J. Appl. Math. & Informatics Vol. **30***(2012), No. 3 - 4, pp. 489 - 498*
Website: http://www.kcam.biz

AN IMPROVED ADDITIVE MODEL FOR RELIABILITY ANALYSIS OF SOFTWARE WITH MODULAR STRUCTURE†

S. CHATTERJEE*, S. NIGAM, J.B. SINGH AND L.N. UPADHYAYA

Abstract. Most of the software reliability models are based on black box approach and these models consider the entire software system as a single unit. Present day software development process has changed a lot. In present scenario these models may not give better results. To overcome this problem an improved additive model has been proposed in this paper, to estimate the reliability of software with modular structure. Also the concept of imperfect debugging has been also considered. A maximum likelihood estimation technique has been used for estimating the model parameters. Comparison has been made with an existing model. χ^2 goodness of fit has been used for model fitting. The proposed model has been validated using real data.

AMS Mathematics Subject Classification : 65H05, 65F10.
Key words and phrases : Software Reliability, Modular Structure, Faults, NHPP.

1. Introduction

With the growing use of computers in various critical areas, the demand of highly reliable software is increasing day by day. To measure the reliability of a software many software reliability models have been developed depending on different aspects of software development process like: perfect debugging, imperfect debugging, immediate removal of faults, effect of learning process etc [15, 17, 18, 21]. In all these models, software systems have been considered as a single unit and its behavior with the outside world has been modeled without considering its internal structure. This approach of modeling is black box based [4]. In this approach most of the models have been developed based on failure nature observed during testing phase. Assuming some probability distribution for past

Received April 4, 2011. Revised September 19, 2011. Accepted October 18, 2011.
*Corresponding author. †Author's acknowledge University Grant Commission (UGC), New Delhi, India, for financial help under the project number F.No.33-115/2007(SR) and Indian School of Mines, Dhanbad, India for providing necessary facilities for this work.

failure data, researchers used to model the future failure behavior of the software and estimate number of faults remaining in a software, its reliability, etc.. Nowadays with the advancement and wide spread use of object oriented design techniques the component based software development is increasing. Commercially available of the shelf (COTS) components can be developed in house, or can be developed contractually. This reflects the fact that, in present days softwares are developed in more heterogeneous fashion, i.e., it is developed by multiple team in different environments. Therefore, black-box approach for developing software reliability models may not produce good results. Thus, predicting reliability of the software in its early life cycle, i.e., design phase is very much essential. This is only possible by using architectural based software reliability estimation technique.

Architecture of a software represents the manner in which the different components, i.e., modules of a software interact. The interaction between the components takes place through transition, i.e., through transfer of execution control. The architecture may also include information about the execution of each component. One must model the interaction of all components under architecture based approach. The failure behavior of each component and their interfaces is specified in terms of reliability and failure intensities (either constant or time dependent). One can estimate the reliability of software by superimposing the failure behavior on architecture based model. The earliest software reliability model based on component utilization and their reliability has been proposed by Cheung [2]. Till date many researchers have proposed various architecture based models [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 19, 20]. Depending on the method of superimposition of the failure behavior on architecture of a software, architecture based software reliability modeling technique can be classified in three categories [8]: (i) state-based modeling, (ii) path-based modeling, and (iii) additive models. In this paper, an improved architecture based software reliability model has been proposed, which falls under category (iii).

In additive models the failure behavior of each components can be modeled using some stochastic process such as non homogeneous poisson process (NHPP). Therefore, the system failure process also follows NHPP. Also, in these category models it has been assumed that, cumulative failures and failure intensity function of the system is the sum of the corresponding functions of each subsystem.

Xie & Wohlin [20] has proposed an additive model, considering the fact that failure process of each subsystem follows a log-power model [22]. They have considered the debugging process as perfect and used linear regression technique to estimate the model parameters. The main drawback of this model is the assumption of perfect debugging. Entire software development process is human dependent. Therefore, while debugging a software one can introduce new errors in the software. Also, as learning of test personnel increases about a software, the failure detection rate increases. This is more realistic situation than perfect debugging. In this paper, a modified additive model based on NHPP

has been proposed considering imperfect debugging and increasing failure detection rate. The paper is organized as follows: Section 2 presents the proposed model. Section 3 presents the results and comparison. Finally conclusion has been presented in Section 4.

2. Proposed Additive Model

In this section an improved component based additive model has been proposed. Any software is composed of various subsystems. For large software system it is a common practice to test these subsystems independently. Since failure of any subsystem will be treated as a software system failure, the software system can be considered as a series system. Therefore, a weighted additive model has been considered here. Based on these assumptions the system failure intensity $\lambda_s(t)$ can be modeled as

$$\lambda_s(t) = \sum_{i=1}^{n} w_i \lambda_i = w_1\lambda_1(t) + w_2\lambda_2(t) + \cdots + w_n\lambda_n(t), \tag{1}$$

where $\lambda_i(t)$ is the failure intensity of each component, i.e., module of a software.

Hence, the expected (mean) number of failures of software system $m_s(t)$ will be also an additive one, i.e.,

$$m_s(t) = \sum_{i=1}^{n} w_i m_i = w_1 m_1(t) + w_2 m_2(t) + \cdots + w_n m_n(t), \tag{2}$$

where w_i are weights ($i = 1, 2, \cdots, n$) based on the frequency of execution of each component and $m_i(t)$ is the expected numbers of failures of each component, i.e., module of a software. The failure intensity $\lambda_i(t)$ and expected number of failures $m_i(t)$ are related by the equation:

$$\frac{d}{dt} m_i(t) = \lambda_i(t) \tag{3}$$

Weighted additive model is more logical because frequency of each component being executed affects over all system reliability [14]. The weights $w_1, w_2, \cdots, w_n$ can be obtained based on the architecture of the software. In architecture based approach all the paths of a software are considered. Each path consists of some modules (i.e., components). In a particular path all the modules are related, and this is due to the interfacing between them. Execution of software for various inputs means execution of different paths. For each run, the execution path is specified in terms of the sequence of components. Hence, a software has series structure. For simplicity it is assumed that components, i.e., modules along a path functions independently. Therefore, the reliability of each path can be calculated using the following equation

$$R_p = \prod_{i=1}^{n} R_i, i = 1, 2, \cdots, n, \tag{4}$$

where R_p is the reliability of each path, R_i is the reliability of each component, i.e., module in a path. Reliability R_i of each component i can be calculated using the equation

$$R_i(t) = e^{-\int_0^t \lambda_i(t)\,\mathrm{d}t}. \tag{5}$$

Here it is considered that, each component behaves independently which may not be true always in reality.

Also, in this paper it has been assumed that failure phenomenon of each component follows NHPP. Hence, the system failure process defined as the sum of all failure process will also follow a NHPP, with the mean value function as the sum of underlying mean value function given in equation (2). NHPP model in the area of software reliability was first proposed by Goel & Okumoto [3] and later various NHPP models have been developed considering perfect and imperfect debugging [18].

Goel-Okumoto assumed that, the counting process $N(t), t \geq 0$ represents the cumulative number of failures by time t follows NHPP with mean value function $m(t)$.

Therefore,

$$Prob(N(t) = n) = \frac{e^{-m(t)}[m(t)]^n}{n!}, n = 0, 1, 2, \cdots \tag{6}$$

According to Goel-Okumoto the failure intensity λ_t is proportional to remaining number of faults. Thus m_t can be obtained by solving the differential equation

$$\lambda(t) = \frac{dm(t)}{dt} = b(a - m(t)) \tag{7}$$

and

$$m(t) = \int_0^t \lambda(t)\,\mathrm{d}t. \tag{8}$$

where a is the initial number of faults present in the software to be detected eventually, b is the failure detection rate.

2.1. Assumptions of the Proposed Model. The proposed model is based on the following assumptions

(i) Failure removal process of a software follows NHPP.

(ii) Software system as well as each component of a software is subject to failure at random time caused by remaining faults.

(iii) the software debugging process is imperfect . Hence, failure detection rate b initially decreases and then increases with time as learning increases.

Thus, failure detection rate $b_i(t)$ for each component i, will be a function of time as follows

$$b_i(t) = b_i(t)^k, 0 < k \leq 1 \tag{9}$$

where b_i and k are constant. Here for the sake of simplicity, the number of errors eventually detected 'a' has been considered as constant.

(iv) The software system failure intensity λ_s is weighted sum of failure intensity $\lambda_i(t)$ of each component i, i.e.,

$$\lambda_s(t) = \sum_{i=1}^{n} w_i \lambda_i = w_1\lambda_1(t) + w_2\lambda_2(t) + \cdots + w_n\lambda_n(t), \tag{10}$$

(v) The expected number of failures $m_s(t)$of the software system is also an weighted sum of expected number of failures $m_i(t)$ of each component i, i.e.,

$$m_s(t) = \sum_{i=1}^{n} w_i m_i = w_1 m_1(t) + w_2 m_2(t) + \cdots + w_n m_n(t), \tag{11}$$

Based on the above assumptions the mean value function and failure intensity for each component of the proposed model is

$$m_i(t) = a_i(1 - e^{\frac{-b_i t^{k+1}}{k+1}}) \tag{12}$$

and

$$\lambda_i(t) = a_i b_i t^k e^{\frac{-b_i t^{k+1}}{k+1}} \tag{13}$$

The conditional reliability $R_i(x|t)$ for each component can be obtained by solving the equation

$$R_i(x|t) = e^{-[m_i(t+x) - m_i(t)]} \tag{14}$$

Though in the proposed model a weighted sum of failure intensity $\lambda_i(t)$ and expected number of failures $m_i(t)$ has been considered, but no data is available on the frequency of execution of each subsystem. Therefore, in this case the weights of each component has been considered as one, i.e., $w_i = 1$ for $i = 1, 2, \cdots, n$.

2.2. Parameter Estimation. The parameters of the proposed model have been estimated using maximum likelihood estimation (MLE) technique given in [18]. The log likelihood function (LLF) is defined as

$$LLF = \sum_{i=1}^{n} (y_i - y_{i-1})\ log[m(t_i) - m(t_{i-1})] - m(t_n) \tag{15}$$

and maximum likelihood equation for estimating the unknown parameter θ is given by

$$\sum_{i=1}^{n} \frac{\frac{\partial}{\partial\theta_i} m(t_i) - \frac{\partial}{\partial\theta_i} m(t_{i-1})}{m(t_i) - m(t_{i-1})} (y_i - y_{i-1}) - \frac{\partial}{\partial\theta_i} m(t_n) = 0 \tag{16}$$

where y_i represents failure corresponding to each time t_i, $i = 1, 2, \cdots, n$. Here, the unknown parameters are the number of failures a_i and failure detection rate b_i of each component i. The parameters a_i and b_i of the proposed model have been estimated using MLE technique given in equation (15) and (16). MLE equations derived using equation (16) has been solved by numerical technique Newton Raphson method. MATLAB software has been used for solving MLE equations.

3. Model Validation and Comparison

In this Section model validation and comparison has been carried out.

3.1. Model Validation. The proposed model has been validated using the failure data of a large communication software given in [20]. The data set has been given in Table 1. The software consists of two components. The subsystem 2 has been introduced in the system at time $t = 23$ months. During the time $t = 25$ to $t = 30$ months the subsystem 2 shows high failure where as subsystem 1 has shown very less failure during the same period. Due to this reason it is difficult to fit a model which can give a very accurate prediction. The failure

Table 1: Software Failure data

Month	Sub-System 1	Sub-System 2	System	Month	Sub-System 1	Sub-System 2	System
1	2	NA	2	26	1	19	20
2	11	NA	11	27	0	12	12
3	18	NA	18	28	0	13	13
4	10	NA	10	29	0	26	26
5	12	NA	12	30	1	32	33
6	4	NA	4	31	0	8	8
7	28	NA	28	32	0	8	8
8	6	NA	6	33	0	11	11
9	7	NA	7	34	0	14	14
10	6	NA	6	35	1	7	8
11	17	NA	17	36	0	7	7
12	31	NA	31	37	1	7	8
13	8	NA	8	38	0	0	0
14	7	NA	7	39	0	2	2
15	10	NA	10	40	0	3	3
16	2	NA	2	41	0	2	2
17	2	NA	2	42	0	5	5
18	0	NA	0	43	1	2	3
19	3	NA	3	44	1	3	4
20	2	NA	2	45	0	4	4
21	1	NA	1	46	0	1	1
22	1	NA	1	47	1	2	3
23	1	3	4	48	0	1	1
24	0	3	3	49	0	0	0
25	1	38	39	50	1	1	2

data has been normalized by taking the log transformation and then it has been used to estimate the parameters. The estimated values of model parameters for subsystem 1 and 2 are $a_1 = 195$, $a_2 = 237$, $b_1 = 0.0808$ and $b_2 = 0.0780$. The actual number of error present in subsystem 1 and subsystem 2 are 198 and 234 respectively. The reason of such a difference has been described earlier. The

final expression of $m(t)$ for subsystem 1, 2 and the software system is given in the following Eq.17, Eq.18 and Eq.19 respectively:

$$m_1(t) = a_1(1 - e^{\frac{-b_1 t^{k+1}}{k+1}}) \tag{17}$$

$$m_2(t) = a_2(1 - e^{\frac{-b_2 t^{k+1}}{k+1}}) \tag{18}$$

$$m(t) = \begin{cases} a_1(1 - e^{\frac{-b_1 t^{k+1}}{k+1}}), \ 0 < t < 23, \\ a_1(1 - e^{\frac{-b_1 t^{k+1}}{k+1}}) + a_2(1 - e^{\frac{-b_2 t^{k+1}}{k+1}}), \ t \geq 23 \end{cases} \tag{19}$$

The graphs between actual and predicted errors for subsystem 1, 2 and software system are plotted in Fig. 1, 2 and 3.

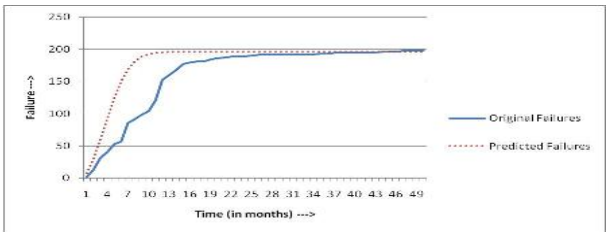

Fig. 1. Prediction graph of Subsystem 1

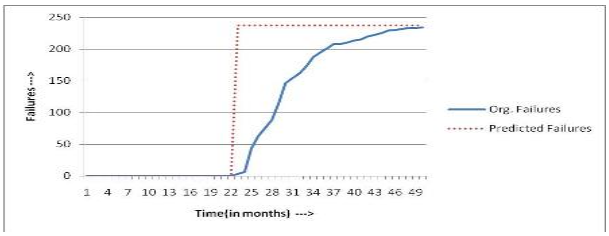

Fig. 2. Prediction graph of Subsystem 2

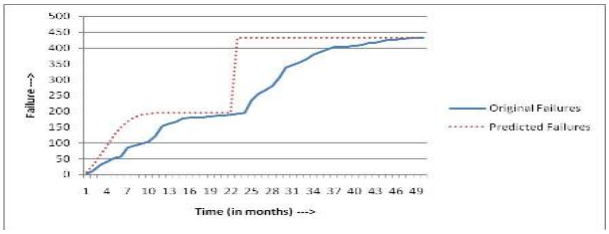

Fig. 3. Prediction graph of the whole system

3.2. Comparison. Here the proposed model has been compared with additive model given in [20]. Akaike information criterion (AIC), root mean squarer error (RMSE) has been used for comparison. χ^2 goodness of fitness has been used for model fitting. The AIC value has been calculated using the formula given in [1] as follows

$$AIC = \frac{-2\,ln(maximum\,likelihood) + 2r}{n} \approx 2\sigma_e^2 + r\frac{2}{n} \tag{20}$$

where r is the number of parameters, n is the total data points and σ_e^2 is the error variance. The RMSE has been calculated using the formula

$$RMSE = \sqrt{\frac{\sum_{j=1}^{n}(y_j - \hat{y_j})}{n}} \tag{21}$$

where y_j denotes the original failure given in the data set and $\hat{y_j}$ is the predicted failure at the time points j,$(j = 1, 2, \cdots, n)$.

The χ^2 goodness of fit has been computed as

$$\chi^2 = \sum_{i=1}^{n} \frac{(f_0(i) - f_e(i))^2}{f_e(i)} \tag{22}$$

where $f_0(i)$, $f_e(i)$ represents actual and predicted faults respectively.

The computed values of AIC and RMSE are compiled in Table 2.

Table 2. AIC and RMSE values of the models

Models	AIC	RMSE
Additive Model by Xie & Wohlin [20]	9.5922	140.2860
Proposed Model	8.2589	84.0898

The computed values of AIC and RMSE in Table 2 shows that the proposed model performs better than the model given in [20]. The computed χ^2 goodness of fit at 1% level of significance for proposed model is 10.6 whereas the tabulated value is 74.90. Hence the proposed model is accepted at 1% level of significance.

4. Conclusion

This paper proposes an improved additive model for estimating reliability of a software with modular structure. The important feature of this paper is the introduction of the concept of imperfect debugging and time dependent error detection rate. The proposed model is more realistic and will be very useful for software reliability analysis. There is a further scope of improvement in the model with time dependent error introduction rate. Also, one can try with different error detection and introduction rate for different components to get better results. The proposed model can further be improved by considering dependency between different software modules. The proposed model will be helpful

for reliability professionals who are interested in architecture based analysis for software. The model can also be used for software consisting of more modules.

References

1. George E.P. Box, Gwilym M. Jenkins and G.C. Reinsel, *Time Series Analysis Forecasting and Control*, San Francisco, California, U.S.A.: Holden-Day, 1976.
2. R.C. Cheung, *A User-Oriented Software Reliability Model*, IEEE Transactions on Software Engineering, SE.-**6(2)** (1980), 118-125.
3. A.L. Goel, K. Okumoto, *A Time-Dependent Error Detection Rate Model for Software Reliability and Other Performance Measure*, IEEE Trans. On Rel., **28** (1979), 206-211.
4. S.S. Gokhale and K.S. Trivedi, *Dependency Characterization in Path-Based Approaches to Architecture-Based Software Reliability Prediction*, Proc. of IEEE Workshop on Application-Specific Software Engineering and Technology, (1998), 86-89.
5. S.S. Gokhale, W. Eric Wong, K.S. Trivedi and J.R. Horgan, *An Analytical Approach to Architectural-Based Software Reliability Prediction*, IEEE International Computer Performance and Dependability Symposium (IPDS), (1998), 13-22.
6. S.S.Gokhale, K.S. Trivedi, *Analytical Models for Architecture-Based Software Reliability Prediction: A Unification Framework*, IEEE Transactions on Reliability, **55(4)** (2006), 578-590.
7. S.S. Gokhale, *Architecture-Based Software Reliability Analysis: Overview and Limitations*, IEEE Transactions on Dependable and Secure Computing, **4(1)** (2007), 32-40.
8. Katerina Goseva-Popstojanova, A.P. Mathur, K.S. Trivedi, *Comparison of Architecture-Based Software Reliability Models*, Proc. 12 th International Symposium on Software Reliability Engineering, (2001), 22-31.
9. Vivek Goswami and Y.B. Achraya, *Method for Reliability Estimation of COTS Components based Software Systems*, Proceedings of 20th International Symposium on Software Reliability Engineering, ISSRE, (2009).
10. S. Krishnamurthy and A.P. Mathur, *On the estimation of Reliability of a Software System Using Reliabilities of its Components*, Proceedings The Eighth International Symposium on Software Reliability Engineering, (1997), 146-155.
11. P. Kubat, *Assessing Reliability of Modular Software*, Operational research Letters, **8** (1989), 35-41.
12. J.C. Laprie, *Dependability evaluation of software systems in operations*, IEEE Transactions on Software Engineering, **10(6)** (1984), 701-714.
13. B. Littlewood, *A Reliability Model for Systems with Markov Structure*, Applied Statistics, **24(2)** (1975), 172-177.
14. J.H. Lo, S.Y. Kuo, M.R. Lyu and C.Y. Huang, *Optimal Resource Allocation and Reliability Analysis for Component-Based Software Applications*, Proceedings of the 26 th Annual International Computer Software and Applications Conference (COMPSAC02), (2002), 7-12.
15. M.R. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill, 1996.
16. Y.K. Malaiya, M.N. Li, J.M Bieman. and R. Karcich, *Software Reliability Growth with Test Coverage*, IEEE Transactions on Reliability, **51(4)** (2002), 420-426.
17. J.D. Musa, A. Iannino, and K. Okumoto, *Software Reliability:Measurement, Prediction, Application*, McGraw-Hill New York, 1987.
18. H. Pham, *System Software Reliability*, Springer, 2006.
19. Wen-Li Wang, D. Pan, M.H. Chen, *Architectural based software reliability modeling*, The Journal of Systems and Software, **79** (2006), 132-146.
20. M. Xie and C. Wohlin, *An Additive Reliability Model for the Analysis of Modular Software Failure Data*, In Proc. 6 th International Symposium on Software Reliability Engineering, (1995), 188-194.

21. M. Xie, *Software Reliability Modeling*, World Scientific, Singapore, 1991.
22. M. Zhao and M. Xie, *On the Log-Power NHPP Software Reliability Model*, Proceedings 3 rd International Symposium on Software Reliability Engineering, (1992), 14-22.

S. Chatterjee has obtained his B.Sc (Mathematics) from T.D.B. College, Raniganj, The University of Burdwan, India. He obtained M.Sc (Mathematics) and Ph.D from IIT Kharagpur, India. His area of research is software reliability modeling. Presently Dr. Chatterjee is working as Associate Professor, in the Dept. of Applied Mathematics, Indian School of Mines (ISM) Dhanbad, India. He has served G.I.E.T, Orissa and SMIT, Sikkim, India, as a faculty. He has total eleven years of teaching and research experience. He has quite a good number of international and national publications. He has reviewed papers for various national and international journals. His areas of interest are Software Reliability, O.R., Stochastic Process, and Fuzzy Set.

Department of Applied Mathematics, Indian School of Mines, Dhanbad-826004, India.
e-mail: chatterjee_subhashis@rediffmail.com

S. Nigam received his M.Sc degree in Mathematics from IIT Kanpur in 2007. He is currently pursuing Ph.D degree in Applied Mathematics from Indian School of Mines Dhanbad. His research interests are software reliability modeling, neural networks, genetic algorithms and statistical modeling.

Department of Applied Mathematics, Indian School of Mines, Dhanbad-826004, India.
e-mail: shobhitngm@gmail.com

J.B. Singh received his M.Sc. degree in Mathematics in 2007 from Banaras Hindu University Varanasi, India. He is currently pursuing Ph.D. in Applied Mathematics from Indian School of Mines Dhanbad, India. His research interests include software reliability modeling, time series, fuzzy time series, genetic algorithm and artificial neural network.

Department of Applied Mathematics, Indian School of Mines, Dhanbad-826004, India.
e-mail: jeetendra01@gmail.com

L.N. Upadhyaya has been a faculty in the University of Gorakhpur, Gorakhpur and Banaras Hindu University, Varanasi, India. He joined Indian School of Mines Dhanbad, India, in 1977 and currently a Professor in the Dept. of Applied Mathematics. He has more than 30 years of teaching/research experience. He has published more than 50 publications in national and international journals. His area of interest is Sampling Strategies, Statistical Inference under Conditional Specification and Reliability Theory.

Department of Applied Mathematics, Indian School of Mines, Dhanbad-826004, India.
e-mail: lnupadhyaya@yahoo.com