ISSN No. 0976-5697

Volume 2, No. 1, Jan-Feb 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Comparative Overview of Software Reliability Growth Models

S. M. K Quadri
Department of Computer Sciences
Kashmir University
Srinagar,India
quadrismk@hotmail.com

Razeef Mohd*
Department of Computer Sciences
Kashmir University
Srinagar,India
m.razeef@gmail.com

Nesar Ahmad
University Dept of Statistics and Computer Applications,
T.M. Bhagalpur University,
Bhagalpur, India.
nesar_bgp@yahoo.co.in

Abstract: A software reliability growth model is one of the fundamental techniques used to assess software reliability quantitatively. The software reliability growth model is required to have a good performance in terms of goodness-of-fit, predictability, and so forth. In this paper, we will summarize some existing Software Reliability Growth Models (SRGM's). Our main focus in this paper will be to provide a comparative overview of the various SRGM's in general. The comparison will mainly be on the basis of what type of Testing-Effort Function (TEF) is used by which type of probability distribution function, to describe Testing-Effort curve. First, we provide overview of Software Reliability and Software Reliability Growth Model. Next, with the intention of comparing various SRGMs we provide the Testing Effort Functions (TEFs) given by various researchers. Then actual software data from the software projects have been used to demonstrate the comparison of SRGMs. The evaluation results from the various SRGMs are analyzed and compared to show which SRGM has fairly better prediction in estimating number of remaining faults, expected number of errors, future failure behavior from present and past failures, optimal release time and which SRGM comparatively describes the actual expenditure pattern more faithfully during software development process.

Keywords: Software Reliability, Testing-Effort Function (TEF), Non-Homogenous Process(NHPP), Probability Distribution Function (PDF), Fault, Failure, Software Reliability Growth Model (SRGM), Testing-Effort curve, Optimal Release Time, Failure-Intensity.

I. INTRODUCTION

Over the last few decades, there has been fast growth in software development process and software management. Efforts are being made to produce quality and reliable software efficiently and effectively. As a result software reliability has become a great concern for both developers and users. Software reliability is one of the important parameters of software quality and system dependability. It is defined as the probability of failure-free software operation for a specified period of time in a specified environment [16], [17], [18], [9]. In highly complex modern software systems, reliability is the most important factor, since it quantifies software failures during the process of software development and software quality control. Software reliability can also be defined for software as the probability of execution without failures for some specified interval of natural units or time [18]. A failure is a departure of system behavior in execution from user requirements and it is the result of a fault. A fault is a defect that causes or can potentially cause the failure when executed [18]. The models applicable to the assessment of software reliability are called SRGMs. SRGM are useful for estimating how software reliability improves as faults are detected and repaired. It can be used to predict when a particular level of reliability is likely to attained and also helps in determining when to stop testing to attain a given reliability level. SRGMs help in decision making in many software development activities such as number of initial faults, failure intensity, reliability within a specified interval of time period, number of remaining faults, cost analysis and release time etc. A software reliability model describes failures as random process as failures are result of two processes: The introduction of faults and then activation through selection of input states, both of these processes are random in nature. So SRGM is generally described as the probability distribution of the value of the random process at each point in time. SRGMs are developed in general by probability distribution of failure times or the number of failures experienced and by the nature of the random process with time. To achieve highly reliable software systems, many software fault detection/ removal techniques can be used by programmers or testing teams. In applying these techniques, the SRGM are important, because they can provide quite useful information for developers and testers during the testing/debugging phase. Many researchers have also tried to compare various SRGMs [25] by using actual failure data.

Numerous SRGMS have been developed during the last three decades and they can provide very useful information about how to improve reliability [16], [18], [29], [21]. The effort index or the execution time is better time domain for software reliability modeling than the calendar-time because the shape of observed reliability growth curve depends strongly on the time distribution of testing-effort [17], [20].

II. TESTING EFFORT BASED SOFTWARE RELIABILITY MODELING

[a] Testing-Effort Function:

In software testing many testing-efforts are consumed, such as the CPU time, the human power and executed test cases. So testing-effort can be measured by the human power, the number of test cases the number of CPU hours, etc [26].Software reliability models should be developed by incorporating the Testing-Effort Functions (TEFs) in real development environment. When applied extensively to real software development projects, these models provide a reasonable fit to the observed data and give insightful interpretations for the resource consumption process during the software development [7][8]. Much testing-effort is consumed during software testing phase. The consumed testing-effort indicated how the errors are detected effectively in the software and can be modified by different distributions [31], [18], [17], [28], [30], [32], and [11]. Many authors have incorporated different Testing-Effort Functions in software reliability growth modeling. Some of them are below:

[b] Different Testing- Effort functions:-

[a] Yamada Exponential Curve [31]: For $\theta = 1$ and $m = 1$, there is an exponential testing-effort function, and the cumulative testing-effort consumed in time (0, t) is:

$W(t)=\alpha.(1-e^{-\beta t}),\alpha>0,\beta>0.$

[b] Yamada Rayleigh curve [32]: For $\theta=1$ and $m=2$ there is a Rayleigh testing-effort function, and the cumulative testing-effort consumed in time $(0,t]$ is:

$W(t)=\alpha.(1-e^{-\beta t^2}),\alpha>0,\beta>0.$

[c] Yamada Weibull curve [24]: For $\theta=1$ there is a Weibull testing-effort function, and the cumulative testing-effort consumed in time $(0,t]$ is:

$W(t)=\alpha.(1-e^{-\beta t^m}),\alpha>0,\beta>0,m>0.$

[d] Generalized Exponential curve [21]: For $m = 2$ there is a generalized exponential testing-effort function, and the cumulative testing-effort consumed in time $(0,t]$ is:

$W(t)=\alpha.(1-e^{-\beta t})^{\theta},\alpha>0,\beta>0,\theta>0.$

[e] Burr Type X curve [22]: For $m = 2$, there is Burr type X testing-effort function, and the cumulative testing-effort consumed in time $(0,t]$ is:

$W(t)=\alpha.(1-e^{-\beta t^2})^{\theta},\alpha>0,\beta>0,\theta>0.$

[f] New Modified Weibull curve [23]: proposed the NMW testing-effort function, and the cumulative testing-effort consumed in time $(0,t]$ is: $W(t)=\alpha.(1-e^{-\beta t^m e^{\delta t}}),\alpha>0,\beta>0,m\geq0,\delta>0.$

Where $\alpha,\beta,m,\delta and,\theta$ are constant parameters, α is the total amount of testing-effort expenditures; β and δ are the scale parameters, and m,θ are shape parameters.

III. SOFTWARE RELIABILITY GROWTH MODEL

The mathematical expression of TE-based is:

$$\frac{dm(t)}{dt}/w(t)=r(t)\cdot[a-m(t)],a>0,\ 0<r(t)<1 \quad (1)$$

$$\frac{dm(t)}{dt}=w(t)\cdot r(t)\cdot[a-m(t)],a>0,\ 0<r(t)<1 \quad (2)$$

The basic SRGM is based on the following assumptions:

[a] The fault removal process is NHPP.

[b] The software system is subject to failures at random times caused by the manifestation of remaining faults in the system.

[c] The mean number of faults detected in $(t,t+\Delta t]$, $\frac{dm(t)}{dt}$ by the current $W(t)$ is proportional to the mean number of remaining faults in the system.

[d] $r(t)$ is function of time (not just a constant).

[e] The time dependent behavior of TE can be modelled by the logistic, Weibull, Rayleigh or Exponential distribution.

[f] Each time a failure occurs, the fault that caused it is immediately and perfectly removed, and no new faults are introduced.

[g] Correction of errors takes only negligible time and detected error is removed with certainty.

Non-Homogeneous Poisson Process (NHPP) as a stochastic process has been successfully used in the reliability study of software system. For stochastic modeling of software error detection phenomenon, a counting process is defined as $N(t),t\geq0$, where $N(t)$ represents the cumulative number of software errors detected by testing time t with mean value function $m(t)$.SRGM based on NHPP under the assumption of Goel and Okumoto (1997) is formulated as:

$$\Pr\{N(t)=n\}=\frac{[m(t)]^n\cdot e^{-m(t)}}{n!},n=0,1,2,\ldots \quad (3)$$

In general, an implemented software system is tested to detect and correct software errors in the software development process. During the testing phase software errors remaining in the system cause software failure and the errors are detected and corrected by test personnel. Based on the above assumptions, if the number of the detected errors by the current testing-effort(TE) expenditure is proportional to the number of remaining errors, then the following different equation [31], [19], [32], [30], [13], [5], [2], [1] is obtained:

$$\frac{dm(t)}{dt}/w(t)=r\cdot[a-m(t)],a>0,\ 0<r<1 \quad (4)$$

Where $m(t)$ represent the expected mean number of errors detected in time $(0,t]$ which is assumed to be a bounded non-decreasing function of t with $m(0) = 0, w(t)$ is the current testing-effort expenditure at time t, a is expected number of initial error in the system, and r is the error detection rate per unit testing-effort at time t. Solving the above differential equation, we get

$$m(t) = a.(1-e^{r\cdot W(t)}) \quad (5)$$

Substituting $W(t)$ from the various cumulative testing-effort expenditure consumed in time $(0,t]$ given by various authors above mentioned, we get

1) $m(t) = a \cdot (1 - e^{r \cdot \alpha \cdot (1 - e^{-\beta \cdot t^m \cdot e^{\delta \cdot t}})})$. (6a)

A NHPP model with mean value function incorporating the NMW testing-effort expenditure (Quadri et al 2010)

2) $m(t) = a \cdot (1 - e^{r \cdot \alpha \cdot (1 - e^{-\beta \cdot t^m})^{\theta}})$. (6b)

A NHPP model with mean value function incorporating the Exponential Weibull (EW) testing-effort expenditure (Quadri et al 2007)

3) $m(t) = a \cdot (1 - e^{-r \cdot \alpha \cdot (1 - e^{-\beta \cdot t^2})^{\theta}})$. (6c)

A NHPP model with mean value function incorporating the Burr Type X testing-effort expenditure. (Quadri et al 2009).

4) $m(t) = a \cdot (1 - e^{r \cdot \alpha \cdot (1 - e^{-\beta \cdot t})^{\theta}})$. (6d)

A NHPP model with mean value function incorporating the Generalized Exponential testing-effort expenditure. (Quadri et al 2006)

In addition, the failure intensity at testing time t of the various NHPP is given by

$$\lambda(t) = \frac{dm(t)}{dt} = a \cdot r \cdot w(t) \cdot e^{-r \cdot W(t)} \quad (7)$$

The expected number of errors to be detected eventually is:

$$m(\infty) = a \cdot (1 - e^{-r \cdot a}) \quad (8)$$

This implies that even if a software system is tested during an infinitely long duration, all errors remaining in the system cannot be detected [19], [32] Thus, the mean number of undetected errors if a test is applied for an infinite amount of time is

$$a - m(\infty) = a - a \cdot (1 - e^{-r \cdot a}) \ = a \cdot e^{-r \cdot a} \quad (9)$$

That is, not all the original errors in a software system can be fully tested with a finite testing effort since the effort expenditure is limited to α.

IV. SOFTWARE RELIABILITY MEASURES

Based on the NHPP model with $m(t)$, we can derive the following quantitative measures for reliability assessments [3], [32]. If $\bar{N}(t)$ represent the number of errors remaining in the system at testing time t, then the mean of $\bar{N}(t)$ and its variance for various SRGM models are given by

$$\begin{aligned} r(t) &= E[\bar{N}(t)] = E[N(\infty)] - N(t)] = m(\infty) - m(t) \\ &= a \cdot (e^{-r \cdot W(t)} - e^{-r \cdot W(\infty)}) = Var[\bar{N}(t)] \end{aligned} \quad (10)$$

The software reliability represents the probability that no failure occurs in the time interval $(t, t+\Delta t)$ given that the last failure occurred at time Δt is given by

$$R = R(\Delta t|t) = e^{[m(t+\Delta t) - m(t)]} = e^{-a[e^{-r \cdot W(t)} - e^{-r \cdot W(t+\Delta t)}]} \quad (11)$$

The instantaneous mean time between failures (MTBF) at arbitrary testing can be defined as a reciprocal of error detection rate in (7). Then, the instantaneous MTBF is given by

$$MTBF(t) == \frac{1}{\lambda(t)} = \frac{e^{r \cdot \alpha \cdot (1 - e^{\beta \cdot t^m \cdot e^{\delta \cdot t}})}}{a \cdot r \cdot \alpha \cdot \beta \cdot (m + \delta \cdot t) \cdot t^{m-1} e^{\delta \cdot t} e^{-\beta \cdot t^m \cdot e^{\delta \cdot t}}} \quad (10)$$

V. PARAMETER ESTIMATION METHODS

MLE and LSE techniques are used to estimate the model parameters [18], [17], [16]. Estimation of maximum likelihood is a general technique that may be applied when the underlying distribution of the data are specified or known and is better in deriving confidence intervals and the asymptotic normal distribution for ML estimates. On the other hand, LSE is fairly general technique which is applied in most practical situations for small or medium size data for better estimates [22], [23], [24], [27], [18], [6], [4]. It minimizes the sum of squares of the deviation between what we expect and what we actually observe. In using LSE one of the common assumptions is that the standard deviation of the error term is constant over all the values of the predictor variable. This assumption, however clearly does not hold in every modeling application. For example, in testing effort data, it may appear that the precision of testing effort varies as the time changes. In such situations, when it may not be reasonable to assume every observation should be treated equally, one could use WLSE to maximize the efficiency of parameters estimation [22]. Sometimes, however, the likelihood equations may be complicated and difficult to solve explicitly. In that case one may have to solve with some numerical methods to obtain the estimates.

[a] Least square method (LSE)

The various parameters e.g. α, β, m, θ and δ in the various testing-effort function [27], [8], [22], [23], can be estimated by the method of LSE. These parameters are determined for n observed data pairs in the form $(t_k, W_k)(k = 1, 2, ..., n;\ 0 < t_1 < t_2 < ... < t_n)$, where W_k is the cumulative testing effort consumed in time $(0, t_k]$

[b] Maximum likelihood method

Suppose that the estimated testing-effort parameters like $\hat{\alpha}$, $\hat{\beta}$, $\hat{m}$ and $\hat{\delta}$ in various current testing-effort functions have been obtained by the method of least squares discussed earlier. The estimators for a and r are determined for n observed data pairs in the form $(t_k, y_k)(k = 1, 2, ..., n; 0 < t_1 < t_2 < ... < t_n)$, where y_k is the cumulative number of software errors detected up to time t_k.

VI. PERFORMANCE ANALYSIS

In order compare performance analysis of various existing/proposed models, experiments on actual software failure data have been performed.

A. Criteria for Model Comparison

To evaluate the performance of a software reliability growth model and to make a fair comparison with the other existing SRGM, we describe the following comparison criteria.

[a] The Accuracy of Estimate (AE) is defined [18], [31], [4], [13] as:

AE $= \frac{M_a - a}{M_a}$, Where M_a is the actual cumulative number of detected errors after the test, and a is the estimated number of initial errors. For practical purposes,

M_a is obtained from software error tracking after software testing.

[b] The mean of Squared Errors (Long-term predictions) is defined [16], [4], [13] as:

MSE = $\frac{1}{k}\sum_{i=1}^{k}[m(t_i)-m_1]^2$, Where $m(t_i)$, is the expected number of errors at time t_i estimated by a model, and $m(t_i)$ is the observed number of errors at time t_i. MSE gives the qualitative comparison for long-term predictions. A smaller MSE indicates a minimum fitting error and better performance [6], [10], [11].

[c] The Coefficient of Multiple Determination is defined [18], [17] as:

$$R^2=\frac{S(\hat{\alpha},0,1,1)-S(\hat{\alpha},\hat{\beta},\hat{m},\hat{\delta})}{S(\hat{\alpha},0,1,1)}$$

, Where $\hat{\alpha}$ is the LSE of α for the model with only a constant term, that is $\beta=0$, $m=1$ and $\delta=1$ in (12). It is given by in $\hat{\alpha}=\frac{1}{n}\sum_{k=1}^{n}\ln W_k$ .Therefore, R^2 measures the percentage of total variation about the mean accounted for by the fitted model and tells us well a curve fits the data. It is frequently employed to compare models and assess which model provides the best fit to the data. The best model is the one which provides the higher R^2, that is, closer to 1 [12]. To investigate whether a significant trend exists in the estimated testing-effort, one could test the hypothesizes $H_0:\beta=0, m=1$ and $\delta=1$, against $H_1:\beta\neq 0$ or at least m or $\delta\neq 0$ using F-test by merely forming the ratio

$$F=\frac{\left[S\{\hat{\alpha},0,1,1\}-S(\hat{\alpha},\hat{\beta},\hat{m},\hat{\delta})\right]/3}{S\{\hat{\alpha},0,1,1\}/(n-4)}$$

If the value of F is greater that $F_\alpha(3,n-4)$, which is the α percentile of the F distribution with degrees of freedom 3 and n-4, we can be $(1-\alpha)100$) percent confident that H_0 should be rejected, that is, there is a significant trend in the testing-effort curve.

[d] The Predictive Validity is defined [18], [17] as the capability of the model to predict future behavior from present and past failure behavior. Assume that we have observed q failures by the end of test time t_q. We use the failure data up to time $t_0(\leq t_q)$ to determine the parameter of $m(t)$. Substituting the estimates of these parameters in the mean value function yields the estimate of the number of failures $\hat{m}(t_q)$ by t_q . The estimate is compared with the actually observed number q. This procedure is represented for various values of t_e .The ratio $\frac{\hat{m}(t_q)-q}{q}$ is called the relative error. Values close to zero for relative error indicate more accurate prediction and hence a better model. We can virtually check the predictive validity by plotting the relative error for normalized test time t_e / t_q .

B. Other Comparison criteria for evaluation [14], [15]:

[a] $\text{Prediction Error}(PE)=\text{Actual}(\text{observed})^i-\text{Predicted}(\text{Estimate})^i$

[b] Variation = $\frac{\sum_{i=1}^{n}(PE_i-Bias)^2}{n-1}$

[c] Bias = $\frac{1}{n}\cdot\sum_{i=1}^{n}PE_i$

[d] RMS-PE = $\sqrt{Bias^2+Variation^2}$

[e] MRE = $\left|\frac{M_{estimated}-M_{actual}}{M_{actual}}\right|$

[f] BMMRE = $\frac{1}{n}\cdot\sum_{i=1}^{n}\frac{|M_{estimated}-M_{actual}|}{\min(M_{estimated},M_{actual})}$

C. Description of the Actual Data Set

Table1: Comparative results of different SRGM for DS1

95% CONFIDENCE LIMIT FOR DIFFERENT SELECTED MODELS (DSI)				
Models	a		r	
	Lower	Upper	Lower	Upper
SGRM with Burr type X TEF [22]	445.7304	685.6162	0.01372	0.02556
SGRM with Exponential Weibull TEF[24]	446.360	684.926	0.01375	0.02554
SGRM with New Modified Weibull TEF[23]	433.29	700.02	0.0130	0.026
SGRM with Gompertz TEF	385.1	489.5	0.02585	0.03917
SGRM with Logistic TEF	358	433.2	0.03399	0.04928
SGRM with Rayleigh TEF	348.6	569.6	0.01651	0.03817
Yamada Delayed shaped Model with Logistic TEF	300.8	361	0.09423	0.1334
Yamada Delayed shaped Model with Rayleigh TEF	291	347.5	0.1088	0.1589
Yamada Delayed S shaped Model with Rayleigh TEF	288.7	377.7	0.07507	0.1258
G-O Model	465.4	1056	0.01646	0.04808
Yamada Delayed S shaped Model	343.7	404.4	0.1748	0.2205

DS 1: The first set of actual data is from the study by Ohba [20]. The system is PL/1 data base application software, consisting of approximately 1,317, 000 lines of code. During the nineteen weeks experiments, 47.65 CPU hours were consumed and about 328 software errors were removed. The study reports that the total cumulative number of detected faults after a long period of testing is 358.

In order to estimate the parameters $\hat{\alpha},\hat{\beta},\hat{m}$ and $\hat{\delta}$ of the various testing-effort function; we fit the actual testing effort data into various current testing effort function [27], [8], [22], [23] and solve it by using the various methods of estimation mentioned before. The estimated parameters for various SRGM's are obtained and there comparative result is shown as:

Table2: Comparison Results for Different TEF Based on DS1

Model	*a*	*r*	AE (%)	MSE
Proposed Model by Quadri et, al 2010 (Eqn. (6a) with New modified Weibull curve)	566.66	0.0196	58.28	103.1
Proposed Model by Quadri et, al 2009 (Eqn. (6d) with Burr Type X TEF)	565.673	0.01964	69.15	123.67
Proposed Model by Quadri et, al 2008 (Eqn. (6b) with Exponential Weibull curve)	565.643	0.01964	57.98	113.10
Yamada exponential model (Eqn. (5) with exponential curve)	828.25	0.0118	131.4	140.7
Yamada Rayleigh model (Eqn. (5) with Weibull curve)	565.35	0.0197	57.91	122.1
Huang Logistic model	394.08	0.0427	10.06	118.6
Ohba exponential mode	455.37	0.0267	27.09	206.9
Inflection S-shaped model	389.1	0.0935	8.69	133.3
Delayed S-shaped model	374.05	0.1977	4.48	168.7
G-O model	760.0	0.0323	112.3	139.8
Delayed S-shaped model with Rayleigh	333.14	0.1004	6.93	798.5

Table3: Summary of estimates of various NHPP model parameters for DS1

Distributions			
Parameter	Logistic	Rayleigh	Exponential
Bias	0.0548	-1.1469	-16.5313
Variation	0.3508	3.4579	6.3495
RMS-PE	0.3551	3.6440	17.7087
MRE	-0.0040	0.2384	-0.5254
PE end of testing	-0.1022	6.0332	-13.2936
BMMRE	24.3718	65.2980	81.4501

VII. CONCLUSION

In this research emphasis is given on comparison of some of the existing SRGMs incorporating various TEFs, we also provided an overview of general SRGM and basic assumptions that is followed by them. The use and applicability of such models during software development and operational phase is important. The objective was to provide a comparative analysis of some of the models that would be helpful in determining which model, if any, to use in a given software development environment. It should be noted that the above analytical models are primarily useful in estimating and monitoring software reliability. The models are compared using actual software data. One unique contribution of this paper is that we do not add any new models to the already large collection of SRGMs, rather we emphasize on the comparison of some of the existing SRGMs used in the software development process.

VIII. REFERENCES

[1] Bokhari, M.U. and Ahmad, N., "Software reliability growth modeling for Exponentiated Weibull function with actual software failures data, " In: Proceedings of 3rd International Conference on Innovative Applications of Information Technology for Developing World (AACC'2005), Nepal, 2005.

[2] Bokhari, M.U. and Ahmad, N., "Analysis of a software reliability growth models: the case of log-logistic test-effort function, "In: Proceedings of the 17th International Conference on Modeling and Simulation (MS'2006), Montreal, Canada, pp. 540-545, 2006.

[3] Goel, A.L. and Okumoto, K. , "Time dependent error-detection rate model for software reliability and other performance measures, " IEEE Transactions on Reliability, Vol. R- 28, No. 3, pp. 206-211, 1979

[4] Huang, C.Y. and Kuo, S.Y. , " Analysis of incorporating logistic testing-effort function into software reliability modeling, " IEEE Transactions on Reliability, Vol. 51, no. 3, pp. 261-270, 2002.

[5] Huang, C. Y. "Performance analysis of software reliability growth models with testing-effort and change-point, "Journal of Systems and Software, Vol. 76, pp. 181-194, 2005.

[6] Huang, C.Y., Kuo, S.Y. and Chen, I.Y., "Analysis of software reliability growth model with logistic testing-effort function, "In: Proceeding of 8th International Symposium on Software Reliability Engineering (ISSRE'1997), Albuquerque, New Mexico, pp. 378-388, 1997.

[7] Huang, C. Y., Lo, J.H., Kuo, S. Y., "A pragmatic study of parametric decomposition models for estimating software reliability growth," in Proc. 9th Int'l. Symp. Software Reliability Engineering (ISSRE'98), 1998, pp. 111-123.

[8] Huang, C. Y., Kuo, S. Y., and Lyu, M. R," Effort-Index-based software reliability growth model and performance assessment," in Proc. 24th Ann. Int'l. Computer Software and Applications Conf. (COMPSAC 2000), 2000.

[9] Kapur, P.K. and Garg, R.B. , "Cost reliability optimum release policies for a software system with testing effort, " Operations Research, Vol. 27, no. 2, pp. 109-116, 1990.

[10] Kapur, P.K. and Garg, R.B. "Modeling an imperfect debugging phenomenon in software reliability, "Microelectronics and Reliability, Vol. 36, pp. 645-650, 1996.

[11] Kapur, P.K., Garg, R.B. and Kumar, S., Contributions to Hardware and Software Reliability, World Scientific, Singapore, 1999.

[12] Kumar, M., Ahmad, N. and Quadri, S.M.K., "Software reliability growth models and data analysis with a Pareto test-effort, "RAU Journal of Research, Vol., 15 (1-2), pp. 124-128, 2005.

[13] Kuo, S.Y., Hung, C.Y. and Lyu, M.R., "Framework for modeling software reliability, using various testing-

efforts and fault detection rates, " IEEE Transactions on Reliability, Vol. 50, no.3, pp 310-320, 2001.

[14] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," IEEE Trans. Software Engineering, vol.21, no. 2, pp. 126–136, 1995.

[15] K. Pillai and V. S. S. Nair, "A model for software development effort and cost estimation," IEEE Trans. Software Engineering, vol. 23, no. 8. Aug. 1997.

[16] Lyu, M.R., Handbook of Software Reliability Engineering, McGraw- Hill, 1996.

[17] Musa J.D., Software Reliability Engineering: More Reliable Software, Faster Development and Testing, McGraw-Hill, 1999.

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

[19] Ohba, M., "Software reliability analysis model" IBM Journal. Research Development, Vol. 28, no. 4, pp. 428-443, 1984.

[20] Pham, H. (2000), Software Reliability, Springer-Verlag, New York, 2000.

[21] Quadri, S.M.K., Ahmad, N., Peer, M.A. and Kumar, M., "Non homogeneous Poisson process software reliability growth model with generalized exponential testing effort function, "RAU Journal of Research, Vol., 16 (1-2), pp. 159-163, 2006.

[22] Ahmad, N., Khan, M.G.M., Quadri, S.M.K., and Kumar, M, "Modeling and analysis of software reliability with Burr type X testing-effort and release determination," Journal of Modeling in Management, vol 4, No 1, 2009, pp. 28-54. Emerald Group Publishing Limited."

[23] Quadri, S.M.K., Ahmad, N, "Software Reliability Growth modeling with New modified Weibull testing–effort and optimal release policy" Int'l Journal of Computer Applications Vol6, No. 12, September 2010.

[24] Ahmad, N., Bokhari, M. U., Quadri, S.M.K., and Khan, M.G.M., "The Exponential Weibull Software Reliability Growth model with testing–effort and optimal release policy ," Int'l Journal of Quality and Reliability Management Vol. 25, No. 2, 2008, pp. 211-235, Emerald Group Publishing Limited.

[25] Quadri, S.M.K., Mohd Razeef," Software Reliability Growth Models: A Comparative Study" JK Science Congress, pp. 299-300, December 2010.

[26] Sy-Yen, Kuo., Chin-Yu Huang. and Michael, R. Lyu., "Framework for modeling software reliability, Using Various Testing-Efforts and Fault-Detection Rates, " IEEE Transactions on Reliability, Vol. 50, no. 3, 2001.

[27] Tohma, Y., Jacoby, R., Murata, Y. and Yamamoto, M., "Hyper-geometric distribution model to estimate the number of residual software fault, "In: Proceeding of COMPSAC-89, Orlando, pp. 610-617, 1989.

[28] Tang, Y. et. al. " Statistical Analysis of a Weibull Extension Model, " Communications in Statistics, Theory and Analysis, pp. 911-916, 2003

[29] Yamada, S. and Osaki, S., "Cost-reliability optimal release policies for software systems, " IEEE Transaction on Reliability, Vol. R-34, no. 5, pp. 422-424, 1985b.

[30] Yamada, S., and Osaki, S., "Software reliability growth modeling: models and applications, " IEEE Transaction on Software Engineering, Vol. SE-11, no. 12, pp. 1431-1437, 1985a.

[31] Yamada, S., Hishitani J. and Osaki, S., "Test-effort dependent software reliability measurement, " International Journal of Systems Science, Vol. 22, no. 1, pp. 73-83, 1991.

[32] Yamada, S., Ohtera, H. and Narihisa, H., "A testing-effort dependent software reliability model and its application, "Microelectronics and Reliability, Vol. 27, no. 3, pp. 507-522, 1987.