM of which mean value function results from the failure rate function in coverage. The proposed failure rate function in coverage and NHPP SRGM are empirically examined in Section 4. Finally concluding remarks are given in Section 5.

2. Derivation of the Failure Rate Function in Coverage

We represent the software under testing by a set Ω, whose elements are all constructs of the software. Suppose that C is a coverage metric defined on Ω such that $C(\phi)=0$, $C(\Omega)=1$ and $C(w_i)<C(w_j)$, where w_i and w_j are subsets of Ω and $w_i \subset w_j$. Constructs may be statements, blocks, branches, p-uses or c-uses depending on the coverage metric employed. Let c be the value of C at the current stage of testing and U be the set of uncovered constructs. Then $c=|\Omega-U|/|\Omega|$, where $|\cdot|$ is the cardinality of a set. Since faults are more likely resident in U, the failure rate of the software is assumed to be a function of the coverage. The failure rate of the software is denoted by $\lambda(c)$. In order to model the behavior of $\lambda(c)$, we may assume that when the current coverage is c, the total potential contribution of U to the failure rate is $\lambda(c)$. This assumption implies that failure rate attains its theoretical minimum value 0 when the coverage reaches 1. However, it is generally accepted that 100% coverage is rarely achieved in practice partly because there exist infeasible constructs and partly because the given testing environment is not capable of covering all the feasible constructs. We refer to as coverable constructs the feasible constructs that can be covered by the given testing environment. We partition U into two sets, UC and UU, which denote the set of uncovered coverable constructs and the set of uncovered uncoverable constructs, respectively. The maximum achievable coverage is then computed as $|\Omega-UU|/|\Omega|$ and denoted by c_{max}. It should be also noted that faults may be found in the covered constructs. That is, finding more faults does not necessarily mean that coverage increases. This and the subsume relationship among coverage metrics indicate that even 100% coverage does not guarantee zero failure rate. Therefore it is reasonable to assume that the minimum achievable failure rate $\lambda_{min}=\lambda(c_{max})$ may be greater than zero and the total potential contribution of UC to the failure rate is $\lambda(c)=\lambda_{min}$.

If further testing covers a portion of U, there will be a coverage increment dc. The coverage increment will result in a failure rate decrement $d\lambda(c)$. We further assume that the failure rate

decrement $d\lambda(c)$ caused by the coverage increment dc is proportional to $(\lambda(c)-\lambda_{min})/|UC|$. That is, the contribution of each construct in UC to the failure rate is homogeneous. Since

$$\frac{\lambda(c)-\lambda_{min}}{|UC|}=\frac{\lambda(c)-\lambda_{min}}{|UC|/|\Omega|}\frac{1}{|\Omega|}=\frac{\lambda(c)-\lambda_{min}}{c_{max}-c}\frac{1}{|\Omega|}, \tag{1}$$

we obtain the following differential equation:

$$d\lambda(c)=-p\frac{\lambda(c)-\lambda_{min}}{c_{max}-c}dc, \tag{2}$$

where $0<p$ is the proportional factor.

There is another important aspect of testing which should be taken into account. The coverage increment due to the execution of the first test case is usually considerably large. For example, the block coverage increment resulting from the execution of the first test case is 0.34 in Mathur data given in Table 3. Therefore coverage values between 0 and c_{min} are not realized, where c_{min} is the coverage achieved by the first test case. Since the failure rate does not change for $0\le c\le c_{min}$, we assume that the differential equation (2) holds only for $c_{min}\le c\le c_{max}$. Parameter c_{min} can be interpreted as the minimum achievable coverage for the given testing environment. Similar assumptions on c_{min} and c_{max} were also adopted by Vouk (1992).

Solving the above differential equation with the initial condition $\lambda(c_{min})=\lambda_{max}$, the failure rate function is obtained as

$$\lambda(c)=\lambda_{min}+(\lambda_{max}-\lambda_{min})\left(1-\frac{c-c_{min}}{c_{max}-c_{min}}\right)^{p}, \tag{3}$$

where $c_{min}\le c\le c_{max}\le 1$, $0\le\lambda_{min}\le\lambda_{max}$ and $0<p$.

3. New Coverage-Based NHPP SRGM

Coverage-based NHPP SRGMs have appeared in Gokhale et al. (1996b), Malaiya et al. (1994, 1996), Piwowarski et al. (1993), and Vouk (1992). These coverage-based NHPP SRGMs are characterized by the mean value function $m(c)$. The NHPP SRGM proposed by

Piwowarski et al. (1993) and Gokhale et al. (1996b) has $m(c)=\alpha c$ for $0\leq c\leq 1$, whereas the mean value function in Malaiya et al. (1994, 1996) is $m(c)=\alpha \ln(1+\beta(\exp(\gamma c-1)$ for $0\leq c\leq 1$. Vouk (1992) derived the mean value function $m(c)=\alpha(1-\exp(-\beta(c-c_{min})^2))$ for $c_{min}\leq c\leq c_{max}$. (Parameters α and β have different meanings in each model.) We now present a coverage-based NHPP SRGM, of which mean value function is derived from the failure rate function given by Equation (3).

Suppose that behavior of the failure rate of a software system due to fault detection and removal phenomenon occurring during testing is described by Equation (3). Then the software system under testing can be regarded as a hardware system which is minimally repaired. The concept of minimal repair was first introduced by Barlow and Hunter (1960). Nakagawa and Kowada (1983) and Murthy (1991) showed that the number of failures of such a hardware system is distributed according to a Nonhomogeneous Poisson distribution with an intensity function equal to the failure rate function. Since fault detection in software testing is equivalent to failure of a hardware system, the number of detected faults follows a Poisson distribution with the intensity function $\lambda(c)$ given by Equation (3). The corresponding mean value function is thus obtained as

$$\begin{aligned} m(c) &= \int_{c_{min}}^{c} \lambda(\tau)d\tau \\ &= \lambda_{min}(c-c_{min})+\frac{1}{p+1}(c_{max}-c_{min})(\lambda_{max}-\lambda_{min})\left(1-\left(1-\frac{c-c_{min}}{c_{max}-c_{min}}\right)^{p+1}\right) \end{aligned} \tag{4}$$

for $c_{min}\leq c\leq c_{max}$. The total number of detectable faults for the given testing environment is computed as

$$m(c_{max})=\frac{(c_{max}-c_{min})(p\lambda_{min}+\lambda_{max})}{p+1} \tag{5}$$

4. Numerical Examples

This section performs an empirical evaluation of the models derived in the previous two section. Two data sets are used for the empirical evaluation. First data set is the one that appeared in Veevers (1990). It includes failure rates of 13 faults and values of three coverage metrics at the instances of fault detection. The three coverage metrics are

respectively statement coverage, block coverage and LCSAJ (linear code sequence and jump) defined in Woodward, Hedley and Hennell (1980). Failure rates of the software system are computed by summing failure rates of the remaining faults at the instances of fault detection. Values of block coverage metric and number of detected faults are reproduced in Table 1. The least squares estimates for $\lambda(c)$ and $m(c)$ are tabulated in Table 2 and the fitted $\lambda(c)$ and $m(c)$ curves are depicted in Figures 1-2. (There are considerable large discrepancy between the estimates of λ_{max} and λ_{min} for $\lambda(c)$ and those for $m(c)$. This is because the failure rates in Table 1 are measured in different unit, not in coverage.) The total number of detectable faults is estimated as 13.0850, which is very close to 13. It seems that the proposed models well describe the behavior of $\lambda(c)$ and $m(c)$. Similar results are obtained for other two coverage metrics.

Second data set was collected from a simulation experiment conducted by Dr. Mathur in Purdue University. The experiment tests a software system with 22 injected faults. 109 test cases were applied and 17 of the 22 injected faults were discovered. Table 3 shows values of the number of detected faults and four coverage metrics. The least squares estimates for $m(c)$ and $m(c_{max})$ are obtained as in Table 4 and the fitted $m(c)$ curve for block coverage is plotted in Figure 3. As for the first data set, the proposed NHPP SRGM works well for the second data set.

Table 1. Values of failure rate, number of detected faults and block coverage: Veevers data.

Failure rate($\times 10^{-4}$)	*Number of detected faults*	c
10.6186	1	0.18
5.5723	4	0.31
3.3352	5	0.32
2.9172	6	0.33
1.7685	7	0.34
1.3463	8	0.36
0.8923	9	0.38
0.5694	10	0.44
0.3048	11	0.59
0.1637	12	0.60
0.0538	13	0.82

Table 2. Least squares estimates for Veevers data.

Parameter	Estimate	
	$\lambda(c)$	m(c)
λ_{min}	0.0000	0.0000
λ_{max}	0.0040	48.9496
c_{min}	0.0020	0.1712
c_{max}	0.9962	0.9546
p	6.5667	1.9307
$m(c_{max})$	-	13.0850

- : not available

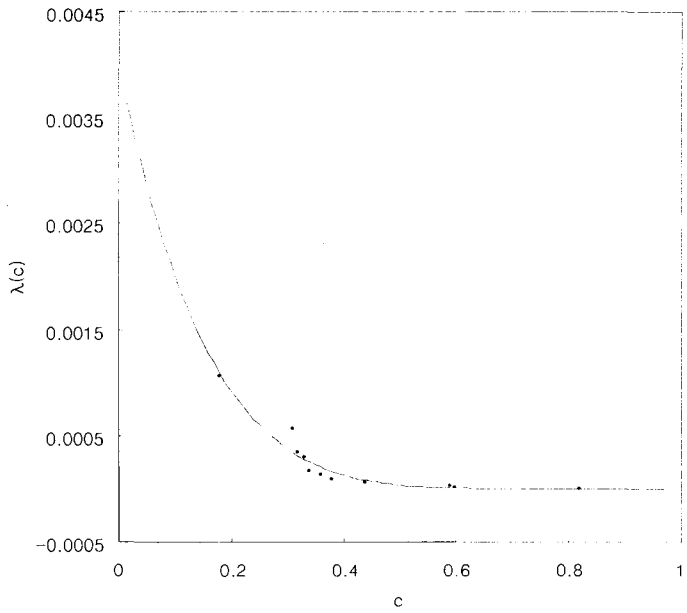

Figure 1. Fitted $\lambda(c)$ curve for Veevers data.

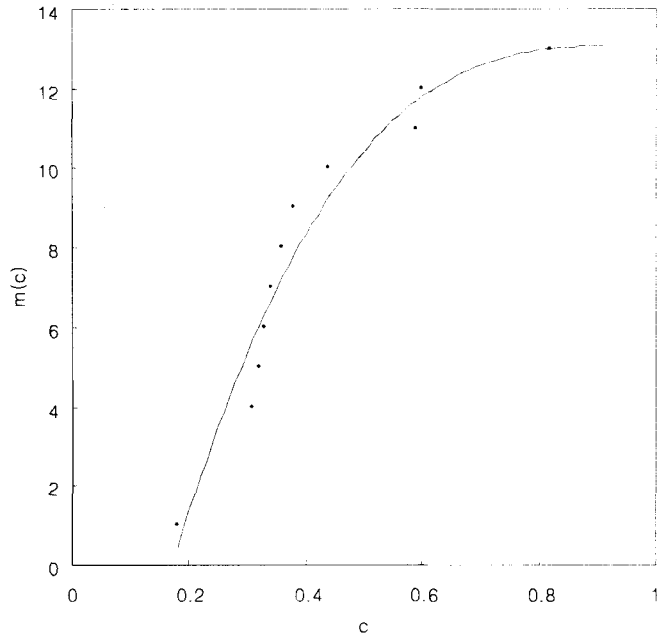

Figure 2. Fitted $m(c)$ curve for Veevers data.

Table 3. Values of number of test cases, number of detected faults and four coverage metrics: Mathur data.

Number of Test Case	Number of Detected Faults	Coverage metric			
		Block	Decision	c-use	p-use
1	1	0.34	0.20	0.26	0.23
2	2	0.42	0.28	0.34	0.30
3	6	0.48	0.33	0.40	0.34
12	7	0.59	0.44	0.51	0.43
14	9	0.60	0.45	0.52	0.44
16	11	0.60	0.46	0.52	0.45
24	12	0.70	0.53	0.62	0.50
67	14	0.73	0.60	0.66	0.56
100	15	0.77	0.65	0.72	0.63
101	16	0.77	0.65	0.72	0.63
109	17	0.78	0.66	0.72	0.64

Table 4. Least squares estimates for Mathur data.

Parameter	Coverage metric			
	Block	Decision	c-use	p-use
λ_{min}	0.0000	0.0000	0.0000	0.0000
λ_{max}	35.5996	34.1283	33.5958	43.8348
c_{min}	0.3355	0.1843	0.2469	0.2216
c_{max}	1.0000	1.0000	1.0000	0.6511
p	0.0000	0.0000	0.0000	0.1360
$m(c_{max})$	23.6573	27.8398	25.3014	16.5722

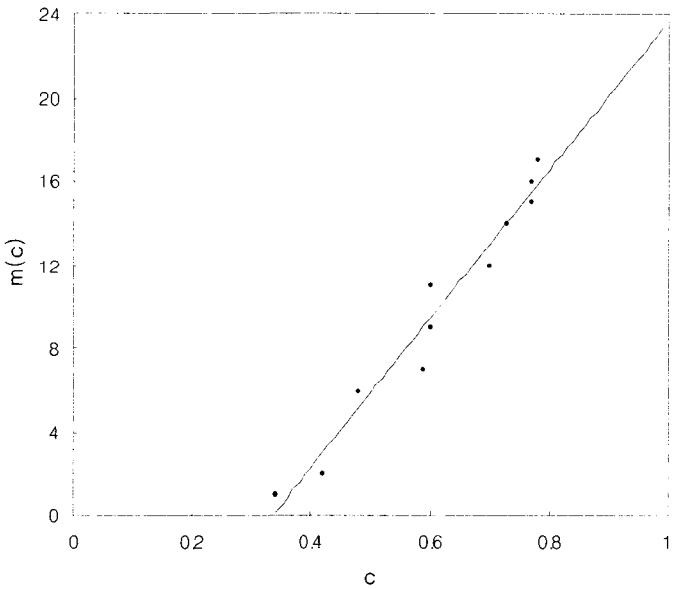

Figure 3. Fitted $m(c)$ curve for block coverage of Mathur data.

5. Concluding Remarks

We first derived the failure rate function relating the software failure rate to software coverage. Then a NHPP SRGM based on the failure rate function was proposed and its practical applicability of the failure rate function and the NHPP SRGM was illustrated. It

was shown that the proposed models work well for two data sets considered in this paper. Apparently more data sets are necessary for further empirical validation. A study of comparing the proposed model and other available models is also required. It has been reported that the software failure rate might increase at the early stage of testing because of the imperfect debugging. The proposed model does not take the imperfect debugging into account. Incorporation of the imperfect debugging into the proposed model will be another future research direction.

References

1. Barlow, R. E. and Hunter, L. C. (1960), Optimum Preventive Maintenance Policies, Operations Research, Vol. 8, pp. 90-100.
2. Chen, M. H., Horgan, J. R. , Mathur, A. P. and Rego, V. J. (1992), A Time/Structure Based Model for Estimating Software Reliability, Technical Report SERC-TR-117-P, Purdue University.
3. Chen, M. H., Lyu , M. R. and Wong, W. E. (1996), An Empirical Study of the Correlation Between Code Coverage and Reliability Estimation, Proceedings of the 3rd IEEE International Symposium on Software Metrics, Berlin, Germany.
4. Chen, M. H., Lyu, M. R. and Wong, W. E. (1997), Incorporating Code Coverage in the Reliability Estimation for Fault-Tolerant Software, Proceedings of the 16th IEEE Symposium on Reliable Distributed System, pp. 45-52, Durham, NC.
5. Chen, M. H., Mathur, A. P. and Rego, V. J. (1995), Effect of Testing Techniques on Software Reliability Estimates Obtained Using A Time Domain Model, IEEE Transactions on Reliability, Vol. 44, pp. 97-103.
6. Dalal, S. R., Horgan, J. R. and Kettenring, J. R. (1993), Reliable Software and Communication: Software Quality, Reliability, and Safety, Proceedings of the 15th IEEE International Conference on Software Engineering, pp. 425-435, Baltimore, MD.
7. Frate, F. D., Garg, P., Mathur, A. P. and Pasquini, A. (1995), On the Correlation Between Code Coverage and Software Reliability, Proceedings of the 6th IEEE International Symposium on Software Reliability Engineering, pp. 124-132, Toulouse, France.
8. Goel, A. L. (1985), Software Reliability Model: Assumptions, Limitations, and Applicability, IEE Transactions on Software Engineering, SE-11, No.12, pp. 1411-1423.
9. Gokhale, S. S., Marinos, P. N. and Trivedi, K. S. (1996), Important Milestones in

Software Reliability Modeling, Communications in Reliability, Maintainability and Serviceability.

10. Gokhale, S. S., Philip, T., Marinos, P. N. and Trivedi, K. S. (1996a), Non-Homogeneous Markov Software Reliability Model with Imperfect Repair, Technical Report TR-96/12, CACC Duke University.
11. Gokhale, S. S., Philip, T., Marinos, P. N. and Trivedi, K. S. (1996b), Unification of Finite Failure Non-Homogeneous Poisson Process Models Through Test Coverage, Technical Report TR-96/36, CACC Duke University.
12. Grottke, M. (1999), Software Reliability Model Study, Technical Report IST-1999-55017, Chair of Statistics, University of Erlangen-Nuremberg, Germany.
13. Grottke, M. (2002), A Vector Markov Model for Structural Coverage Growth and the Number of Failure Occurrences, Proceedings of the 13th International Symposium on Software Reliability Engineering.
14. Horgan, J. R., Mathur, A. P., Pasquini, A. and Rego, V. J. (1995), Perils of Software Reliability Modeling, Technical Report SERC-TR-160-p, Software Engineering Research Center, Purdue University.
15. Lyu, M. R. (1996), Handbook of Software Reliability Engineering, McGraw-Hill.
16. Malaiya, Y. K., Li, N., Bieman, J., Karcich, R. and Skibbe, R. (1994), The Relationship Between Test Coverage and Reliability, Proceedings of the 5th International Symposium on Software Reliability Engineering, pp. 186-195, Monterey, CA.
17. Malaiya, Y. K., Li, N., Bieman, J., Karcich, R. and Skibbe, R. (1996), Software Test Coverage and Reliability, Technical Report CS-96-128, Colorado State University.
18. Malaiya, Y. K. and Srimani, P. K. (1990), Software Reliability Models: Theoretical Developments, Evaluation and Applications, IEEE Computer Society Press, Los Alamos, California.
19. Murthy, D. N. P. (1991), A Note on Minimal Repair, IEEE Transactions on Reliability, Vol. 40, pp. 245-246.
20. Musa, J. D., Iannino, A. and Okumoto, K. (1987), Software Reliability: Measurement, Prediction, Application, McGraw-Hill.
21. Nakagawa, T. and Kowada, M. (1983), Analysis of A System with Minimal Repair and Its Application to A Replacement Policy, Euro. J. Operations Research, Vol. 12, pp. 176-182.
22. Piwowarski, P., Ohba, M. and Caruso, J. (1993), Coverage Measurement Experience During Function Test, Proceedings of the 15th International Conference on Software Engineering, pp. 287-300, Baltimore, MD.

23. Ramamoorthy, C. V. and Bastani, F. B. (1982), Software Reliability-Status and Perspective, IEEE Transactions on Software Engineering, SE-8, 8, pp. 354-371.
24. Shanthikumar, J. G. (1983), Software Reliability Model: A Review, Microelectronics and Reliability, Vol. 23, No. 5, pp. 903-943.
25. Varadan, G. S. (1995), Trends in Reliability and Test Strategies, IEEE Software, Vol. 12, No. 3, p.10.
26. Veevers, A. (1990), Software Coverage Metrics and Operational Reliability, Proceedings of the IFAC/EWICS/SARS Symposium, Gatwick, UK, Vol. 17, pp. 67-69.
27. Veevers, A. and Marshall, A. (1994), A Relationship Between Software Coverage Metrics and Reliability, Software Testing, Verification and Reliability, Vol. 4, pp. 3-8.
28. Vouk, M. A. (1992), Using Reliability Models During Testing with Non-Operational Profiles, Proceedings of the 2nd Bellcore/Purdue Workshop on Issues in Software Reliability Estimation, pp. 103-111.
29. Wood, A. (1997), Software Reliability Growth Models: Assumptions vs. Reality, Proceedings of the 8th International Symposium on Software Reliability Engineering, pp. 136-141, Albuquerque, New Mexico.
30. Woodward, M. R., Hedley, D. and Hennell, M. A. (1980), Experience with Path Analysis and Testing of Programs, IEEE Transactions on Software Engineering, Vol. 6, pp. 278-286.