Journal of the Korean Institute
of Industrial Engineers
Vol. 15, No. 2, Dec., 1989

A Comparison of Software Reliability Models[†]

Chi-Hyuck Jun *

소프트웨어 신뢰성 모형의 비교에 관한 연구[†]

전 치 혁*

Abstract

A general software reliability model is developed, which includes the Jelinski-Moranda model, the Goel-Okumoto model, the Shanthikumar model and the Ross model as special cases. In each of above models estimators of the software failure rate and the number of remaining errors are presented and compared in terms of the expected absolute error loss and the expected squared error loss by a Monte Carlo simulation.

1. Introduction

The importance of quality assurance of software and software reliability has been recognized recently as computer and software industries grow. In accordance with this, many software reliability models are developed to predict software performance. See Kim[3], Musa, Iannino and Okumoto [5], and Shanthikumar[8] for a review of software reliability models. In most studies software failure rate and/or the number of remaining errors are usually considered as performance measures. Therefore the study on software reliability is reduced to the estimation of these performance measures. However, ther are few attempt to compare these models with each other by investigating behavior of the estimating measures. This is important in practice where we should decide which model to choose.

In this paper we present a general

* Department of Industrial Engineering Phang Institute of Science and Technology

† This work was supported by KOSEF Grant No. 883-0915-012-1.

software reliability model which includes some well-known models such as the Jelinski-Moranda model and the Goel-Okumoto model, and derive the software failure rate and the number of remaining erros as performance measures in each model. Then we perform a Monte Carlo simulation to obtain estimates of these measures and compare those existing models.

2. The Model

Suppose that a software under development initially contains m unknown number of errors and that error i will cause failures independently of other errors in accordance with a nonhomogeneous Poisson process with intensity function $\lambda_i(s)$, $i=1, 2,\cdots,m$. Suppose also that when a failure occurs, the error causing that failure will be identified and it will be removed instantly. That is, one error at a time will be debugged. We are interested in estimating the (revised) software failure rate and the number of remaining erros after testing the software for t time period.

Let us define for $i=1, 2,\cdots,m$,

$$I_i(t) = \begin{cases} 1 & \text{if error i has not caused a failure by time t} \\ 0 & \text{otherwise.} \end{cases}$$

Then the software failure rate at time t, K(t), is given by

$$K(t)=\Sigma_{i=1}^{m} \lambda_i(t)\, I_i(t),$$

and the number of remaining errors at time t, U(t), can be expressed by

$$U(t)=\Sigma_{i=1}^{m} I_i(T).$$

Note that the expectations of K(t) and U(t) are evaluated, respectively, by

$$\begin{aligned} E[K(t)] &= \Sigma_{i=1}^{m} \lambda_i(t)\, E[I_i(t)] \\ &= \Sigma_{i=1}^{m} \lambda_i(t) \exp(-G_i(t)) \end{aligned} \qquad \cdots\cdots\cdots\cdots (1)$$

and

$$E[U(t)] = \Sigma_{i=1}^{m} \exp(-G_i(t)), \qquad \cdots\cdots\cdots\cdots (2)$$

where for $i=1,\cdots,m$,

$$G_i(t)=\int_0^t \lambda_i(x)dx.$$

If N(t) is defined as the number of erros detected by time t with $N(0)=0$, then $\{N(t), t \geq 0\}$ is a counting process and its expectation is given by

$$\begin{aligned} E[N(t)] &= m-\Sigma_i E[I_i(t)] \\ &= m-\Sigma_i \exp(-G_i(t)). \end{aligned} \qquad \cdots\cdots\cdots\cdots (3)$$

The probability distribution function of N(t), $P_n(t)=Pr\{(N(t)=n\}$, can be obtained from the corresponding forward Kolmogorov equations. If we denote failure rates of the detected error by $\Lambda_1(t),\cdots,\Lambda_n(t)$ when $N(t)=n$, and software failure rate by $k_n(t)$, which is defined as

$$\begin{aligned} k_n(t) &= E[K(t) \mid N(t)=n] \\ &= \Sigma_{i=1}^{m} \lambda_i(t)-\Sigma_{i=1}^{n} \Lambda_i(t), \end{aligned} \qquad \cdots\cdots\cdots\cdots (4)$$

then we have the following differential equations:

$$\frac{dP_0(t)}{dt} = -\Sigma_{i=1}^{m} \lambda_i(t)\, P_0(t) \qquad \cdots\cdots\cdots\cdots (5)$$

$$\frac{dP_n(t)}{dt} = k_{n-1}(t)P_{n-1}(t)-k_n(t)P_n(t),\ 1 \leq n \leq m,$$

with the boundary conditions $P_n(0)=0$, $n \geq 1$ and $P_0(0)=1$.

Let X_j be the time between failures (j-1) and j, and S_j be the time to the j-th failure, that is,

$$S_j = S_{j-1} + X_j = \Sigma_{i=1}^{j} X_i,$$

with $S_0=0$. Then the joint probability density function of $S_1, S_2, \cdots, S_n$, $f(s_1,\cdots,s_n)$, can be obtained by

$$f(s_1,\cdots,s_n) = \Pi_{j=1}^{n} k_{j-1}(s_j)\ \exp\{-(H_{j-1}(s_j) - H_{j-1}(s_{j-1}))\}, \quad \cdots\cdots\cdots\cdots\cdots\cdots (6)$$

where

$$H_j(s) = \int_0^s k_j(x)dx.$$

When $\lambda_1(t) = \cdots = \lambda_m(t) = \phi(t)$, our model is reduced to the Shanthikumar model[7]. Shanthikumar also shows that when $\phi(t) = \alpha b \exp(-bt)$ with parameters α and b, it is equivalent to the Goel-Okumoto model as $m \to \infty$, $\alpha \to 0$, $m\alpha \to a$, and $m\alpha^k \to 0$ for all $k>1$. Goel and Okumoto[1] considers $\{N(t),\ t \geq 0\}$ as a nonhomogeneous Poisson process with intensity function of $\mu(t) = ab\ \exp(-bt)$. It is clear that when $\phi(t) = \lambda$, we have the Jelinski-Moranda model. If we assume that $\lambda_i(t) = \lambda_i$, $i=1,\cdots,m$, then we will have the Ross model[6].

In this paper we are interested in estimating the software failure rate given n failures by time t, that is, $k_n(t)$ of Eq.(4), and the number of remaining errors at time t, $u_n(t)$, which is defined by

$$u_n(t) = E[U(t) \mid N(t)=n] = m-n. \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (7)$$

3. Estimators of Interest in Existing Models

3-1. Jelinski-Moranda Model

This model is introduced by Jelinski and Moranda[2], which is one of the earliest software reliability models. They assume that each error has the identical failure rate, that is, $\lambda_1(t) = \cdots = \lambda_m(t) = \lambda$. Therefore the conditional expectation of K(t) given N(t)=n in Eq.(4) is reduced to

$$k_n(t) = (m-n)\lambda. \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (8)$$

Solving differential equations in Eq.(5) for this model, we can show that the distribution of N(t) is binomial:

$$P_n(t) = \binom{m}{n}(1-e^{-\lambda t})^n (e^{-\lambda t})^{m-n},\ 0 \leq n \leq m.$$

Noting that $H_j(s) = (m-j)\lambda s$ in this model and $S_j - S_{j-1} = X_j$, from Eq.(6) the log-likelihood function for m and λ given data $x_1,\cdots, x_n$ of n inter-failure times is

$$L(m, \lambda \mid x_1,\cdots,x_n) = \Sigma_{j=1}^{n} \{\ln((m-j+1)\lambda) - (m-j+1)\lambda x_j\}. \quad \cdots\cdots\cdots\cdots\cdots\cdots (9)$$

Therefore the maximum likelihood estimates(MLE) of m and λ will be obtained by solution of the following two equations;

$$\Sigma_{j=1}^{n} 1/(m-j+1) - \Sigma_{j=1}^{n} \lambda x_j = 0 \quad \cdots\cdots (10a)$$

and

$$n/\lambda - \Sigma_{j=1}^{n} (m-j+1)x_j = 0. \quad \cdots\cdots\cdots (10b)$$

Given the MLE of m and λ, $\hat{m}$ and $\hat{\lambda}$, the software failure rate, $k_n(t)$, and the remaining number of errors at time t, $u_n(t)$, can be estimated by δ_{JM} and η_{JM}, respectively, as follows when $N(t)=n$;

$$\delta_{JM}=(\hat{m}-n)\hat{\lambda}, \quad (11)$$

and

$$\eta_{JM}=\hat{m}-n. \quad (12)$$

3-2. Moranda Model

Moranda[4] modified the Jelinski-Moranda model by introducing a geometrically decreasing hazard function. That is, he assumed the software failure rate given $N(t)=n$ as follows;

$$k_n(t)=Dr^n, \quad (13)$$

where D and r are two model parameters to be estimated. Note that Eq.(13) is not a special case of Eq.(4) except $n=0$. But comparing Eq.(4) and Eq.(13) when $n=0$, we notice that D can be regarded as the initial software failure rate.

Since $H_j(t)=Dr^j t$ for this model, the log-likelihood function of D and r given data $x_1,\cdots,x_n$ is reduced from Eq.(6) to

$$L(D, r \mid x_1,\cdots,x_n)=\Sigma_{j=1}^{n}\{\ln(Dr^{j-1})-Dr^{j-1}x_j\},$$

and the MLE of D and r are solution of two equations shown below:

$$n/D-\Sigma_{j=1}^{n} r^{j-1}x_j=0 \quad (14a)$$

and

$$\Sigma_{j=1}^{n}(j-1)-D\Sigma_{j=1}^{n}(j-1)r^{j-1}x_j=0. \quad (14b)$$

If $\hat{D}$ and $\hat{r}$ denote the MLE of D and r, respectively, then the estimate of the software failure rate at time t given $N(t)=n$ is obtained by

$$\delta_M=\hat{D}\,\hat{r}^n. \quad (15)$$

But an estimate of the number of remaining errors is not available since the initial error content, m, can not be estimated directly.

3-3. Shanthikumar Model

Shanthikumar[7] developed a Markov process model which includes the Jelinski-Moranda model and the Goel-Okumoto model as special cases. He assumed that $\lambda_1(t)=\cdots=\lambda_m(t)=\phi(t)$ and he chose $\phi(t)=\alpha b\exp(-bt)$. Therefore the software failure rate at time t given $N(t)=n$ will be

$$k_n(t)=(m-n)\alpha b\exp(-bt). \quad (16)$$

He showed by solving Eq.(5) that the distribution of N(t) is binomial given below:

$$P_n(t)=\binom{m}{n}(1-\exp(-G(T)))^n$$
$$(\exp(-G(t))^{m-n},\ 0\le n\le m,$$

where $G(t)=\int_0^t\phi(x)dx=\alpha(1-e^{-bt})$.

From Eq.(6) with $H_j(t)=(m-j)\alpha(1-\exp(-bt)$, the log-likelihood function of m, α and b given data $s_1,\cdots,s_n$ of n software failure times will be

$$L(m,\alpha,b\mid s_1,\cdots,s_n)$$
$$=\Sigma_{j=1}^{n}\ln(m-j+1)+n\ln(\alpha b)-b\,\Sigma_{j=1}^{n}s_j$$
$$-\Sigma_{j=1}^{n}(m-j+1)\alpha\{\exp(-bs_{j-1})-\exp(-bs_j)\}.$$

Hence the MLE of parameters m, α, and b will be obtained by solution to the following three equations:

$n/\alpha-\Sigma_{j=1}^{n}(m-j+1)\{\exp(-bs_{j-1})$
$-\exp(-bs_j)\}=0,$ ·················· (17a)

$n/b-\Sigma s_j-\Sigma(m-j+1)\alpha\{s_j\exp(-bs_j)$
$-s_{j-1}\exp(-bs_{j-1})\}=0,$ ················· (17b)

and

$\Sigma_{j=1}^{n}1/(m-j+1)-\alpha\{1-\exp(-bs_n)\}=0.$
·· (17c)

The quantities of $k_n(t)$ and $u_n(t)$ can be estimated, respectively, by

$\delta_S(t)=(\hat{m}-n)\hat{\alpha}\hat{b}\exp(-\hat{b}t)$ ·············· (18)

and

$\eta_S(t)=\hat{m}-n,$ ······························ (19)

where $\hat{m}$, $\hat{\alpha}$ and $\hat{b}$ are the MLE of m, α and b, respectively.

3-4. Goel-Okumoto Model

Goel and Okumoto[1] assume that the process $\{N(t), t\geq0\}$ is a nonhomogeneous Poisson process(NHPP) with the intensity function $\mu(t)$ as follows:

$\mu(t)=abe^{-bt},$

where a and b are unknown parameters. Therefore the distribution of N(t) is Poisson with mean $m(t)=a(1-\exp(-bt))$, that is,

$$P_n(t)=\frac{(m(t))^n\exp(-m(t))}{n!},\quad n=0,1,2,\cdots$$

The quantity of our interest will also be

$k_n(t)=abe^{-bt},$ ························· (20)

Notice that the software failure rate is unaltered by error occurrences. We can regard parameters a and b as the initial error content and the faiure rate of each error, respectively. when $m=a$ and $\lambda_i(s)=b$ for all i, $E[K(t)]$ of Eq.(1) is same as $k_n(t)$ in this model.

From Eq.(6) with $H_j(t)=a(1-\exp(-bt))$, the log-likelihood function of a and b given data $s_1,\cdots,s_n$ of n software failure times is given by

$L(a,b\mid s_1,\cdots,s_n)=n\ \ln(a)+n\ \ln(b)$
$-a(1-\exp(-bs_n))-b\ \Sigma_j s_j,$

and the MLE of parameters a and b can be calculated from the two equations given below:

$n/a=1-\exp(-bs_n)$ ····················· (21a)
$n/b=\Sigma_{j=1}^{n}s_j+as_n\exp(-bs_n).$ ········ (21b)

Therefore the estimates for the software failure rate and the number of remaining errors at time t will be

$\delta_{GO}(t)=\hat{a}\hat{b}\exp(-\hat{b}t)$ ················ (22)
$\eta_{GO}=\hat{a}-n,$ ······························ (23)

where $\hat{a}$ and $\hat{b}$ are MLE of a and b, respectively.

3-5. Ross Model

Ross[6] assumed that errors have different failure rates from each other and that each rate is unchanged over time, that is, $\lambda_i(s)=\lambda_i$, $i=1,\cdots,m$. If we assume that the failure rate of an error becomes known once the error has been detected and let $\Lambda_1,\cdots,\Lambda_n$ be failure rates of detected errors when $N(t)=n$ by time t, his estimator of the software failure rate at time t is given by

$$\delta_o(t) = \Sigma_{i=1}^{N(t)} \frac{\Lambda_i \exp(-\Lambda_i t)}{1-\exp(-\Lambda_i t)}$$

Notice that

$$E[\delta_o(t)] = E[K(t)] = \Sigma_{i=1}^{m} \lambda_i e^{-\lambda_i t},$$

and therefore $\delta_o(t)$ is an estimator of E[K(t)] actually.

In this model $k_n(t)$ of Eq.(4) is reduced to

$$k_n(t) = \Sigma_{i=1}^{m} \lambda_i - \Sigma_{i=1}^{n} \Lambda_i,$$

which can also be estimated by $\delta_o(t)$. He also suggested that Λ_i, i=1,…,n can be estimated by $1/S_i$. Therefore the software failure rate at time t can be estimated by

$$\delta_R(t) = \Sigma_{i=1}^{n} \frac{\exp(-t/s_i)}{s_i(1-\exp(-t/s_i))}, \quad \cdots\cdots\cdots (24)$$

where s_i, i=1,…,n are data of n failure times.

The estimator of his type for the number of remaining errors at time t can be given by

$$\eta_o(t) = \Sigma_{i=1}^{N(t)} \frac{\exp(-\Lambda_i t)}{1-\exp(-\Lambda_i t)}.$$

Note that $E[\eta_o(t)] = E[U(t)] = \Sigma_{i=1}^{m} \exp(-\lambda_i t)$. Therefore the number of remaining errors at time t when the data $s_1,\cdots,s_n$ are available can be estimated by

$$\eta_R(t) = \Sigma_{i=1}^{n} \frac{\exp(-t/s_i)}{1-\exp(-t/s_i)}. \quad \cdots\cdots\cdots\cdots (25)$$

4. A Numerical Example

We again analyze the NTDS(Naval Tactical Data System) data which has been considered by Jelinski and Moranda[2] and by Goel and Okumoto[1]. We fit this software failure data to each model in Sec. 3, estimate relevant parameters, and compare models by computing the software failure rate and the number of remaining errors. The NTDS data is shown in Table 1.

Table 1. NTDS Software Failure Data

failure no. j	inter-failure times Xj (day)	time to jth failure Sj (day)
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

We obtain parameters estimates of each model by solving simultaneous equations numerically and compute quantities of our interest after the 26th failure has been detected (t=250). We use the ZSCNT routine of IMSL[9] in a program with FORTRAN language. Table 2 summarizes the result.

We observe from Table 2 that the estimated software failure rate is largest in the Moranda model and the smallest in the Ross model and that the estimated number of remaining errors is largest in the Goel-Okumoto model and the smallest in the Ross model. We also notice that the Jelinski-Moranda model and the Shanthikumar model estimate both measures almost identically. Note that in the NTDS data eight more errors were detected after the 26th failure. Hence in this example the Goel-Okumoto model precisely estimates the number of remaining errors.

5. A Monte Carlo Simulation

One approach to investigating whether one estimator is better than the other is the decision-theoretic one with some error loss, which evaluates and compared the risk of each estimator. Here we will consider the two kinds of error loss functions, that is, the absolute error loss and the sqared error loss. Hence we want to evaluate the expected absolute error $R_1(\delta)$ and the expected squared error $R_2(\delta)$ associated with the estimator of the software failure rate δ. Similarly we want $R_1(\eta)$ and $R_2(\eta)$ associated with the estimator of number of remaining failure rate η. These quantities are expressed by

$$R_1(\delta)=E[\,|\,\delta-k_n(t)\,|\,] \quad \cdots\cdots\cdots\cdots\cdots (26)$$

$$R_2(\delta)=E[\,(\,\delta-k_n(t))^2] \quad \cdots\cdots\cdots\cdots\cdots (27)$$

Table 2. Estimation Results by Models

Model	Parameters estimates	failure rate (δ)	no. of remaining errors (η)
Jelinski-Moranda	$\hat{m}$ = 31.22		
	$\hat{\lambda}$ = 0.00685	0.0357	5.22
Moranda	$\hat{D}$ = 0.2020		
	$\hat{r}$ = 0.95471	0.0605	-
Shanthikumar	$\hat{m}$ = 31.40		
	$\hat{\alpha}$ = 13.71		
	$\hat{b}$ = 0.00051	0.0335	5.40
Goel-Okumoto	$\hat{a}$ = 33.99		
	$\hat{b}$ = 0.00579	0.0463	7.99
Ross		0.0199	3.10

$R_1(\eta)=E[\,|\,\eta-u_n(t)\,|\,]$ .... (28)

$R_2(\eta)=E[(\eta-u_n(t))^2]$, ... (29)

where n is considered as a random quantity. But it is almost impossible to evaluate them analytically. Therefore we will pursue this by a computer simulation.

5-1. Simulation Procedure

In simulation we assume that error i causes a failure according to a Poisson process with rate λ_i, $i=1,\cdots,m$(λ_i's may be equal) independently with other errors. Our procedure is as follows:

Step 1. Choose m and λ_i's.

Step 2. Generate the time to failure caused by each error and obtain n, the number of errors detected by time t. Note that the time to failure caused by error i is exponentially distributed with mean $1/\lambda_i$. Then we obtain the data s_j's by sorting n failure times in ascending order. Next we obtain $k_n(t)$ and $u_n(t)$, and calculate the estimate of the software failure rate δ and the estimated remaining erros η for each model. Then we compute $|\,k_n(t)-\delta\,|$, $(k_n(t)-\delta)^2$, $|\,u_n(t)-\eta\,|$ and $(u_n(t)-\eta)^2$.

Step 3. Repeat Step 2 N times and obtain an estimate of $R_1(\delta)$ of Eq.(26) by averaging $|\,k_n(t)-\delta\,|$ over N and an estimate of $R_2(\delta)$ of Eq.(27) by averaging $(k_n(t)-\delta)^2$ over N. Obtain estimates of $R_1(\eta)$ and $R_2(\eta)$ analogously.

Without loss of generality we assume that $t=1$ in this simulation. Also in Step 3, we choose $N=100$. In step 2, we solve nonlinear simultaneous equations by using the ZSCNT routine of IMSL which solves equations iterativley until some conditions are met. Due to the computation time we limit the number of iteration to 1,000 and set fnorm to 0.0001, which is defined by

$\text{fnorm}=\Sigma_i f_i^2(X)$,

where $f_i(x)$'s are equations to be solved and $\mathbf{x}$ is a vector of unknowns.

5-2. Simulation Results

We consider six cases(case I to case VI) with different number of errors(m) and failure rates(λ_i's) assumed. Case I assumes $m=10$ and the equal failure rates of errors which are set to 0.5 and the Case II assumes $m=10$ and failure rates of errors equally as 1.0. Case III and IV assume $m=20$ and equal failure rates 0.5 and 1.0, respectivley. In Case V and VI, $m=20$ is chosen but failure rates are assumed differently among errors, that is, they are chosen randomly from the interval(0.1, 1.0) and (0.1, 2.0), respectively. Table 3 summarizes the simulation results of all six cases.

Note that for each case 100 replications were made but some of them were deleted in calculating estimates since they failed in convergence during solving nonlinear simultaneous equations. Particularly in the Shanthikumar model convergence problem frequently occurs. It should be mentioned that this simulation is quite time-consuming. Simulation of one case requires about 2 hours in CPU time at the VAX

Table 3. Summary of Simulation Results

Case No.	initial no. of errors and failure rates	avg. no. of errors detected and true values	Average Estimates Values and Expected Error Losses					
				J-M	Shanthi	G-M	Moranda	Ross
I	$m=10$	$n=3.99$						
	$\lambda=0.5$	$k_n(t)=3.01$	$E[\delta]=$	4.24	3.06	4.23	5.31	1.03
			$R1(\delta)=$	1.93	1.67	1.89	3.70	2.02
			$R2(\delta)=$	5.74	4.85	5.24	34.02	5.30
		$u_n(t)=6.01$	$E[\eta]=$	13.29	33.71	17.13	-	0.71
			$R1(\eta)=$	8.11	32.21	11.32	-	5.30
			$R2(\eta)=$	130.26	2359.27	218.92	-	31.27
II	$m=10$	$n=6.18$						
	$\lambda=1.0$	$k_n(t)=3.82$	$E[\delta]=$	5.55	2.88	5.61	6.55	1.41
			$R1(\delta)=$	2.75	2.09	2.47	4.38	2.47
			$R2(\delta)$	12.55	6.91	9.06	37.29	9.01
		$u_n(t)=3.82$	$E[\eta]=$	11.31	12.39	15.26	-	0.93
			$R1(\eta)=$	8.47	11.90	11.53	-	2.89
			$R2(\eta)=$	143.71	691.65	185.42	-	11.10
III	$m=20$	$n=7.58$						
	$\lambda=0.5$	$k_n(t)=6.21$	$E[\delta]=$	7.37	5.24	7.24	10.24	1.97
			$R1(\delta)=$	2.42	2.54	2.23	5.80	4.29
			$R2(\delta)=$	9.01	9.25	8.36	67.01	21.22
		$u_n(t)=12.42$	$E[\eta]=$	25.78	31.29	30.05	-	1.35
			$R1(\eta)=$	14.87	28.76	18.20	-	11.12
			$R2(\eta)=$	382.28	2040.63	524.20	-	130.30
IV	$m=20$	$n=12.37$						
	$\lambda=1.0$	$k_n(t)=7.63$	$E[\delta]=$	9.76	6.15	9.99	9.69	2.80
			$R1(\delta)=$	3.51	3.23	3.24	4.31	4.90
			$R2(\delta)=$	19.80	19.17	16.70	29.04	29.56
		$u_n(t)=7.63$	$E[\eta]=$	19.47	6.09	22.31	-	1.84
			$R1(\eta)=$	13.23	4.78	14.90	-	5.81
			$R2(\eta)=$	358.17	66.59	345.65	-	38.90
V	$m=20$	$n=9.69$						
	$\lambda=(0.1,\ 1.0)$	$k_n(t)=6.22$	$E[\delta]=$	8.99	5.58	8.69	10.96	2.49
			$R1(\delta)=$	3.59	2.58	3.43	6.18	3.78
			$R2(\delta)=$	19.21	10.83	16.65	72.83	19.02
		$u_n(t)=10.31$	$E[\eta]=$	25.24	7.43	28.26	-	1.67
			$R1(\eta)=$	15.73	7.20	18.54	-	8.64
			$R2(\eta)=$	475.00	134.32	556.81	-	82.29
VI	$m=20$	$n=12.91$						
	$\lambda=(0.1,\ 2.0)$	$k_n(t)=6.67$	$E[\delta]=$	9.62	5.42	9.70	9.60	2.77
			$R1(\delta)=$	4.70	3.21	4.46	4.76	4.05
			$R2(\delta)=$	30.38	16.91	27.21	34.46	23.65
		$u_n(t)=7.09$	$E[\eta]=$	18.04	4.56	19.46	-	1.78
			$R1(\eta)=$	13.13	3.97	13.64	-	5.32
			$R2(\eta)=$	338.33	23.40	317.59	-	34.58

-8800.

We notice in Table 3 that almost all models give quite good estimates for the software failure rate but not good ones for the remaining number of errors. The Shanthikumar model and the Ross model tend to underestimate the software failure rate and the rest models tend to overestimate it. When failure rates of errors are assumed as different with each other, the Shanthikumar model surprisingly shows good performance on estimating the software failure rate and the remaining number of errors. However the Shanthikumar model gives very poor results on estimating the remaining number of erros when equal(particularly small) failure rates of errors are assumed. We can also see that the Jelinksi-Moranda model and the Goel-Okumoto model perform almost equally although the Goel-Okumoto model estimates the software failure rate slightly better than the Jelinski-Moranda model does and the Jelinski-Moranda model shows slight better performance on estimating the number of remaining errors. The Ross model has an advantage of not requiring nonlinear equations to calculate estimates but unfortunately it severely underestimates both performance measures. Some cure should be needed in further research.

6. Concluding Remarks

We present a general software reliability model which includes existing models such as the Jelinski-Moranda model, the Goel-Okumoto model, the Shanthikumar model, and the Ross model. In addition, some other models which assume the software failure rate as special cases of Eq.(4) can be included in model. Furthermore the joint probability density function of $S_1, S_2, \cdots, S_n$ given in Eq.(6) will be valid even when the software failure rate $k_n(t)$ cannot be written by Eq.(4).

Simulation results presented here are not quite extensive but they show that there is no dominating model in estimating the software failure rate and the number of remaining errors. Particularly a further study should be necessary in estimation of the number of remaining errors.

References

1. Goel, A.L. and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures", *IEEE Transactions on Reliability,* Vol. R-28, 206-211, 1979.
2. Jelinski, Z. and P. Moranda, "Software Reliability Research", in W. Freiberger(ed.), *Statistical Computer Performance Evaluation,* Academic Press, 465-484, 1972.
3. Kim, Y.H., "Software Reliability: A State-of-the-Art Survey", *Journal of the Korean Institute of Industrial Engineers,* Vol. 13, No. 1, 39-52, 1987.

4. Moranda, P.B., "Event-Altered Rate Model for General Reliability Analysis", *IEEE Transactions on Reliability,* Vol. R-28, 376-381, 1979.

5. Musa, J.D., A. Iannino, and K. Okumoto, *Software Reliability-Measurement, Prediction, Application,* McGraw-Hill Book Co., New York, 1987.

6. Ross, S.M., "Statistical Estimation of Software Reliability", *IEEE Transactions on Software Engineering,* Vol. SE-11, 479-483, 1985.

7. Shanthikumar, J.G., "A General Software Reliability Model for Performance Prediction", *Microelectronics and Reliability,* Vol. 21, 671-682, 1981.

8. Shanthikumar, J.G., "Software Reliability Models: A Review", *Microelectronics and Reliability,* Vol. 23, 903-943, 1983.

9. IMSL, Inc., IMSL Library-User's Manual, Vol. 4, 1984.