Unified Framework for Developing Testing Effort Dependent Software Reliability Growth Models

P.K. KAPUR[1], OMAR SHATNAWI[2], ANU G. AGGARWAL[1], RAVI KUMAR[1]
[1]Department of Operational Research, University of Delhi, Delhi 110007, INDIA
[2]Department of Computer Science, Al al-Bayt University, Mafraq 25113, JORDAN
pkkapur1@gmail.com

Abstract: - Several software reliability growth models (SRGMs) have been presented in the literature in the last three decades. These SRGMs take into account different testing environment depending on size and efficiency of testing team, type of components and faults, design of test cases, software architecture etc. The plethora of models makes the model selection an uphill task. Recently, some authors have tried to develop a unifying approach so as to capture different growth curves, thus easing the model selection process. The work in this area done so far relates the fault removal process to the testing/execution time and does not consider the consumption pattern of testing resources such as CPU time, manpower and number of executed test cases. More realistic modeling techniques can result if the reliability growth process is studied with respect to the amount of expended testing efforts. In this paper, we propose a unified framework for testing effort dependent software reliability growth models incorporating imperfect debugging and error generation. The proposed framework represents the realistic case of time delays between the different stages of fault removal process i.e Failure Observation/Fault Detection and Fault Removal/Correction processes. The Convolution of probability distribution functions have been used to characterize time differentiation between these two processes. Several existing and new effort dependent models have been derived by using different types of distribution functions. We have also provided data analysis based on the actual software failure data sets for some of the models discussed and proposed in the paper.

Key-Words: - Software reliability growth model, Testing effort function, Imperfect debugging, Error generation, Convolution, Probability distribution function.

1 Introduction

The role of software is expanding rapidly in many aspects of modern life, ranging from critical infrastructures, such as transportation, defense, and telecommunication systems, to work-place automation, productivity enhancement, education, health-care, publishing, on-line services, entertainment, etc. Given the potentially costly impact of software failures for many of these applications, it is important to have sound methods of developing reliable software as well as accurate methods of quantitatively certifying software reliability. Hence it is crucial that software reliability engineering techniques should play a central role in the planning and control of software development projects. In particular, it is important to document the times and nature of bug occurrences, and their correction times, throughout the design and implementation phases as well as testing phase. With such data it is possible to estimate the time at which the software product will have reached a target level of reliability, or to devise methods to decrease that time.

A large number of software reliability growth models (SRGMs), which relate the number of failures (faults identified/corrected) and execution time, have been discussed in the literature [10,13,16,23]. These SRGM assume diverse testing environment like distinction between failure and correction processes, learning of the testing personnel, possibility of imperfect debugging and fault generation, constant or monotonically increasing / decreasing fault detection rate (FDR) or randomness in the growth curve. But no SRGM can be claimed to be the best as the physical interpretation of the testing and debugging changes due to numerous factors *e.g.,* design of test cases, defect density, skills and efficiency of testing team, availability of testing resources *etc*. The plethora of SRGM makes the model selection a tedious task. To reduce this difficulty, unified modeling approaches have been proposed by many researchers. These schemes have proved to be successful in obtaining several existing SRGM by following single methodology and thus provide an

insightful investigation for the study of general models without making many assumptions.

The work in this area started as early as in 1980s with Shantikumar [19] proposing a Generalized birth process model. Gokhale and Trivedi [6] used Testing coverage function to present a unified framework and showed how NHPP based models can be represented by probability distribution functions of fault–detection times. Dohi et al. [3] proposed a unification method for NHPP models describing test input and program path searching times stochastically by an infinite server queuing theory. Inoue [7] applied infinite server queuing theory to the basic assumptions of delayed S-shaped SRGM [22] *i.e.* fault correction phenomenon consists of successive failure observation and detection/correction processes and obtained several NHPP models describing fault correction as a two stage process.

Another unification methodology is based on a systematic study of fault detection process (FDP) and fault correction process (FCP) where FCPs are described by detection process with time delay. The idea of modeling FCP as a separate process following the FDP was first used by Schneidewind [18]. More general treatment of this concept is due to Xie *et al* [20,21] who suggested modeling of fault detection process as a NHPP based SRGM followed by fault correction process as a delayed detection process with random time lag. The recent unification scheme (due to Kapur *et al* [11]) is based on cumulative distribution function for the detection/correction times and incorporates the concept of change point in fault detection rate.

These unification schemes predict the fault content and reliability of the software with respect to the calendar time and do not consider the consumption pattern of resources such as computer time, manpower and number of executed test cases etc. More realistic unifying techniques can result if the reliability growth process is related to the amount of expended testing efforts. In this paper, we propose a generalized framework for deriving several existing as well as new testing effort dependent software reliability growth models with the possibility of imperfect debugging and error generation.

In practical software development scenario, As soon as a failure is observed, the efforts are made to correct the cause of the failure. It is quite possible that the testing team may not be able to remove/correct fault completely and the original fault may remain leading to a phenomenon known as imperfect debugging, or replaced by another fault resulting in error generation. In case of imperfect debugging the fault content of the software is not changed, but because of incomplete removal, the original detected fault is not corrected perfectly. But in case of error generation, the total fault content increases as the testing progresses because new faults are introduced in the system while removing the old original faults.

It was Goel [4] who first introduced the concept of imperfect debugging. Model due to Obha and Chou [12] is an error generation model applied on G-O model and has been also named as Imperfect debugging model. Kapur and Garg [10] introduced the imperfect debugging in Goel and Okumoto [5]. They assumed that the FDR per remaining faults is reduced due to imperfect debugging. Thus the number of failures observed/detected by time infinity is more than the initial fault content. Pham [15] developed an SRGM for multiple failure types incorporating error generation. Recently, Kapur *et al.* [9] proposed a flexible SRGM with imperfect debugging and error generation using a logistic function for fault detection rate which reflects the efficiency of the testing/removal team.

In this paper, we present a unified framework for software reliability growth modeling with respect to testing effort expenditure and incorporate the concept of imperfect debugging and error generation. This unified scheme is based on probability distribution functions. It is also shown that previously reported non-homogeneous Poisson process (NHPP) based SRGMs with imperfect debugging and error generation are special cases of the proposed framework. From this approach, we can not only obtain existing models but also develop some new NHPP models. The proposed models are formulated for the case when there is a time differentiation between failure observation / detection and fault removal/correction processes. Here we have used different standard probability distribution functions for representing failure observation and fault correction times. These distribution functions have been discussed briefly to demonstrate their utility and applicability for representing these random times.

The existing and new models derived here have been validated and evaluated on two actual software failure data sets. Non-linear regression based on least square method has been used for parameter estimation and MSE (mean squared error) and R^2 (coefficient of multiple determination) has been used as the performance comparison criteria. For faster and accurate calculations, the statistical package SPSS has been

utilized for the purpose. The goodness of fit curves have been drawn to illustrate the fitting of the models to the data graphically.

Rest of this paper is organized as follows: Section 2 mentions the basic assumptions made followed by the model development under imperfect debugging and error generation. This section describes the unified framework for testing effort dependent software reliability growth models by considering time differentiation between failure observation/detection and fault removal/correction processes. In section 3 we derive many existing and new software reliability growth models by using different probability distribution functions. Section 4 shows numerical examples for the proposed models based on two real software failure data sets. Finally, conclusions are drawn in section 5.

Notations

$m(W_t)$	Mean value function (MVF) or the expected number of faults corrected by time t.
a	Expected number of faults lying dormant in the software when the testing starts i.e at $t=0$.
W_t	Amount of testing effort expended by time t.
$a(W_t)$	Total fault content of software dependent on testing effort expended
$\lambda(W_t)$	Intensity function for fault correction process (FCP) or fault correction rate per unit time.
$G(W_t)$, $F(W_t)$	Testing effort dependent probability distribution function for failure observation and fault correction Times
$g(W_t)$, $f(W_t)$	Testing effort dependent probability density function for failure observation and fault correction times
*	Convolution.
$\otimes$	Steiltjes convolution.

2 Unified Framework for Modeling Reliability Growth with Time Differentiation between Failure Observation/Detection and Fault Removal/Correction

2.1 Basic Assumptions

The model is based on the following assumptions:

1. Software system is subject to failure during execution caused by faults remaining in the system.
2. The number of faults detected at any time instant is proportional to the remaining number of faults in the software. Each time a failure is observed, immediate correction effort starts and the following may occur:
   (a) Fault content is reduced by one with probability (p).
   (b) Fault content remains unchanged with probability ($1-p$).
3. During the fault correction process, whether the fault is removed successfully or not, new faults are generated with a constant probability α.
4. The Fault correction times are i.i.d. random variables with probability distribution function
   $$F(W_t) = \int_0^{W_t} f(x)dx$$
   where $F(W_t)$ is testing effort dependent distribution function.
5. The fault correction process is modeled by NHPP.
6. The initial number of failure observed in the software system at $t=0$ is a Poisson random variable with mean of a.

2.2 Model Development

Let the counting processes $\{X(W_t),\ t\geq 0\}$ and $\{N(W_t),\ t\geq 0\}$ represent the cumulative number of failures observed and faults corrected up to time t respectively and let the test begun at time $t=0$. Then the distribution of $N(W_t)$ is given by

$$\Pr\{N(W_t)=n\} = \sum_{j=0}^{\infty} \Pr\{N(W_t)=n \mid X(0)=j\} \Pr\{X(0)=j\} \quad (1)$$

Here it can be noted that the conditional probability $\Pr\{N(W_t)=n \mid X(0)=j\}$ is zero for $j<n$. For $j\geq n$ it is given by

$$\Pr\{N(W_t)=n \mid X(0)=j\} = \binom{j}{n} (F(W_t))^n (1-F(W_t))^{j-n} \quad (2)$$

Therefore, we have

$$\Pr\{N(W_t)=n\} = \sum_{j=0}^{\infty} \binom{j}{n} (F(W_t))^n (1-F(W_t))^{j-n} \frac{a^j \exp(-a)}{j!}$$

$$= \frac{[aF(W_t)]^n}{n!} \exp(-a) \sum_{j=0}^{\infty} \frac{[a(1-F(W_t))]^{j-n}}{(j-n)!}$$

Here it can be noted that

$$\sum_{j=0}^{\infty} \frac{[a(1-F(W_t))]^{j-n}}{(j-n)!} = \exp(a(1-F(W_t)))$$

From above we obtain

$$\Pr\{N(W_t)=n\} = \frac{(a\ F(W_t))^n \exp(-a\ F(W_t))}{n!} \quad (3)$$

Hence we can conclude that the fault correction process is poison with mean value function (MVF) as given by:

$$m(W_t) = E[N(W_t)] = a\ F(W_t) \quad (4)$$

As specified before, here $F(W_t)$ is the testing effort dependent probability distribution function for fault correction times. It may be noted that $F(W_t)$ is so defined that it satisfies all the properties of probability distribution functions.

1. At t=0, W_t=0 and $F(W_t)$. In this paper, we have used three types of testing effort function namely Exponential, Rayleigh and Weibull type. All these functions satisfy the property that at t=0, W_t=0. It can be verified from their expressions, discussed in detail in appendix at the end of the paper.
2. For t>0, W_t>0 and $F(W_t)$>0. In this paper we have assumed $F(W_t)$ to be either Exponential, Gamma, Weibull or Normal type. As t increases, W_t also increases indicating monotonically increasing nature of $F(W_t)$. Similarly the continuity of $F(W_t)$ can also be explained.
3. As testing continues for an infinitely large time, i.e., $t\to\infty$, $W_t\to \overline{W}$, the corresponding value of distribution function $F(W_t)$ is $F(\overline{W})$. Here $\overline{W}$ is very large positive number representing the upper bound on the availability of the amount of testing resources available. Therefore, $F(\overline{W})$ can be assumed to be of order 1.

From Equation (4), the instantaneous failure intensity function $\lambda(W_t)$ is given by:

$$\lambda(W_t) = aF'(W_t)$$

Or we can write

$$\lambda(W_t) = \frac{dm/dt}{dW_t/dt} = [a - m(W_t)]\frac{F'(W_t)}{1-F(W_t)} \quad (5)$$

Let us define

$$s(W_t) = \frac{F'(W_t)}{1-F(W_t)}$$

Here $s(W_t)$ represents hazard rate function or failure occurrence rate per remaining fault of the software, or the rate at which the individual faults manifest themselves as failures during testing or hazard rate function. The expression of hazard rate function $s(W_t)$ in terms of probability distribution function gives the directions for incorporating the case of time differentiation between the stages of failure observation and fault correction.

2.3 Proposed Testing Effort Dependent Modeling

Let us consider the case when there is a time delay between the observation of the failure and the correction of the underlying fault. This time delay can be due to various factors e.g. severity / complexity of the faults, change in defect density, skill of the testing team etc. Then FCP is no longer a one-stage process. The correction may be a two / three stage process namely failure observation, fault detection followed by the fault removal/correction. This division of fault correction into different processes defines the complexity of faults present in software. More the delay in removal/correction of a fault on its observation/detection, more complex is the fault. In that case, Equation (5) can be modified as:

$$\lambda(W_t) = \frac{dm/dt}{dW_t/dt} = \frac{(f*g)(W_t)}{1-(F\otimes G)(W_t)}[a - m(W_t)] \quad (6)$$

or, $$\lambda(W_t) = h(W_t)[a - m(W_t)]$$

where $h(W_t) = \frac{(f*g)(W_t)}{1-(F\otimes G)(W_t)}$ is the failure observation/detection-fault removal/correction rate.

Upon solving we get:

$$m(W_t) = a(F\otimes G)(W_t) \quad (7)$$

By selecting suitable probability distribution functions, we can derive MVF for several existing and new Finite failure count models. This equation represents two stage fault correction under perfect debugging conditions.

Now let us consider the case when faults can be introduced during the debugging phase with a constant fault introduction rate α. Therefore, the fault content rate function $a(W_t)$ is a linear function of the expected number of faults detected by time t. That is,

$$a(W_t) = a + \alpha\ m(W_t)$$

Now incorporating imperfect debugging and error generation in proposed modeling, we have

$$\lambda(W_t)=\frac{dm/dt}{dW_t/dt}=p\frac{(f*g)(W_t)}{1-(F\otimes G)(W_t)}\left[a+\alpha m(W_t)-m(W_t)\right] \quad (8)$$

where p is the probability of perfect debugging.

Solving Equation (8) with initial condition that at t=0, $m(0)$=0 and W_0=0, we get:

$$m(W_t)=\frac{a}{(1-\alpha)}\left[1-(1-(F\otimes G)(W_t))^{p(1-\alpha)}\right] \quad (9)$$

Here If p=1 and α=0, i.e. perfect debugging, Equation (9) is nothing but Equation (7).

The mean value functions $m(W_t)$ for various models can be derived by using different types of distribution functions $F(W_t)$ and $G(W_t)$.

2.4 Particular Cases

Here if we define failure observation times distribution, i.e., $G(W_t)$ is unit function, then Equation (7) is same as the Equation (4) and Equation (9) becomes

$$m(W_t)=\frac{a}{1-\alpha}\left[1-(1-F(W_t))^{p(1-\alpha)}\right] \quad (10)$$

This equation defines the removal process as one stage process where no time is lost between the failure observation and its removal. This case has been discussed in detail in [1].

3 Derivation of Existing and New SRGM

3.1 Probability Distribution Functions for Modeling Detection/Correction Times

In this paper we have used the following probability distributions functions for random failure observation / detection and correction times.

Distribution	Description / Application
Exponential	This is the most simple and widely used distribution in reliability engineering modeling because it has a constant rate. It indicates the uniform distribution of faults in the software code where each and every fault has same probability for its removal. Though in most of the software testing projects, for sake of simplicity, the removal times are assumed to follow exponential distribution, but to achieve a more flexible modeling of removal times, we can use Weibull or Gamma distribution. Both of these distributions are generalization of Exponential distribution only and have very similar shapes.
Weibull	It can represent different types of curves depending on the values of its shape parameter and hence extremely flexible. It is very appropriate for representing the processes with fluctuating rate i.e. increasing /decreasing rates.
Gamma / Erlang	Gamma and Erlang distributions are extensions of Exponential distribution where the fault removal consists of multiple steps e.g. generation of failure report, its analysis and correction time followed by verification and validation.
Normal	During testing, there are numerous factors, which affect the fault correction process. These factors can be internal e.g. defect density, complexity of the faults, the internal structure of the software or the factors can be external and come from the testing environment itself e.g. design of the test cases, skill of the testers / test case designers, testing effort availability/consumption. etc. This two-parameter distribution can describe the correction times quite well for the cases where correction time depends on multiple factors.

By combining above-mentioned distributions in our proposed UM approach; we can explain a number of existing SRGM formulated for different T&D scenario. In the next section we discuss how to obtain MVF of the various existing SRGM and propose few new models also.

3.2 MVF for Various New and Existing Models

The mean value functions $m(W_t)$ corresponding to different forms of distribution functions $F(W_t)$ and $G(W_t)$ are summarized in Table I.

Table I

Model	$F(W_t)$	$G(W_t)$	$m(W_t)$
SRGM-1	$W_t \sim \exp(b)$	$1(W_t)$	$\frac{a}{1-\alpha}\left[1 - e^{-bp(1-\alpha)W_t}\right]$
SRGM-2	$W_t \sim \exp(b)$	$W_t \sim \exp(b)$	$\frac{a}{1-\alpha}\left[1-\left((1+bW_t)e^{-bW_t}\right)^{p(1-\alpha)}\right]$
SRGM-3	$W_t \sim \exp(b_1)$	$W_t \sim \exp(b_2)$	$\frac{a}{1-\alpha}\left[1-\left\{\frac{1}{b_1-b_2}\left(b_1e^{-b_2t} - b_2e^{-b_1t}\right)\right\}^{p(1-\alpha)}\right]$
SRGM-4	$W_t \sim$Erlang-2 (b)	$W_t \sim \exp(b)$	$\frac{a}{1-\alpha}\left[1-\left(\left(1+bW_t+\frac{b^2W_t^2}{2}\right)e^{-bW_t}\right)^{p(1-\alpha)}\right]$
SRGM-5	$W_t \sim Wei(b,k)$ (Weibull Distribution)	$1(W_t)$	$\frac{a}{1-\alpha}\left[1 - e^{-bp(1-\alpha)W_t^{k}}\right]$
SRGM-6	$W_t \sim N(\mu,\sigma^2)$ (Normal Distribution)	$1(W_t)$	$\frac{a}{1-\alpha}\left[1-\left(1-\phi(W_t,\mu,\sigma)\right)^{p(1-\alpha)}\right]$
SRGM-7	$W_t \sim N(\mu,\sigma^2)$	$W_t \sim \exp(b)$	$\frac{a}{1-\alpha}\left[1-\left(\begin{array}{l}1-\varphi(t,\mu,\sigma)\\ +e^{\left(-bt+\mu b+\frac{(b\sigma)^2}{2}\right)}\varphi(t,\mu+b\sigma^2,\sigma)\end{array}\right)^{p(1-\alpha)}\right]$
SRGM-8	$W_t \sim \gamma(\alpha_1,\beta_1)$	$W_t \sim \exp(b)$	$\frac{a}{1-\alpha}\left[1-\left(\begin{array}{l}1-\Gamma(t,\alpha_1,\beta_1)\\ +\frac{e^{-bt}}{(1-b\beta_1)^{\alpha_1}}\Gamma\left(t,\alpha_1,\frac{\beta_1}{1-b\beta_1}\right)\end{array}\right)^{p(1-\alpha)}\right]$

4 Model Validation, Comparison Criteria and Data Analyses

4.1 Model Validation

To illustrate the estimation procedure and application of the SRGM (existing as well as proposed) we have carried out the data analysis of the following two real software data sets.

Data set	Software Project / Program Description
DS-1	The first data set (DS-1) had been collected during 35 months of testing a radar system of size 124 KLOC and 1301 faults were detected during testing [2].
DS-1I	The second data set (DS-2) had been collected during 19 weeks of testing a real time command and control system of size 1317 KLOC and 328 faults were detected during testing [14].

4.2 Comparison Criteria for SRGM

The performance of SRGM are judged by their ability to fit the past software fault data (goodness of fit) and predicting the future behavior of the fault.

Goodness of Fit criteria

The term goodness of fit is used in two different contexts. In one context, it denotes the question if a sample of data came from a population with a specific distribution. In another context, it denotes the question of "How good does a mathematical model (for example a linear regression model) fit to the data"?

a. The Mean Square-Error (MSE):

The model under comparison is used to simulate the fault data, the difference between the expected values, $\hat{m}(t_i)$ and the observed data y_i is measured by MSE as follows.

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

where k is the number of observations. The lower MSE indicates less fitting error, thus better goodness of fit [10].

b. Coefficient of Multiple Determination (R^2):

We define this coefficient as the ratio of the sum of squares resulting from the trend model to that from constant model subtracted from 1.

$$R^2=1-\frac{\text{residual SS}}{\text{corrected SS}}$$

R^2 measures the percentage of the total variation about the mean accounted for the fitted curve. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well. The larger R^2, the better the model explains the variation in the data [10].

c. Bias:

The difference between the observation and prediction of number of failures at any instant of time i is known as PE_i (prediction error). The average of PEs is known as bias. Lower the value of Bias better is the goodness of fit [17].

d. Variation:

The standard deviation of prediction error is known as variation.

$$Variation=\sqrt{\left(1/N-1\right)\sum\left(PE_i-Bias\right)^2}$$

Lower the value of Variation better is the goodness of fit [17].

e. Root Mean Square Prediction Error:

It is a measure of closeness with which a model predicts the observation.

$$RMSPE=\sqrt{\left(Bias^2+Variation^2\right)}$$

Lower the value of Root Mean Square Prediction Error better is the goodness of fit [17].

In other words, we evaluate the performance of the models under comparison using MSE, R^2, Bias, Variation, and RMSPE metrics. For MSE, Bias, Variation, and RMSPE, the smaller the metric value the better the model fits relative to other models run on the same data set. For R^2, the larger the metric value the better.

4.3 Data Analyses

The SRGM with mean value function $m(t)$ given in Table I are estimated for finding their unknown parameters. For testing effort estimation we have worked out results on all three effort functions namely Exponential, Rayleigh and Weibull. But for model parameter estimation we have used Weibull function as it gives best results as compared to other two effort functions. The estimated values of testing effort function parameters are given in Tables II and III for DS-1 and DS-2 respectively.

For DS-1

The parameter estimation and comparison criteria results for DS-1 of all the models under consideration can be viewed through Table IV and Table V. The fitting of the models to DS-1 is graphically illustrated in Fig. 1.1 and Fig.1.2. SRGM-5 shows a poor fitting to the actual values of the real time data set while all other models fit the data excellently well.

For DS-2

The parameter estimation and comparison criteria results for DS-2 of all the models under consideration can be viewed through Table VI and Table VII. The fitting of the models to DS-2 is graphically illustrated in Fig.2.1 and Fig.2.2. SRGM-4 shows a poor fitting to the actual values of the real time data set while all other models fit the data quite well.

Tables: Parameters Estimation and Comparison Criteria

Table II: Estimation of Testing Effort Function Parameters for DS-I

Testing Effort Function	Parameter Estimation		
	$\overline{W}$	v	l
Exponential	1030421	.0000461	-
Rayleigh	2873	.00173	-
Weibull	2669	.0007729	2.07

Table III: Estimation of Testing Effort Function Parameters for DS-II

Testing Effort Function	Parameter Estimation		
	$\overline{W}$	v	$\overline{l}$
Exponential	8544	.000288	-
Rayleigh	49	013681	-
Weibull	799	.002328	1.11

Table IV: Parameter Estimates for DS-1

Models	a	b/b_1	b_2/k	p	α	μ	σ	α_1	β_1
SRGM-1	1193	.0104	-	.1362	.2279	-	-	-	-
SRGM-2	1484	.0956	-	.0131	.0025	-	-	-	-
SRGM-3	1357	.0447	.0453	.0330	.0676	-	-	-	-
SRGM-4	1484	.3779	-	.0032	.0147	-	-	-	-
SRGM-5	1446	.0991	.5718	.1327	.9741	-	-	-	-
SRGM-6	1252	-	-	.1118	.0147	1.242	181.34	-	-
SRGM-7	1522	.2008	-	.0055	.0147	.2147	.1245	-	-
SRGM-8	1435	.0164	-	.0733	.0828	-	-	.2304	.0136

Table V: Model Comparison Results for DS-1

Models	R^2	MSE	BIAS	VARAITION	RMSPE
SRGM-1	.99470	1121	-7.59343	33.09411	33.95409
SRGM-2	.99490	1085	-5.19371	33.01314	33.41919
SRGM-3	.99473	1121	-7.59343	33.09411	33.95409
SRGM-4	.99482	1102	-3.95571	33.45488	33.68793
SRGM-5	.97470	5387	17.54	72.31	74.407
SRGM-6	.99684	672	1.63	26.25	26.300
SRGM-7	.99471	1127	-0.28057	34.07326	34.07442
SRGM-8	.99470	1129	-2.19771	34.02571	34.09661

Table VI: Parameter Estimates for DS-2

Models	a	b/b_1	b_2/k	p	α	μ	σ	α_1	β_1
SRGM-1	332	.1647	-	.2026	.4111	-	-	-	-
SRGM-2	406	.9536	-	.0364	.1147	-	-	-	-
SRGM-3	379	.9853	.3818	.0856	.2368	-	-	-	-
SRGM-4	332	.8958	-	.0592	.1824	-	-	-	-
SRGM-5	427		.8441	.2067	.6478	-	-	-	-
SRGM-6	318	-	-	.1945	.0741	2.842	8.027	-	-
SRGM-7	367	.4368	-	.0721	.3158	.4535	.2578	-	-
SRGM-8	418	.1392	-	.1896	.6475	-	-	.0417	.0293

Table VII: Model Comparison Results for DS-2

Models	R^2	MSE	BIAS	VARAITION	RMSPE
SRGM-1	.98867	116	0.563	11.095	11.109
SRGM-2	.98871	122	-1.365	11.262	11.345
SRGM-3	.98862	138	-2.043	11.90	12.074
SRGM-4	.98458	159	-2.66	12.67	12.95
SRGM-5	.98487	156	1.42	12.76	12.83
SRGM-6	.99196	82	0.21	9.35	9.36
SRGM-7	.98901	92	0.047	9.85	9.855
SRGM-8	.99081	94	0.238	10.00	10.01

Figures: Goodness of Fit Curves

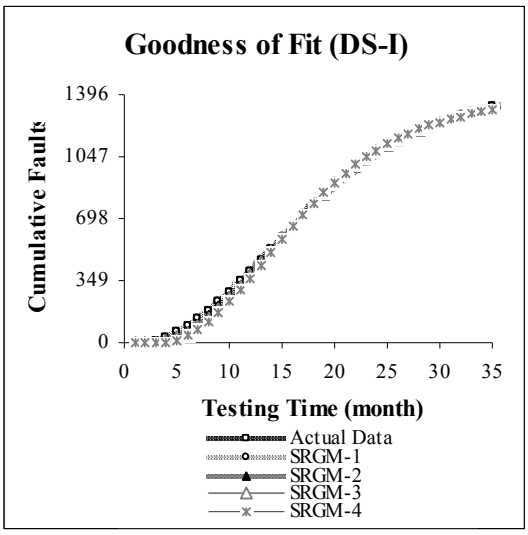

Figure.1.1

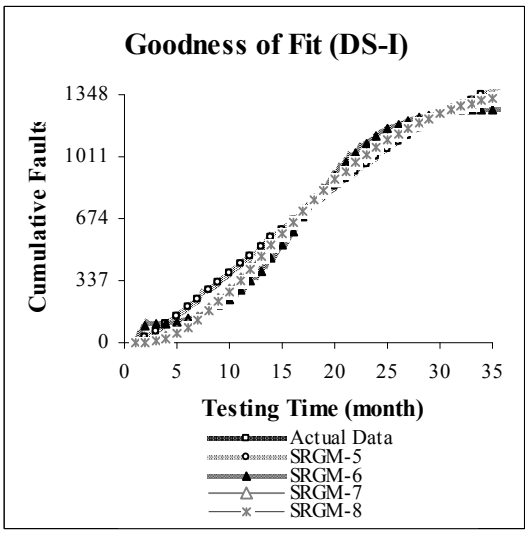

Figure.1.2

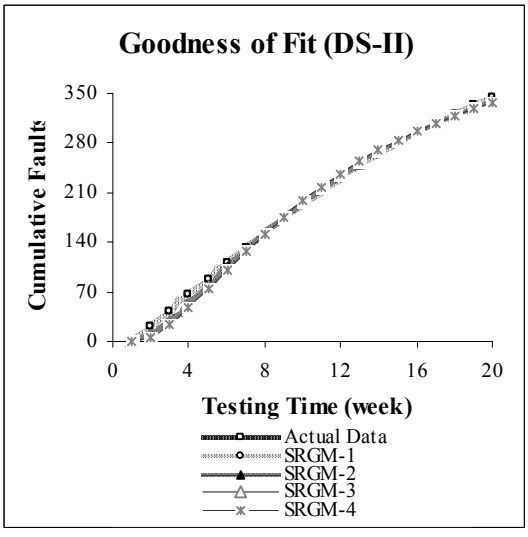

Figure.2.1

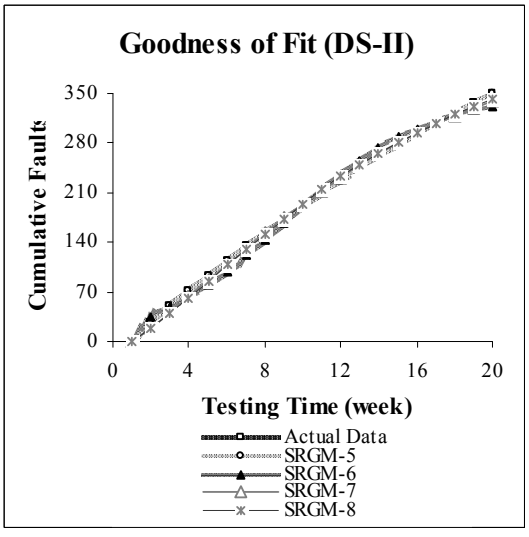

Figure.2.2

5 Conclusion

In this paper, a unified framework for testing effort dependent software reliability growth models has been discussed under the assumption that failure observation/detection has time difference to the fault correction process. More realistic software testing scenario has been modeled by incorporating the possibility of two types of imperfect debugging i.e. imperfect debugging and error generation. The framework presented here proves to be excellent for deriving a wide variety of effort dependent models by using different probability distribution functions. The technique is simple and presents a unique methodology for developing many new as well as existing models for different design environment. The scope for future research in this area lies for the case when reliability growth is studied with respect to number of test-cases executed i.e. discrete time unified modeling framework. In this paper we have used standard distributions e.g. Exponential, Weibull, Erlang k-type, Normal and Gamma for correction times. Their validity and accuracy have been carried out on two real software failure datasets. The results obtained are quite encouraging as can be viewed through the numerical illustrations shown in tables obtained after the parameter estimation. In future work the possibility of including change point or the modeling using stochastic differential equations can be worked out. The concept of unification provides an area of interesting study which can ease out the problem of model selection for the software developer and thus make these techniques more accessible and applicable.

References:

[1] Aggarwal G Anu, Kumar Ravi and PK Kapur, A New Approach for Developing Testing Effort Dependent Software Reliability Growth Models, *In: Proc. of the 3rd National Conference on Computing for Nation Development* (INDIACoM-2009), New-Delhi, India, 2009, pp.425-432.

[2] Brooks WD and Motley RW, *Technical Report*, Rome Air Development Center, New York, 1980.

[3] Dohi T, Osaki S, and Trivedi KS, An Infinite Server Queuing Approach for Describing Software Reliability Growth: Unified Modeling and Estimation Framework, In: *Proc. of the 11th Asia-Pacific Software Engineering Conference* (APSEC'04), Pusan, Korea, 2004, pp.110-119.

[4] Goel AL, Software Reliability Models: Assumptions, Limitations and Applicability, *IEEE Transactions on Software Engineering*, Vol.11, No.21, 1985, pp.1411-1423.

[5] Goel AL and Okumoto K, Time Dependent Error Detection Rate Model for Software Reliability and other Performance Measures, *IEEE Transactions on Reliability*, Vol.28, No.3, 1979, pp.206-211.

[6] Gokhale SS, Philip T, Marinos PN and Trivedi KS, Unification of Finite Failure Non-Homogeneous Poisson Process Models through Test Coverage, In: Proc. *Int'l Symposium on Software Reliability Engineering* (ISSRE 96), NY, USA, 1996, pp. 289-299.

[7] Inoue S, A study on Stochastic Modelling for Accurate Software Reliability Assessment, *Ph.D. Thesis*, Doctoral Program of Graduate School of Engineering, Tottori University, Japan, 2006.

[8] Kapur PK, Aggarwal G Anu and Anand Sameer, A New Insight into Software Reliability Growth Modeling, *International Journal of Performability Engineering*, Vol.5, No.3, 2009, pp. 267-274.

[9] Kapur PK, Kumar D, Gupta A and Jha PC, On How To Model software Reliability Growth in the Presence Of Imperfect Debugging and error Generation, In: *Proc. of the 2nd Int'l Conference on Reliability and Safety Engineering*, Chennai, India, 2006, pp.515-523.

[10] Kapur PK, Garg RB and Kumar S, *Contributions to Hardware and Software Reliability*, World Scientific, 1999.

[11] Kapur PK, Kumar J and Kumar R, A Unified Modeling Framework Incorporating Change Point for Measuring Reliability Growth During Software Testing, *OPSEARCH*, Special Issue on Quantitative Assessment of Software Reliability (Eds. Kapur PK and Pham H), Vol.45, No.4, 2008, pp.317-334.

[12] Ohba M and Chou XM, Does Imperfect Debugging Effect Software Reliability Growth, In: *Proc. of 11th Int'l Conference of Software Engineering*, Pittsburgh, Pennsylvania, USA, 1989, pp.237-244.

[13] Musa JD, Iannino A and Okumoto K, *Software Reliability*: *Measurement, Prediction, Applications*, McGraw-Hill, 1987.

[14] Ohba M, Software Reliability Analysis Models, *IBM Journal of Research and*

Development, Vol.28, No.4, 1984, pp.428-443.

[15] Pham H and Zhang X, An NHPP Software Reliability Models and its Comparison, *International Journal of Reliability, Quality and Safety Engineering*, Vol.4, No.3, 1997, pp.269-282.

[16] Pham H, *System Software Reliability*, Reliability Engineering Series, Springer, 2006.

[17] Pillai K and Nair VSS, A Model for Software Development Effort and Cost Estimation, *IEEE Transactions on Software Engineering*, Vol.23, No.8, 1997, pp.485-497.

[18] Schneidewind NF, Analysis of Error Processes In Computer Software, *Sigplan Notices*, Vol.10, No.6, 1975, pp.337–346.

[19] Shanthikumar JG, A General Software Reliability Model for Performance Prediction, *Microelectronics Reliability*, Vol.21, No.5, 1981, pp.671–682.

[20] Xie M and Zhao M, The Schneidewind Software Reliability Model Revisited, In: *Proc. of the 3rd Int'l Symposium on Software Reliability Engineering*, Research Triangle Park, NC, USA 1992, pp.184–192.

[21] Xie M, QP Hu, Wu YP and Ng SH, A Study of the Modeling and Analysis of Software Fault-Detection and Fault-Correction Processes, *Quality and Reliability Engineering International*, Vol.23, No.4, 2007, pp.459-470.

[22] Yamada S, Ohba M and Osaki S, S-shaped Reliability Growth Modelling for Software Error Detection, *IEEE Transactions on Reliability*, Vol.32, No.5, 1983, pp.475-478.

[23] Shatnawi Omar and Kapur PK, A Generalized Software Fault Classification Model, *WSEAS Transactions on Computers*, Vol.7, No.9, 2008, pp.1375-1384.

[**24**] Junhong G, Hongwei L, Xiaozong Y, and Cheng ZD, A Software Reliability Time Series Growth Model with Kalman Filter, *WSEAS Transactions on Computers*, Vol.5, No.1, 2006, pp.1-8.

[**25**] Junhong G, Hongwei L, Xiaozong Y, A Software Reliability Time Series Growth Model Transformed from Goel-Okumoto Model, *WSEAS Transactions on Signal Process*, Vol.1, No.1, 2005, pp.39-46.

Appendix

Description of Testing Effort function

The testing resources spent during testing of any software basically, include manpower used for fault detection/removal and CPU time spent in executing software under test. Greater the amount of testing effort faster is the testing process. The testing effort (resources) that govern the pace of testing for almost all the software projects are [13]:

1. Manpower
2. Computer time.

The key function of manpower engaged in software testing is to design and run test cases and compare the test results with desired specifications. Any departure from the specifications is termed as a failure. On a failure the fault causing it is identified and then removed by failure correction personnel. During testing continuous monitoring is done to analyze the progress of testing and quality achieved. The computer facilities represent the computer time, which is necessary for failure identification and correction.

The Functions which have been used in this paper to explain the testing effort are- Exponential, Rayleigh and Weibull.

They can be derived from the assumption that, "The testing effort rate is proportional to the testing resources available".

$$\frac{dW_t}{dt} = v(t)\left[\overline{W} - W_t\right]$$

where $v(t)$ is the time dependent rate at which testing resources are consumed, with respect to remaining available resources.

For solving this differential equation, we use initial condition that at

Case 1: When $v(t)=v$, a constant, we get Exponential function:

$$W_t = \overline{W}\left(1 - e^{-vt}\right)$$

Case 2: If $v(t)= v.t$, we get Rayleigh type curve:

$$W_t = \overline{W}\left(1 - e^{-v t^2/2}\right)$$

Case 3: If $v(t)=v.l.t^{l-1}$, we get Weibull function:

$$W_t = \overline{W}\left(1 - e^{-v\, t^l}\right)$$

To study the testing effort process, one of the above functions can be selected.