http://dx.doi.org/10.5516/NET.04.2012.067

A SOFTWARE RELIABILITY ESTIMATION METHOD TO NUCLEAR SAFETY SOFTWARE

GEE-YONG PARK[1*] and SEUNG CHEOL JANG[2]
[1]Instrumentation & Control and Human Factors Korea Atomic Energy Research Institute
Daedeok-Daero 989-111, Yuseong-Gu, Daejeon, Korea 305-353
[2]Integrated Safety Analysis Korea Atomic Energy Research Institute
Daedeok-Daero 989-111, Yuseong-Gu, Daejeon, Korea 305-353
Corresponding author. E-mail : gypark@kaeri.re.kr

Received September 28, 2012
Accepted for Publication June 13, 2013

A method for estimating software reliability for nuclear safety software is proposed in this paper. This method is based on the software reliability growth model (SRGM), where the behavior of software failure is assumed to follow a non-homogeneous Poisson process. Two types of modeling schemes based on a particular underlying method are proposed in order to more precisely estimate and predict the number of software defects based on very rare software failure data. The Bayesian statistical inference is employed to estimate the model parameters by incorporating software test cases as a covariate into the model. It was identified that these models are capable of reasonably estimating the remaining number of software defects which directly affects the reactor trip functions. The software reliability might be estimated from these modeling equations, and one approach of obtaining software reliability value is proposed in this paper.

KEYWORDS : Software Reliability; Bayesian Inference; Software Reliability Growth Model

1. INTRODUCTION

As digital systems are increasingly being used in nuclear instrumentation and control systems, the reliability of software (S/W) is becoming an important issue as it is software that executes the required functions that a digital safety system should perform.

Hardware faults usually occur due to aging or wearing out, but software faults causing a digital system failure are a design fault introduced during the software development or maintenance phase. Hence, it is recommended that a software safety analysis shall be performed along the software development lifecycle [1] in order to remove hazards from design faults at an early stage. The methods of performing the software safety analysis had been devised and applied to nuclear safety software for a digital reactor protection system (DRPS) [2][3]. All previous works for software safety analysis were based on a qualitative basis.

This paper presents a quantitative reliability assessment of software that has passed through various test phases. Most of the existing software reliability methods have been developed in non-nuclear industries based on the software reliability growth model (SRGM) [4][5]. The SRGM modeling methods have some problems when applied to nuclear safety software. The accuracy of the existing SRGM models is heavily dependent on software failure data. That is, it is hard for the SRGM to make statistical inference on rare event data where the software failure detection frequency is extremely low. It is a prerequisite for the SRGM to produce a reasonably accurate estimate with acceptable uncertainty that an appropriate and sufficient amount of software failure data is to be acquired [6].

Son, et al. proposed a procedure of the software reliability assessment based on the SRGM modeling for safety-critical software [7]. Even though the procedure proposed by Son, et al. provides a systematic way to select a few candidate SRGM models to be used in the probabilistic safety analysis (PSA), the requirement of the sufficient software failure data is inevitable. In the nuclear field, however, software failure data are not accumulated sufficiently resulting in the SRGM model's inability to produce a confident result.

On the contrary, there exists plentiful indirect information on the quality of nuclear safety software in such a way that there can be found sufficient qualitative information about the software quality in the verification and validation (V&V) reports. Or, the indirect quality information can be obtained from a large amount of quantitative test efforts in the V&V test reports.

In order to apply an SRGM modeling method to nuclear safety software that contains rare software failure data, the Bayesian statistical inference is employed in this paper in order to incorporate a covariate such as software test cases into a software reliability model whose analysis foundation is based on the software failure data. The Bayesian statistical inference is performed by the Markov-chain Monte Carlo (MCMC) sampling method in finding an optimal point for model parameters with very rare failure data.

2. DESCRIPTION OF TARGET SOFTWARE AND FAILURE DATA

The target software from which the software failure data are acquired is the trip-functioning software in the bistable processor (BP) software of a digital reactor protection system (DRPS) whose configuration is depicted in Fig.1 [Ref.8]. Each channel of the four-redundant-channel-based DRPS is composed of four main digital processors; i.e., the bistable processor (BP), the coincidence processor (CP), the automatic test and interface processor (ATIP), and the cabinet operator module (COM). The trip functions are performed in the BP and the CP, and the other processors are not directly related with the trip functions. Thus any malfunction in the BP (or the CP) software may induce a significant consequence on the DRPS.

It is very important to reduce the introduction of faults during the software development process because software failures are caused by design faults of the designer or programmer. To circumvent this, independent software V&V activities following a rigorous procedure are performed on the development outputs for the case of nuclear safety software. Fig.2 shows the V&V activities that had been actually performed during the development of the BP software [9]. As can be seen in Fig.2, the main activity in the V&V works after the design phase (i.e., from the implementation phase to the validation phase) is the V&V testing efforts going through the component test (CT), the integration test (IT), and the system test (ST).

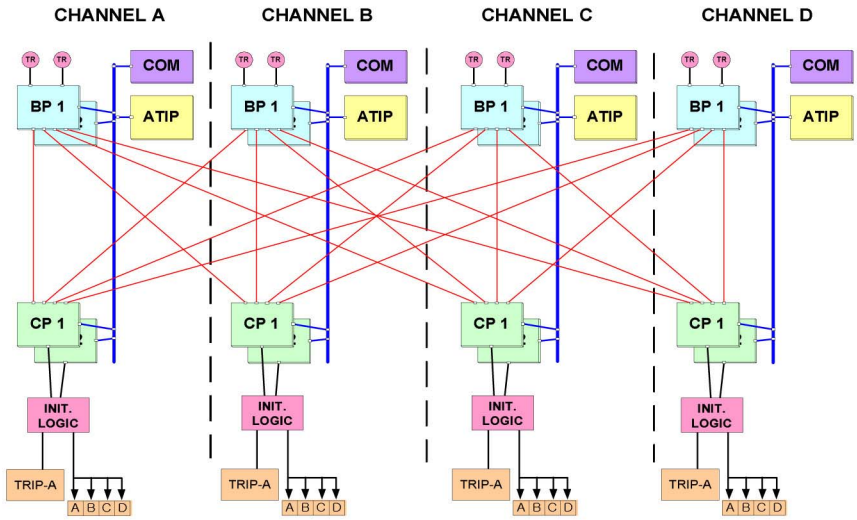


Fig. 1. Configuration of a DRPS

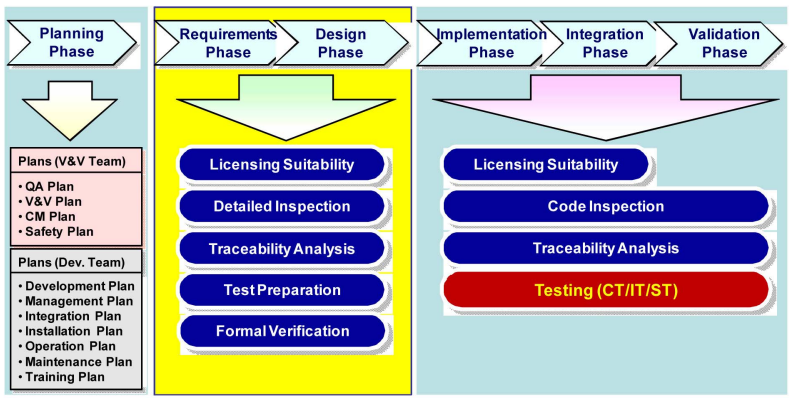


Fig. 2. Software V&V Activities for BP Software

From our experience, both the V&V activities described above and the software safety analysis [2][3] could discover various defects during the software development lifecycle. Specifically, numerous software defects were discovered and corrected at the requirements and design phases by the software V&V activities together with the software safety analysis.

In support of the above argument, according to Gaffney's empirical formulae [10], the number of software bugs (B) can be approximately anticipated when the lines of code (S) for a source program are identified. Table 1 presents the anticipated numbers of bugs in the target software (that is, the trip-functioning software of the BP) that is 13,047 lines of source code (LOC). From Table 1, it is anticipated that a minimum of over 200 software defects are going to be detected during the testing process for the target software.

Actually, by reviewing the test reports, the total number of software failures from both the component and integration tests for the BP trip-functioning software was identified to be 6, as shown in Table 2. Table 2 presents time-based failure occurrences and the number of test cases executed only for the trip-functioning software modules among all the BP software modules. The type of test cases for the BP trip-functioning software is on-demand type.

From Table 1 and Table 2, the frequency of failure occurrences in the BP software is supposed to be extremely lower than that of commercial software in non-nuclear fields. From the above description, it is reasonable to suppose that the V&V activities performed at the phases before the implementation phase is remarkably effective at removing the software defects, otherwise being detected in the V&V test process, if these V&V test activities are performed with a low uncertainty. This also implies that the existing SRGM models are difficult to apply to such infrequent failure occurrences, because although it is not formally validated yet, the conventional SRGM models usually work well for software whose failure occurrences are near the anticipated range of some existing empirical formulae such as Gaffney's empirical formulae. Hence, any other information reflecting the software quality than the failure data is needed and must be incorporated into an SRGM model in order to improve the model's performance when it is employed for the failure behavior as presented in Table 2.

Table 1. Anticipated Software Defects for BP Trip-Functioning Program

Gaffney's Formulae [10]	BP Program (# of LOC: 13,047)
$B = 0.021 \times S$	274
$B = 4 + 0.0014 \times S^{\frac{4}{3}}$	488
$B = 4.2 + 0.0015 \times S^{\frac{4}{3}}$	522

In commercial non-nuclear software, the SRGM models based on the statistical inference are typically used and they usually estimate two quantities, such as the remaining software faults and the software failure rate, as a quantitative measure. For a finite number of software failures collected during a particular test interval with equal length, the SRGM models with the non-homogeneous Poisson process (NHPP) for describing the software failure process are known to be the most famous method. One of the representative methods for these models was described in a paper by Goel and Okumoto [11].

As mentioned in Section 1, it is a prerequisite that a sufficient number of failure data be acquired to accurately estimate the parameters of the SRGM models. Insufficient failure data cause the maximum likelihood estimation to be unable to reach the maximum point, which makes the statistical inference of the SRGM models result in the number of remaining software faults with a high uncertainty. In order to identify the performance of existing SRGM models for the failure data in Table 2, some SRGM modeling methods had been applied to the data in the second row of Table 2.

There are so many modeling methods proposed in the software reliability literature, including over 200 modeling methods for the S/W reliability [4]. Also, many computer-aided tools are developed. Some of the tools widely used in the world are the CASRE (Computer Aided Software Reliability Estimation) tool [12] and the SMERFS III (Statistical Modeling and Estimation of Reliability Functions for Systems) tool [13].

It was identified that most of the existing modeling methods based on the statistical inference could not find the optimal point during the estimation of model parameters. Fig.3 shows application results by SMERFS III. From Fig.3, most models are inappropriate to apply to such rare failure data in Table 2, except the S-T2 (Schnidewind Type #2) method [13]. Though the S-T2 method is able to fit the trend of the software failure data, the uncertainty of the estimation results is too high to be considered a reliable result.

Table 2. Numbers of Test Failures and Test Cases along (Arbitrary) Time-Basis

Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Failure No.	0	0	0	0	2	0	2	0	0	0	0	1	0	0	0	1	0	0	0	0
Test Cases	12	33	29	42	169	1	76	21	74	20	62	40	22	17	15	85	13	30	4	10

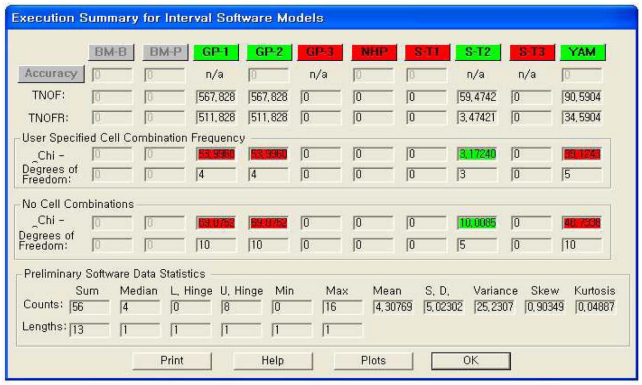


Fig. 3. Application Results of Existing SRGM Models by SMERFS III

Bayesian reliability modeling is able to make inference stably for a rare event as shown later in Section 4. In addition, this method has a very attractive advantage in that it can easily incorporate a so-called covariate into its modeling scheme. The covariate indicates diverse sources of reliability evidence, which includes qualitative information such as the V&V efforts as well as the testing efforts. Regarding this fact, Bayesian modeling is employed for the software reliability estimation in this paper.

Even though the software failure behavior in the BP trip-functioning software is a rare event system, the information of improving software quality is abundant in the V&V reports. This information is also acquired from various test reports. In order to incorporate the plentiful information about the software quality into the estimation procedure for the software reliability, Park, et al. [14] proposed an S/W reliability assessment method in which the V&V inspection results are incorporated into a Bayesian reliability model. In this paper, an NHPP-based SRGM model using the Bayesian inference is employed. As a covariate, independent V&V test cases for the component and integration tests are incorporated into the model, because both the time-based failure data together with the test cases can be identified and arranged from the component and integration test reports.

3. DESCRIPTION OF SOFTWARE RELIABILITY MODELS

In this paper, the distribution of software failures is assumed to follow the non-homogeneous Poisson process. Thus, at an arbitrary time interval $[t_{k-1}, t_k]$, the distribution of a random failure M(k) is represented as

$$\Pr\{M(k) = y_k\} = \frac{m(k)}{y_k!} e^{-m(k)}. \quad (1)$$

In Eq.(1), y_k is the number of failures detected in the interval of $[t_{k-1}, t_k]$ and m(k) is the mean value function of M(k). The mean value function m(k) is in this paper represented by

$$m(k) = r(k-1)\left[1 - \exp\left\{-b(k) \times (t_k - t_{k-1})\right\}\right], \quad (2)$$

where b(k) is the defect discovery rate in the interval of $[t_{k-1}, t_k]$ and r(k) denotes the expected number of defects remaining at time t_k. The r(k) is represented recursively as

$$r(k) = r(k-1)\exp\left\{-b(k) \times (t_k - t_{k-1})\right\}. \quad (3)$$

3.1 Ray, et al.'s Method [15]

In order to incorporate a covariate into the above SRGM model, Ray, et al. [15] proposed that the defect discovery rate b(k) in Eq.(2) or Eq.(3) is formulated by the following log-linear model.

$$\log\left[b(k)\right] = \gamma_0 + \gamma_1 \times TC[k], \quad (4)$$

where γ_0 and γ_1 are the regression coefficients and TC[k] is the number of test cases executed at the time interval of $[t_{k-1}, t_k]$.

Although this method has shown good estimation and prediction results for some particular software failures [15], it was identified in our case that this modeling method could not make a reasonably good result for the data in Table 2. Hence, in order to more accurately estimate the failure behavior in Table 2, we modified the SRGM modeling equations in Eq.(2) and Eq.(3) and applied modified models to the software failure data of Table 2. Among them, two modeling methods, which are called Model #1 and Model #2 in this paper, showed reasonably good estimation results. These two models are presented in the following subsections.

3.2 Model #1

In Model #1, the mean value function m(k) is modified as

$$m(k) = r(k-1)\left\{1 - \exp\left[-b(k) \times \left(d(k) \times \Delta t_k\right)\right]\right\}, \quad (5)$$

where b(k) is the defect discovery rate at the time interval of $[t_{k-1}, t_k]$, d(k) is the term reflecting the test efforts, and $\Delta t_k = t_k - t_{k-1}$.

In Eq.(5), the time domain Δt_k in Eq.(2) is divided into two separate domains, i.e., the domain of test efforts and the domain of testing time. This is similar to the Cobb-Douglas type function [16] which was previously employed in a so-called two-dimensional software reliability assessment. Inoue and Yamada [16] proposed this two-dimensional software reliability model based on the statistical inference and this model is applicable to the so-called "S-curve" software failures (which means software failure data are abundant).

The defect discovery rate b(k) is given by the following simple log-linear model such as

$$\log\left[b(k)\right] = \gamma_0. \quad (6)$$

In Eq.(6), γ_0 is the regression coefficient and thus the defect discovery rate is constant. The test effort term d(k) is formulated by

$$\log[d(k)] = \eta_1 \times TE[k], \quad (7)$$

where η_1 is also the regression coefficient. In Eq.(7), TE[k] indicates the testing efforts. In this model, TE[k] is given by the test cases accumulated from the test start to the current time, t_k.

The expected number of defects remaining at time t_k, r(k), is represented as

$$r(k) = r(k-1)\exp\{-b(k)\times(d(k)\times\Delta t_k)\}.$$

3.3 Model #2

In Model #2, the defect discovery rate of b(k) is also constant. The mean value function is represented by a type of Weibull distribution such as

$$m(k) = r(k-1)\left[1-\exp\left\{-\left(b(k)\times\Delta t_k\right)^{d(k)}\right\}\right], \quad (8)$$

where b(k) is represented by Eq.(6) and d(k) is formulated as

$$\log[d(k)] = \eta_1 \times TC[k], \quad (9)$$

where η_1 is the regression coefficient and TC[k] is the number of test cases executed during $[t_{k-1}, t_k]$.

The expected number of defects r(k) is, in this model, represented as

$$r(k) = r(k-1)\exp\left\{-\left(b(k)\times\Delta t_k\right)^{d(k)}\right\}.$$

4. APPLICATION OF SOFTWARE RELIABILITY MODELS

The SRGM models described in the previous section are implemented by the Bayesian inference tool, called the WinBUGS [17]. The parameters (r(k), γ_0, γ_1) or (r(k), γ_0, η_1) in modeling equations presented in the previous section are estimated by the Bayesian inference.

The WinBUGS program for the modeling equations of Model #1 in Section 3.2 is shown in Fig.4. As can be seen in the last three expressions in Fig.4, the prior distributions of the three model parameters, i.e., (r_0, γ_0, η_1) are all given the Normal distribution ("dnorm") with zero mean and a precision of 0.01 (roughly speaking, the precision is the reciprocal of the variance). These prior distributions indicate the non-informative distribution.

After editing the program as shown in Fig.4, we need to compile this program and run the MCMC algorithm. Fig.5 shows some windows in the WinBUGS program used for this procedure. Some menus in the menu bar and the button functions in the windows are briefly explained below following our estimation procedure.

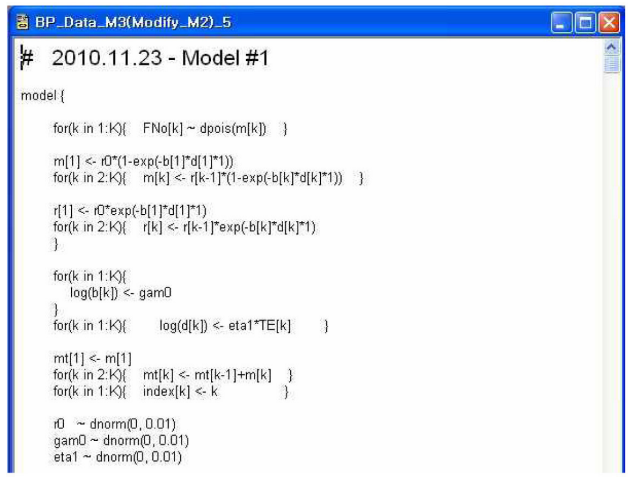


Fig. 4. WinBUGS Program for Modeling Equations of Model #1

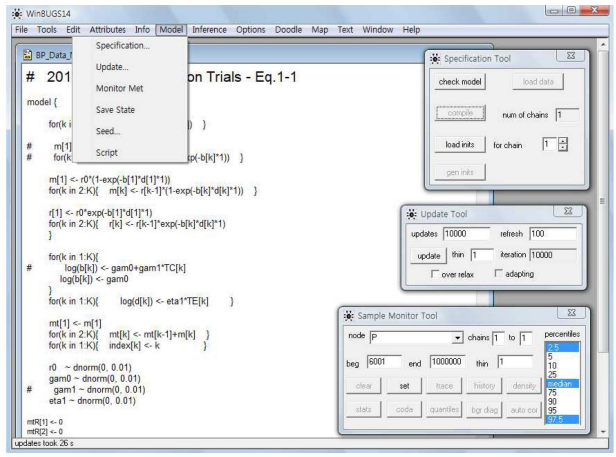


Fig. 5. Compiling and Running Tools in the WinBUGS

When the model is programmed, we first select the "Model" menu in the menu bar and then click the "Specification…" item, resulting in the generation of a small window of "Specification Tool" as can be seen in the upper right portion of Fig.5. In the "Specification Tool" window, the procedure using buttons is as follows:

- We need to press the "check model" button in order to identify whether there is a syntax error in our program,
- We load the data of the failure occurrences and the number of test cases as presented in Table 2 through the "load data" button,
- By clicking the "compile" button, the program is compiled,
- Then initial values for the MCMC algorithm are loaded by clicking the "load inits" button. (in our case, the initial values for model parameters are all set to zero).

Completing the above procedure, the Bayesian inference based on the MCMC sampling algorithm is ready to run.

This is activated by selecting the "Update…" item in the "Model" menu in Fig.5. Then the "Update Tool" window appears as in the middle right area of Fig.5. After inputting a desired sampling value into the field titled "updates" and clicking the "update" button, the Bayesian inference begins. In this case, the Bayesian sampling is updated until the number of updates reaches 10,000 sampling steps as specified in the "updates" editing field.

When the Bayesian inference is completed, the results can be seen through the "Inference" menu in the menu bar that creates the "Sampler Monitor Tool" window when it is selected, as shown in the lower right portion of Fig.5. The results in Fig.6 and Fig.7 below were actually obtained using the functions provided by the "Sampler Monitor Tool" window. The detailed explanation of how to use the WinBUGS menus and their corresponding functions are presented in Ref.17.

Fig.6 shows some of the posterior distributions for the model parameters of Model #1 obtained from the MCMC sampling with the sampling number of 10,000 (only 4,000 samples are used for the posterior distribution, discarding the first 6,000 samples to remove a transient phenomenon during the convergence process of model parameters). The statistical information for the posterior distributions in Fig.6 is presented in Fig.7.

In Fig.7, the first column titled "node" presents the model parameters to be monitored. The second column ("mean") represents the mean value for each of the corresponding parameters. The third column ("sd") means the standard deviation and the fourth column ("MC error") is the errors related to the MCMC sampling algorithm. The last three columns represent the percentile of a distribution.

As can be seen in Fig.7, the expected number of remaining software faults at the end of the tests is 1.324 (r[20] in Fig.7), while the number of remaining software faults is estimated to be 4.878 for the Bayesian modeling without test cases (to save space, this is not shown in this paper). From this result, it is identified that the software quality

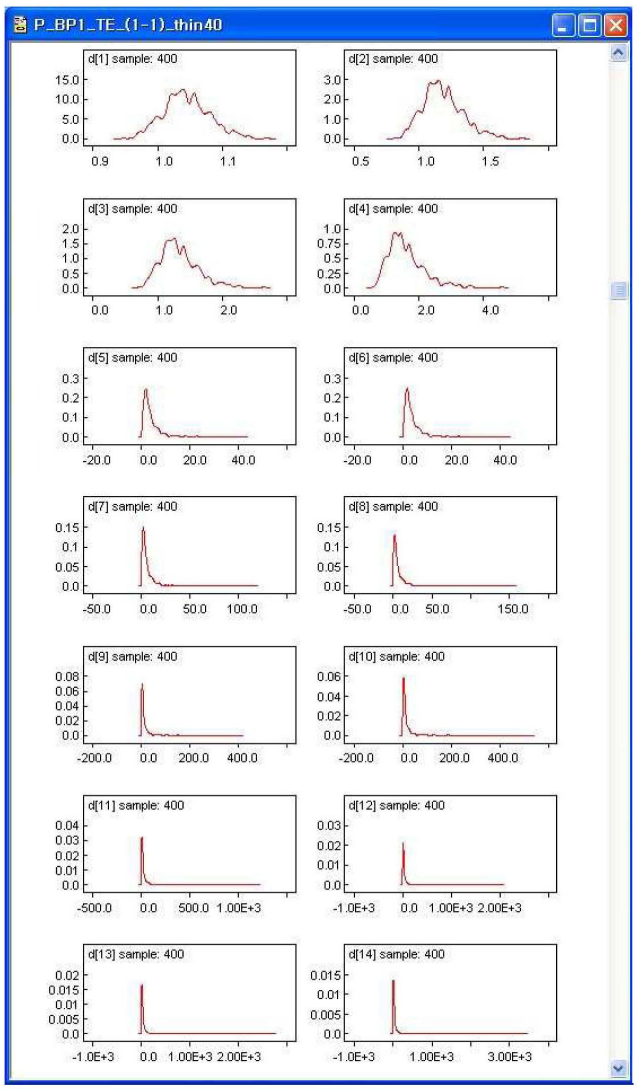


Fig. 6. Posterior Distributions for Model #1 Parameters

S_BP1_TE_(1-1)

node	mean	sd	MC error	2.5%	median	97.5%
P	0.9397	0.08272	0.003098	0.7133	0.9816	1.0
P5	0.9394	0.08324	0.003106	0.7117	0.9813	1.0
b[1]	0.03536	0.03222	0.002045	0.001467	0.0258	0.1184
d[1]	1.046	0.03477	0.00279	0.9813	1.044	1.119
d[2]	1.19	0.1496	0.01233	0.9318	1.175	1.526
d[3]	1.343	0.2805	0.02363	0.8903	1.304	2.004
d[4]	1.618	0.5402	0.04695	0.8334	1.516	2.973
d[5]	3.942	3.678	0.353	0.6392	2.782	14.54
d[6]	3.966	3.716	0.3568	0.6382	2.792	14.67
d[7]	6.373	8.14	0.8056	0.5664	3.667	29.96
d[8]	7.323	10.1	1.007	0.548	3.954	36.49
d[9]	12.27	21.71	2.207	0.4879	5.158	73.12
d[10]	14.2	26.72	2.729	0.4728	5.541	88.23
d[11]	22.75	51.05	5.274	0.4289	6.923	157.9
d[12]	31.25	77.73	8.075	0.4028	7.992	230.0
d[13]	37.37	98.03	10.21	0.3891	8.648	282.7
d[14]	42.99	117.3	12.24	0.3789	9.193	331.7
d[15]	48.72	137.5	14.37	0.37	9.701	381.8
d[16]	101.3	339.9	35.73	0.3238	13.16	848.4
d[17]	113.7	390.6	41.09	0.3173	13.79	958.6
d[18]	148.8	538.6	56.73	0.3027	15.36	1271.0
d[19]	154.3	562.2	59.22	0.3008	15.58	1319.0
d[20]	169.0	625.9	65.95	0.2961	16.15	1449.0
eta1	0.0037	0.002764	2.197E-4	-0.001571	0.00359	0.009392
gam0	-3.799	1.078	0.09007	-6.525	-3.657	-2.134
r[1]	7.364	3.464	0.1142	2.88	6.646	16.48
r[2]	7.101	3.401	0.1102	2.751	6.367	16.12
r[3]	6.837	3.352	0.1082	2.599	6.11	15.64
r[4]	6.563	3.317	0.1081	2.407	5.867	15.38
r[5]	6.138	3.302	0.1112	2.135	5.38	15.02
r[6]	5.751	3.293	0.1142	1.857	4.972	14.85
r[7]	5.288	3.293	0.1177	1.511	4.475	14.61
r[8]	4.845	3.296	0.1206	1.197	3.975	14.21
r[9]	4.323	3.307	0.1235	0.8971	3.366	13.99
r[10]	3.85	3.316	0.1261	0.6435	2.825	13.59
r[11]	3.344	3.33	0.1299	0.3912	2.246	13.21
r[12]	2.884	3.342	0.1348	0.2216	1.711	12.98
r[13]	2.514	3.339	0.1387	0.0998	1.273	12.64
r[14]	2.225	3.319	0.1407	0.03648	0.9393	12.47
r[15]	1.999	3.284	0.1409	0.01336	0.6736	12.26
r[16]	1.779	3.244	0.1408	0.002283	0.4073	12.03
r[17]	1.623	3.192	0.1389	2.853E-4	0.2495	11.87
r[18]	1.499	3.135	0.1364	1.919E-5	0.1409	11.69
r[19]	1.404	3.076	0.1337	7.36E-7	0.08108	11.46
r[20]	1.324	3.018	0.1309	2.404E-8	0.04726	11.23
r0	7.626	3.549	0.1205	2.983	6.902	16.9

Fig. 7. Statistics for Posterior Distributions of Model #1

is increased when the information of the test efforts is incorporated into the SRGM model based on the Bayesian inference in regard to the estimation of the number of accumulated failures, remaining software faults, and so on.

Fig.8 shows the number of accumulated real and estimated failure occurrences along the time index. The table to the left in Fig.8 represents the accumulated number of failure occurrences of Table 2. In Fig.8, the four types of estimations for failure occurrences are displayed, and these four graphs plot the mean values.

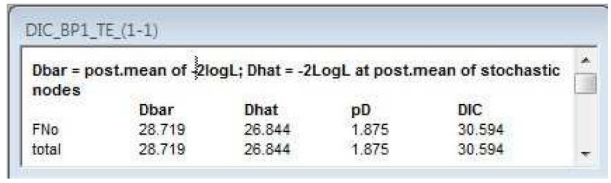
The line with solid diamond symbol with the legend of "Fail_No" represents real software failure occurrences as presented in the table to the left in Fig.8. The line with solid rectangular symbol ("w/o TC") indicates the Bayesian estimation based solely on the software failure data, which means the term including TC[k] in the most right-hand side of Eq.(4) is removed during the Bayesian estimation. The line with the solid triangular symbol ("w/ TC") represents Ray, et al.'s model described in the subsection 3.1. The line with the cross symbol ("Model #1) is for Model #1 and the line with the snow symbol ("Model #2") is for Model #2.

Both methods, the Bayesian inference purely based on the failure data and the Bayesian model in the subsection 3.1, show the inferior performance in that they underestimate the failure occurrences as the testing time elapses. In order words, they are going to more optimistically estimate the software reliability. Ray, et al.'s model estimates the failure trend reasonably well at the initial stage, but as time elapses, this model underestimates the failure trend. It is supposed that this inferiority of Ray, et al.'s model is due to some mismatch between the number of test cases and the software failure occurrences discovered. This mismatch is supposed to be caused by the effect of the V&V activities before the start of the V&V tests, or this may be due to an uncertainty of testing activities such as the right recording of the failure occurrences and test cases at the right time.

The estimation results of Model #1 initially underestimates the failure occurrences, but slightly overestimates as time goes on. Model #2 overestimates the failure occurrences over the entire range of the time index. From the results of Fig.8, Model #1 and Model #2 estimate the trend of real failure occurrences reasonably well.

Fig.9 shows the results of deviance information criterion (DIC) [17] for Model #1 and Model #2, respectively. Also the mean-square error (MSE) values were calculated for both models from the estimation results of Fig.8. For Model #1, the DIC and MSE are 30.594 and 0.136, respectively. The DIC and MSE are 22.674 and 0.122, respectively, for Model #2. From these results, the Model #2 estimates are better than the Model #1 estimates for the software failure data as presented in Table 2.

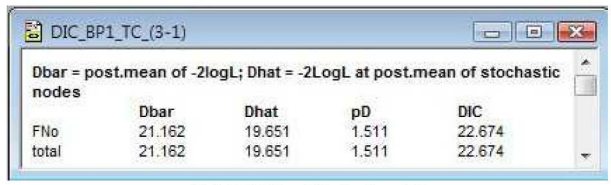
From Fig.7, the number of remaining software defects, r[20], estimated by Model #1 is 1.324. The number of remaining software defects for Model #2 is 1.69 (though not shown in this paper). The fact that the number of remaining software defects is 1.324 or 1.69 implies that additional testing efforts are required for the integration test process, or more rigorous testing activity must be performed in the system test in order to improve the software quality.


DIC_BP1_TE_(1-1)

Dbar = post.mean of -2logL; Dhat = -2LogL at post.mean of stochastic nodes

	Dbar	Dhat	pD	DIC
FNo	28.719	26.844	1.875	30.594
total	28.719	26.844	1.875	30.594

(a) DIC of Model #1


DIC_BP1_TC_(3-1)

Dbar = post.mean of -2logL; Dhat = -2LogL at post.mean of stochastic nodes

	Dbar	Dhat	pD	DIC
FNo	21.162	19.651	1.511	22.674
total	21.162	19.651	1.511	22.674

(b) DIC of Mode #2

Fig. 9. Deviance Information Criterion (DIC) for Two Models

Time Index	Failures Detected
1	0
2	0
3	0
4	0
5	2
6	2
7	4
8	4
9	4
10	4
11	4
12	5
13	5
14	5
15	5
16	6
17	6
18	6
19	6
20	6

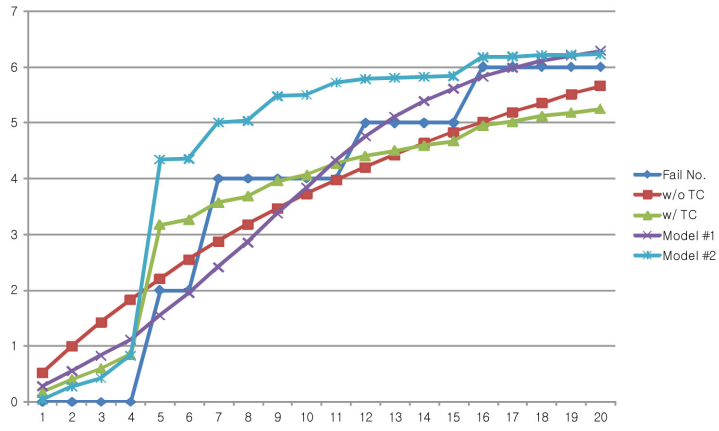


Fig. 8. Estimations of S/W Failure Behavior

The "P" in the second row of Fig.7 represents the value of a type of reliability. The value of P is 0.9397 in Model #1 and 0.9953 in Model #2, respectively. This value is calculated from

$$P = \exp\left\{-r[K]\cdot\left(1-e^{-b[K+1]\cdot(d[K+1]\cdot\Delta t_{K+1})}\right)\right\}. \quad (10)$$

In Eq.(10), the index K indicates the last test time interval (in this case, K=20) and b[K+1] equals b[K] because the defect discovery rate for both models is constant for time intervals of equal length. d[K+1] is obtained from

$$d[K+1] = \exp\left\{\eta_1\cdot(TE[K]+1)\right\}. \quad (11)$$

From Eqs.(10) and (11), the reliability P means the probability that there is no failure when just one more demand of a trip is applied to the trip-functioning software during the next future time interval $[t_{K+1}-t_K]$. Thus if just one single trip condition occurs during the time interval $[t_{K+1}-t_K]$, the probability of actuating the trip function is 0.9397 according to Model #1 and 0.9953 for Model #2, respectively.

5. CONCLUSIONS

In this paper, a quantitative software reliability assessment is described, which is based on the NHPP-based software reliability growth model. For safety software for use in the DRPS, it was identified that the V&V activities before the beginning of the testing phase could reduce software defects significantly, and hence the failure detection during testing phase might become a rare event.

It was identified that most of the existing SRGM modeling methods were not appropriate to apply to this rare failure data. To circumvent this problem, the Bayesian inference is used in the estimation of the SRGM parameters. The number of test cases as a time-varying covariate is incorporated into the Bayesian inference.

The SRGM models based on the Bayesian inference could provide an accurate estimate for the failure trend, resulting in a reduced number of remaining software defects. And the software reliability for a specified future time interval is proposed from the modeling equations. Based on these results, the software quality can surely be represented quantitatively and the current level of the software reliability can be identified.

In addition, when a new dataset is acquired from other tests, e.g., the system test or a real operation, by the aid of the peculiar characteristic of the Bayesian inference, a new posterior distribution is easily calculated from the current posterior distribution using this dataset through the Bayes theorem.

REFERENCES

[1] IEEE Std. 1228, IEEE Standard for Software Safety Plan, Institute of Electrical and Electronics Engineers, 1994.
[2] G. Y. Park, J. S. Lee, S. W. Cheon, K. C Kwon, E. Jee, and K. Y. Koh, "Safety Analysis of Safety-Critical Software for Nuclear Digital Protection System," LNCS (Lecture Notes in Computer Science) 4680, pp.148-161, 2007.
[3] G. Y. Park, K. Y. Koh, E. Jee, P. H. Seong, K. C. Kwon, and D. H. Lee, "Fault Tree Analysis of KNICS RPS Software," Nuclear Engineering and Technology, Vol.40, No.5, pp.397-408, 2008.
[4] Ch. A. Asad, M. I. Ullah, and M. J. Rehman, "An Approach for Software Reliability Model Selection," Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04), Hong Kong, China, September 27-30, 2004.
[5] IEEE Std. 1633-2008, IEEE Recommended Practice on Software Reliability, Institute of Electrical and Electronics Engineers, 2008.
[6] M. C. Kim, S. C. Jang, and J. Ha, "Possibilities and Limitations of Applying Software Reliability Growth Models to Safety-Critical Software," Nuclear Engineering and Technology, Vol.39, No.2, pp.145-148, 2007.
[7] H. S. Son, H. G. Kang, and S. C. Jang, "Procedure for Application of Software Reliability Growth Models to NPP PSA," Nuclear Engineering and Technology, Vol.41, No.8, pp.1065-1072, 2009.
[8] J. H. Park, D. Y. Lee, and C. H. Kim, "Development of KNICS RPS Prototype," Proceedings of ISOFIC (International Symposium on Future I&C) 2005, Tongyeong, Korea, Nov. 1~4, Session 6, pp. 160-161, 2005.
[9] G. Y. Park, S. W. Cheon, K. C. Kwon, K. Y. Koh, P. H. Seong, E. Jee, and S. Cha, "Software Qualification Activities for Safety Critical Software," Sixth American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC&HMIT 2009), Knoxville, Tennessee, U.S.A, April 5-9, 2009. (CD-ROM).
[10] J. E. Gaffney, Jr., "Estimating the Number of Faults in Code," IEEE Transactions on Software Engineering, Vol.SE-10, pp.459-464, 1984.
[11] A. L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures," IEEE Transactions on Reliability, Vol.R-28, No. 3, pp.206-211, 1979.
[12] CASRE 3.0, http://www.openchannelfoundation.org/projects/CASRE_3.0.
[13] SMERFS III, http://www.slingcode.com/smerfs.
[14] G. Y. Park, H. S. Eom, S. C. Jang, and H. G. Kang, "Software Failure Probability Assessment by Bayesian Inference," Nuclear Technology, July, (to be published) 2013.
[15] B. K. Ray, Z. Liu, and N. Ravishanker, "Dynamic Reliability Models for Software Using Time-Dependent Covariates," TECHNOMETRICS, Vol.48, No.1, pp.1-10, 2006.
[16] S. Inoue and S. Yamada, "Two-Dimensional S/W Reliability Assessment with Testing-Coverage," The 2nd Int. Conf. on Secure System Integration and Reliability Improvement, 2008. (DOI 10.1109/SS1RI.2008.21 IEEE)
[17] D. Spiegelhalter, A. Thomas, N. Best, and D. Lunn, WinBUGS User Manual, Ver. 1.4, January, 2003. (http://www.mrc-bsu.cam.ac.kr/bugs)