PROCEDURE FOR APPLICATION OF SOFTWARE RELIABILITY GROWTH MODELS TO NPP PSA

HAN SEONG SON*, HYUN GOOK KANG[1] and SEUNG CHEOL CHANG[1]
Joongbu University, Department of Game Engineering
101 Daehak-ro, Chubu-myeon, Geumsan-gun, Chungnam, 312-702, Korea
[1] Korea Atomic Energy Research Institute
P. O. Box 105, Yuseong, Daejeon, 305-600, Korea
*Corresponding author. E-mail : hsson@joongbu.ac.kr

Received January 19, 2009
Accepted for Publication June 19, 2009

As the use of software increases at nuclear power plants (NPPs), the necessity for including software reliability and/or safety into the NPP Probabilistic Safety Assessment (PSA) rises. This work proposes an application procedure of software reliability growth models (RGMs), which are most widely used to quantify software reliability, to NPP PSA. Through the proposed procedure, it can be determined if a software reliability growth model can be applied to the NPP PSA before its real application. The procedure proposed in this work is expected to be very helpful for incorporating software into NPP PSA.

KEYWORDS : Probabilistic Safety Analysis, Software Reliability, Reliability Growth Models, Application Procedure, Nuclear Power Plants

1. INTRODUCTION

Software reliability and/or safety [1] have become an important issue for instrumentation and control systems in nuclear power plants (NPPs). Software is one of the important safety issues in digital system safety assessment [2]. When using safety-critical software, various methods like formal verification and validation [3,4] play critical roles in demonstrating compliance with several regulatory requirements. Meanwhile, for the past several decades, Probabilistic Safety Assessment (PSA) techniques have been used in the nuclear industry to assess the relative effects of contributing events on plant risk and system reliability. More realistic PSA results provide more reasonable and accurate risk-informed decisions. The field experience of U.S. nuclear plants during the period of 1990 through 1993 shows that software error caused a significant number of digital system failures [5]. Software errors (30 failures) cause the majority of digital system failure events in comparison with the fact that only 9 events were caused by random component failures. In addition, since these failures could cause common-mode or cause failure, which might remove the redundancy effect if the same software is installed in redundant systems and can lead to significant safety events, software failure analysis is inevitable for realistic PSA results. Based on the results of sensitivity studies, a precedent research [6] pointed out that quantification of the 'possibility of software error' and 'imperfection of a fault-tolerant mechanism' is very important and an inevitable factor for realistic reliability evaluation. In summary, quantitative software reliability can give confidence for the use of software, especially when it is incorporated with NPP PSA.

It is notable that there has been much discussion among software engineering researchers about whether a software failure can be treated in a probabilistic manner [7]. Software faults are design faults by definition. It means that software is deterministic and its failure cannot be represented by 'failure probability.' However, a fault in software causes a system failure only when the input sequence activates the fault. If we assume the randomness of the input sequences in the real use of software, its failure could be treated based on a probabilistic method. If the input profile of the software can be determined statistically, we can estimate the software reliability in a probabilistic manner based on the input profile [2].

Software faults are integrated into PSAs to statistically analyze system reliability and/or safety. Li et al. [8] developed the software failure taxonomy to integrate software into the PSA process, which also identifies software related failures. They pointed out that software failure events appear as the initiating and intermediate events in event sequence diagrams for event tree analysis or as the elements of fault trees. For example, a software fault which results in spurious activation of a reactor protection system causes a transient of a NPP and will be

treated as an initiating event. In addition, a software fault which results in the unavailability of a safety injection pump control will be modeled in a fault tree for the safety function failure. Nevertheless, how to quantify the rate of software failure events is still an open issue. There have been various attempts to quantify software failure rate. Among them, software reliability growth models (RGMs) are the most practical and popular. Even though software RGMs have a few limitations, careful application of them would raise their applicability to industry. In this work, an application procedure of software RGM to NPP PSA is proposed. Through the proposed procedure, it can be determined if a software reliability growth model can be applied to NPP PSA before its real application. The procedure proposed in this work is expected to be very helpful for incorporating software into NPP PSAs.

The remainder of the paper is organized as follows: Section 2 briefly describes software RGMs. General features and limitations of software RGMs are introduced. In Section 3, the proposed application procedure of the software RGM is described. The example application of the procedure is reported in Section 4. Listing the application criteria of software RGM, which are derived from the proposed procedure and its example application, Section 5 concludes this work.

2. SOFTWARE RELIABILITY GROWTH MODELS

Software RGMs are one of the most mature techniques for software dependability assessment. A Software RGM is a mathematical model which estimates the increment of the reliability as a result of a fault removal by regarding the number of software faults detected in a time-interval and the software failure-occurrence time interval as random variables. Software RGMs capture failure behavior of software during testing and extrapolate to determine its behavior during operation. The key parameters of them are a and b: a means the expected number of faults in the software when the testing starts, and b stands for the fault detection rate per remaining fault in the software. Software RGMs use failure data information and trends observed in failure data to estimate the parameters and derive reliability predictions. There have been various kinds of classifications for software RGMs [9,10,14]. They include Musa Basic Model, Goel Okumoto Non-Homogeneous Poisson Process (NHPP) Model, Musa Okumoto NHPP Model, Musa Poisson Execution Time Model, Jelinski Moranda Model, Littlewood Verall Model, Weibull Model, Raleigh Model, Delayed S-shaped Growth Model, Exponential Model, Logarithmic Poisson Model, Imperfect Debugging Model, Inflection S-shaped Growth Model, and Logistic Model. Each model has its own best fitting failure data. This means that the model selection is very crucial to producing meaningful reliability predictions.

Software RGMs must meet many assumptions in view of software testing and repair [11]. Some software RGMs assume testing follows an operational profile, but it is particularly difficult to define operation profile and perform operational tests. They also assume that software does not change, except that defects are fixed when discovered, and repair is immediate and perfect. However, in practice, defects may not be repaired immediately and the repair of defects may introduce new defects. In summary, the assumptions for software RGMs are an open problem because they are often violated in one way or another.

Limitations of software reliability growth models that are applied to a safety-critical software system also need to be considered [12]. One of the most serious limitations is the expected total number of inherent software faults calculated by the software reliability growth models that are highly sensitive to time-to-failure data. After long time-to-failures, drastic decreases in the estimated total number of inherent software faults are frequently observed for software RGMs. This sensitivity to time-to-failure data indicates that the resultant high software reliability could be a coincidence in the calculation process. One other limitation is that, although many failure data are needed, we cannot be sure that a sufficient amount of failure data is revealed during the development and testing of a safety-critical software system.

Nevertheless, many studies reported that software RGMs perform well in practice. Two industrial applications of reliability measurement are described in [13]. In both cases, software RGMs predicted failure rates well despite distributional assumption violations. This demonstrated that reliability measurement based on software RGMs may be used in industry. Software RGMs showed quite a good performance in predicting the additional test effort needed to achieve a desirable quality level during the system test and predicting the number of remaining failures that could be reported after release at the end of the system test. From these applications, we can see the possibility that reliability measurement based on software RGMs may be used for the NPP PSA. Therefore, through careful application of software RGMs with a systematic procedure, their applicability to NPP PSAs would be raised considerably.

3. PROPOSED APPLICATION PROCEDURE

This work proposes a systematic procedure for applying software RGMs to NPP PSAs. The purpose of this procedure proposition is to enhance the applicability of software RGMs to NPP PSAs. The procedure consists of model selection, reliability goal setting, determination of application range, acquisition of software failure data, and application of the data to the PSA. Figure 1 shows the proposed application procedure.

3.1 Model Selection

Generally, it is necessary to select appropriate software

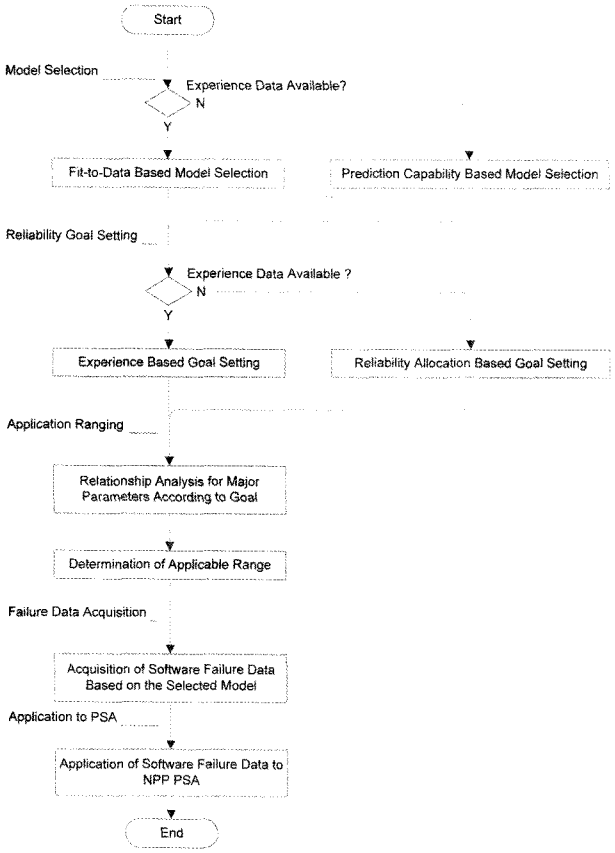


Fig. 1. Proposed Application Procedure of Software RGMs to NPP PSAs

reliability models, considering many criteria, to estimate software reliability. Criteria used for software reliability model selection have been proposed [9], including life cycle phase, output desired by the user, input required by the model, trend exhibited by the data, validity of assumptions according to the data, nature of the project, structure of the project, test process, and development process. This work proposes an application procedure and criteria of software RGMs to NPP PSAs assuming that the result of the model selection is software RGMs. As mentioned in Section 2, there exist various kinds of software RGMs. Thus, additional model selection criteria are required specifically for software RGMs.

P. H. Seong et al. introduced a case where an expected software failure rate based on a software reliability growth model is very close to the required reliability of safety critical software [12]. This shows that a certain failure data set is appropriate to the use of a specific software reliability growth model. If a set of failure data can be obtained from other similar projects, the software reliability that is fit to the data set shall be selected. However, if data gathering is difficult, the degree of fitment on available failure data and the prediction capability of models can be selection criteria for the software RGM. D. R. Prince Williams evaluated six popular software RGMs in view of the fitment degree and the prediction capability [14]. The models include Delayed S-shaped Growth Model, Exponential Model, Logarithmic Poisson Model, Imperfect Debugging Model, Inflection S-shaped Growth Model, and Logistic Model. The former three models are two parameter models and the last three models are three parameter models. The prediction capability of each model was analyzed using four famous data sets. This work estimated the parameters and evaluated the goodness of fit of each model using 80% of the failure data. It also compared prediction capability of the models by validating against the remaining 20% of the data. D. R. Prince Williams pointed out that the Delayed S-shaped Growth Model can be a good candidate though the Logistic Model shows the best overall failure data prediction capability. This is because the prediction variation between the two models is very small (less than 5%) and the implementation of the Delayed S-shaped Growth Model is easier than that of the Logistic Model.

3.2 Reliability Goal Setting

Reliability goal, so to speak, or required reliability can be obtained by operational experiences or reliability allocation. Unavailability of a safety-grade component in a NPP is usually around 10^{-3}, and the control system for this component must have less unavailability. Based on this value, P. H. Seong et al. assumed unavailability due to software failure did not exceed 10^{-4}, which is the same requirement that is used for proving the unavailability requirement of the system that the software runs on [12]. The unavailability requirement reflects the operational experiences of the system in previously constructed NPPs. From the safety viewpoint, the function of the safety software properly responds to deviation of a NPP. With the assumption that the testing is performed in a real operation profile, the mission time of the software can be calculated from the frequency of the deviation arrival, which has a value in Ulchin NPP 3 & 4 licensing PSA model of around 4.0/yr. The software failure rate is calculated using the unavailability requirement (10^{-4}) and the mission time (1/4 yr). In this case, the reliability goal of the software is 4.6E-8/hr. If the same or similar software systems are operated in a sufficient number of fields, failure data can be obtained to estimate this ultra-high software reliability. The estimated software reliability is a reliability goal in this case.

Reliability allocation deals with the setting of reliability goals for individual components such that a specified system reliability goal is met and the component goals are 'well balanced' among themselves [15]. Generally, the component goals are balanced based on development time, difficulty, or risk. For a NPP instrumentation and control system, however, reliability shall be as high as possible and not be optimized or compromised by development effort. Thus, reliability allocation for a NPP PSA shall be

conservatively performed. For example, if the target failure rate of a system is in the order of 10^{-4}, the target failure rate of software should be in the same order.

3.3 Determination of Application Range

In this step, the relationship analysis for major parameters according to reliability goal is performed and the applicability of the selected model is determined. As mentioned in Section 2, the key parameters of a software RGM are *a* and *b*. When a software reliability goal is set, the relationship between *a* and *b* meeting the reliability goal is uniquely derived. The derivation can be performed by a mathematical simulation like the MATLAB simulation. If the relationship between *a* and *b* is the curve shown in Figure 2, the curve divides the whole region into 'Below' region, 'Adjacent' region, and 'Far Above' region. If the collected information and/or failure data drop *a* and *b* in the 'Below' region or 'Adjacent' region, the selected model can be applied to the NPP PSA. On the other hand, if the parameters estimated based on the collected information and/or failure data fall into the 'Far Above' region, the selected software reliability growth model cannot be applied to the NPP PSA. The border between the regions is determined by engineering judgment.

It should be noted that the key parameters can be estimated based on the collected information from various sources and/or failure data from testing of previous similar projects. The information sources include software quality measures, development processes, verification and validation (V&V) results, and so forth. Let us assume that the estimation of *a* is available from V&V results. When the estimation produces a_1 as shown in Figure 3(a) - Step (1), the boundary of the applicability region produces b_1 as a criterion for determining the applicability of the selected model - Step (2). If the experts predict that the *b* of the model is less than or equal to b_1, the selected model is determined to be applicable. Figure 3(b) shows the case that the estimation of *b* is available. When the estimation produces b_1 based on the V&V results - Step (1), the boundary of the applicability region produces a_1 as a criterion for determining the applicability of the selected model - Step (2). If the experts predict that the *a* of the model is less than or equal to a_1, the selected model is determined to be applicable.

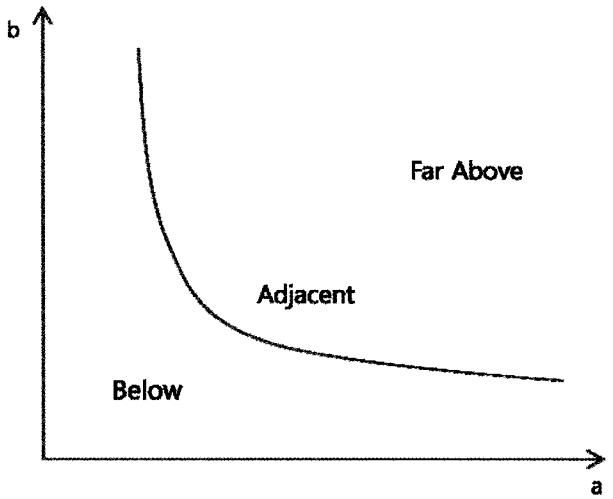
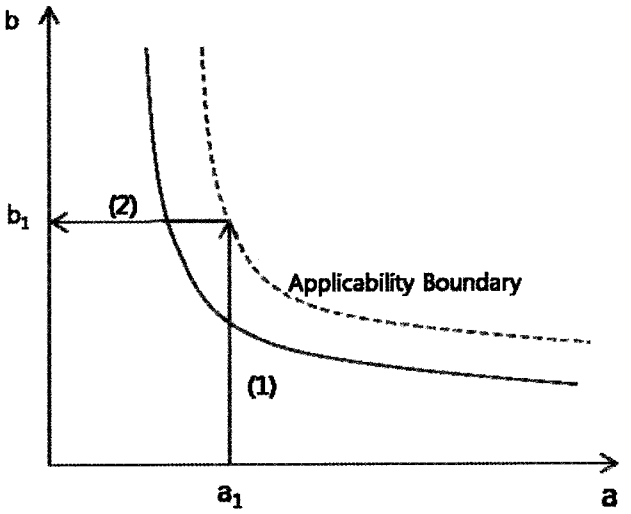
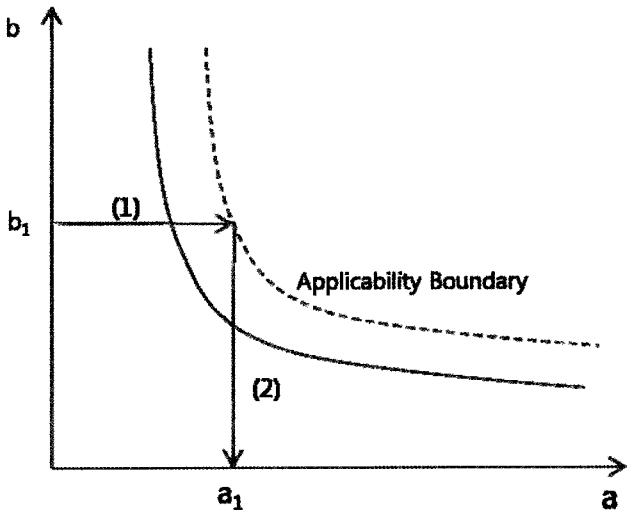


Fig. 2. Key Parameter Relationship and Applicability Regions



(a) In the case that the estimation of *a* is available



(b) In the case that the estimation of *b* is available

Fig. 3. Determination of Applicability

3.4 Failure Data Acquisition and Application to NPP PSA

Once it is proven that the selected model can be applied to the NPP PSA, reliability prediction based on the selected model is performed. The NPP PSA usually requires failure rate data. Thus the selected model shall capture failure behavior of software during testing and extrapolate to determine its behavior during operation in terms of failure rate. The derived software failure rate is directly applied to the NPP PSA as a corresponding basic event frequency.

If the selected model is the Delayed S-shaped Growth Model, the expected value function of the model, μ(t), representing the cumulative number of failures expected to occur after the software has executed for time t, is:

$$\mu(t) = a(1 - (1 + bt)e^{-bt}) \tag{1}$$

From Eq. (1), the fault-detection intensity is derived as follows:

$$\lambda(t) = \frac{d\mu(t)}{dt} = ab^2te^{-bt} \tag{2}$$

The error detection rate per error at time t is:

$$D(t) = \frac{d\mu(t)}{dt}/[a - \mu(t)] = b^2t/[1 + bt] \tag{3}$$

The expected number of errors remaining in the system is:

$$n(t) = E\{N(\infty) - N(t)\} = a(1 + bt)e^{-bt} \tag{4}$$

Assuming that the most recent failure occurred at time t, reliability R(x|t) is the probability that no failure occurs during time x. R(x|t) is expressed in terms of the expected value function as follows:

$$R(x \mid t) = \exp(\mu(t) - \mu(t + x)) \tag{5}$$

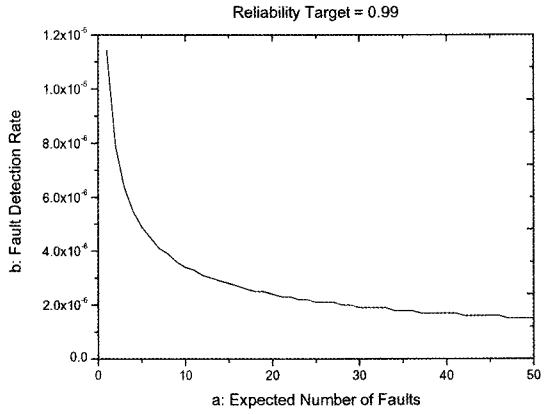
From Eq. (2), the software failure rate at time t can be calculated only if a and b are estimated. Let us assume that the test data were gathered as shown in Table 1. Through the nonlinear fitting with these data, a and b are estimated to be 247.221 and 0.191014, respectively [16].

At this time point, experts are requested to determine whether the parameters can be used to predict the software failure rate. If they think the test data are sufficient to estimate a and b, the estimated a and b are used to predict the software failure rate using Eq. (2). Let us assume that the software operates just after the 25 week testing described in Table 1. The software failure rate at 13 weeks (1/4 yr, an expected deviation arrival time of a NPP) after the testing revealed the most recent failure (22nd week) should be predicted. In this case, t is 35 weeks (22 weeks + 13 weeks) and the calculated failure rate λ(t=35 weeks) is 0.394286934. This means that, if the software system is tested for 35 weeks, the failure rate right after the test is 0.394286934. In this case, the predicted software failure rate can be directly used for the NPP PSA in that the calculated failure rate can be considered as the operational failure rate because there is no bug-fixing during the operation. On the other hand, if they do not feel the test data are enough to estimate the parameters, the predicted reliability can be an alternative to determine usability. R(x=13|t=35) means the probability that no failure occurs during the time x (13 weeks) after the most recent failure occurred at the time t (35nd week). By Eq. (1) and Eq. (5), R(x=13|t=35) is calculated to be 0.121135457. If this figure

Table 1. Example Test Data [1), 2)]

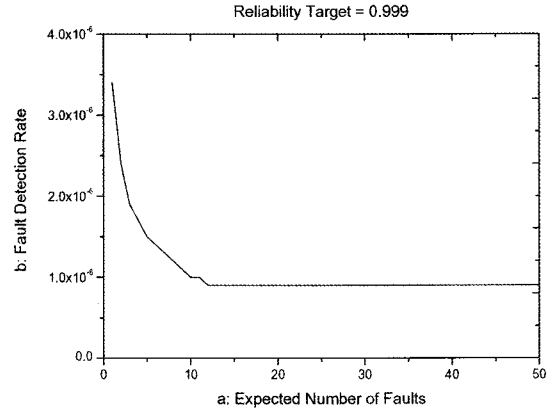
WEEK	CNF	WEEK	CNF	WEEK	CNF	WEEK	CNF
1	44	8	100	15	197	22	230
2	75	9	124	16	205	23	230
3	75	10	130	17	214	24	230
4	75	11	130	18	215	25	230
5	75	12	159	19	225		
6	75	13	175	20	227		
7	75	14	181	21	228		

[1)] CNF: Cumulative Number of Failures
[2)] These data were extracted from the proceedings of COMSAC 2004 [15]



a	1	2	5	10	20	30	40	50
b	1.14E-5	7.90E-6	4.90E-6	3.40E-6	2.40E-6	1.90E-6	1.70E-6	1.50E-6

(a) Relationship between a and b (Reliability Target = 0.99)



a	1	2	3	4	5	6	10	20
b	3.40E-6	2.40E-6	1.90E-6	1.70E-6	1.50E-6	1.40E-6	1.00E-6	0.90E-6

(b) Relationship between a and b (Reliability Target = 0.999)

Fig. 4. Relationships between a and b According to Reliability Targets

is acceptable to the experts, the predicted software failure rate can be used for the NPP PSA. However, if not, it cannot be used. More efforts to gather information and/or data should be taken in this case.

4. EXAMPLE APPLICATION

In order to exemplify the proposed procedure for applying software RGMs to NPP PSAs, it is applied to an artifact project. For the first step, assuming that any experience or failure data could not be obtained from other similar projects, the Delayed S-shaped Growth Model is selected according to the result of [14]. Once the reliability model is determined, the reliability goal (or reliability target) is required. When the reliability target is 0.99, the relationship between a and b is as shown in Figure 4(a). When the reliability target is 0.999, the relationship between a and b is as shown in Figure 4(b). Major points on the curves are also shown in the corresponding figures. These relationships are derived from the MATLAB simulation for the Delayed S-shaped Growth Model. After combining Eq. (1) and Eq. (5), a MATLAB program simulated the combined equation for R(x|t) by increasing a with the increment of one. Once a reliability target (R) is set, the simulation produces a-b relationship graphs like Figure 4.

In this artifact project, the reliability goal was set to be 0.999 through a reliability allocation process. The

Table 2. Test Data of Example Application [1]

MONTH	CNF	MONTH	CNF	MONTH	CNF
1	1	5	18	9	30
2	3	6	21	10	31
3	11	7	24		
4	15	8	27		

[1] CNF: Cumulative Number of Failures

collected information from various sources and test data from a previous project made it possible to predict that the number of remaining faults is within 5. Expert opinion showed that the fault detection rate can be estimated to be less than 5.0E-3. According to Figure 4, if the reliability goal is 0.999 and the expected number of faults in the software is 5, the fault detection rate per remaining fault in the software is estimated to be 1.5E-6. Through the process described in Figure 3(a), the collected information and failure data drive a and b into the 'Adjacent' region and the Delayed S-shaped Growth Model is proven to be applied to the NPP PSA. Since the Delayed S-shaped Growth Model was selected, the failure behavior of software during the remaining testing was captured based on the model.

The failure behavior was extrapolated to determine its behavior during operation in terms of failure rate. The extrapolation was performed with the failure data collected by a testing process. The data are shown in Table 2. From the table, it can be known that there was no failure revealed by the tests performed after 10 months. Through a nonlinear fit using the collected failure data, a and b of the Delayed S-shaped Growth Model were estimated to be 37.0281 and 0.326899, respectively. The fitting was performed by the 'nonlinear fit' function of the SAS JMP tool [17]. The system experts decided to estimate the software failure rate at the time point of the 56th week (10 months (43 weeks) + 13 weeks). This is reasonable in the sense that the software should be tested at least for an additional 13 weeks (1/4 yr, an expected deviation arrival time of NPP) even though there was no failure revealed by the tests performed after 10 months. The estimated software failure rate is around 4.436E-8 per week (Eq. (2)).

As mentioned in Section 3.4, system experts shall determine whether the estimated parameters and the predicted software failure rate can be used for the NPP PSA or not. In this project, they determined that the estimated parameters and the predicted software failure rate could be used for the NPP PSA. When the a and b used to determine the applicability region were compared with the a and b estimated with the test data, it might be said that the test should have been designed to produce more software failures even after 10 months. However, in spite of the lack of test data, the reliability during the expected deviation arrival time (i.e., R(x=13|t=56)) was estimated to be 0.999992125 by Eq. (5) and this figure was acceptable to the experts. Therefore, the predicted software failure rate could be directly applied to the NPP PSA as a corresponding basic event frequency.

5. CONCLUSIONS

This work proposed an application procedure of software RGM, which is used practically to quantify software reliability to NPP PSAs. Reliability goal setting and application ranging are essential steps of the procedure. Through the steps, a selected software reliability growth model can be efficiently validated in view of adequacy for the project before real application. Based on the proposed procedure and an example application, the criteria for the application of software RGM to NPP PSA are derived as follows:

1) Possible to set a reliability goal of software
2) Possible to collect information and/or failure data in evidence of the major parameters of the selected model (i.e., the expected number of faults and the fault detect rate per remaining fault)
3) Possible to gather failure data during testing to determine its behavior during operation based on the selected model

If a software reliability growth model can meet the above criteria, it can produce reliability data which are directly applied to NPP PSAs. Therefore, the procedure proposed in this work is expected to be very helpful for incorporating software into NPP PSAs. Furthermore, although the proposed procedure was developed for software RGMs, a similar procedure can be developed for other kinds of software reliability models. The application of the procedure to real projects remains as a further study.

ACKNOWLEDGEMENTS

This research was supported by the Nuclear Research & Development Program of the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korean Government (MEST) (Grant Code: M20702030002-08M0203-00210).

REFERENCES

[1] N.G. Leveson, *SAFEWARE, System safety and Computers*, Addison Wesley, 1995.

[2] H.G. Kang, et al., "An Overview of Risk Quantification Issues of Digitalized Nuclear Power Plants using Static Fault Tree," *Nuclear Engineering and Technology*, Vol.41, 2009.

[3] D.A. Peled, *SOFTWARE RELIABILITY METHODS*, Springer, 2001.

[4] J. Yoo, T. Kim, S. Cha, J.S. Lee, and H.S. Son, "A formal software requirements specification method for digital

nuclear plants protection systems," *Journal of Systems and Software*, 74, 1, pp.73–83, 2005.
[5] NEA/CSNI/R(97)23, Operating and maintenance experience with computer- based systems in nuclear power plants, 1998.
[6] Kang, H.G and Sung, T, "An analysis of safety-critical digital systems for risk-informed design," *Reliability Engineering and Systems Safety*, Vol. 78, No. 3, 2002.
[7] White, R.M and Boettcher, D.B, "Putting Sizewell B digital protection in context," *Nuclear Engineering International*, pp. 41-43, 1994.
[8] B. Li, M. Li, S. Ghose, and C. Smidts, "Integrating software into PRA," *Proceedings of the 14th International Symposium on Software Reliability Engineering*, IEEE Computer Society Press, 2003.
[9] C.A. Asad, M.I. Ullah, M.J. Rehman, "An approach for software reliability model selection," *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMSAC'04)*, IEEE, 2004.
[10] M. R. Lyu, *HANDBOOK OF SOFTWARE RELIABILITY*, IEEE Computer Society Press and McGraw-Hill, 1996.
[11] A. Wood, "Software Reliability Growth Models: Assumptions vs. Reality," *Proceedings of the International Symposium on Software Reliability Engineering*, **23**, **11**, pp. 136-141, 1997.
[12] P.H. Seong, et al., *RELIABILITY AND RISK ISSUES IN LARGE SCALE SAFETY-CRITICAL DIGITAL CONTROL SYSTEMS*, Springer-Verlag, 2008.
[13] J. Musa, A. Ackerman, "Quantifying Software Validation: When to Stop Testing," *IEEE Software*, May, pp. 19-27, 1989.
[14] D.R. Prince Williams, "Prediction Capability Analysis of Two and Three Parameters Software Reliability Growth Models," *Information Technology Journal*, **5**, **6**, pp. 1048-1052, 2006.
[15] J.D. Musa, A. Iannino, and K. Okumoto, *SOFTWARE RELIABILITY, Measurement, Prediction, Application*, McGraw-Hill, 1987.
[16] C. Huang, C. Lin, S. Kuo, M.R. Lyu, and C. Sue, "Software Reliability Growth Models Incorporating Fault Dependency with Various Debugging Time Lags," *Proceedings of The 28th Annual International Computer Software and Application Conference(COMSAC'04)*, IEEE, 2004.
[17] J. Sall, A. Lehman, "JMP Start Statistics," SAS Institute, Duxbury Press.