International Journal of Mathematical, Engineering and Management Sciences
Vol. 8, No. 3, 504-513, 2023
https://doi.org/10.33889/IJMEMS.2023.8.3.028

Modeling the Role of Testing Coverage in the Software Reliability Assessment

Sudeep Kumar
Department of Mathematics,
AIAS, Amity University, Uttar Pradesh, 201303, India.
Corresponding author: sudeepdeepak777@gmail.com

Anu G. Aggarwal
Department of Operations Research,
University of Delhi, Delhi, India.
E-mail: anuagg17@gmail.com

Ritu Gupta
Department of Mathematics,
AIAS, Amity University, Uttar Pradesh, 201303, India.
E-mail: rgupta@amity.edu

(Received on October 10, 2022; Accepted on February 28, 2023)

Abstract
To assure the reliability and quality of the final product, testing is an essential and crucial part in the software development cycle. During this process, fault correction/detection activities are carried out to increase the reliability of the software. The non-homogeneous Poisson Process (NHPP) is the basis of the investigated software reliability growth models (SRGMs), which are based on the supposition that the number of faults found is affected by the amount of code covered during testing and that the amount of code covered during testing depends on the testing effort expended. This research takes into consideration several testing coverage functions: exponential, delayed S-shaped and logistic distributions, to propose three SRGMs that are based on testing efforts. For testing effort expenditure Weibull distribution has been employed. Two real failure datasets have been utilised to validate the proposed models, and their performance is evaluated using four goodness-of-fit metrics, including predictive ratio risk (PRR), coefficient of determination (R^2), predictive power (PP) and mean square error (MSE). Sensitivity analysis of cost requirement-based release time of software for exponential function has been done by using a genetic algorithm, which minimized the overall cost of the software subject to the requirement for reliability.

Keywords- Testing Coverage, Software reliability growth models, Non-homogeneous poisson process, Software reliability, Release planning, testing effort.

1. Introduction

In recent times, the entire world has witnessed a revolution in information technology with significant advancements at an unprecedented rate. Technological breakthroughs are happening at an exponential rate and these new developments frequently manifest in the shape of a new product. As a reason, the requirement for high-quality software systems has increased to keep up with technology advancements. With computers and technology infiltrating every aspect of our lives, software failure or inappropriate functioning can result in major losses. As the demand for digitalization grows, most industries and sectors such as education, aviation, healthcare and banking are shifting toward computerization. As software systems have grown in complexity, the importance of well-planned and effective testing has grown many folds. There have already been numerous cases where the failure of computer-controlled systems has resulted in massive losses of money and human lives.

Software reliability engineering is crucial across the software life cycle to enhance the software systems quality. It is necessary to have high reliability while designing a complicated and large real-time software-embedded system. As we all know, software can fail for a variety of reasons, but none of them entail wear and tear. Typically, the software fails due to a design issue. The reliability of software is one of its most significant quality attributes. It is also crucial to evaluate the software's reliability before its release in order to avoid risk and maintenance costs. A software system's reliability refers to its ability to function for a fixed period of time under certain operational and environmental circumstances (Haque and Ahmad, 2021). As a result, there is a significant need for high-quality software. However, many systems suffer from poor performance due to faulty software. To ascertain the ideal timing to release a software system, it is also necessary to examine software reliability and accuracy.

Software development is a complicated process with many challenges that have yet to be resolved. In the literature, SRGMs have been designed for certain testing conditions and assumptions. During testing, the majority of them do not take into account the pattern of testing resource consumption such as human resources and computer time. However, in practice, software models should include testing effort, which can be measured human work in hours, test cases completed, hours of CPU and so on. The 'Testing Effort Control' analysis aids in long-term user-client relationships by ensuring a high-quality product. Models based on SRGMs are used to forecast reliability and software fault content over time. There is a direct correlation between software's reliability and the quantity of testing it has undergone. The more time spent testing, the more reliable the software becomes. Furthermore, the testing process accelerates, when the amount of testing effort increases. Yamada et al. (1987) studied applications of SRGM based on testing efforts.

It is quite accurate that software cannot be made perfect/fault-free, which is why testing consumes approximately half of all software development costs. One of these difficulties is testing coverage. Test coverage is a software metric that can be used to evaluate a testing process's effectiveness, ineffectiveness and completeness. A testing effort and time-dependent consecutive release modeling of software have been established in the presence of testing coverage. Both consumers and developers value testing coverage because it can help them enhance the performance of tested software and determine what more effort is required to enhance the software's reliability. It is critical to continue testing until all the program's structures acquire the appropriate level of coverage in order to verify the software's quality. On the other side, when customers expect to buy or utilize software items, testing coverage might provide a quantitative confidence criterion. In the presence of testing coverage, a testing effort and time-dependent consecutive release modeling of software have been established by Chatterjee et al. (2022).

An extensive introduction to the proposed study is followed by a summary of the remaining work in this paper as follows: the review of the literature is covered in the second section. In section three, we create the SRGM based on NHPP for three models. Thus, in models 1, 2, and 3, the exponential, delayed S-shaped, and logistic functions, respectively, represent the testing coverage function. With the use of the data set, we demonstrate the numerical findings in section four to validate the suggested model. On the basis of the proposed models, section five presents the optimal release timing. In section six, the managerial implication of the real-world situation is given. In section seven, we summarize the entire research and outline the paper’s aims.

2. Review of Literature

In the past few decades, there has been substantial research on how test coverage affects reliability measurement, and many statistical models have been proposed out for the prediction and assessment of software reliability. Software reliability growth with testing coverage was explored by Malaiya et al. (2002).

Performance analysis of the SRGM has been performed using change point and testing effort by Huang (2005). An SRGM was proposed by Kapur et el. (2009) and Li et al. (2010) with testing effort. Optimal release policy and NHPP-based SRGM with imperfect debugging and logistic-exponential test coverage were proposed by Chatterjee and Singh (2014). A NHPP based SRGM by incorporating testing coverage and imperfect debugging has been developed by Li and Pham (2017). Gupta et al. (2019) proposed a debugging time lag based SRGM. Change point was used by Arora and Aggarwal (2020) to explore how testing effort affected the software reliability evaluation. Tandon and Aggarwal (2020) proposed a multi-release SRGM with a fault reduction factor dependent on testing coverage. Huang et al. (2022) developed an SRGM by incorporating imperfect debugging. Kim et al. (2022) proposed the software reliability model for the study of optimal release time.

According to the literature review, there have been few research articles published on testing coverage-based SRGMs with effort expenditure. According to the authors' best knowledge, there is a research gap in the domain of software reliability modeling with different test coverage functions and effort expenditure as Weibull distribution. The proposed study gives three SRGMs insight into the notion of testing coverage and effort expenditure. First, the Weibull distribution function is used for testing effort expenditure because it is flexible to fit a wide variety of failure data and has been employed to address a variety of problems in a broad range of fields (Bokhari and Ahmad, 2007). Second, in this study, testing coverage is considered because it gives quantifiable metrics that may be used to show how testing is going. Third, by including optimization problems, software developers can better determine the appropriate the cost components and project delivery time that have a significant impact on both project delivery time and overall cost.

3. Model Description and Assumptions

For the past five decades, Many researchers have found software reliability modeling to be an interesting area of study, and multiple SRGMs have been proposed in the literature. SRGMs are mathematical functions that describe the relationship between the no. of faults detected by the testing team and testing duration. By taking into account a variety of testing-related criteria, such as kind of faults, team’s experience and skill, resources available, test case design, team size, project complexity, etc., this aids in analysing the reliability growth as the testing continues.

3.1 Basic Assumptions

(i) The occurrence of software faults is governed by a NHPP.
(ii) The number of detected faults in the system is proportional to the faults remaining in it.
(iii) Number of faults detected is affected by the amount of code covered during testing.
(iv) Faults are removed using testing efforts.
(v) Code covered during testing is modeled by using three types of distribution functions namely exponential, delayed S-shaped and logistic.
(vi) Testing coverage is a function of efforts spent.

3.2 Notations

a	– no. of initial faults (constant)
b	– fault detection rate (constant)
β	– logistic function scale parameter (constant)
t	– calendar time
$W, W(t)$	– testing effort function
$\bar{W}$	– total available testing effort (constant)

$c(W), c$	– testing coverage function
$m, m(W), m(t)$	– mean value function (MVF)
ζ/γ	– Weibull testing effort function shape/scale parameter
ϕ	– constants.

An NHPP-based model of software reliability growth is developed as a result of this assumption and the following expression provides the basis for our proposed model's mean value function.

$$\frac{dm(t)}{dt} = \frac{dm}{dc}\frac{dc}{dW}\frac{dW}{dt} \tag{1}$$

where, $\frac{dm}{dc}$ = no. of faults detected with respect to amount of testing coverage,

$\frac{dc}{dW}$ = rate of change of testing coverage with respect to testing effort expenditure.

These two components may be modeled as:

$$\frac{dm}{dc} = \frac{c'(W)}{1-c(W)}(a - m) \tag{2}$$

$$\frac{dc}{dW} = \phi \text{ (constant)} \tag{3}$$

Here, $\frac{c'(W)}{1-c(W)}$ is fault detection rate. By combining and solving differential equations (1), (2) and (3), the model's MVF can be determined. The MVF is given as follows:

$$m(W) = a\left[1 - \left(1 - c(W)\right)^{\phi}\right] \tag{4}$$

The Weibull distribution function is used for testing effort expenditure.

$$W(t) = \bar{W}\left(1 - e^{-\gamma t^{\zeta}}\right) \tag{5}$$

The MVF in equation (4) is a generalized model. In this paper three different types of testing coverage functions have used to derive three models given as follows in Table 1.

Table 1. Testing coverage and mean value function for three different models.

Model no.	Model name	$c(W)$	$m(W(t))$
1.	Exponential Model	$[1 - e^{-bW}]$	$a[1 - (e^{-bW})^{\phi}]$
2.	Delayed S-Shaped Model	$\left[1 - \left(1 + b(W)\right)e^{-bW}\right]$	$a\left[1 - \left(\left(1 + b(W)\right)e^{-bW}\right)^{\phi}\right]$
3.	Logistic Model	$\left[\frac{1 - e^{-bW}}{1 + \beta e^{-bW}}\right]$	$a\left[1 - \left(\frac{(1 + \beta)\, e^{-bW}}{1 + \beta e^{-bW}}\right)^{\phi}\right]$

4. Numerical Results

To assess the performance of effort-dependent SRGM proposed with testing coverage functions, the parameter estimation of these models has been performed on two data sets: DS I- software testing data acquired from Tandem Computers and DS II- Radar systems given by Wood (1996) and Brooks and Motley (1980) respectively, by using a non-linear regression technique. The estimated values for the parameters of the Weibull testing effort function are shown in Table 2.

Table 2. Parameter estimation of Weibull testing effort function.

Testing effort function	Parameters estimated	DS-I estimated values	DS-II estimated values
Weibull	W'	11740.754	2669.916
	γ	0.024	0.001
	ζ	1.460	2.069

Now the parameters of the three proposed models (MVF) given in Table 1 will be estimated. Table 3 presents the estimated results of parameters of the purposed models 1, model 2 and model 3 respectively, and Figure 1-3 depicts the fitting of the proposed model with different testing coverage functions to a given data set. The goodness of fit curves for a different type of coverage function as shown in Figure 1-3 for DS-I and Figures 4-6 for DS-II, clearly indicate that the models fit the actual data excellently.

Table 3. Parameter estimation results of developed models 1, 2 and 3.

Testing coverage function		DS-I			DS-II		
		Exponential	Delayed S-shaped	Logistic	Exponential	Delayed S-shaped	Logistic
Estimated values	a	122	135	131	1661	1693	1402
	b	0.008	0.071	0.071	0.003	0.335	0.002
	ϕ	0.017	0.002	0.002	0.308	0.002	0.669
	β	-	-	75.601	-	-	1.288

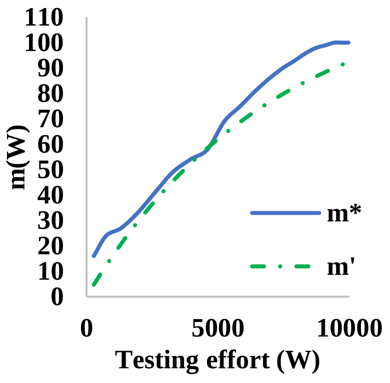

Figure 1. Goodness of fit curve for proposed model 1 for DS-I.

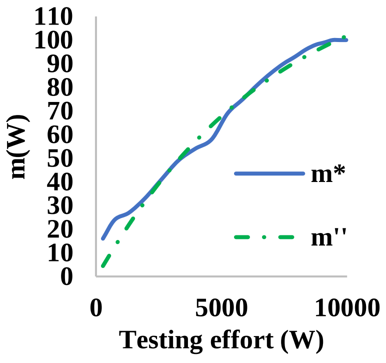

Figure 2. Goodness of fit curve for proposed model 2 for DS-I.

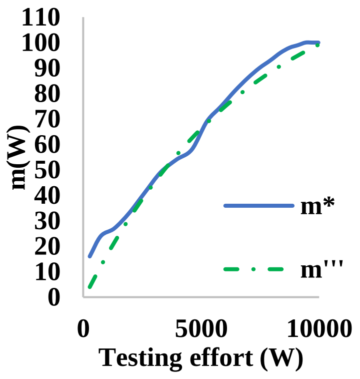

Figure 3. Goodness of fit curve for proposed model 3 for DS-I.

where, the actual and estimated no. of faults of models 1, 2 and 3 are m*, m', m'' and m''' respectively for the Tandem data set.

The estimation is done using the SPSS tool and four goodness-of-fit criteria, namely predictive ratio risk (PRR), coefficient of determination (R^2), predictive power (PP) and MSE are used to demonstrate the estimation capabilities of proposed models (Table 4 and 5). The total no. of observations is k and $m(t_j)$ and x_j are estimated and actual values of the mean value function.

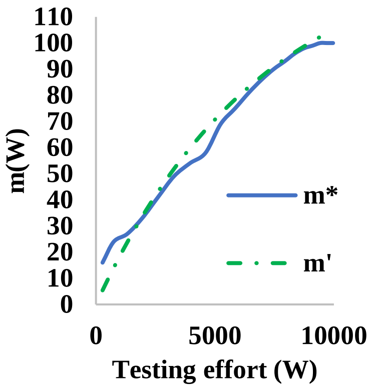

Figure 4. Goodness of fit curve for proposed model 1 for DS-II.

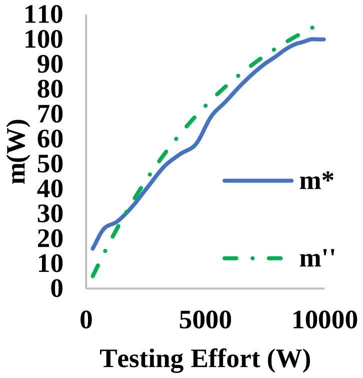

Figure 5. Goodness of fit curve for proposed model 2 for DS-II.

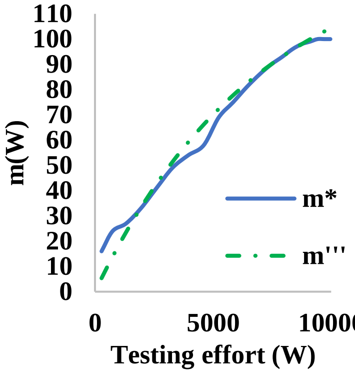

Figure 6. Goodness of fit curve for proposed model 3 for DS-II.

Table 4. Goodness of fit criteria.

Performance Criteria	R^2	MSE	PP	PRR
Expression	$1-\frac{\text{residual ss}}{\text{actual ss}}$; ss= sum of squares	$\sum_{j=1}^{k}\frac{\left(m(t_j)-x_j\right)^2}{k}$	$\sum_{j=1}^{k}\left(\frac{m(t_j)-x_j}{x_j}\right)^2$	$\sum_{j=1}^{k}\left(\frac{m(t_j)-x_j}{m(t_j)}\right)^2$
Interpretation	The model fits the data better with a higher value of R^2.	Lower the value of MSE, less fit of error i.e. better the model fits to data	With a lower PP value, the model fits the data better.	The model fits the data better with a lower value of PRR.

Table 5. Goodness of fit criteria for developed models 1, 2 and 3.

	DS-I			DS-II		
Models	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3
R^2	0.976	0.978	0.976	0.994	0.994	0.997
MSE	68.49883	18.70733	24.8695	6420.484	6534.017	6075.331
PP	0.932923	0.523161	0.883065	1.291244	2.708052	2.720831
PRR	6.895268	6.818445409	10.24172601	2.098579	66.06595	18.87345

5. Cost Requirement based Release Time of Software

Determination of optimal release time for a software project is one of the most difficult problems to solve. Since cost and reliability are such important considerations, when to finish software testing and release it is frequently decided using them. Many researchers; Pham and Zhang (2003), Li et al. (2010) and Gupta et al. (2019) discussed the release time optimization problem in the study. This section discusses about the proposed SRGM's cost model. During the software operational and testing phases, when determining the optimum release time, we can also consider the overall cost into account.

Let $TC(W)$ be the software system's cost function for testing effort and is given by as:

$$TC(W) = C_1 m(W) + C_2\left(a - m(W)\right) + C_3 W \tag{12}$$

where,

$C_1 -$ unit cost for removal of a fault during testing phase.

$C_2 -$ unit cost for removal of a fault during operational phase; $C_2 > C_1$.
$C_3 -$ testing cost per unit effort.

A software system's release time is a trade-off between the cost and reliability. Our objective is to identify a release time problem that minimises testing costs while still achieving reliability standards.

Minimize $TC(W) = C_1 m(W) + C_2\big(a - m(W)\big) + C_3 W$
subject to $R_S(W) \geq R_0; W \geq 0.$

where, R_0 $(0 < R_0 < 1)$ is the pre-specified reliability and software reliability $R_S(W)$ is given by

$$R_S(W) = \frac{m(W)}{a}.$$

Here $R_S(W)$ basically represents the proportion of faults removed by spending W amount of effort and is taken as a measure of software reliability (cf. Huang et al., 2002).

5.1 Numerical Example

The cost requirement-based release time of software for exponential function (Model 1) given in Table 1 for the Tandem data set has been done by using a genetic algorithm, which minimizes the overall cost of the software subject to the requirement for reliability by setting the cost parameters as $C_1 = 35$ units, $C_2 = 100$ units, $C_3 = 1$ unit. The required reliability to be obtained by the release time is also assumed to be 0.8, i.e., R_0 =0.8 The optimal amount of effort before release is 11334 units. The minimum expected software cost's 17094 units. The minimum cost of software for distinct levels of reliability is given in Table 6.

Table 6. Minimum cost and amount of efforts required for different levels of reliability.

R_0	0.7	0.75	0.8	0.85	0.9	0.95
TC(W)	16518.68	16662.64	17094.07	17980	19695.39	23436.71
W'	8478.681	9762.636	11334.07	13360	16215.38	21096.7

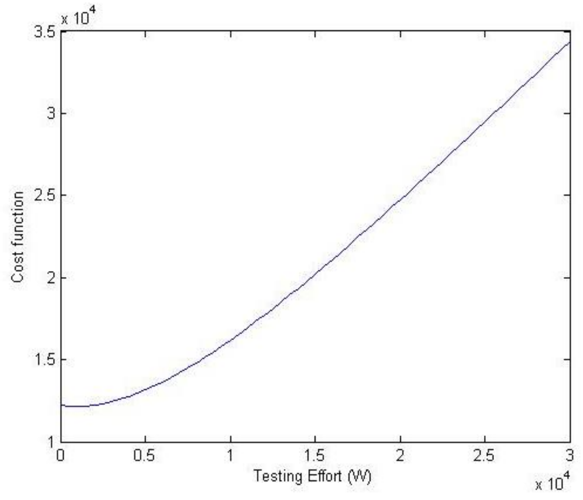

Figure 7. Variation of cost w.r.t. testing effort (W).

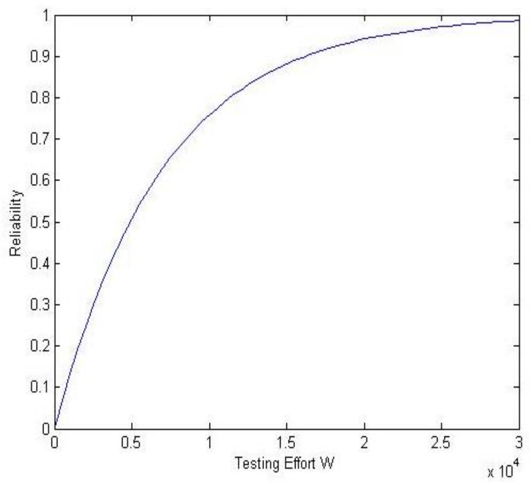

Figure 8. Variation of reliability w.r.t. testing effort (W).

Table 6 clearly indicates that the total software cost and amount of testing efforts increase by increasing the required reliability of the software. Figures 7 and 8 show the influence of testing effort expenditure on the software's total cost and reliability respectively.

In Figures 7-8, we may observe the effect of testing efforts expenditure on the cost and reliability of the software. The total software cost first decreases by increasing the efforts and after attaining the optimal value it is increasing gradually with increasing testing efforts as shown in Figure 7. Also, Figure 8 depicts the effect of testing efforts expenditure on the software reliability. The software reliability increases as we increase the testing efforts expenditure.

6. Managerial Implications

The attention of a reliability-focused problem is on the requirements and expectations of the customer, whereas a cost-focused problem that tries to lower development costs fully which is taken into account for the developer's perspective. According to the study, reliability and cost should both be priorities for a high-quality software product. The study's findings have a number of implications for how the development team should go about developing software under reliability and financial limitations. The objective is to increase the product's reliability while developing high-quality software. Testing duration and cost are influenced by reliability level; for example, if software's reliability needs to be increased, more testing will be required, which will cost more money and resources. More time is needed to reach a goal that has been set too high. Therefore, companies must concentrate on reliability, testing duration, and budget. However, the companies must test the software thoroughly to ensure that a high degree of reliability which has been attained. Incorporating both of the aforementioned factors into an optimization model aids in addressing both management and user perspectives by satisfying their aspirations and needs.

7. Conclusion

Software developers strive to increase the reliability of the software by offering desirable features to customers and to assist developers in achieving this goal, many SRGMs have previously been proposed. The testing coverage is frequently used by customers and developers to demonstrate how fully the software has been tested and to express their confidence in its readiness. The use of reliability growth projections helps software engineers meet their deadlines while maintaining high reliability levels. We have proposed testing effort based SRGM incorporating the testing coverage function in the model development. In models 1, 2 and 3 testing coverage function is represented as Exponential, Delayed S-shaped and Logistic functions respectively, whereas the testing effort function is assumed to be Weibull. The accuracy and validity of proposed models have been tested on two real failure data sets provided by Wood (1996) and Brooks and Motley (1980) by using estimated values found by the SPSS tool. The goodness-of-fit of all three models was tested using software testing data. Four criteria namely predictive ratio risk (PRR), coefficient of determination (R^2), predictive power (PP) and MSE have been examined to find the goodness of fit of models. Sensitivity analysis of cost requirement-based release time of software for exponential function has been done by using a genetic algorithm, which minimized the total cost of the software subject to reliability requirement. The findings of this research into creating a model for software validation datasets are pretty promising. These models will aid developers in making sensible software development decisions. This paper only mentioned the optimization problem for an exponential model but in the future, researchers can study other models by adding more constraints and objectives to the problem. In future models, this work could be extended by incorporating multiple change points, fault reduction factor, multi-release problems and a variety of other factors in the environment.

Conflict of Interest

No conflict of interest.

Acknowledgements

The author (Sudeep Kumar) gratefully acknowledges financial support from the CSIR, New Delhi (File No. 09/915(0018)/2019-EMR-I).

References

Arora, R., & Aggarwal, A. (2020). Testing effort based software reliability assessment incorporating FRF and change point. *Yugoslav Journal of Operations Research*, *30*(3), 273-288. https://doi.org/10.2298/YJOR190315022A.

Bokhari, M.U., & Ahmad, N. (2007). Software reliability growth modeling for exponentiated Weibull function with actual software failures data. In *Innovative Applications of Information Technology for The Developing World* (pp. 390-395). https://doi.org/10.1142/9781860948534_0062.

Brooks, W., Motley, R. (1980). Analysis of discrete software reliability models. *IBM Federal Systems Division.* ADA086334. Gaithersburg, Maryland.

Chatterjee, S., & Singh, J.B. (2014). A NHPP based software reliability model and optimal release policy with logistic–exponential test coverage under imperfect debugging. *International Journal of System Assurance Engineering and Management*, *5*(3), 399-406. https://doi.org/10.1007/s13198-013-0181-6.

Chatterjee, S., Saha, D., Sharma, A., & Verma, Y. (2022). Reliability and optimal release time analysis for multi up-gradation software with imperfect debugging and varied testing coverage under the effect of random field environments. *Annals of Operations Research*, *312*(1), 65-85. https://doi.org/10.1007/s10479-021-04258-y.

Gupta, R., Jain, M., Jain, A. (2019). Software reliability growth model in distributed environment subject to debugging time lag. In: Deep, K., Jain, M., Salhi, S. (eds) *Performance Prediction and Analytics of Fuzzy, Reliability and Queuing Models*. Asset Analytics. Springer, Singapore. https://doi.org/10.1007/978-981-13-0857-4_7.

Haque, M.A., & Ahmad, N. (2021). An effective software reliability growth model. *Safety and Reliability*, *40*(4), 209-220. https://doi.org/10.1080/09617353.2021.1921547.

Huang, C.Y. (2005). Performance analysis of software reliability growth models with testing-effort and change-point. *Journal of Systems and Software*, *76*(2), 181-194. https://doi.org/10.1016/J.JSS.2004.04.024.

Huang, C.Y., Lo, J.H., Kuo, S.Y., & Lyu, M.R. (2002). Optimal allocation of testing resources for modular software systems. In *13th International Symposium on Software Reliability Engineering* (pp. 129-138). IEEE. Annapolis, USA.

Huang, Y.S., Chiu, K.C., & Chen, W.M. (2022). A software reliability growth model for imperfect debugging. *Journal of Systems and Software*, *188*, 111267. https://doi.org/10.1016/j.jss.2022.111267.

Kapur, P.K., Shatnawi, O., Aggarwal, A.G., & Kumar, R. (2009). Unified framework for developing testing effort dependent software reliability growth models. *WSEAS Transactions on Systems*, *8*(4), 521-531.

Kim, Y.S., Song, K.Y., Pham, H., & Chang, I.H. (2022). A software reliability model with dependent failure and optimal release time. *Symmetry*, *14*(2), 343. https://doi.org/10.3390/sym14020343.

FLi, Q., & Pham, H. (2017). A testing-coverage software reliability model considering fault removal efficiency and error generation. *Plos One*, *12*(7), e0181524. https://doi.org/10.1371/journal.pone.0181524.

Li, X., Xie, M., & Ng, S.H. (2010). Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points. *Applied Mathematical Modelling*, *34*(11), 3560-3570. https://doi.org/10.1016/j.apm.2010.03.006.

Malaiya, Y.K., Li, M.N., Bieman, J.M., & Karcich, R. (2002). Software reliability growth with test coverage. *IEEE Transactions on Reliability*, *51*(4), 420-426. https://doi.org/10.1109/TR.2002.804489.

Pham, H., & Zhang, X. (2003). NHPP software reliability and cost models with testing coverage. *European Journal of Operational Research*, *145*(2), 443-454. https://doi.org/10.1016/S0377-2217(02)00181-9.

Tandon, A., & Aggarwal, A.G. (2020). Testing coverage based reliability modeling for multi-release open-source software incorporating fault reduction factor. *Life Cycle Reliability and Safety Engineering*, *9*, 425-435. https://doi.org/10.1007/s41872-020-00148-7.

Wood, A. (1996). Predicting software reliability. *Computer*, *29*(11), 69-77. https://doi.org/10.1109/2.544240.

Yamada, S., Ohtera, H., & Narihisa, H. (1987). A testing-effort dependent software reliability model and its application. *Microelectronics Reliability*, *27*(3), 507-522. https://doi.org/10.1016/0026-2714(87)90469-0.