A Software Reliability Growth Model Considering Imperfect Debugging and Disagreement between Operation Environments

Ji Won Pak^{1}*, Kwang Chol Kim^{2}, Kwang Song Han^{3}

*Faculty of Mathematics, **Kim Il Sung** University, Taesong District, Pyongyang, DPR Korea*

†*The corresponding author. Email address: gc.Kim@star-co.net.kp*

SUMMARY

Many software reliability growth models are proposed to be used in practice. However, most software reliability growth models suffer in the realistic software testing environment due to the unrealistic assumptions, such as perfect debugging, constant fault detection rate and regular changes.

In fact, considering more reasonable assumptions in the reliability modeling may further improve the fitting and predictive power of software reliability growth models. It is affected by many factors, such as tester's skill, test plans, testing tools and runtime environment. Thus, software debugging is an imperfect process. And software testing for getting fault data set is done under the assumption that user's operation environment is the same as the testing one. However, in practice, it is exactly the same. This paper deals with a software reliability growth model which considers imperfect debugging and disagreement between operation environments. The better performance of proposed model is illustrated with fault data sets from software development project.

KEY WORDS : software reliability growth model; imperfect debugging; disagreement between operation environment

1. Introduction

Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment [1]. Reliability is one of the most important quality attributes of software and a primary concern for both software developers and software users. We can quantitatively measure and assess the software reliability by using a mathematical model. The Software Reliability Growth Model (SRGM) is a tool that can be used to evaluate the software reliability quantitatively and track the growth of reliability [5].

In the last four decades, with the increasing demand to high quality software, many SRGMs have been published [3,6,8-19]. An important class in SRGMs is the Non-homogeneous Poisson Process (NHPP) model, which has been widely used in analyzing the reliability of software products in practice [10]. These models consider the debugging process as a counting process characterized by its mean value function. Software reliability can be estimated after the mean value function is determined. Every model has been built upon various assumptions.

According to the assumption for debugging, SRGMs can be divided into two categories. Models in first category considered perfect debugging, i.e., the fault detected is immediately removed and no new errors are introduced. As increasing of size and complexity of software product, these models supper in the realistic software testing environment due to the ideal assumptions [4,21].

The others in second category are not only considered imperfect debugging, i.e., faults are not always fully repaired, and can be introduced during the fault remove process, but also several factors, such as fault detection rate and learning phenomenon of the software developers [24].

Meanwhile, according to change of fault detection rate, SRGMs can be divided into three categories.

First category includes models having a constant fault detection rate. For example, the models published by Jelinski and Modranda, Goel and Okumoto [13] supposed that the fault detection rate is constant and no new fault is introduced.

Second category includes models based on regular changes of fault detection rate. The delayed S-shaped (DSS) model published by Yamada et al. [9] and the inflection S-shaped model published by Ohba [3] assumed that fault detection rate is increasing smoothly over time.

Many models, such as Wang Model, Kapur Models, Pham Models considered testing effort, fault removal effect, learning phenomenon and so on. Pham and Zhang [17] assumed that the fault detection rate is non-decreasing and the fault content function is an exponential function and supposed an imperfect debugging model. Pham et al. [18] proposed a general imperfect debugging model, which assumed that the fault content function is a linear function and the fault detection rate is non-decreasing. Zhang et al. [24] proposed a software reliability growth model considered fault removal efficiency, which assumes that fault introduction rate to be constant and the fault detection rate function is non-decreasing with an inflection S-shaped. Kapur et al. [8,11] proposed two SRGMs, which assume that fault content function is proportional to the number of fault detections and fault detection rate is non-decreasing and fault removal rate is constant.

We can say that these models all consider regular changes of fault detection rate.

Third category includes some models considering irregular changes of fault detection rate. Yamada supposed that fault detection process is a stochastic one and proposed software reliability growth models by applying stochastic differential equations [6,12].

Yamada agreed that if the size of software is very large, the number of faults detected during testing phase becomes large, in such a case, a model in a stochastic process with a continuous state space is useful.

As seen above, more accurate software reliability growth models are publi shed to estimate the optimal software release time and the cost, due to no c ommon model in all software development.

Many researchers, in fact, agree that software debugging is very complicated and uncertain and it is affected by many factors, such as the testing tool, operation profile, skill of tester and runtime environment [10]. The model built upon better reasonable assumptions can improve the fitting and predictive power in practice.

Now, most of models have a common assumption that software testing environment is the same as user's operation environment. But, software product is used in different environment after they are released and there is a disagreement between operation environments. So we incorporate disagreement between operation environments and imperfect debugging into modeling and propose a new model.

The organization of study is as follow. In Section 2, the basic assumptions for new model are introduced and proposed a software reliability growth model. In Section 3, the performance of new model is compared with several models by using public fault data sets, which are from real software development process. The conclusion is given in Section 4.

2. Software Reliability Growth Modeling

Notations

$m(t)$	mean value function that presents the expected number of detected faults by time t.
$a(t)$	Total fault content function, i.e., the sum of expected number of initial faults and introduced faults till time t.
$b(t)$	Fault detection rate function.

Assumptions for the proposed model

In general, fault detection and removal are complicated. Reasonable assumption is needed in building a good software reliability growth model.

Proposed model in this paper has some assumptions as follows.

1. The occurrence of fault follows NHPP.
2. Software failure is at any times as a result of faults remaining in the software.
3. The number of faults remaining in the software gradually decreases as the testing procedure goes on.
4. Learning phenomenon is considered in fault detection process.
5. Whenever a fault is detected, it is removed immediately and new fault can be introduced into software. The number of fault introduced is directly proportional to ones of fault detected at any time.
6. The disagreement between operation environments is random.

New model development

We can have the following differential equation directly from assumption 1, 2.

$$\frac{dm(t)}{dt} = b(t)[a(t) - m(t)] \tag{1}$$

From the assumption 5, we can denote changing of fault introduction as follow.

$$\frac{da(t)}{dt} = \delta \frac{dm(t)}{dt} \tag{2}$$

where δ is the fault introduction rate.
Thus

$$a(t) = a + \delta m(t) \tag{3}$$

where a is the expected number of initial faults in software.
The disagreement between operation environments is expressed by using a random variable as follow[20].

where η is a random variable that presents the disagreement between operation environments.

$$\frac{dm(t)}{dt} = \eta b(t)[a(t) - m(t)] \tag{4}$$

Under the initial condition $m(0) = 0$, we can get a general solution of equation 4 as follow.

$$m_\eta(t) = e^{-\eta B(t)}[\int_0^t \eta a(\tau) b(\tau) e^{\eta B(t)} d\tau] \tag{5}$$

where $B(t) = \int_0^t b(\tau) d\tau$.

Substituting equation 3 into equation 5:

$$m_\eta(t) = \frac{a}{1-\delta} e^{-\eta B(t)}[\int_0^t \eta b(\tau) e^{\eta B(t)} d\tau] \tag{6}$$

From assumption 6, we can assume that η follows the gamma distribution.

$$g(x) = \frac{\beta^\alpha x^{\alpha-1} e^{-\beta x}}{\Gamma(\alpha)}, \alpha, \beta > 0; x \geq 0 \tag{7}$$

where $\Gamma(t)$ is a gamma function and $g(x)$ is a probability density function.

From equation 5and 6, $m_\eta(t)$ is as follow.

$$m_\eta(t) = \int_0^{+\infty} m_x(t) g(x) dx = \int_0^{+\infty} e^{-xB(t)} \left(\int_0^t x b(\tau) e^{xB(\tau)} d\tau \right) g(x) dx \tag{8}$$

Now, mean value function $m(t)$ that presents the number of detected faults by time t is the same as the expectation of $m_\eta(t)$ as a random variable η.

$$m(t) = m_\eta(t) = \int_0^{+\infty} m_x(t) g(x) dx = \int_0^{+\infty} e^{-xB(t)} \left(\int_0^t x b(\tau) e^{xB(\tau)} d\tau \right) g(x) dx \qquad (9)$$

From assumption 3 and 4, we can assume that fault detection rate is non-decrease due to learning phenomenon of tester and decreasing of number of faults contented in software. It is expressed as follow.

$$b(t) = bt^d e^{ct} \qquad (10)$$

Substituting equation 10 into equation9

$$m(t) = m_\eta(t) = \frac{a}{1-\delta}\left(1 - \left(\frac{\beta}{\beta + \sum_{i=0}^{k} \frac{bc^i t^{i+d+1}}{i!(i+d+1)}}\right)^{\alpha}\right) \qquad (11)$$

3. Illustrative Examples

In this section, we examine the good-of-fit and predictive power of the proposed model and compare it with existing models.

The procedure is as follow [18]:

First, fit each model to the data; estimate the model parameters and obtain the mean value functions. Most of the data points are used to fit the models and estimate the parameters.

Second, Compare models with each other within a data set, using some criteria, the remaining points from data are used to illustrate the short-term predictive power of the model.

Criteria for model comparison

Two common criteria are used for model comparison. Below is a brief description of the criteria.

The mean square of error (MSE) measures the deviation between the predicted values with the actual observation and is defined as follow [15,28].

$$MSE = \frac{\sum_{i=1}^{n}\left(y_i - \widehat{m}(t_i)\right)^2}{n-k} \tag{12}$$

where n and k are the number of observations and number of parameters in the model.

When the sample size of fault data set is small, we give the other form of MSE as follow[19]:

$$MSE = \frac{\sum_{i=1}^{n}\left(y_i - \widehat{m}(t_i)\right)^2}{n-k} \tag{13}$$

The Akaike information criterion(AIC) calculates the ability of a model to maximize the likelihood function that is directly related to the degrees of freedom during fitting; increasing the number of parameters usually gives a better fit [2,23].

This criterion can be defined as follows:

AIC = -2log(likelihood function at its maximum value)+2k; (14)

Where k denotes the number of parameters in the model.

The AIC criterion uses the degrees of freedom by assigning a larger penalty to a model with more parameters. For AIC and MSE, the smaller the value, the better the model performance [10,15].

Data set

To validate the performance of the proposed model in an actual test, we use two public fault data sets from real software development process.

The first data set (DS1) was obtained from an small operating system, during 148 days, 112 faults were detected and removed.

Table Ⅰ. DS1[14].

Test Period(day)	**Detected faults**	**Test Period(day)**	**Detected faults**	**Test Period(day)**	**Detected faults**	**Test Period(day)**	**Detected faults**
1	0	38	27	75	68	112	85
2	1	39	27	76	68	113	86
3	1	40	29	77	69	114	87
4	1	41	29	78	69	115	90

5	1	42	32	79	70	116	91
6	1	43	34	80	73	117	92
7	1	44	35	81	73	118	92
8	1	45	37	82	73	119	95
9	1	46	37	83	73	120	95
10	1	47	37	84	74	121	98
11	2	48	38	85	74	122	99
12	2	49	40	86	75	123	99
13	2	50	43	87	76	124	99
14	2	51	43	88	76	125	99
15	2	52	43	89	76	126	100
16	2	53	44	90	76	127	103
17	3	54	44	91	76	128	103
18	5	55	44	92	77	129	103
19	6	56	44	93	77	130	104
20	9	57	45	94	78	131	104
21	12	58	51	95	79	132	104
22	14	59	52	96	79	133	104
23	14	60	53	97	79	134	105
24	14	61	54	98	79	135	105
25	14	62	55	99	79	136	105
26	14	63	57	100	80	137	106
27	14	64	58	101	81	138	106
28	14	65	59	102	81	139	106
29	16	66	59	103	81	140	106
30	18	67	61	104	81	141	106
31	20	68	61	105	83	142	108
32	21	69	61	106	83	143	109
33	21	70	62	107	84	144	110
34	24	71	65	108	84	145	110
35	25	72	66	109	85	146	110
36	26	73	66	110	85	147	111
37	26	74	68	111	85	148	112

TableⅡ. DS2[25].

Test Period(week)	Detected faults	Test Period(week)	Detected faults	Test Period(week)	Detected faults	Test Period(week)	Detected faults
1	12	6	97	11	116	16	141
2	23	7	109	12	123	17	144
3	43	8	111	13	126		
4	64	9	112	14	128		
5	84	10	114	15	132		

The second data set (DS2) was obtained from middle-size software project during 17 weeks, 144 faults were detected and removed.

Experimental results and comparison

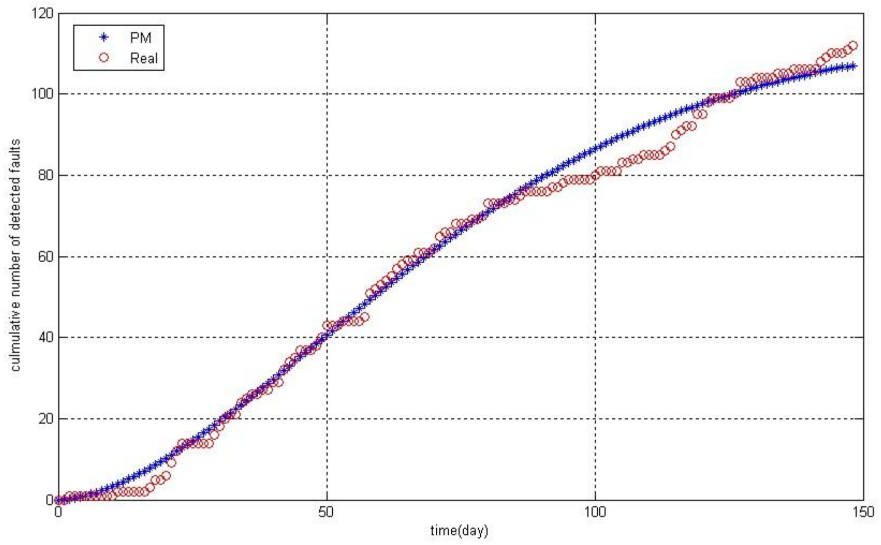
We divide data sets into 50%, 75% subsets to compare the performance of models. We use 50%, 75% of data sets to estimate the model parameters and the remaining 50%, 25% of the data sets to compare the model predictive power. We select 11 different SGRMs as the comparison models.

DS1: As shown in TableⅢ, our model has the best fitting and predictive power among all models when 50% of the fault data from DS1 are applied. Its MSE$_{predict}$, AIC are 19.3, 190.5 respectively. When 75% of the fault data from DS1 are applied, our model has the smallest MSE$_{predict}$, AIC values among all models.

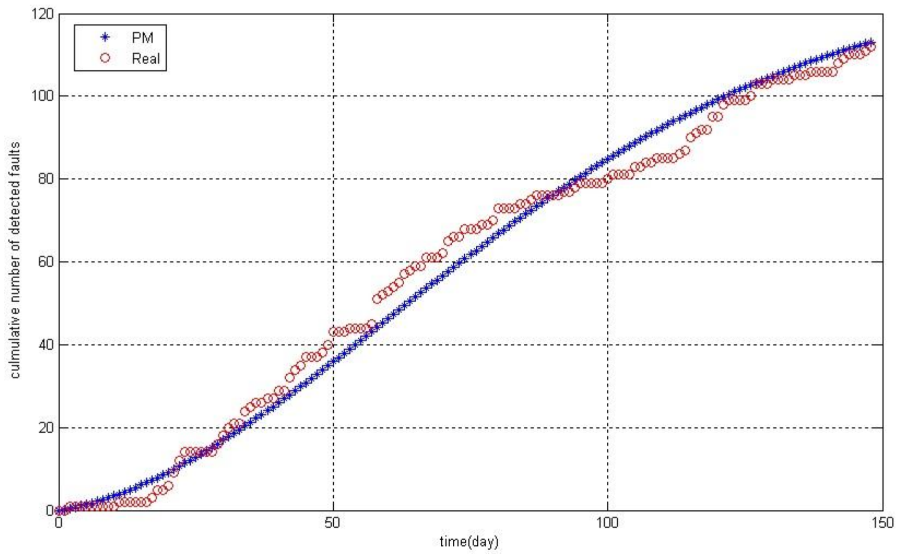
TableⅢ. Comparison of fitting and predictive power of SRGM using DS1.

Model	**50%**		**75%**	
	MSE$_{predict}$	**AIC**	**MSE$_{predict}$**	**AIC**
G-O[13]	215.98	202.84	19.77	275.17
DSS[9]	275.3	190.81	120.7	259.8
ISS[3]	215.26	204.82	255.74	260.85
P-N-Z[18]	319.1	192.83	132.96	260.87
P-Z[17]	177.61	196.49	173.09	262
Zhang-Teng Pham[24]	133.57	198.67	221.35	265.68
Kapur-1[8]	8617.31	206.82	325.43	279.17
Kapur-2[11]	378.42	194.05	150.35	261.72
Yamada SDE[12]	169.7	204.9	20.47	277.2
Yamada DSS SDE[6]	282.17	192.8	124.55	261.8

Yamada ISS SDE[6]	277.5	206.9	21.06	279.2
Proposed Model	**19.3**	**190.5**	**16.8**	**259.8**



(a)



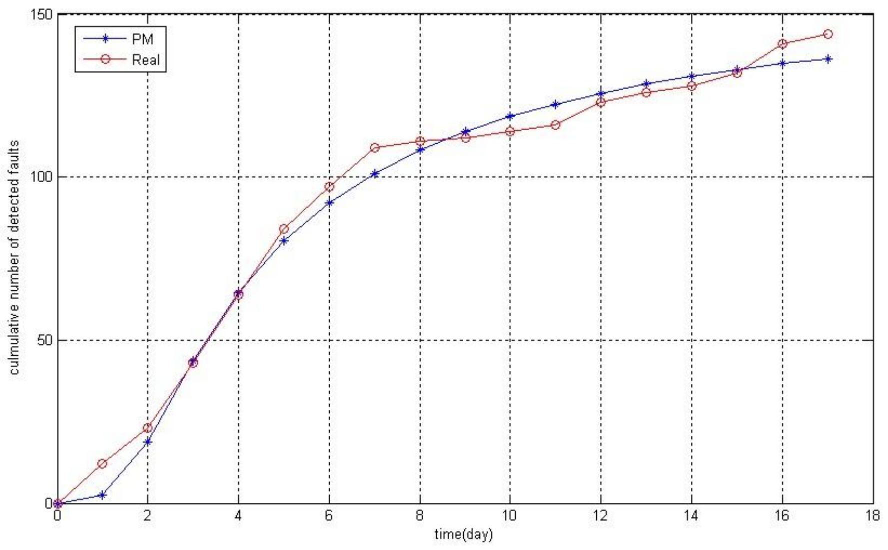
(b)

FigureⅠ. Actual and estimated number of fault detected in the time interval. (a) and (b) illustrate the case for 50% and 75% of DS1.

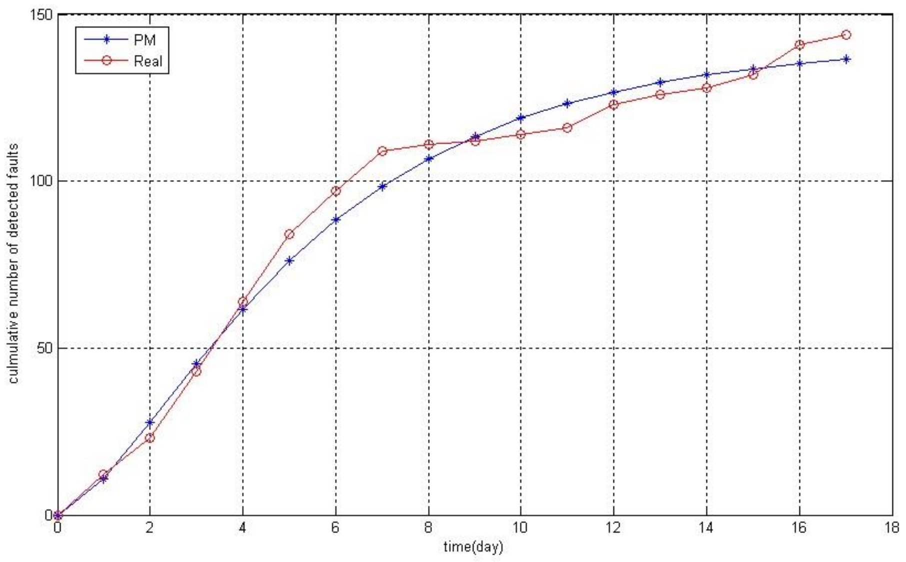
DS2: Similarly, As shown in Table Ⅳ, our model has the best fitting and predictive power among all models when 50% of the fault data from DS1 are applied. Its $MSE_{predict}$, AIC are 20.7, 47.5 respectively. When 75% of the fault data from DS1 are applied, our model has the smallest $MSE_{predict}$, AIC values among all models.

TableⅣ. Comparison of fitting and predictive power of SRGM using DS2.

Model	**50%**		**75%**	
	$MSE_{predict}$	**AIC**	**$MSE_{predict}$**	**AIC**
G-O[13]	32.83	72.6	34.98	91.5
DSS[9]	152.76	56.6	100.77	79.7
ISS[3]	297.58	48.7	113.1	82.6
P-N-Z[18]	297.96	50.7	113.38	84.6
P-Z[17]	306.48	52.7	113.05	86.6
Zhang-Teng Pham[24]	152.03	64.9	149.45	121.9
Kapur-1[8]	32.86	76.6	34.17	95.5
Kapur-2[11]	32.84	78.6	94.27	82
Yamada SDE[12]	31.92	74.5	35.5	93.1
Yamada DSS SDE[6]	145.75	58.9	103.77	81.8
Yamada ISS SDE[6]	288.63	50.2	112.3	84.8
Proposed Model	**20.7**	**47.5**	**27.3**	**77.4**



(a)



(b)

FigureⅡ. Actual and estimated number of fault detected in the time interval. (a) a nd (b) illustrate the case for 50% and 75% of DS2.

As seen above experiments, we can see that proposed model has a better performance that other models.

4. Conclusion

In this paper, disagreement between testing environment and user's operation one and imperfect debugging are incorporate into software reliability growth modeling.

We are faced with disagreement between testing environment and realistic operation one in software testing due to using software by different users after they are released. And software debugging is an imperfect process, too. These problems are studied here. The proposed model considers that the introduced fault is directly proportional to number of detected fault at any time. The gamma distribution is used in our model to present the disagreement between operation environments. The fitting and predictive power of proposed model has been done on two public fault data sets with several models. The results show better fitting and predictive power than some other ones.

References

1. Musa J D, Iannino A, Okumoto K. Software Reliability Measurement Pre diction Application McGraw-Hill New York 1989: 32-241.
2. Akaike H, A New Look at Statistical Model Identification. IEEE Transacti ons on Automatic Control 1974;AC-19(6):716-723.
3. Ohba M. Inflection S-shaped Software Reliability Growth Models. In: Osa ki S, Hatoyama Y, eds. Proc. of the Stochastic Models in Reliability The ory. Berlin: *Springer-Verlag* 1984; 144-162.
4. Ohba M, Chou X M. Does Imperfect Debugging Affect Software Reliability Growth? *11th International Conference on Software Engineerin g* 1989: 237–244.
5. William W. Everett. Software Reliability Measurement. *IEEE Journal on S elected Areas in Communications* 1990; 8(2):247-252.
6. Yamada S, Nishigaki A, Kimura M. A Stochastic Differential Equation M odel for Software Reliability Assessment and its Goodness-of-Fit. *Internat ional Journal of Reliability and Application* 2003; 4(1):1-11.
7. Jinyong Wang, Zhibo Wu, Yanjun Shu, Zhan Zhang. An Imperfect Softwa re Debugging Model Considering Log-logistic Distribution Fault Content Function. *Journal of Systems and Software* 2015; 100:167-181.
8. Kapur P K, Gupta D, Gupta A, Jha P C. Effect of Introduction of Faults and Imperfect Debugging on Release Time. Ratio Mathematica 2008; 18: 62-90.
9. Yamada S, Ohba M, Osaki S. S-shaped Reliability Growth Modeling for Software Error Detection. *IEEE Transactions on Reliability* 1983; 32:475-484.
10. Jinyong Wang, Zhibo Wu, Yanjun Shu, Zhan Zhang. A General Imperfec t Software Debugging Model Considering the Nonlinear Process of Fault Introduction[C]. *In Proceedings of the 14th International Conference on Quality Software* 2014:222-227.
11. Kapure P K, Pham H, Anand S, Yadav K. A Unified Approach for Dev eloping Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation. *IEEE Transactions on Reliability* 2011; 60(1):331-340.

12. Shigeru Yamada, Mitsuhiro Kimura. Software Reliability Measurement an d Assessment with Stochastic Differential Equations. IEICE Trans. Funda mentals 1994; E77-A(1):109-116.
13. Goel AL, Okumoto K. Time-Dependent Error-Detection Rate Model for Software Reliability and other Performance Measures. *IEEE Trans. On Rel iability* 1979; 28:206-2011.
14. Musa J D. Software Reliability Data. Cyber Security and Information Sy stems Information Analysis Center, January 1980.
15. Kapur PK, Sameer Anand, Shigeru Yamada, Venkata SS. Stochastic Diffe rential Equation-Based Flexible Software Reliability Growth Model. *Mathe matical Problems in Engineering* 2009; [doi:10.1155/2009/581383].
16. Qiuying Li, Haifeng Li, Minyan Lu. Incorporating S-shaped Testing-effor t Functions into NHPP Software Reliability Model with Imperfect debug ging. *Journal of Systems Engineering and Electronics* 2015; 26(1):190-2 07.
17. H. Pham, L. Nordamann, X. Zhang. A general imperfect software debug ging model with S-shpaed fault detection rate. IEEE Trans. On Engineeri ng 1997; 14(3):269-282.
18. Pham H, Zhang X. A General Imperfect-Software-Debugging Model with S-Shaped Fault-Detection Rate. *IEEE Transactions on Reliability* 1999; 48 (2):169-175.
19. Jinyong Wang. An imperfect software reliability model considering irregu lar fluctuation of fault introduction rate. *Quality Engineering* 2017; 29(3): 377-394.
20. Hoang Pham. A generalized gault-detection software reliability model sub ject to random operating environments. *Journal of Computer Science* 201 6; 3:145-150.
21. Tao Li, Kaigui Wu. A NHPP Software Reliability Growth Model Consid ering Learning Process and Number of Residual Faults. Journal of Conve rgence Information Technology 2012; 7(13):127-134.
22. Norman F.Schneidewind. Software Reliability Model with optimal selectio n of failure data. *IEEE Transactions on Software Engineering* 1993; 19(1 1):1095-1104.
23. Joseph E. Cavanaugh, Robert H. Shumway. An Akaike information criter

ion for model selection in the presence of incomplete data. *Journal of St atistical Planning and Inference* 67(1998):45-65.

24. Zhang X M, Teng X L, Pham H. Considering Fault Removal Efficiency in Software Reliability Assessment. *IEEE Trans. on S ystems Man and Cybernetics Part A-Systems and Humans*2 003;33(1):114−120.
25. Xie M, Hu Q P, Wu Y P, Ng S H. A Study of the Modeling and Anal ysis of Software Fault-detection and Fault-correction Processes. Quality a nd Reliability Engineering International 2007; 23(4):459-470.
26. Xiaolin Teng, Pham Hoang. A New Methodology for Predicting Softwar e Reliability in the Random Field Environments. *IEEE Transactions on R eliability* 2007; 55(3):458-468.
27. Kai Yuan Cai, Ping Cao, Zhao Dong, Ke Liu. Mathematical Modeling o f Software Reliability Testing with Imperfect Debugging. *Computers and Mathematics with Applications* 2010; 59:3245-3285.
28. Kapur P K, Garg R B, Kumar S. Contributions to hard ware and softw are reliability. *World Scientific Singapore* 1999.