Transactions of the Korean Nuclear Society Spring Meeting
Chuncheon, Korea, May 25-26 2006

Possibilities and Limitations of Applying Software Reliability Growth Models to Safety-Critical Software

Man Cheol Kim, Seung Cheol Jang, Jaejoo Ha
Integrated Safety Assessment Division, Korea Atomic Energy Research Institute,
150 Deokjin-dong, Yuseong-gu, Daejeon, 305-353, Korea, charleskim@kaeri.re.kr

1. Introduction

As digital systems are gradually introduced to nuclear power plants (NPPs), the need of quantitatively analyzing the reliability of the digital systems is also increasing. Kang and Sung [1] identified (1) software reliability, (2) common-cause failures (CCFs), and (3) fault coverage as the three most critical factors in the reliability analysis of digital systems.

For the 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 most widely used. The use of BBNs in 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 the software using various software reliability growth models such as Jelinski-Moranda model [2] and Goel-Okumoto's non-homogeneous Poisson process (NHPP) model [3]. Even though it is generally known that software reliability growth models cannot be applied to safety-critical software due to small number of expected failure data from the testing of safety-critical software, we try to find possibilities and corresponding limitations of applying software reliability growth models to safety-critical software.

2. Methods and Results

2.1 Required Software Reliability

We first calculate a required reliability of safety-critical software. 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 programmable logic comparators (PDCs) of Wolsung NPP unit 1. The testing period is assumed to be 1 month, which is the same assumption that is used in the unavailability analysis of digital plant protection system (DPPS) of Ulchin NPPs unit 5 and 6. Based on the two values, the required reliability of safety-critical software can be calculated as follows:

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

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

where,

U : required unavailability
λ : failure rate (of the software)
T : test period.

2.2 Selection of Example Failure Data

To demonstrate the possibilities and limitations of applying software reliability growth models to safety-critical software through an example application, we select an example failure data. The criteria for the selection of the example data are *reasonability* (the failure data can reasonably represent the expected failures of safety-critical software) and *accessibility* (other researchers can easily get the example failure data). The selected example failure data are those from Goel and Okumoto [3].

2.3 Selection of Software Reliability Growth Models

Musa [4] summarized various software reliability growth models and categorized them into two groups: (1) binomial-type models and (2) Poisson-type models. Most basic and most well-known models in the two groups are Jelinski-Moranda model [2] and Goel-Okumoto's NHPP model [3]. For this reason, we decide to apply the two representative models to the selected example failure data.

2.4 Analysis of Example Failure Data

After analysis of example failure data, we found that the software reliability growth model produce software reliability results after 22 failures. Figure 1 shows the change of estimated total number of inherent software faults (which is a part of 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 observed software failures. For example, the 24th failure was observed 91 days after the occurrence and correct repair of the 23rd software failure.

The number of already observed failures is represented with the pink line. Because the total number of inherent software faults should not be less than the number of already observed failures, the green line and the blue line should not be below the pink line.

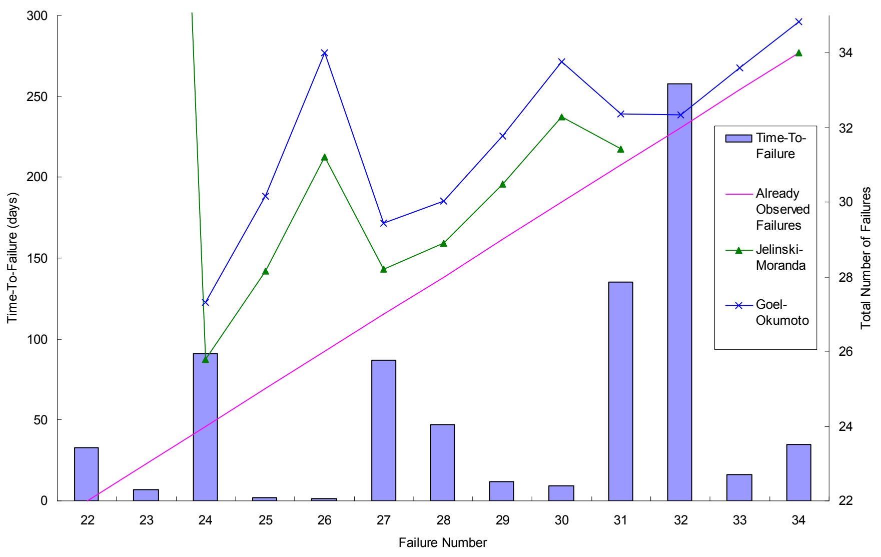

Figure 1 Change of estimated total number of inherent software faults calculated by Jelinski-Moranda model and Goel-Okumoto's NHPP model

The estimated total number of software inherent faults by Jelinski-Moranda model and Goel-Okumoto's NHPP model are represented with the green line and the blue line, respectively.

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

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

When comparing Eq.(3) with Eq.(2), it can be said that there are some possibilities that software reliability growth models can be applied to prove the high reliability of a safety-critical software when almost all inherent software faults are identified and correctly repaired.

2.6 Limitations of Software Reliability Growth Models

But, there are several limitations for the software reliability growth models to be applied to safety-critical software. One of the most serious limitations is that the expected total numbers of inherent software faults calculated by software reliability growth models are highly sensitive to the time-to-failure data. As shown in Figure 1, after long time-to-failures such as shown in 24th failure, 27th failure, and 31st failure, drastic decreases in the estimated total number of inherent software faults can be observed for both software reliability growth models. This sensitiveness to time-to-failure data gives an impression that the resultant high software reliability as shown in Eq.(3) can be a coincidence in the calculation process. One of other limitations is that it seems that we need at least 20 failure data, but we cannot make sure that that amount of failure data will be generated during development and testing of safety-critical software.

3. Conclusion

In this paper, we demonstrate 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 inherent software faults are identified and correctly repaired. But, we also describe the limitations of the possibility caused by the high sensitiveness of the estimated total number of inherent software faults to the time-to-failure data, and the uncertainty on the availability of enough software failures for the safety-critical software to be applied to software reliability growth models.

REFERENCES

[1] H. G. Kang, T. Sung, An Analysis of Safety-Critical Digital Systems for Risk-Informed Design, Reliability Engineering and System Safety, Vol.78, p.307, 2002.

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

[3] A. L. Goel, K. Okumoto, Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures, IEEE Transactions on Reliability, Vol.R-28(3), p.206, 1979.

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