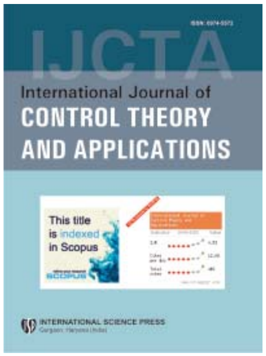

International Journal of Control Theory and Applications

ISSN : 0974–5572

Volume 9 • Number 40 • 2016

Selection of Right Software Reliability Growth Models for Every Software Project

IndhuraniLakshmanan[a] and SubburajRamasamy[a]

[a]*School of Computing, SRM University, Kattankulathur-603203,Tamil Nadu, India.*

E-mail: indhurani.a@gmail.com, subburaj.r@ktr.srmuniv.ac.in

Abstract: Software reliability is one of the attributes of software quality. Due to the increasing complexity of the software systems, delivering reliable software in a timely manner becomes a challenging task. Software Reliability Growth Models (SRGMs) are used to estimate the reliability of the software systems during testing. Although large number of SRGMs have been proposed, it appears that no single model can be considered to be suitable to describe every software failure data set. The research is still continuing to develop more robust models. However, the success of reliability modeling for a given project depends on selection of appropriate SRGM that will fit the software failure data adequately. This paper presents a brief review of existing SRGMs, model selection methods.

Keywords: *Software Reliability Growth Models, Selection Method, Non-Homogeneous Poisson Process, Combinational Model, Soft computing techniques.*

1. INTRODUCTION

The reliability of a software system is defined as the probability that the software will not fail during stated period of time.Many SRGMs have been proposed during past three decades and used both by the software industry and researchers. To use these SRGMs, Software practitioners have to estimate the parameters of the SRGM using software failure data during testing. Using these software failure data, SRGMs can estimate future failure occurrence times, total number of initial faults, number of faults remaining at the time of release, the failure intensity, software reliability achieved at any given time during testing and release time determination [1]. A single SRGM could give varying degree of goodness of fit statistic for different software failure data sets because the characteristics of software failure data sets may vary [2]. Hence, it appears that no single model is available to provide accurate result in all situations. On the other hand, researchers have suggested that combining more than one model may improve estimation accuracy than selecting a single model [2].Estimation accuracy may also vary depending on the different parameter estimation methods and model evaluation criteria [3]. However, the success of software reliability modeling depends on selecting an appropriate SRGM that produces better estimation accuracy in all cases. Hence, this paper presents a review on selection methods to select appropriate SRGM suitable for a given project.The process for selection of appropriate SRGM for a given projectcomprise four stages: study the data, choose SRGMs that may be suitable for the data and the type of metrics to be collected, find goodness of fit and accept or reject the chosen SRGM using model selection methods.This paper presents a brief review on already proposed SRGMs with their parameter estimation methods, evaluation criteria and selection methods. We briefly discuss about different model selection methods used in existing SRGMs.

This paper is organized as follows: Section 2 gives a review of existing SRGMs with different parameter estimation techniques. Section 3 presents the review ofestimation evaluation criteria. Section 4 presents review on selection of appropriate SRGM for a given project. Summary and conclusions are given in section 5.

2. REVIEW OF EXISTING SRGMS

The traditional software reliability growth models are proposed based on a set of assumptions and distributions[4]. Hudson published the first paper on software reliability in 1967 for Markov Birth-Death process [5]. Later, the early developed models include Jeliski-Moranda Model[6], Littelwood-Verral Model[7], Schneidewind model[8] and Goel-okumoto model[3] etc. More than 300 SRGMs have been proposed in the last three decades. When there are a large number of models, they need to be grouped according to chosen characteristics in order to have a better understanding.Thus, this review presents how the existing software reliability growth models have been classified into different categories.This classification covers simple and flexible SRGMs which are widely used by software practitioners under different conditions.

According to Musa[1], the software reliability models could be classified as shown in Table 1.

Table 1
Musa- Software Reliability Models Classification

Category	*Description*
Time	Clock time or execution time
Category	Finite failure or Infinite failure
Type	Probability distribution of the number of failures experienced at time t.

Goel [4] classified the software reliability models as shown in Table 2.

Table 2
Goel- Software Reliability Models Classification

Catogory	*Description*
Time between failure models	Time between two successive failures follows a distribution whose parameters depend on the number of faults remaining in the software during this interval.
Failure or fault count models	Number of failures or faults in specified time interval which follows a stochastic process with a time dependent discrete or continuous failure rate.
Fault seeding models	Seed a set of identified faults in a program which is assumed to have an unidentified set of indigenous faults.
Input domain models	Createa number of test cases from a model distribution as input which is assumed to be representative of the operational usage of the program.

Xie [9] classified the software reliability models based on the failure occurrence process as shown in Table 3.

Table 3
Xie- Software Reliability Models Classification

Category	*Description*
Markovian models	A stochastic process in which its future action depends on the present state of the process and not on the past. Ex. Jelinski-Moranda(J-M) model [6].
Bayesian models	Bayesian models used knowledge on the previous performance of the system. They are described by two distributions. The failure times which follows one distribution with a certain failure rate and the failure rate follows another one distribution. Ex. Littlewood-Verrall(L-V) model [7].
Non-Homogeneous Poisson Process (NHPP) models	A process that follows Poisson distribution with a time dependent failure rate [4].

Yamada et al.[10] classified NHPP based SRGMs into two categories as shown in Table 4.

Table 4
Yamada et al. - NHPP based Software Reliability Growth Models Classification

Category	*Description*
Continuous time SRGM	It uses machine execution time/CPU time or calendar time as a unit of fault detection period which varies with time.
Discrete time SRGM	It uses the number of test cases as a unit of fault detection period without considering time and unit of fault detection period is countable.

Subburaj[11] classified NHPP based continuous time SRGM as shown in Table 5.

Table 5
Subburaj- NHPP based Software Reliability Growth Models Classification

Category	*Description*
Failure based and Fault based SRGMs	Failure based models assume perfect debugging that a failure is caused by one fault. In fault based models, a failure is caused by one or more faults.
Exponential growth and S-shaped growth mean value function	Mean value function either follows exponential growth or S-shaped growth.
Testing Effort models	Since time-based SRGMs assume efforts are constant during entire testing period, it transforms into effort based SRGM using time transform property.
Graphical models	Models which are generalized and flexible to address both exponential and S-shaped mean value function.
Quality metrics producing models	Models estimate the quality of debugging and other measures such as learning index, total number of faults etc. observed in the project that is useful to the software management to improve its assessment.

The parameters on the above models are estimated by either one of the two commonly used statistical parameter estimation methods namely: Maximum Likelihood Estimation(MLE) and Least Square Estimation(LSE).NHPP models are widely used by researchersandthey possess other two important properties called superposition and time transformation [9]. Instead of developing new SRGMs, we can use existing SRGMs effectively by incorporating these two properties. Time transform property develops testing effort based SRGMs as given in table 5.We can develop new combinational SRGMs by summing up two or more SRGMswith their respective mean value functions using superposition property.

2.1. Review of NHPP based combinational SRGMS

While combining the models, the assumptions behind each parameter and model become lost. Hence a non-parametric distribution-free modeling technique may come out [2]. Non-parametric models can produce better estimation accuracy than classical parametric models[12].Combinational SRGM may give accurate parameter estimation than single component model alone [2]. It combines the results of individual component models. It performs well for a few data sets and poor for some other data sets based on the component models.

Almering et al. [13] proposed parametric and non-parametric classification of SRGMs as shown in Table 6.

Table 6
Almering et al. - Parametric and non-parametric classification of SRGMs

Category	*Description*
Parametric SRGM	Parametric or traditional SRGMs assume a predetermined behavior for parameters during model evolution. The parameters are explicitly defined in the model and have a physical interpretation.
Non-parametric SRGM	Although the non-parametric SRGMs include parameters in their model evolution, they don’t have any physical interpretation.

Instead of depending on the result of any single model, Lyu[2] introduced SRGM combination model that combine the results of selected candidate models based on assigned weights. The weights are assigned using equally weighted linear combination, dynamic weighted linear combination, median-oriented combination approaches. Keene et al.[14] proposed an approximation approach for software reliability combinational model and applied for software and hardware failure rate to predict the availability. Popenitiu et al.[15] proposed a linear combination model using supermodel approach and Li et al.[16] suggestted a hierarchical mixture approach for software reliability combination model. Subburajet al.[17, 18] proposed dynamic weighted combination approach for fault-based and failure-based SRGMs respectively. The parameters on these combinational models are estimated using statistical parameter estimation methods like MLE, LSE and Expectation-Maximization [16] algorithm. The results from these combinational models show that, the more component models we combine, the better estimation and prediction.

To improve the parameter estimation accuracy, different methods and algorithms using soft computing techniques have been proposed to estimate the parameters.Karunanidhiet el. [12] introduces Artificial Neural Network (ANN) to software reliability models and proposed feed-forward and Jordon’s semi-recurrent connectionist neural networks for software reliability estimation and prediction. Cai et al.[19] and Yogesh et al.[20] proposed feed-forward neural network approaches to estimate and predict software reliability. These authors built ANN using sigmoid activation function and compare the results with existing classical SRGMs. The results from these approaches concludes that parameter estimation using ANN may give better accuracy than statistical parameter estimation methods. However, the results from the above ANN approaches show that the estimation accuracy using ANN for SRGMs depends on the selection of network architecture by determining the number of neurons is a kind of art.To address this issue, Huang et al.[21], Jung [22], Wang et al.[23], Roy et al.[24] and Indhurani et al.[25] proposed ANN based combinational model for software reliability estimation using existing classical SRGMs. They have combined more than two SRGMs and implemented feed-forward and recurrent neural network architectures by designing activation functions from selected SRGMs.

Guo et al.[26], Sultan et al.[27] and Pachauri et al.[28] explored the use of fuzzy logic and applied fuzzy set theory to build SRGM. They estimate the parameters using fuzzy modeling and calculated the total software cost. Costa et al.[29] , Huang et al.[30], Kim et al. [31] and Jung et al.[32] proposed genetic algorithm to estimate the parameters of combinational SRGM.Xie et al.[33], Zheng et al.[34] Mohandy et al.[35] and Roy et al.[36] proposed hybrid intelligent system by combining neural network and genetic algorithm to estimate the parameters of combinational SRGM. The other soft computing techniques used to estimate the parameters of combinational SRGM are Ant Colony Optimization(ACO) [37], Particle Swarm Optimization(PSO) [38], Cuckoo Search [39] and Bacterial Foraging Optimization Algorithm(BFOA) [40]. Jin et al.[41] and Subburaj et al.[42] proposed parameter estimation using PSO and ANN respectively with testing effort function and concludes that it is flexible and effective than existing methods. Although the results using soft computing techniques give incrementally better parameter estimation, they also depend on the selected SRGMs [21, 24, 25]. Hence, the selection of appropriate flexible SRGMs will reduce the number of component models in the combinational SRGM and produce accurate results for parameter estimation [25, 42].

3. REVIEW OF EVALUATION CRITERIA METHODS

Some meaningful measures are used by researchers to evaluate the SRGMs in terms of goodness of fit and predictive validity. Table7.shows different evaluation criteria used by reliability researchers.

Table 7
Evaluation criteria used by reliability researchers

	Bias	*Variance*	*Noise*	*MSE*	*SSE*	*RMSE*	*R2*	*RE*	*AE*	*MSF*
Lyu&Nikora [2]	*		*							
Xie et al. [33]				*						
Li et al. [16]		*								
Huang et al. [21, 30]	*	*		*		*		*	*	
Cai et al. [19]								*		
Roy et al. [24]								*	*	
Subburaj et al. [11, 17, 18, 25, 42]	*	*	*	*	*	*	*	*	*	*
Zheng et al. [34]				*				*	*	
Guo et al. [26]								*		
Jung et al. [22]								*	*	*

4. REVIEW OFSRGM SELECTION METHODS FOR A GIVEN PROJECT

Although many SRGMs are proposed, it appears that there is no clear guide to select appropriate SRGM for a given project. The success of reliabilitymodeling depends on the selection of appropriate SRGMs. Khoshgoffaer et al.[43] suggested to use Akaike Information Criterian (AIC) to select the best SRGM. Stringfellow et al.[44] proposed an empirical selection method for choosing the best SRGM in terms of goodness of fit, stability and predictive validity. Sharma et al.[45] and Liang et al. [46] proposed distance based approach to select optimal

SRGMs.Kharchenko et al. [47] proposed a method to select an SRGM using assumptions matrix by taking software engineering features and testing processes. Rana et al.[48] suggested a method to select appropriate SRGM by predicting the expected shape of on-going project data and also observing the software process. For example, it is observed that Gompertz SRGM is best for either V-model process or agile type software development process and logistic SRGM is best for waterfall software development process. Park et al.[49] proposed a systematic reliability prediction framework using decision trees for dynamic model selection and combination.

Current practice to select best SRGMs is to apply several SRGMs by fitting models and evaluate their respective goodness-of-fit using software failure data and select appropriate model based on comparison criteria such as Mean Squared Error (MSE), Bias, Noise, Variance and Relative Error (RE) etc. But this approach shows that different methods of model selection criteria result in different model being chosen. Hence, software practitioners may end up with conflict in the model evaluation like best MSE and worst bias etc. All these approaches focus on the goodness of fit to the software failure data, and it may cause the under and over-fitting problems in the predictive validity [49]. Thus, it is necessary to develop a selection method that can easily adopt and produce accurate estimation results in all cases.

5. CONCLUSION

Software Reliability Growth Model is essential to assess the growth of reliability during software testing and it is used to estimate future failure occurrence times, number of faults remaining at the time of release etc. This paper reviewsthe various classifications of SRGMs and the selection of appropriate SRGM for a given project. We present a review of existing SRGMs anddifferent parameter estimation methodsusing statistical and soft computing techniques. We also discuss about different evaluation criteria used to measure the fitting error of the model for the chosen data and model selection methods proposed by various reliability researchers.

6. APPENDIX

Table 8. provides some simple and flexible NHPP based continuous time SRGMs.

Table 8
Simple and flexible NHPP based SRGMs

S.No.	*Model*	*Mean value function μ(t) Equation*
1	Goel-okumotto-1979 [3]	$m(t) = a(1 - e^{-bt})$
2	S-shaped by Yamada-1983 [50]	$m(t)\ a(1 - (1 + bt)e^{-bt})$
3	Ohba Inflection S-Shaped-1984[51]	$m(t) = \dfrac{a(1-e^{-bt})}{1-\beta e^{-bt}}$
4	Kapur- Garg imperfect debugging-1990 [52]	$m(t) = \dfrac{a}{p}(1-e^{(-bpt)})$
5	Logistic Growth by Huang- 2002 [53]	$m(t) = \dfrac{a}{1+ce^{bt}}$
6	Musa BET,LPET-1989 [54] Basic Execution Time (BET) Logarithmic Poisson Execution Time (LPET)	$m(t) = \gamma\left(1-e\left(-\dfrac{\lambda}{\gamma}t\right)\right)$ $m(t) = \alpha \ln(1 + bt)$

S.No.	*Model*	*Mean value function μ(t) Equation*
7	Yamada - Imperfect Debugging 1 & 2-1992 [55]	$m(t)=\frac{ab}{b+\alpha}(c^{at}-e^{bt})$ $m(t)=a(1-e^{bt})\left(1-\frac{\alpha}{b}\right)+\alpha at$
8	P-Z (Pharm and Zuang)-1997[56]	$m(t)=\frac{1}{1+\beta e^{-bt}}\Big((c+a)(1-e^{-bt})$ $-\frac{a}{b-\alpha}(e^{\alpha t}-e^{-bt})$
9	P-N-Z (Pharm, Nordmann and Zuang)-1999[57]	$m(t)=\frac{a}{1+\beta e^{-bt}}((1-e^{-bt})\left(1-\frac{\alpha}{b}\right)+\alpha at$
10	Duane - Power law-1964 [58]	$m(t) = at^{b}$
11	Modified Duane - Littlewood-1984 [59]	$m(t)=a\left(1-\left(\frac{b}{b+t}\right)^{c}\right)$
12	Logarithmic law-1993 [60]	$m(t) = a \ln\{bt\}$
13	Log-power-law(mixure of power-law and logarithmic law)-1993 [60]	$m(t) = a (\ln t)^{b}$
14	Log-power-1993 [60]	$m(t) = a \ln^{b}(1+t)$
15	Exponential law-1993 [60]	$m(t) = ae^{bt}$
16	Inverse-exponential law-1993 [60]	$m(t) = ae^{-b/t}$
17	Combination of logarithmic and log power model-1993 [60]	$m(t) = a \ln^{c}(1+bt)$
18	Goel generalized-1985 [4]	$m(t) = a(1-e^{-btc})$
19	S-G (Subburaj and Gopal) GE-2006 [61]	$m(t)=\mathrm{N}\left(1-e^{\left(-\frac{t}{\theta}\right)}\right)^{\beta}$
20	SGK (Subburaj, Gopal and Kapur)-2007 [62]	$m(t)=\mathrm{N}\left(1-e^{\left(-\left(\frac{t-\gamma}{\theta}\right)^{\beta}\right)}\right)$
21	Generalized NHPP with sWF ROCOF-2008 [63]	$m(t)=\left(1-e^{\left(-\frac{ct}{\theta}\right)}\right)^{\beta}$
22	SGK-2012 [64]	$m(t)=\frac{a}{c}\left(1-e^{\left(-c\left(-\frac{t-\gamma}{\theta}\right)^{\beta}\right)}\right)$

S.No.	*Model*	*Mean value function μ(t) Equation*
23	Yamada et al. (Exponential and Rayleigh Testing efforts)-1986 [65]	$m(t) = a(1 - e^{-b\alpha 1(1 - e^{\beta 1 t})})$ $m(t) = a(1 - e^{-b\alpha 1(1 - e^{-\frac{\beta 1}{2}t^2})})$
24	Yamada et al. (WeibullTesting effort)-1993 [66]	$m(t) = a(1 - e^{-b\alpha 1(1 - e^{\beta 1 t^{\gamma}})})$
25	Huang et al. (Logistic Testing effort)-2002 [53]	$m(t) = a\left(1 - e^{-b\left(\frac{\alpha 1}{1 + \gamma 1 e^{(-\beta 1 t)}}\right)}\right)$

REFERENCE

[1] J. D. Musa, A. Iannino, andK. Okumoto, Software Reliability, Measurement, Prediction and Application, McGraw-Hill, 1987.

[2] M. R. Lyu and A. Nikora, "Applying reliability models more effectively," IEEE Software, vol.9, pp. 43–52, 1992.

[3] A.L.Goel, and K. Okumoto, "Time Dependent Error Detection Rate Model for SoftwareReliability and Other Performance Measures," IEEE Transactions on Reliability, vol. 28,206-211, 1979.

[4] A.L.Goel, "Software Reliability Models: Assumptions, Limitations and Applicability," IEEE Transactions of Software Engineering, vol. 11, pp. 1411-1423, 1985.

[5] I. L. Hudson, "Large sample inference for Markovian exponential families with applicatiom to branching processes with immigration," Australian Journal of Statistics, vol. 24, pp. 98-112, 1982.

[6] W. Farr, "Handbook of Software Reliability Engineering," McGraw-Hill PublishingCompany, New York, 1996.

[7] B. Littlewood, and J. L. Verrall, "A Baysian Reliability Growth model for computer software," Applied Statistics, vol. 22, pp. 332- 346, 1973.

[8] M. Xie, and M. Zhao, "The Schneidewind Reliability Model Revisited," Proceedings IEEE, 1992.

[9] M. Xie, Software Reliability Modeling, World Scientific, Singapore, 1991.

[10] S. Yamada, and S. Osaki, "Discrete SoftwareReliability Growth Models," Applied StochasticModels and Data Analysis, vol. 1, pp. 65-77, 1985.

[11] R. Subburaj, Software Reliability Engineering. McGraw-Hill Professional, New Delhi, 2015.

[12] N. Karunanithi, D. Whitley, and Y. K.Malaiya, "Using neural networks in reliability prediction," IEEE Software, Vol. 9, pp. 53–59, 1992.

[13] V. Almering, M. V. Genuchten, G. Cloudt, andP. J. Sormemans, "Using software reliability growth models in practice,"IEEE Software, 82-88, 2007.

[14] S. Keene and C. Lane, "Combined hardware and software aspects of reliability,"Quality Reliability Engineering International, vol. 8, pp. 419–426, 1992.

[15] F. Popentiu and D. N. Boros, "Software reliability growth supermodels," Microelectronics Reliability, vol. 36, pp. 485–491, 1996.

[16] S. Li, Q. Yin, P. Guo, and M. R. Lyu, "A Hierarchical mixture modelfor software reliability prediction," Applied Mathematical Computing, vol. 185, pp. 1120–1130, 2007.

[17] R. Subburaj, and A. M. J. Muthukumaran,"Dynamically Weighted Combination of Fault - based Software Reliability Growth Models," Indian Journal of Science and Technology, vol. 9, 2016.

[18] R.Subburaj, andC. A. S. DeivaPreetha,"Dynamically Weighted Combination Model for Describing Inconsistent Failure Data of Software Projects," Indian Journal of Science and Technology, vol. 9, 2016.

[19] Kai- Yuan Cai, Lin Cai, Wei-Dong Wang, Zhou-Yi Yu, and David Zhang, "On the neural network approach in software reliability modeling," Journal of Systems and Software, Vol. 58, pp. 47-62, 2001.

[20] Yogesh Singh, and Pradeep Kumar, "Prediction of Software Reliability Using Feed ForwardNeural Networks," in Proceedings IEEE, 2010.

[21] Yu- Shen Su, Chin- Yu Huang, "Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models," Journal of Systems and Software, Vol. 80, pp. 606-615, 2007.

[22] Jung- Hua LO, "The Implementation of Artificial Neural Networks Applying to Software Reliability Modeling", in Proceedings IEEE, 2009.

[23] Gaozu Wang, and Weihuai Li, "Research of Software Reliability Combinational Model Based on Neural Net," Second WRI World Congress on Software Engineering, 2010.

[24] Pratik Roy, G. S. Mahapatra, Pooja Rani, S. K. Pandey, and K. N.Dey, "Robust feed forward and recurrent neural network based dynamic weighted combination models for software reliability prediction," Applied Soft Computing, Vol. 22, pp. 629-637, 2014.

[25] IndhuraniLakshmanan, and SubburajRamasamy, "An Artificial Neural Network approach to Software Reliability Growth Modelling,"Elsevier Procedia Computer Science, vol. 57, pp. 695-702, 2015.

[26] GuoJunhong, Yang Xianozong, and Liu Hongwei, "Software Reliability Nonlinear Modeling and Its Fuzzy Evaluation," WSEAS International Conference on Non-Linear Analysis, Non-Linear Systems and Chaos, Bulgeria, pp.49-54, 2005.

[27] Sultan Aljahdali, and Alaa. F. Sheta, "Predicting the Reliability of Software Systems Using Fuzzy Logic," IEEE Proceedings of International Conference of Information Technology: New Generation, 2011.

[28] BhoopendraPachauri, Ajay Kumar, and JoydipDhar, "Modeling Optimal release policy under fuzzy paradigm in imperfect debugging environment," Information and Software Technology, vol. 55, pp. 1974-1980, 2013.

[29] Eduardo Oliveira Costa, Silvia R. Vergilio, Aurora Pozo, and Gustavo Souza, "Modeling Software Reliability Growth with Genetic Programming," Proceedings of IEEE International Symposium on Software Reliability Engineering, 2005.

[30] Chao-Jung Hsu, and Chin-Yu Huang, "Reliability Analysis Using Weighted Combinational Models for Web-based Software," Proceedings of WWW MADRID, ACM, Spain, 2009.

[31] Taehyoun Kim, Kwangkyu Lee, and JongmoonBaik, "An effective approach to estimating the parameters of software reliability growth models using a real-valued genetic algorithm," Journal of Systems and Software, vol.102, pp. 134-144, 2015.

[32] Chao-Jung Hsu, and Chin-Yu Huang, "Optimal Weighted Combinational Models for Software Reliability Estimation and Analysis," IEEE Transactions on Reliability, vol. 63, pp. 731-749, 2014.

[33] Q. P. Hu, M. Xie, S. H. Ng, and G. Levitin, "Robust recurrent neural network modeling for software fault detection and correction prediction," Reliability Engineering and System Safety, vol. 92, pp. 332-340, 2007.

[34] Jun Zheng, "Predicting Software reliability with neural network ensembles," Expert Systems with Applications, vol. 36, pp. 2116-2122, 2009.

[35] RamakantaMohanty, V. Ravi and M. R. Patra, "Hybrid intelligent systems for predicting software reliability," Applied Soft Computing, vol.13, pp. 189-200, 2013.

[36] Pratic Roy, G. S. Mahapatra, and K. N. Dey, "Neuro-genetic approach on logistic model based software reliability prediction," Expert systems with applications, vol. 42, pp. 4709-4718, 2015.

[37] LathaShanmugam, and Lilly Florence. "Enhancement and comparison of ant colony optimization for software reliability models," Journal of Computer Science, vol. 9, pp. 1232-1240, 2013.

[38] Changyou Zheng, Xiaoming Liu, Song Huang, and Yi Yao, "A parameter estimation method for software reliability models," Elsevier Procedia Engineering, vol. 15, pp. 3477-3481, 2011.

[39] A. A. S. Najla, and A. A. Marwa, "The use of Cuckoo search in estimating the parameters of software reliability growth models," International Journal of Computer Science and Information Security, vol. 11, 2013.

[40] Bushra Khalid, and Kapil Sharma, "Ranking of software reliability growth models using bacterial foraging optimization algorithm," Proceedings IEEE, 2015.

[41] Cong Jin, and Shu-Wei Jin, "Parameter optimization of software reliability growth model with S-Shaped testing-effort function using improved swarm intelligent optimization," Applied Soft Computing, vol. 40 pp. 283-291, 2016.

[42] SubburajRamasamy and IndhuraniLakshmanan, "Application of artificial neural network for software reliability growth modeling with testing effort," Indian Journal of Science and Technology, vol. 9, 2016.

[43] T. M. Khoshgoftaar,"Non-Homogeneous Poisson Processfor Software Reliability," COMPSTAT, pp. 13–14, 1988.

[44] C. Stringfellow, and A. Amschler Andrews, "An empirical method for selection software reliability growth models,"Empirial Software Engineering, vol. 7, pp.319-343, 2002.

[45] K. Sharma, R. Garg, C. K. Nagpal, R. K. Garg, "Selection of optimal software reliability growth models usingadistancebasedapproach," IEEETransactions on Reliability, vol.59, pp.266–276, 2010.

[46] Liang Fuh Ong, MohdAdham Isa, N. A. Dayang, Jawawi, Shahliza, and Abdul Halim, "Improving software reliability growth model selection ranking using particle swarm optimization," Journal of Theoritical and Applied Information Technology, vol. 95, 2017.

[47] V. Kharchenko, O. Tarasyuk, V. Sklyar, and V. Dubnitsky, "The method of software reliability growth models choice using assumptions matrix," IEEE Proceedings, 2002.

[48] Rakesh Rana et al., "Selecting software reliability growth models and improving their predictive accuracy using historical projects data," The Journals of Systems and Software, vol. 98, pp.59-78, 2014.

[49] Jinhee Park, and JohgmoonBaik, "Improving software reliability prediction through multi-criteria based dynamic model selection and combination," The Journal of Systems and Software, vol. 101, pp. 236-244, 2015.

[50] S. Yamada, M. Ohba, and S. Osaki, "S-Shaped reliability growth modeling for software error detection," IEEE Transactions on Reliability," vol. R-32, pp.475-478, 1983.

[51] M. Ohba, "Inflection s-shapedsoftwarereliabilitygrowthmodel," Stochastic models inreliabilitytheory, SpringerBerlinHeidelberg, pp.144–62, 1984.

[52] P. K. Kapur, and R. B. Garg, "Optimal software release policies for software reliability growth models under imperfect debugging," Operations Research, vol. 24, pp. 295-305, 1990.

[53] C. Y. Huang, and S. Y. Kuo, "Analysis of incorporating logistic testing effort function into software reliability modeling," IEEE Transactions on Reliability, vol. 51, pp. 261–270, 2002.

[54] J. D. Musa, and A. F. Ackerman, "Quantifying software validation: When to stop testing?," IEEE Software, pp. 19-27, 1989.

[55] S.Yamada,K.Tokuno,andS.Osaki,"Imperfectdebuggingmodelswithfaultintroductionrateforsoftwarereliabilityassessment," International Journal of System Science, vol. 23, pp. 2241–2252, 1992.

[56] H. Pham, and X. Zhang, "AnNHPPsoftwarereliabilitymodelanditscomparison," International Journal of ReliabilityQuality and Safety Engineering, vol. 4, pp. 269–282, 1997.

[57] H. Pham, L. Nordmann, and X. Zhang, "Ageneralimperfectsoftwaredebuggingmodel withs-shapedfault-detectionrate," IEEETransactions on Reliability, vol. 48, pp. 169–175, 1999.

[58] J.T. Duane, "Learning curve approach to reliability monitoring," IEEE Transactions on Aerospace," vol. AS-2,pp. 563-566, 1964.

[59] B. Littlewood, "Rationale for a modified Duane model," IEEE Transactions on Reliability, vol. R-33, pp. 157- 159, 1984.

[60] M. Xie, and M. Zhao,"On some reliability growth models with simple graphical interpretations," Microelectronics and Reliability, vol. 33,pp. 149-167, 1993.

[61] R. Subburaj, and Gopal G,"Generalized exponential Poisson model for software reliability growth," International Journal of Performability Engineering, vol. 2, pp. 291–301, 2006.

[62] R. Subburaj, G. Gopal, and P. K. Kapur, "A software reliability growth model for Vital Quality Metrics," South African Journal of Industrial Engineering, vol. 18, pp. 93-108, 2007.

[63] R. Subburaj, and G. Gopal, "Software Reliability Growth Model Addressing Learning," Journal of Applied Statistics, vol. 35, pp. 1151 – 1168, 2008.

[64] R. Subburaj, G. Gopal, and P. K. Kapur,"A software reliability growth model for estimating debugging and the learning indices," International Journal of Performability Engineering, vol. 8, pp. 539–549, 2012.

[65] S. Yamada,"Software reliability growth model with testing effort," IEEE Transactions on Reliability, vol. 35, pp. 422-424, 1986.

[66] S. Yamada,"Software reliability growth model with Weibull testing effort," IEEE Transactions on Reliability,vol. 42, pp. 100–106, 1993.