POSSIBILITIES AND LIMITATIONS OF APPLYING SOFTWARE RELIABILITY GROWTH MODELS TO SAFETY-CRITICAL SOFTWARE

MAN CHEOL KIM*, SEUNG CHEOL JANG and JAEJOO HA
Korea Atomic Energy Research Institute
150 Deokjin-dong, Yuseong-gu, Daejeon, 305-353 Korea
*Corresponding author. E-mail : charleskim@kaeri.re.kr

Received August 16, 2006
Accepted for Publication December 16, 2006

It is generally known that software reliability growth models such as the Jelinski-Moranda model and the Goel-Okumoto's non-homogeneous Poisson process (NHPP) model cannot be applied to safety-critical software due to a lack of software failure data. In this paper, by applying two of the most widely known software reliability growth models to sample software failure data, we demonstrate the possibility of using the software reliability growth models to prove the high reliability of safety-critical software. The high sensitivity of a piece of software's reliability to software failure data, as well as a lack of sufficient software failure data, is also identified as a possible limitation when applying the software reliability growth models to safety-critical software.

KEYWORDS : Software Reliability, Safety-critical Software, Software Reliability Growth Model

1. INTRODUCTION

As digital systems are gradually being introduced into nuclear power plants (NPPs), the need to quantitatively analyze the reliability of the digital systems is also increasing. Kang and Sung [1] identified (1) a piece of software's reliability, (2) common-cause failures (CCFs), and (3) fault coverage as the three most critical factors during the reliability analysis of digital systems. For a reliability estimation of the safety-critical software (the software that is used in safety-critical digital systems), the use of Bayesian Belief Networks (BBNs) seems to be the most widely implemented [2-6]. The use of BBNs in a reliability estimation of safety-critical software is basically a process of indirectly assigning a reliability based on various observed information and experts' opinions. When software testing results or software failure histories are available, we can use a process of directly estimating the reliability of a piece of software by using various software reliability growth models, such as the Jelinski-Moranda model [7] and the Goel-Okumoto's non-homogeneous Poisson process (NHPP) model [8]. Even though it is generally known that the software reliability growth models cannot be applied to safety-critical software due to the extremely long time required for the reliability to grow to acceptable levels [9], in this paper, we try to explore the possibilities and corresponding limitations of applying the software reliability growth models to safety-critical software.

2. METHODS AND RESULTS

2.1 Required Software Reliability

We calculated the required reliability of safety-critical software first. It is assumed that the unavailability due to software failures must not exceed 10^{-4}, which is the same requirement that was used for proving the unavailability requirement of the programmable logic comparators (PDCs) of Wolsung NPP Unit 1. The testing period is assumed to be one month, which is the same assumption that was used in the unavailability analysis of the digital plant protection system (DPPS) of Ulchin NPPs Units 5 and 6. Based on the two values, the required reliability of safety-critical software can be calculated as follows:

$$\frac{\lambda T}{2} \le U \tag{1}$$

$$\lambda \le \frac{U}{2T} = \frac{10^{-4}}{2 \times 1\,month} = 2.78 \times 10^{-7}\,hr^{-1} \tag{2}$$

where,

U: required unavailability

λ: failure rate (of the software)

T: test period.

2.2 Selection of Sample Failure Data

To demonstrate the possibilities and limitations of applying the software reliability growth models to safety-critical software through a sample application, we selected sample failure data. The criteria for selection of the sample data is reasonability (the failure data can reasonably represent the expected failures of safety-critical software) and accessibility (other researchers can easily obtain the sample failure data). The selected sample failure data is that from Goel and Okumoto [8], which is summarized in Table 1.

2.3 Selection of the Software Reliability Growth Models

Musa [10] summarized various software reliability growth models and categorized them into two groups: (1) binomial-type models and (2) Poisson-type models. The most basic and well-known models in the two groups are the Jelinski-Moranda model [7] and the Goel-Okumoto's NHPP model [8]. For this reason, we decided to apply these two representative models to the selected sample failure data.

2.4 Analysis of the Sample Failure Data

After an analysis of the sample failure data, we found that after 24 failures both software reliability growth models produced software reliability results. Table 2 summarizes the analysis results of the sample failure data with the two software reliability growth models. In Table 2, $\hat{N}$ and a mean the estimated total number of inherent software faults. The meanings of the other notations ϕ, λ, b, $\lambda(t)$ can be found in [7,8].

Figure 1 shows the changes of the estimated total number of inherent software faults, which is a part of a software reliability result, calculated by the two software reliability growth models, as software failures are observed one by one. In Figure 1, the time-to-failure data (blue bar) represents the time-to-failure of the observed software failures. For example, the 24th failure was observed 91 days after the occurrence, and a correct repair of the 23rd software failure was implemented.

The number of previously observed failures is represented by the pink line. Because the total number of inherent software faults should not be less than the number of previously observed failures, neither the green line nor the blue line should be below the pink line. The estimated total number of software inherent faults by the Jelinski-Moranda model and the Goel-Okumoto's NHPP model are represented with a green line and a blue line, respectively.

Table 1. Software Failure Data from Goel and Okumoto [8]

Failure Number	Time-to-failure x_k (days)	Cumulative time s_k (days)
1	9	9
2	12	21
3	11	32
4	4	36
5	7	43
6	2	45
7	5	50
8	8	58
9	5	63
10	7	70
11	1	71
12	6	77
13	1	78
14	9	87
15	4	91
16	1	92
17	3	95
18	3	98
19	6	104
20	1	105
21	11	116
22	33	149
23	7	156
24	91	247
25	2	249
26	1	250
27	87	337
28	47	384
29	12	396
30	9	405
31	135	540
32	258	798
33	16	814
34	35	849

2.5 Possibilities of the Software Reliability Growth Models

For the Jelinski-Moranda model, the estimated total software faults ($\hat{N}$) and the single hazard rate (ϕ) after 34 failures (n) are calculated to be 34.003 and 4.845×10^{-3}

Table 2. Change of Estimation Parameters for the Sample Data Calculated with the Jelinski-Moranda Model and the Goel-Okumoto's NHPP Model

Failure Number	Jelinski-Moranda Model			Goel-Okumoto's NHPP Model		
	$\hat{N}$	ϕ	λ	a	b	$\lambda(t)$
22	59.719	3.051E-03	4.795E-03			
23	66.118	2.715E-03	4.877E-03			
24	25.781	9.865E-03	7.319E-04	27.324	8.529E-03	2.835E-02
25	28.157	8.256E-03	1.086E-03	30.175	7.081E-03	3.664E-02
26	31.216	6.849E-03	1.489E-03	33.994	5.790E-03	4.628E-02
27	28.194	8.355E-03	4.158E-04	29.429	7.402E-03	1.798E-02
28	28.911	7.859E-03	2.982E-04	30.036	7.009E-03	1.427E-02
29	30.493	6.905E-03	4.294E-04	31.765	6.165E-03	1.705E-02
30	32.291	6.071E-03	5.794E-04	33.766	5.416E-03	2.040E-02
31	31.425	6.481E-03	1.147E-04	32.371	5.856E-03	8.026E-03
32				32.346	5.686E-03	1.969E-03
33				33.590	4.965E-03	2.930E-03
34	34.003	4.845E-03	6.056E-07	34.827	4.441E-03	3.565E-03

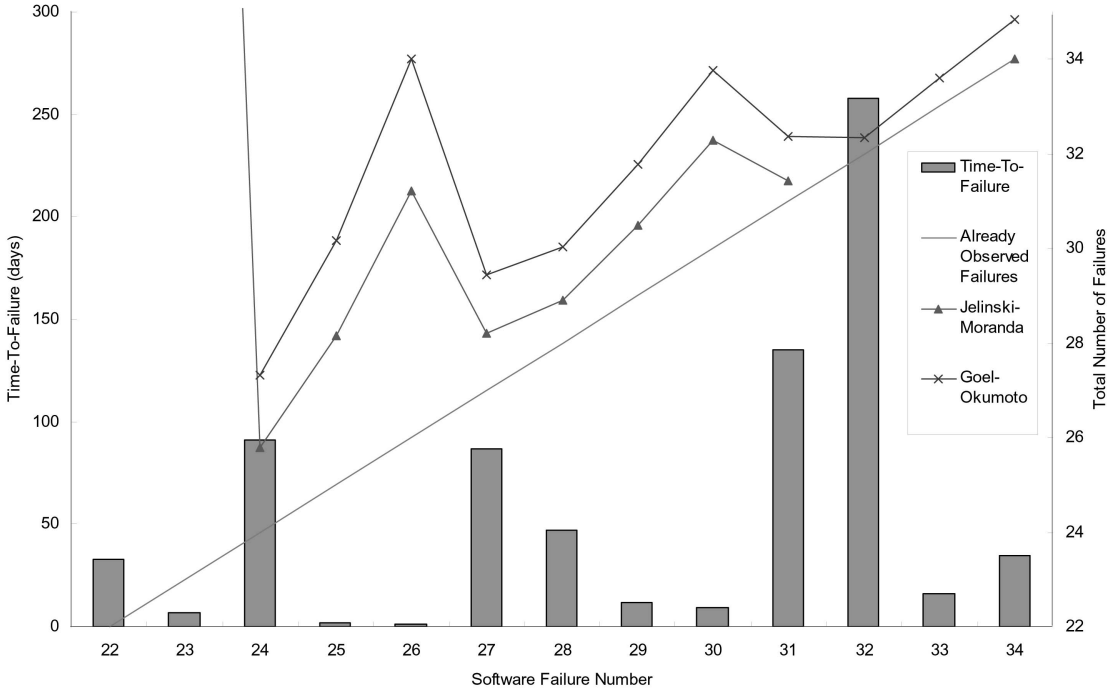

Fig. 1. Change of Estimated Total Number of Inherent Software Faults Calculated by the Jelinski-Moranda Model and the Goel-Okumoto NHPP Model

month^{-1}, respectively. Therefore, after 34 failures and a correct repair of the software, the expected failure rate of the software (λ_{34}) becomes

$$\lambda_{34} = \phi(\hat{N} - n) = 6.056 \times 10^{-7}\, hr^{-1} \qquad (3)$$

When comparing Eq.(3) with Eq.(2), the two calculation results are in the same order of a magnitude. This means that there are some possibilities that the software reliability growth models can be applied to prove the high reliability of safety-critical software when almost all the inherent software faults are identified and correctly repaired.

2.6 Limitations of the Software Reliability Growth Models

However, there are several limitations when applying the software reliability growth models to safety-critical software. One of the most serious limitations is that the expected total number of inherent software faults calculated by the software reliability growth models is highly sensitive to the time-to-failure data. As shown in Figure 1, after long time-to-failures, such as shown in the case of the 24th, 27th, and 31st failures, drastic decreases in the estimated total number of inherent software faults can be observed for both of the software reliability growth models. For the software reliability growth models to be successfully applied to safety-critical software, the estimated total number of inherent software faults (the blue line and green line in Figure 1) should be almost constant as the software failure number (x-axis) increases. The interpretation of Figure 1 suggests that the estimated total number of inherent software faults varies from 25.78 to 34.00 for the Jelinski -Maranda model and from 27.32 to 34.83 for the Goel-Okumoto's NHPP model as the software failure numbers change from 24 to 34. This sensitivity to the time-to-failure data gives an impression that the resultant high software reliability as shown in Eq.(3) could be a coincidence during the calculation process. One of the other limitations is that it seems that we need at least 20 failure data sets, but we cannot be sure that the amount of failure data results will be generated during the development and testing of safety-critical software.

3. CONCLUSIONS

In this paper, we demonstrated that there are some possibilities that software reliability growth models can be applied to prove the high reliability of safety-critical software at the point where all the inherent software faults are identified and correctly repaired. However, we also described the limitations of the possibilities caused by the high sensitivity of the estimated total number of inherent software faults to the time-to-failure data and the uncertainty of the availability of sufficient software failure sets for the safety-critical software to be applied to the software reliability growth models.

ACKNOWLEDGEMENTS

This work was partially supported by Korean Nuclear Mid- and Long-Term Research and Development Program of the Korean Ministry of Science and Technology (MOST).

REFERENCES

[1] H. G. Kang and T. Sung, "An Analysis of Safety-Critical Digital Systems for Risk-Informed Design," *Reliability Engineering and System Safety*, **78**, 3 (2002).

[2] N. Fenton, B. Littlewood, M. Neil, L. Stringini, A. Sutcliffe, and D. Wright, "Assessing Dependability of Safety Critical Systems using Diverse Evidence," *IEE Proceedings on Software*, ***145***, 1 (1998).

[3] N. E. Fenton and M. Neil, "A Critique of Software Defect Prediction Models," *IEEE Transactions on Software Engineering*, **25**, 5 (1999).

[4] N. E. Fenton and M. Neil, "Software Metrics: Successes, Failures and New Directions," *The Journal of Systems and Software*, **47**, 2-3 (1999).

[5] M. Bouissou, F. Martin, and A. Ourghanlian, "Assessment of a Safety-Critical System Including Software: A Bayesian Belief Network for Evidence Source," *Proceedings on Annual Reliability and Maintanability Symposium*, pp.142-150, 1999.

[6] H. S. Eom, H. G. Kang, S. C. Jang, and J. J. Ha, "A Study on the Quantitative Reliability Assessment Method for Safety Critical Software Using Bayesian Belief Nets," KAERI/TR-2437/2003, Korea Atomic Energy Research Institute (2003).

[7] Z. Jelinski, P. B. Moranda, "Software Reliability Research" (W. Freiberger, Editor), *Statistical Computer Performance Evaluation*, Academic, New York, p.465 (1972).

[8] A. L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Transactions on Reliability*, **R-28**, 3 (1979).

[9] R.W. Butler and G.B. Finelli, "The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software," *IEEE Transactions on Software Engineering*, **19**, 1 (1993)

[10] J. D. Musa, A. Iannino, and K. Okumoto, *Software* Reliability – *Measurement, Prediction, Application*, McGraw-Hill Book Company, Singapore (1987).