International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

Analysis of Software Reliability Growth Models for Quantitative Evaluation of Software Reliability and Goodness of Fitness Metrics

Harminder Pal Singh Dhami[1], Vaibhav Bansal[2]

[1, 2]Department Computer Science & Engineering, OPJS University, Churu, Rajasthan, India

Abstract: ***Till now there have been many Software Reliability models developed for assessing the reliability of software product. Most of these are based upon past failure data gathered during the testing phase. These models have been utilized to evaluate the quality of the software and for future predication of reliability. They have been used in many critical management decision making problems that occur during the testing phase. But none of these models can claim to be the best and hence there is a need for further research. In this paper different modeling approaches are briefly studied and we present procedures to estimate the parameters of SRGMs and a critical analysis of Goodness of Fit using some existing Software Reliability Growth Models. On the basis of our observations we recommend the methodologies for estimating parameters of SGRM and the various metrics used for comparison of Goodness of Fit and predictive validity.***

Keywords: SRGM, SRE, NHPP, Mean Value Function, Parameter Estimation, Failure Intensity Rate, MSE, AIC, R^2, MLE, LSE, RPE

1. Introduction

Software Reliability Engineering (SRE) is a conventional area of software engineering research and is concerned with the improvement and measurement of reliability. SRE works by quantitatively characterizing and applying two things about the product, the projected relative use of its functions and its required major quality characteristics. The major quality characteristics are reliability, availability, delivery date and life-cycle cost. IEEE has suggested the following standard definition of a fault (may be referred as a defect) [29]. A fault occurs when a user makes a mistake, called an error, while performing some software activity. Many designers do not distinguish between faults, errors or bugs, as their effects are the same, the dreaded failures. A failure is a departure from the system's required behavior and it can occur any time in a system deliver, during testing, or operation or maintenance. Few faults may never turn into failures, if faulty code is never executed or a particular state never occurs. Nevertheless software cannot be made fault free and these faults certainly lead to failures. This demands application of Software Engineering tools, techniques and procedures, in an effective way.

The basic aim of software engineering is to produce high quality efficient software at low cost. With growth in size and complexity of software, management issues began dominating. An optimal design strategy without any compromises e.g. cost & time, for the system does not develop an optimal design. The reason for this is the changes in requirements that may occur in later development cycles. Such changes may cause design decisions taken earlier to be less optimal. Design erosion is inevitable with the current way of developing software. Refined methods only contribute by delaying the moment that a system needs to be withdrawn or retired. These approaches do not address the fundamental problems that cause design erosion and makes system unreliable [3]. Component based design is expected to have a strong impact on the quality of software development: Due to the simplicity, the software development speeds up. The shorter development time results in reduced costs. The extensibility and resolvability of software systems is improved, because components can flexibly be substituted by another component that satisfies the requirements. Software components enhance the reliability of the software, as they are improved, tested and debugged over years [2].

Software Engineering Methodologies constitute the framework that guides software developers in optimally developing the software systems. These frameworks define the different phases of software development. The selection of which methodology to apply in a specific development process is closely related to the size, complexity, reliability and maintainability of the software, and to the environment it is supposed to function. Now a days the fusion of all these methodologies is incorporated. All the developers look at the low cost & risk, high quality and small cycle of time, so that the productivity and quality of the product can be optimized. There should be a tradeoff between the development time and the quality of the product [1].

The ability and quality of software depends greatly on its reliability. The reliability of any software relies on testing phase. While execution, if a system fails, it implies that there are faults inherent in the system. This reduces the reliability of the software system. Software Reliability improves as faults are detected and corrected [4]. Therefore, the testing phase emphasizes on detecting and correcting the faults in the software. Pham [5] reviewed and compared various NHPP based SRGMs on their fit and predictive power. The comparison of SRGMs on defect inflow data is studied by Wood [6] and he found it correlated with past released defects. Even though a number of studies have compared and evaluated SRGM within different context. We are still not capable to make a consensus on how to choose SRGMs for specified purpose and which models are best for given process characteristics.

2. Software Reliability

Software Reliability is one of the main features of software quality. As the definition of reliability is user oriented, it has become the fundamental quality attribute of any product, be it software or hardware. Software Reliability models can be classified in a number of ways. For the analysis normally stochastic software reliability models are used. They model the failure process of the software and use other software failure data as a basis for parameter estimation. The models are capable (a) to estimate the current reliability and (b) to predict future failure behavior.

Software error detection method can be considered as a software reliability growth process. A software reliability growth model has been studied by many researchers, as a mathematical model for the reliability growth process. The software reliability growth model describes the relationship between the cumulative number of detected errors in software and the time of software testing. Xie[33] categorized the Software Reliability Models into following groups according to their probabilistic assumptions.

1. Software Reliability Models which describe the dynamic aspects of the failure occurrence process.
   Markovian Model, Failure Count Models, and Model Based on Bayesian Analysis.
2. Software Reliability Models which do not make any dynamic assumptions of the failure process.
   The Input Domain Models, Fault Seeding Models, Software Metrics Models, and some Software Reliability Growth Models Based on NHPP Goel-Okumoto Model, Delayed S-Shaped SRGM and Inflection S-Shaped SRGM.

3. Models with Dynamic Aspects of Failure Occurrence Process

A. Markovian Models

The memory less property of the Markovian process implies that the time between consecutive failures follows an exponential distribution. When a Markov process represents the failure process, the resultant model is called Markovian model. The software can attain several states at any particular time with respect to number of faults remaining or number of faults already removed. The transition between states depends on the current state of the software and the transition probability. Numerous attempts have been made to develop Markovian models; especially in earlier days due to similar theory in hardware reliability was already well developed. One of the popular reliability models developed by Jelinski and Moranda [19] is a Markov process model. Littlewood [25] proposed a model, based on semi-Markovian process to describe the failure phenomenon of software with module structure. Cheung [14] has also proposed a Markovian model to describe module-structured software. Kremer [24] proposed birth-death model, which incorporates the probabilities of fault removal and introduction. Goel [17] modified Jelinski-Moranda model by introducing the concept of imperfect debugging. Xie and Bergman [32] proposed a model relating the fault detection probability to the fault size. Kapur et al.[21] proposed a model incorporating imperfect debugging and fault introduction with upper bound on the number of faults.

The Markov model is analysed in order to measure the probability of being in a given state at a given point of time, the amount of time a system is expected to spend in a given state, as well as the expected number of transitions between states: for instance representing the number of failures and repairs.

Markov models provide great flexibility in modeling the timing of events. They can be applied when simple parametric time-based models, such as exponential or Weibull time-to-failure models, are not sufficient to describe the dynamic aspects of a system's reliability or availability behavior; as may be the case for systems incorporating standby redundancy.

B. Fault Counting Model

The majority of these failure count models are based upon the NHPP. Schneidewind [31] proposed a fault detection model based on NHPP. Different NHPP models are distinguished by their unique mean value functions. Yamada et al. [35] proposed the delayed S-shaped model. This category includes the models, which describe the occurrence of failure method by stochastic processes like Homogeneous Poisson Process (HPP), Non Homogeneous Poisson Process (NHPP), and Compound Process (CPP) etc. Ohba [28] proposed the infection S-shaped model. Musa et al. [26],[27] have proposed the basic execution time model and Log-Poisson model. Goel [17] modified Musa's originals model by introducing the test quality parameter. Yamada et al. [36] also proposed a discrete time model. The effect of testing effort on failure process was taken into consideration by Yamada et al. [37]. In recent times efforts have been directed towards development of general SRGMs [22],[30]. General SRGMs are flexible models and many of the above models can be derived from them. Kapur and Garg [20] modified Goel-Okumoto [15] model by introducing the concept of imperfect debugging. Kareer et al. [23] proposed a model with two types of faults where each fault type is module as by an S-shaped curve model. Xie and Zhao [34] have illustrate how the Schneidewind model can be modified to result in many of the above SRGMs. Zeephongsekul et al. [39] proposed a model describing the case when a major fault introduces secondary faults. Attempts as listed above for new models were made with the primary intention of getting flexible models that could describe a range of failure count curves or reliability growth curves like exponential curves and highly S-shaped curves. Models with such property are termed as flexible SRGMs [13],[22],[28].

Recently reliability modeling for distributed development environment has caught the attentions of many researchers [38]. Large software systems have modular design. A system is said to be modular when each activity of the system is carry out by exactly one component, and when the inputs and outputs of each component are well defined [29]. Often such components of software are developed separately by different development teams with availability of communication networks at cheaper rates. Some software components are developed at separate geographic location also. Software developed under this distributed development

environment has proved to be economical. Many times, components from other software projects are also reused. Therefore SRGMs for software developed under distributed environment needs to have different approach. But very few attempts have been made in this regard [38].

C. Models Based on Bayesian Analysis

In the earlier two categories, the unknown metrics or parameters of the models are estimated either by the Maximum Likelihood Method or by the Least Squares Method (will be discussed later). But in this, the Bayesian analysis technique is used to estimate the unknown parameters of the models. Littlewood and Verral, recommended the first software reliability model based on Bayesian Analysis [25]. Singpurwalla [40] have proposed a number of Bayesian software reliability models for various testing environments. This technique assists the use of gathered information by developing similar software project. Based on this information the parameter of given model are assumed to pursue some distribution, known as priori distribution. Given the software test data and based on a priori distribution, a posterior or subsequent distribution can be acquired, which in turn describes the failure phenomenon.

4. Models Without any Dynamic Assumptions of Failure Process

The following four categories of Software Reliability models are defined briefly here.

A. The Input Domain Models

The basic approach in this category is to generate a set of tests from a distribution. The distribution should be chosen so that it is representative of the operation of the software and the reliability is estimated from the outcome of the test cases.

B. Fault Seeding Models

In this class a known number of faults are introduced into the software. During the software testing, these seeded faults as well as original faults are visible. The proportion of the seeded faults found compared to the unseeded faults found can given an estimate of the total fault content in the software.

C. Software Metrics Models

The models in this class relate the fault content in the software to some features of the software program such as program length, complexity, volume etc. These models are empirically built and the result obtained by a model is dependent on the software development process environment, which may not be the same in the other projects.

D. Some SRGMs Based On NHPP

In this segment few SRGMs are presented briefly. These SRGMs and their hypothesis have been presented during the development of some of the models and theories of research. Some of the basic assumption or postulates, apart from some special ones for specific models discussed, for the models are as follows:

1) Software system is subject to failure during execution, caused by faults left behind in the systems.
2) Failure rate of the software is equally influenced by faults left behind in the software.
3) The number of faults detected at any instant of time is proportional to the remaining number of faults in the software.
4) Repair effort starts and fault causing the failure is removed with certainty, on a failure.
5) From failure detection point of view, all faults are mutually free.
6) The proportionality of failure detection/ fault removal/ fault isolation is constant.
7) The fault detection/removal phenomenon is modeled by NHPP.

Notations

m(t): Expected number of faults identified in (0,t), mean value function of NHPP

a, b : Constants, representing initial fault content and rate of fault removal per remaining faults for a software.

p, q: Proportionality constants

Non Homogeneous Poisson Process (NHPP)

Let (N(*t*); t $\geq$ 0) be a counting process denoting the cumulative number of failures (or faults isolated as the case may be) by time t, N(*t*) is a random variable and (N(*t*); t $\geq$ 0) is a Non Homogeneous Poisson Process (NHPP) if N(0) = 0, (N(*t*); t $\geq$ 0) has independent increments

P (two or more failures in (t, t +Δt)) = 0(Δt)

P (exactly one failure in (t, t +Δt)) = λ(t) + 0(Δt)

Where λ(t) is intensity function of N(t). If we let m (t) = $\int_0^t \lambda(x)\,dx$ represent the mean of number of faults removed in (0, *t*). It can be shown that

$$P[N(t) = n] = \frac{[m(t)]^n e^{-m(t)}}{n!},\ n = 0,1,2 \qquad \ldots (1)$$

i.e. N *(t)* has a Poisson distribution with expected value *E[N(t)]= m(t)* for t > 0 and m(*t*) is called the mean value function of NHPP.

1) Goel-Okumoto Model :

The differential equation results from assumption-3 are as follows

$$\frac{d}{dt} m(t) = b[a-m(t)] \qquad (2)$$

The first order linear differential equation, as above, when solved with the initial condition as m(0) = 0 gives the following mean value function for NHPP (2)

$$m(t) = a\,(1-e^{-bt}) \qquad (3)$$

The mean value function is exponential in nature and does not offer a good-fit to the S-shaped growth curves that normally occur in Software Reliability. But the model is popular due to its straightforwardness.

2) S-Shaped SRGMs

Few S-shaped SRGMs which will be discussed are Delayed S-Shaped SRGM and Inflection S-Shaped SRGM

i. Delayed S-Shaped SRGM

Fault detection in the model is assumed to be a two-phase process comprising of failure detection and its eventual removal by isolation. It considers the time taken to isolate and remove a fault and, so it is important that the data to be used at this time should be that of fault isolation. In addition, it is assumed that the number of faults isolated at any time instant is proportional to the remaining number of faults in the software. The failure rate and isolation rate per fault are assumed to be similar and equal to b.

Thus

$$\frac{d}{dt} mf(t) = b[a-mf(t)] \quad (4)$$

$$\frac{d}{dt} m(t) = b\,[mf(t) - m(t)] \quad (5)$$

$m_f(t)$ is the expected number of a failures in $(0,t)$. Solving (4) and (5), we get the mean value function as

$$m(t) = a\{1-(1+bt)e^{-bt}\} \quad (6)$$

Alternately, the model can also be devised as one stage process directly as follows:

$$\frac{d}{dt} m(t) = \left(\frac{b^2 t}{1+bt}\right)(a-m(t)) \quad (7)$$

It is observed that $\frac{b^2 t}{1+bt} \to b$ as $b \to \infty$. This model was specifically developed to account for delay in the failure observation and later its removal. This kind of derivation is unusual to software reliability only. Xie and Zhao have also proposed alternative ways of deriving the above model.

ii. Inflection S-Shaped SRGM

The model features S-shaped ness to the mutual dependency between software faults. Other than assumption-3 it is also assumed that the software contains two types of faults, namely mutually independent and mutually dependent. The mutually independent faults are those, which are located on different execution paths of the software, thus they are equally likely to be detected and removed. On the other hand, the mutually dependent faults are those faults, which are located on the same execution path. According to the order of the software execution, some faults in the execution path will not be removed until their earlier faults are removed. According to the assumptions of the model, the fault removal intensity per unit time can be written as

$$\frac{d}{dt} m(t) = b(t)\,[a-m(t)] \quad (8)$$

b is the fault removal rate in the steady state. Solving (8) under the initial condition M(0) = 0 we get

$$m(t) = a\,\frac{1-e^{-bt}}{1+\frac{1-r}{r}e^{-bt}} \quad (9)$$

If $r = 1$, the model reduces to the Goel-Okumoto model. For different values of r, different growth curves can be attained and in that case the model is flexible.

5. Parameter Estimation

The parameters of the SRGMs are estimated based upon these (input failure) data. Therefore, efforts should be made to make the data gathering more precise and scientific. Usually data is collected in one of the following two ways. In the first case, Time-Domain data the time between successive failures are recorded. The accomplishment of mathematical modeling approach to reliability estimation depends greatly upon quality of failure data gathered. Though this type of data gathering is more preferable, it may not be simple. Problems can come in measuring the testing effort for each fault and it may be very inconvenient to note the time at each failure report. The other convenient and commonly collected data type is known as the grouped data or Interval-Domain data. Here testing intervals are specified and number of failures observed during each such interval is noted. Some existing software reliability models can handle both type of data but Time- Domain data provides better accuracy of parameters estimation with current existing Software Reliability Models [5]. For both these data type method of Maximum Likelihood and Method of Least Squares have been recommended and widely used for estimation of parameters of SRGMs.

A. Parameter Estimation Methods

Parameter Estimation is of major importance in Software Reliability predication. Once the mean value function m(t) of analytical model is known, the parameter in the solution is required to be determined. During the testing and initial operational phases of Software Development Life Cycle (SDLC), failure events are encountered. They are recorded and underlying faults that caused them are removed, which results in process called Reliability Growth. The basic idea behind the SRGM is simple; if the history of fault detection and removal follows a certain recognizable pattern, it is possible to describe the mathematical form of the pattern. The function that represent this pattern is called mean value function m(t), which is cumulative number of faults describe in a given time t. If we are able to fit this function to the existing historical fault detective data, we can predict the future failure behaviour of software. The mean value function is often transferred to failure intensity (rate) function $\lambda(t)$ by formula $\lambda(t) = \frac{d}{dt} m(t)$. The parameters of SRGM are estimated by one of the following methods.

1) Least Square Estimation (LSE)
2) Maximum Likelihood Estimation (MLE)

1) Method of Least Squares

In this method the square of the difference between observed response and value predicted by the model is minimized. If the expected value of the response variable is given by $m(t)$ (can be a mean value function of an SRGM), then the least square estimators of the parameters of the model may be obtained from n pairs of sample values $(t_1, y_1), (t_2, y_2), \ldots\ldots (t_n, y_n)$ by minimizing J given by

$$J = \sum_{i=1}^{n} [y_i - m(t)]^2 \quad (10)$$

y_i and t_i dependent variables and observed values of explanatory respectively. For small and average size samples Least Square Estimation is preferred [27]. For estimation of

the parameters of the analytical models, Method of Least Square (Non Linear Regression) has been used. Non Linear Regression is a technique to find a nonlinear model of the relationship between the dependent variable and a set of independent variables. Unlike conventional linear regression, which is restricted to estimate linear models, nonlinear regression can estimate models with arbitrary relationships between independent and dependent variables.

2) Maximum Likelihood Estimation

Maximum Likelihood Estimation(MLE) method has been broadly adopted for estimation of parameters of SRGMs based upon NHPP. We briefly discuss below the MLE procedure for two types of software failure data as discussed above. For the first type of data, suppose that estimation is to be performed at a specified time t_k, not essentially corresponding to a failure and with total of m_k failures being experienced at time t_1, t_2, t_{mk}. Then the likelihood function for the NHPP is:

$$L = \left[\prod_{i=1}^{k} \lambda(t_i)\right] e^{-\int_0^{tk} \lambda(x)\,dx} \qquad (11)$$

The MLE of the Parameters can be obtained by maximizing Likelihood function or its Log likelihood function (log L).

If the software failure data is grouped into k points (t_i, y_i); $i=1, 2, \ldots k$, where y_i is the cumulative number of failure reports at time t_i. Then the Likelihood function L is given as follows:

$$L = \prod_{i=1}^{k} \frac{[m(t_1) - m(t_{i-1})]^{y_i - y_{i-1}}}{(y_i - y_{i-1})} e^{-\{m(t_i) - m(t_i - 1)\}} \qquad (12)$$

Taking natural logarithm of (12) we get the log likelihood function

$$LogL = \sum_{i=1}^{k} (y_i - y_{i-1}) \ln[m(t_i) - m(t_{i-1})] - m(t_k) - \sum_{i=1}^{k} \ln[(m(y_i - y_{i-1})!] \qquad (13)$$

The MLE of the parameters of SRGM can be obtained by maximizing (13) with respect to the model parameters. Estimation of parameters using MLE requires solving set of simultaneous equations to maximize the likelihood defect data coming from given function to find parameters.

Wood [6] applied both MLE and LSE and found LSE to be more stable and better correlated to field data although MLE results were more reasonable. It can be safely assumed that statistically MLE is much better parameter predictions method than LSE as LSE is much easier and provide consistent results in wider data sets than preferred methods [9],[10],[5].

Both the estimation methods can also be used to other stochastic processes. Maximum Likelihood Estimators possess many desirable properties such as efficiency, asymptotic normality, consistency, and the invariance property. Hence, it is the most preferred estimation procedure for reasonably large sample size. So we conclude MLE is more suitable for the large sample of data and the LSE for small to medium size sample.

6. Comparison Criteria for SRGMS

The performance of SRGMs are evaluated by their ability to fit the history software fault data (goodness of fit) and to forecast satisfactorily the future performance of the software fault removal process (predictive validity)[11],[12]. Musa et al. [8] have suggested the following attributes for choosing an SRGM.

a) Capability: The model should possess the ability to estimate with satisfactory accuracy metrics needed by the software managers.
b) Quality of assumptions: The assumptions should be plausible and must depict the testing environment.
c) Applicability: A model can be adjudged as the better one if it can be applied across software products of different sizes, structures, platforms and functionalities.
d) Simplicity: The data required for an ideal SRGM should be simple and inexpensive to collect. The parameter of estimation should not be too complex and easy to understand and apply even for persons without extensive mathematical background.

Other than the above qualitative aspect the following indices help in comparing SRGMs.

A. Goodness of Fit Metrics

The common used metrics for model comparison of goodness of fit and the predictive power are discussed as below

1) The Mean Square Fitting Error (MSE): The model under comparison is used to create the fault data, the difference between the expected values, $m(t_i)$ and the true or observed data y_i is measured by MSE as,

$$MSE = \sum_{i=1}^{k} \frac{\hat{m}(t_i) - y_i)^2}{k} \quad \ldots \quad \ldots \qquad (14)$$

where k is the number of observations.

The lesser MSE indicates, lesser the fitting error, thus improved goodness of fit.

2) The Akaike Information Criterion (AIC): It is defined as

AIC = -2 (The value of the max. log likelihood function) + 2 (The no. of the parameters used in the model) … (15)

This index takes into account both the number of parameters that are estimated and the statistical goodness of fit.

Lesser values of AIC indicate the preferred model, i.e. the one with the minimum number of parameters that still offers a good fitness to the data.

3) Coefficient of Multiple Determination (R^2) – They are calculated using the parameters estimated using LSE estimators.

$$R^2 = 1 - (\text{residual SS} / \text{corrected SS}) \qquad (16)$$

If value of R^2 is close to 1 then the model provides better goodness of fit.

B. Predictive Validity Criterion

The number of faults removed by time t_k can be anticipated by the SRGM and compared to the reported fault removal, i.e. y_k. The difference between the anticipated/predicted

value $\hat{m}(t_k)$ and the reported value measures the fault in anticipated/prediction. The ratio $[(\hat{m}(t_k) - y_k)/y_k]$ is called the Relative Prediction Error (RPE). If the RPE is positive (negative) the SRGM is said to overestimate (underestimate) the fault removal process. Segments of the failure data are serially chosen to calculate the RPE. Values nearer to zero for RPE indicate more accurate prediction, thus more confidence in the model and better predictive validity [8] [12]. The various metrics such as MSE, AIC, R^2, SSE and PRR can be used for predicting the validation of SRGM.

7. Parameters Estimations Numerical Implementation

In this section we will illustrate the procedure for estimation of unknown SRGM model parameters with practical considerations.

A. Models Used With Data Set

The data set used for this study is Time-Domain data for a real time control system provided by Ohba [16]. In data, fifteen (15) faults have been reported with their time between the failures. In this study we use the following two widely used SRGMs as given in Table I.

Table 1: SRGM Models Used

Model	Name of SRGM	Mean Value Function *m(t)*	Parameters to be Estimated
Model-1	Exponential Goel-Okumoto(G-O)	$m(t) = a(1 - exp[-bt]),$ $a > 0,\ b > 0$	Two, (a, b)
Model-2	Inflection S-Shaped Model [16]	$m(t) = a * \frac{1 - exp\,-bt}{1 + \beta(r) * exp[-bt]},$ $\beta(r) = \frac{1-r}{r},$ $,\ a > 0, b > 0, r > 0$	Three, a, b, β

B. Parameter Estimation and Analysis

In this study the parameters of software reliability growth model mentioned in the Table I are estimated by using Least Square Estimation (Non- Linear Regression). The statistical package IBM SPSS is used for estimation of parameters for goodness of fit and prediction of the models. The values of parameters estimated by using Least Square Estimation (LSE) of Non Linear Regression (NLR) and parameters values using Maximum Likelihood Estimation (MLE) obtained with same data and models by Pham-Zhang [18] are summarized in Table II for comparison. The presentation of the models given in Table I using LSE and MLE estimators to the observed data is also presented in Figure 1 and Figure 2. It is observed from these figures the curves of the presented models using LSE and MLE Estimators are almost consistent which show Goodness of Fit. It is also observed that for both curve and growth rate our estimates using LSE are much closer to the parameters estimates obtained using MLE.

Table 2: Estimated Parameters

Model	*Mean Value Function m(t)*	*Estimated values of Parameters*	
		LSE	**MLE**
Model-1	$m(t) = a(1 - exp[-bt]),$ $a > 0,\ b > 0$	a=18.538 b=0.0049	a=19.544 b=0.00488
Model-2	$m(t) = a * \frac{1 - exp\,-bt}{1 + \beta(r) * exp[-bt]},$ $\beta(r) = \frac{1-r}{r},$ $,\ a > 0, b > 0, r > 0$	a=28.3062 b=0.0000231 β= -0.9939	a=28.5811 b=0.000132 β= -0.9654

C. Parameter Validity and Accuracy

To predict the goodness of fit and validity of Model using LSE and MLE estimators, the various metrics are evaluated such as Sum of Squared Errors (SSE), Mean Squared Error (MSE), (AIC) and Root Mean Square Predictive Error (RMPSE) summarized in Table III. We now compare the predictive accuracy of the value of a metrics obtain using MLE and NLR estimators as shown in Table III. It is observed that the values of SSE, MSE, PRR and RMSPE obtained with MLE estimator are less than the values obtained with LSE estimator in Model-1 and Model-2 which indicates that MLE is better estimator.

Table 3: Comparison of Goodness of Fit Metrics

Metrics → / **Model ↓**	MSE		AIC		RMSPE	
	LSE	MLE	LSE	MLE	LSE	MLE
Model-1	0.14	0.04	0.089	0.038	0.521	0.212
Model-2	0.03	0.02	0.011	0.012	0.212	0.173

The coefficients of Multiple Determination (R^2) are calculated using the parameters estimated using LSE estimators. The Table IV summed up the values of R^2 using LSE and growth rate of the models with LSE and MLE estimators to predict the validity of the models. The values of R^2 obtained for various models lie close between 0 and 1. The values are very close to 1 or equal to 1, which indicate the validity of the models. It is also observed that growth

rates in both the cases of various models are almost equal which indicates the Goodness of Fit of these models.

Table 4: R^2 and Growth Rate to Predict Validity and Accuracy of the Model

Model	*Model-1*	*Model-2*
Value of R^2 using LSE	0.9999	0.9999
Growth Rate with LSE	0.0054	0.000023
Growth Rate with MLE	0.0049	0.000133

D. Models Asymptote for Goodness of Fit

Asymptotic view of goodness of fit for the Model-1 i.e. Goel-Okumoto (G-O) and Model-2 i.e. Inflection S-Shaped Model, is shown below in Figure-1 and Figure-2 respectively.

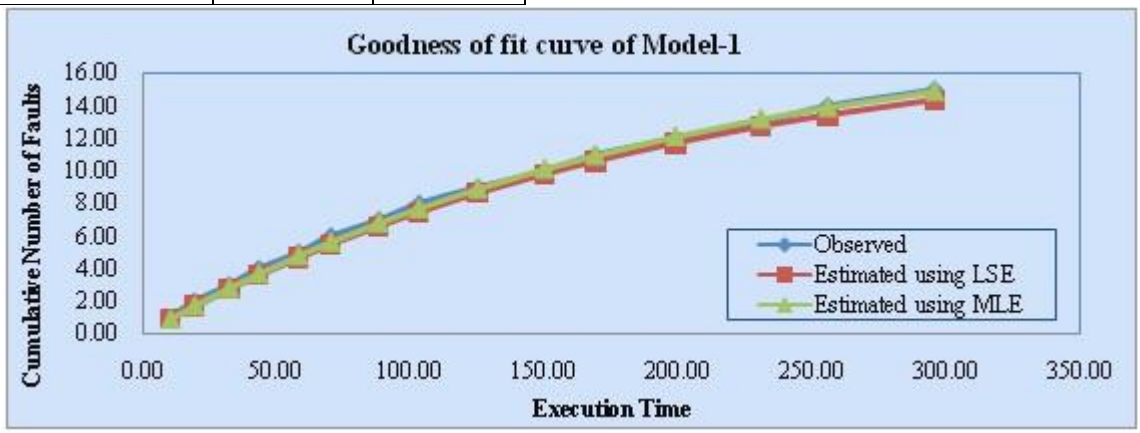

Figure 1: Model 1 - Goel-Okumoto (G-O)

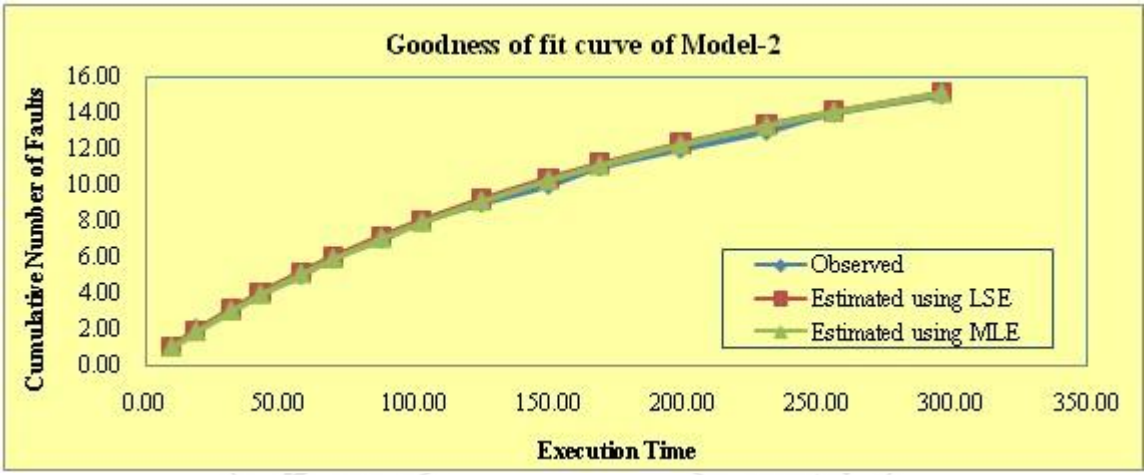

Figure 2: Model 2 - Inflection S-Shaped Model

8. Conclusion

In this paper we have discussed, when a software system is designed, the major concern is the software quality. The quality of software depends on various factors such as software reliability, efficiency, date of delivery and development cost etc. This paper analyzed various existing software reliability models with their failure intensity function and the mean value function. Therefore, we recommend that most important process, Least Square Estimation and Maximum Likelihood Estimation for estimating parameters of Software Reliability Growth Models and the various metrics used for comparison of Goodness of Fit and predictive validity. Using the failure data set from the Literature and existing SRGMs for implementation of LSE and MLE practically for comparison of Models, it was noted that MLE gives the better results as compared to LSE for Goodness of Fit and predictive validity of Models. It was observed that MLE is difficult to apply which limits its use in industry, especially due to lack of tools support whereas LSE is easy to use due to availability of compatible tools. It was concluded from the results presented here and the properties of LSE and MLE estimators suggested that Maximum Likelihood Estimator is better for prediction of Reliability Growth Models whereas Least Square Method for Non Linear Regression is a good estimator for fitting the data to observed failure data. In future the above recommendations can be implemented and results of the model can be compared with the existing model results.

References

[1] A. Kaur, H. P. S. Dhami and J. Kaur (2009): Component Based Software Engineering, IEEE-SOFT-3140, IEEE International Advance Computing Conference IACC-2009.
[2] H. P. S. Dhami (Sept.2016): Comparative Study and Analysis of Software Process Models on Various Merits, International Journal of Advanced Research in Computer Science and Software Engineering, Vol 6(9), pp. 234-243.

[3] H. P. S. Dhami and Dr Anuj Kumar (July 2013): Analysis of Software Design Erosion Issues, International Journal of Advanced Research in Computer Science and Software Engineering 3 (7), pp. 1-7.
[4] Quadri, S.M.K. and Ahmad, N.(2010), Software Reliability Growth Modelling with new modified Weibull testing- effort and optimal release policy, International Journal of Computer Applications, Vol. 6, No. 12.
[5] Pham, H. (2003), Software reliability and cost models: Perspectives, comparison and practice, European Journal of Operation Research, Vol. 149, No. 3, 475-489.
[6] Wood, A.(1996), Predicting Software reliability, Computer, Vol. 29, No.11, pp. 69-77.
[7] Harminder Pal Singh Dhami (Dec.2016): Improving Software Reliability, Productivity and Quality using software metrics, International Journal of Computer Science & Technology (IJCST) Vol.7 Issue: 4(1), ISSN 0976 – 8491 (online), ISSN 2229 – 4333 (Print).
[8] Musa, J.D., Iannino(1987), A. and Komodo, K., Software Reliability: Measurement, Prediction and Application, McGraw-Hill.
[9] Ullah, N., Moriso, M. and Vetro, A.(2012), A Comparative Analysis of Software Reliability Growth Models Using Defects Data of Closed & Open Source Software, in Software Engineering Workshop (SEW), 2012, 35th Annual IEEE, pp 187-192.
[10] Rana, R., Staron, M., Berger, C., Hanson, J., Nilsson, M. and Torner, F.(2013), Evaluating the long-term predictive power of standard reliability growth models on automotive systems, Pasadena, CA, USA.
[11] Musa JD (1999), Software Reliability Engineering, McGraw- Hill.
[12] Kapur PK, Garg RB and Kumar S (1999), Contributions to Hardware and Software Reliability, World Scientific, Singapore.
[13] Bittanti S, Bolzern P, Pedrotti E, Scattolini R (1988). A flexible modelling approach for software reliability growth. Software Reliability Modelling and Identification (Ed.) G. Goos and J. Harmanis. Springer Verlag. Berlin. pp.101-140.
[14] Cheung RC (1980). A User Oriented Software Reliability Growth Model. IEEE Transactions on Software Engineering. SE-6: pp.118-125.
[15] Goel AL, Okumoto K (1979). Time dependent error detection rate model for software reliability and other performance measures. IEEE Transactions on Reliability. R-28(3): pp.206-211.
[16] Ohba (July 1984), Software reliability-analysis models, IBMJ. Res. Develop.,vol. 28, pp. 428-443.
[17] Goel AL (1985), Software reliability models: assumptions, limitations and applicability. IEEE Transactions on Software Engineering. SE-11, pp.1411-1423.
[18] Pham, H., Zhang X.(1997), An NHPP Software reliability model and its comparison, International Journal of Reliability, Quality and Safety Engineering, Vol. 4, No.3, pp 269-282.
[19] Jelinski Z. Moranda PB(1972). Software Reliability Research. In: Freiberger W. (Ed.) Statistical Computer performance Evaluation. New York: Academic Press: pp.465-497.
[20] Kapur PK, Garg RB.(1990c). Optimal release policies for software systems with testing effort. Int. Journal System Science. 22(9), pp.1563-1571.
[21] Kapur PK, Sharma KD, Garg RB. (1992). Transient solutions of software reliability model with imperfect debugging and error generation. Microelectronics and Reliability. 32: pp.475-478.
[22] Kapur PK, Garg RB, Kumar S (1999). Contributions to hardware and software reliability. Singapore: World Scientific Publishing Co. Ltd.
[23] Kareen N, Grover PS, Kapur PK (1990). An S-shaped Reliability growth model with two types of errors. Microelectronics and Reliability. 30(6):pp.1085-1090.
[24] Kremer W (1983). Birth–death bug counting. R-32: pp.32-47.
[25] Littlewood B, Verrall JL (1997). A Bayesian reliability growth model for computer software. Applied Statistics. 22: pp.332-346.
[26] Musa JD (1979). Validity of Execution Time Theory of Software Reliability. IEEE Transactions on Reliability. R-28: pp.181-191.
[27] Musa JD, Iannino A, Okumoto K (1987). Software reliability: Measurement, prediction, Applications. New York: MC Graw Hill.
[28] Ohba M (1984). Software Reliability Analysis Models. IBM Journal of Research and Development. 28: pp.428-443.
[29] Pfleeger SL(1998). Software Engineering- Theory and Practice. New Jersey: prentice Hall.
[30] Pham H(2000). Software Reliability. Springer-Verlag Pte. Singapore. 2000
[31] Schneidewind NF(1975). Analysis of Error Process in Computer Software. Sigplan Notices. 10(6): pp.337-346.
[32] Xie M, Bergman B (1980). On Modelling Reliability Growth For Software. Proceedings 8th IFAC symposium on Identification and Parameter estimation, Beijing; 1980; China: pp.27-31.
[33] Xie M (1991). Software Reliability Modeling. World Scientific.
[34] Xie M, Zhao M (1992). The Schneidewind Software Reliability Growth Model Revisited. IEEE proceedings. pp.184-192.
[35] Yamada S, Ohba M, Osaki S(1983). S-Shaped Software Reliability Growth Modelling for Software Error Detection. IEEE Trans. On reliability. R-32(5): pp.475-484.
[36] Yamada S, Osaki S, Narihisa H (1986). Discrete Models for Software Reliability Evaluation. In: Basu AP (Editor) Reliability and Quality Control. North Holland: Elsevier science Publishers.
[37] Yamada S, Hishitani J, Osaki S (1993). Software-Reliability Growth Model with a Weibull Test Effort: A Model And Application. IEEE Transactions on Reliability 42(1): pp.100-106.
[38] Yamada S, Tamura Y, Kimura M (2003). A Software Reliability Growth Model for A Distributed Development Environment. Electronics & Communications in Japan. part 3,Vol. 83, pp 1446-53.
[39] Zeephongsekul p, Xia G, Kumar S (1992). A Software Reliability Growth Model With Primary Errors

Generating Secondary Errors. Research report Number-2, Department of Mathematics; Royal Melbourne Institute of Technology, Australia.

[40] Singpurwalla ND, Wilson SP (1999). Statistical Methods in Software Engineering-Reliability and Risk. New York: Springer-Verlag.