Discrete Time NHPP Models for Software Reliability Growth Phenomenon

Omar Shatnawi
Prince Hussein bin Abdullah Information Technology College, al-Bayt University, Jordan

Abstract: *Nonhomogeneous poisson process based software reliability growth models are generally classified into two groups. The first group contains models, which use the machine execution time or calendar time as a unit of fault detection/removal period. Such models are called continuous time models. The second group contains models, which use the number of test occasions/cases as a unit of fault detection period. Such models are called discrete time models, since the unit of software fault detection period is countable. A large number of models have been developed in the first group while there are fewer in the second group. Discrete time models in software reliability are important and a little effort has been made in this direction. In this paper, we develop two discrete time SRGMs using probability generating function for the software failure occurrence / fault detection phenomenon based on a NHPP namely, basic and extended models. The basic model exploits the fault detection/removal rate during the initial and final test cases. Whereas, the extended model incorporates fault generation and imperfect debugging with learning. Actual software reliability data have been used to demonstrate the proposed models. The results are fairly encouraging in terms of goodness-of-fit and predictive validity criteria due to applicability and flexibility of the proposed models as they can capture a wide class of reliability growth curves ranging from purely exponential to highly S-shaped.*

Keywords: *Software engineering, software testing, software reliability, software reliability growth model, nonhomogeneous poisson process, test occasions.*

Received August 19, 2007; accepted December 9, 2007

1. Introduction

Software reliability assessment is important to evaluate and predict the reliability and performance of software system. The models applicable to the assessment of software reliability are called Software Reliability Growth Models (SRGMs). An SRGM provides a mathematical relationship between time span of testing or using the software and the cumulative number of faults detected. It is used to assess the reliability of the software during testing and operational phases.

An important class of SRGM that has been widely studied is NonHomogeneous Poisson Process (NHPP). It forms one of the main classes of the existing SRGM; due to it is mathematical tractability and wide applicability. NHPP models are useful in describing failure processes, providing trends such as reliability growth and fault-content. SRGM consider the debugging process as a counting process characterized by the mean value function of a NHPP. Software reliability can be estimated once the mean value function is determined. Model parameters are usually determined using either Maximum Likelihood Estimate (MLE) or least-square estimation methods [6, 13, 18].

NHPP based SRGM are generally classified into two groups [2, 6, 17]. The first group contains models, which use the execution time (i.e., CPU time) or calendar time. Such models are called continuous time models. The second group contains models, which use the number of test cases as a unit of fault detection period. Such models are called discrete time models, since the unit of software fault detection period is countable. A test case can be a single computer test run executed in an hour, day, week or even month. Therefore, it includes the computer test run and length of time spent to visually inspect the software source code [2, 6]. A large number of models have been developed in the first group while fewer are there in the second group due to the difficulties in terms of mathematical complexity involved.

The utility of discrete time models cannot be underestimated since the number of test cases (i.e., executed test run) is more appropriate measure of the fault detection/removal period than the CPU/calendar time used by continuous time model. These models generally provide a better fit than their continuous time counterparts because they are more realistic. Therefore, in spite of difficulties in terms of mathematical complexity involved, discrete time models are proposed regularly.

Yamada and Osaki [17] discussed two classes of models. One class describes the fault detection process in which the expected number of faults per test case is geometrically decreasing whereas, in the second is proportional to the current fault-content and this class is known as exponential model. Kapur *et al.* [6] proposed a model in which the testing phase is assumed to have two different processes namely, fault isolation and fault detection processes, this model is known as delayed S-shaped model. Further they proposed another model based on the assumption that software may contain several types of faults. In these models the fault removal process (i.e., debugging process) is assumed to be perfect. But due to the complexity of the software system and the incomplete understanding of software requirements, specifications and structure, the testing team may not be able to remove the fault perfectly and the original fault may remain or replaced by another fault. While the first phenomenon is known as imperfect debugging. The second is called error/fault generation.

The concept of imperfect debugging was first introduced by Goel [3]. He introduced the probability of imperfect debugging in Jelinski and Moranda model [5]. Kapur *et al.* [6] introduced the imperfect debugging in Goel and Okumoto model [4]. They assumed that the fault removal rate per remaining faults is reduced due to imperfect debugging. Thus the number of failures observed by time infinity is more than the initial fault content. Although these two models describe the imperfect debugging phenomenon yet the software reliability growth curve of these models is always exponential. Moreover, they assume that the probability of imperfect debugging is independent of the testing time. Thus, they ignore the role of the learning process during the testing phase by not accounting for the experience gained with the progress of software testing. Actually, the probability of imperfect debugging is supposed to be a maximum in the early stage of the testing phase and is supposed to reduce with the progress of testing [1, 6, 7, 8, 10, 12, 13, 14, 15, 16].

In this paper, we propose two discrete time NHPP based SRGMs for the situation given above. The assumptions in this case are with respect to number of test cases instead of time. The rest of this paper is organized as follows. Section 2 derives the proposed two flexible discrete time models. Sections 3 and 4 discuss the methods used for parameter estimation and the criteria used for validation and evaluation of the proposed models. The applications of the proposed discrete to actual software reliability data through data analyses and model comparisons are shown in section 5. We conclude this paper in section 6.

2. Software Reliability Modelling

2.1. Basic Discrete Time Model

2.1.1. Model Development

The main assumption of SRGM is that the failure observation / fault detection depends linearly upon the number of remaining faults [4]. This assumption implies that all faults are equally likely to be detected.

In practical situation it has been observed that large number of simple faults is easily detected at the early stages of the testing, while the fault detection may become extremely difficult in the later stages of the testing phase. In this case, the fault detection rate has a high value at the beginning of the testing as compared to the value at the end of the testing. On the contrary, it may happen that the detection of faults increases the skill of the testing team and thus leading to increase in efficiency. In other words, the testing team detects more faults in less time. This implies that the value of the fault detection rate at the end of the testing phase is higher than it is value at the beginning of the testing. Bittanti *et al.* [1] exploited this change in fault detection rate, which they termed as the Fault Exposure Coefficient (FEC) for their SRGM.

Through real data experiments and analysis on several software development projects, the fault detection rate has three possible trends as time progresses; increasing, decreasing or constant [1, 10]. It decreases when the software has been used and tested repeatedly, showing reliability growth. It can also increase if the testing techniques/requirements are changed, or new faults are introduced due to new software features or imperfect debugging. Thus, we treat the fault detection rate as a function of the number of test cases to interpret these trends.

2.1.2. Model Assumptions, Notations, and Formulation

- Failure observation/fault detection phenomenon is modelled by NHPP.
- Software is subject to failures during execution caused by faults remaining in the software.
- Each time a failure occurs, the fault that caused it is immediately and perfectly detected, and no new faults are introduced, i.e., the debugging process is perfect.

a	Initial fault content of the software.
$b(n+1)$	Fault detection rate function is dependent on the number of test cases and behaves linearly on the number of faults remaining.
b_i	Initial fault detection rate.
b_f	Final fault detection rate.
$m(n)$	The expected mean number of faults detected by the n^{th} test case.

Under the given model assumptions, the expected cumulative number of faults detected between the n^{th} and $(n+1)^{th}$ test cases is proportional to the number of faults remaining after the execution of the n^{th} test run, satisfies the following difference equation:

$$m(n+1) - m(n) = b(n+1)\big(a - m(n)\big) \qquad (1)$$

The fault detection rate is given as a function of the number faults detected, as in the following equation,

$$b(n+1) = b_i + (b_f - b_i)\frac{m(n+1)}{a} \qquad (2)$$

According to the values of b_i and b_f, we can distinguish between the following cases [1]:

- Constant fault detection rate; $b_i=b_f=b$
- Increasing fault detection rate; $b_f>b_i$
- Decreasing fault detection rate; $b_i>b_f$
- Vanishing fault detection rate; $b_f=0$, $b_i>0$

By substituting equation 2 in the difference equation 1 and then solving it using Probability Generating Function (PGF) with initial condition $m(n=0)=0$, one can get the closed form solution as given below.

$$m(n) = \frac{a\left(1-(1-b_f)^n\right)}{1+\frac{b_f-b_i}{b_i}(1-b_f)^n} \qquad (3)$$

The structure of the model is flexible. The shape of the growth curve is determined by the parameters b_i and b_f and can be both exponential and S-shaped for the four cases discussed above. In case of constant fault detection rate, equation 3 can be written as

$$m(n) = a\left(1-(1-b)^n\right) \qquad (4)$$

If $b_f<b_i$, it is apparent from equation 1 that $b(n+1)$ decreases to zero more rapidly than linearly. The smaller ratio b_f/b_i, the larger rate convergence of equation 1. If $b_f=0$, then equation 1 becomes:

$$m(n+1)-m(n) = \frac{b_i}{a}\left(a-m(n+1)\right)\left(a-m(n)\right) \qquad (5)$$

The solution of which is given by:

$$m(n) = \frac{ab_i n}{1+b_i n} \qquad (6)$$

If $b_f>b_i$, then equation 3 has an inflection point and the growth curve is S-shaped. Such behaviour of the SRGM can be attributed to the increased skill of the test team or a modification of the testing strategy.

The proposed discrete time model defined in equation 3 is very interesting from various points of view. Besides the above-mentioned interpretation as a flexible S-shaped fault detection model, this model has the exponential model [17] as special cases as given in equation 4.

Note that this proposed basic discrete time model is able to model both cases of strictly decreasing failure intensity and the case of increasing-and-decreasing failure intensity. None of the exponential model [17] and the ordinary delayed S-shaped model [6] can do both.

2.2. Extended Discrete Time Model

2.2.1. Model Development

During debugging process faults are identified and removed upon failures. In reality this may not be always true. The corrections may themselves introduce new faults or they may inadvertently create conditions, not previously experienced, that enable other faults to cause failures. This results in situations where the actual fault removals are less than the removal attempts. Therefore, the fault detection rate is reduced by the probability of imperfect debugging [6, 8, 13, 14]. Besides there is a good chance that some new faults might be introduced during the course of debugging.

The learning-process of the software developers has also been studied [1, 6, 12, 16]. Leaning occurs if testing appears to improve dynamically in efficiency as one progress through the testing phase. Learning usually manifests itself as a changing fault detection rate. By assuming the fault detection rate is dependent on the number of test cases, the role of the learning process during the testing phase can be established. The extended model proposed below incorporates these three factors.

To avoid mathematical complexity many simplifying assumptions have been taken while developing the basic model. One of which is that the number of initial fault-content is constant. Modifications to the basic model have been proposed for increasing over-all fault content, where a of equation 1 is substituted by $a(n)$ as given in equation 8. The nature of $a(n)$ depends upon various factors like the skill of test team or testing strategy, number of test cases, software size, and complexity. Hence no single functional form can describe the growth in number of faults during testing phase and debugging process. This necessitates a modelling approach that can be modified without unnecessarily increasing the complexity of the resultant model [8].

Here, we show how this can be achieved for the proposed basic discrete time model by changing the fault detection rate. Before the spread of flexible models, a number of models were proposed that linked the fault detection rate to the initial fault content. But basic model due to it is inherent flexibility could describe many of them.

Hence it is imperative to capture this flexibility and in this paper we do so by proposing a logistic number of test cases dependent for fault detection rate as proposed in equation 9.

2.2.2. Model Assumptions

In addition to basic discrete time model assumptions except for assumption 3, we have:

- Over-all fault content is linearly dependent on the number of test cases.

- Fault detection rate is number of test cases dependent function with an inflection S-shaped model.
- Fault introduction rate is a linear function of the number of test cases.
- Faults can be introduced during the testing phase and the debugging process may not lead to the complete removal of the faults, i.e., the debugging process is imperfect.

2.2.3. Model Notations

In addition to the basic discrete time model notations we include the following:

$a(n)$	Over-all fault-content dependent on the number of test cases, which includes initial fault-content and the number of faults introduced.
$b(n+1)$	Fault detection rate dependent on the number of test cases.
α	Fault introduction rate per detected faults per test case.
p	The probability of fault removal on a failure, i.e., the probability of perfect debugging Fault detection rate function is dependent on the number of test cases and behaves linearly on the number of faults remaining.

2.2.4. Model Formulation

Under the above extended model assumptions, the expected cumulative number of faults detected between the nth and (n+1)th test cases is proportional to the number of faults remaining after the execution nth test run, satisfies the following difference equation:

$$m(n+1) - m(n) = b(n+1)\big(a(n) - m(n)\big) \quad (7)$$

Both $a(n)$ and $b(n+1)$ are number of test cases dependent functions. An increasing $a(n)$ implies an increasing total number of faults, and thus reflects fault generation. Whereas, $b(n+1)$ is an S-shaped curve that can capture the learning process of the software testers, and this function is affected by the probability of fault removal on a failure and they are giving as:

$$a(n) = a(1+\alpha n) \quad (8)$$

and

$$b(n+1) = \frac{b_f p}{1 + \frac{b_f - b_i}{b_i}(1-b_f p)^{n+1}} \quad (9)$$

By substituting equations 9 and 8 in 7 and then solving it using PGF with initial condition $m(n=0)=0$, after tedious algebraic manipulations, one can get the closed form solution as given below.

$$m(n) = \left(\frac{a}{1+\frac{b_f - b_i}{b_i}(1-b_f p)^n} \times \left[\left(1-(1-b_f p)^n\right)\left(\frac{b_f p - \alpha}{b_f p}\right) + \alpha n\right]\right) \quad (10)$$

Fault generation and imperfect debugging with leaning are integrated to form the extended discrete time model as given in equation 10.

According to the values of parameters, we can distinguish between the following cases:

- Constant fault detection rate (b_i=b_f=b), no faults introduced (α=0), and prefect debugging (p=1)
- No faults introduced (α=0), and prefect debugging (p=1).

In case 1, equation 10 can reduce to equation 4. Whereas in case 2, it reduces to equation 3.

3. Parameter Estimation

MLE method is used to estimate the unknown parameters of the developed framework. Since all data sets used are given in the form of pairs $(n_i,x_i)(i=1,2,\ldots,f)$, where x_i is the cumulative number of faults detected by n_i test cases $(0<n_1<n_2<\ldots<n_f)$ and n_i is the accumulated number of test run executed to detect x_i faults.

The likelihood function L for the unknown parameters with the superposed mean value function is given as

$$L\big(parmaters \mid (n_i,x_i)\big) = \prod_{i=1}^{f} \frac{\left[m(n_i)-m(n_{i-1})\right]^{x_i-x_{i-1}}}{(x_i-x_{i-1})!} \times \exp\big(-\left(m(n_i)-m(n_{i-1})\right)\big) \quad (11)$$

Taking natural logarithm of equation 11 we get

$$\ln\ L = \sum_{i=1}^{f}(x_i - x_{i-1})\ln\left[m(n_i)-m(n_{i-1})\right] - \left\{m(n_i)-m(n_{i-1})\right\} - \sum_{i=1}^{f}\ln\left[(x_i-x_{i-1})!\right] \quad (12)$$

The MLE of the SRGM parameters can be obtained to by maximizing L in equation 11 or 12 with respect to the following constraints: ($a,b_i,b_f>0$, $0<p\le 1$, $\alpha\ge 0$).

4. Model Validation and Comparison Criteria

4.1. Model Validation

To check the validity of the proposed discrete time models to describe the software reliability growth, they have been tested on four Data Sets (DS) cited from real software development projects.

The first DS-I was collected from test of a network-management system at AT&T Bell Laboratories, after it was tested for 20 weeks in which 100 faults were detected [13]. The second DS-II was collected during 21 days of testing, 46 faults were detected [12]. The third DS-III was for a radar system of size 124 KLOC after it was tested for 35 months in which 1301 faults were detected [2]. The fourth DS-IV had been collected during 25 days of testing in which 136 faults were detected [13].

4.2. Comparison Criteria

The performance of an SRGM judged by its ability to fit the past software reliability data (goodness-of-fit) and to predict satisfactorily the future behavior from present and past data (predictive validity) [6, 11].

4.2.1. Goodness of Fit

- The Sum of Squared Error (SSE). The difference between the simulated data $m^\wedge(n_i)$ and the observed (reported data) x_i is measured by the SSE as,

$$SSE = \sum_{i=1}^{f} \left(\hat{m}(n_i) - x_i\right)^2 \tag{13}$$

where f is the number of observations. The lower value of SSE indicates less fitting error, thus better goodness-of-fit.

- The Akaike Information Criterion (AIC). This criterion was first proposed as SRGM model selection tool by [9]. It is defined as:

AIC= -2(value of max. log likelihood function) + 2(number of parameters used in the model) (14)

Lower value of AIC indicates more confidence in the model thus a better fit and predictive validity.

- Coefficient of multiple determination (R^2). This measure can be used to investigate whether a significant trend exists in the observed failure intensity. This coefficient is defined as the ratio of the Sum of Squares (SS) resulting from the trend model to that from a constant model subtracted from 1, that is:

$$R^2 = 1 - \frac{residual\ \ SS}{corrected\ \ SS} \tag{15}$$

R^2 measures the percentage of the total variation about the mean accounted for by the fitted curve. It ranges in value from 0 to 1. Small values indicate that the model does not fit the data well [11].

4.2.2. Predictive Validity

Predictive validity is defined as the ability of the model to determine the future failure behaviour from present and past failure behaviour [11]. The relative prediction fault (RPF) is defined as,

$$RPF = \frac{\hat{m}(n_f) - x_f}{x_f} \tag{16}$$

where x_f is the cumulative number of faults removed after the execution of the last test run n_f and $m^\wedge(n_f)$, is the estimated value of the SRGM $m(n_f)$, which determined using the actually observed data up to an arbitrary test case $n_e(\leq n_f)$.

If the RPF value is negative/positive the model is said to underestimate/ overestimate the fault removal process. A value close to zero indicates more accurate prediction, thus more confidence in the model. The value is said to be acceptable if it is within (±10%) [6].

5. Data Analysis and Model Comparison

5.1. For Basic Discrete Time Model

5.1.1. Goodness of Fit Analysis

Using MLE method, the estimated values of the proposed basic discrete time model parameters for DS-I and DS-II are given in Table 1. According to the estimated values of the initial and final fault detection rates (b_i and b_f), the skill of the test-team does improve with time in both DS-I and DS-II. That is why the fault detection/removal process resembles an S-shaped growth curve in both DS-I and DS-II.

Table 1. Parameters estimations for DS-I and DS-II.

Model	Data Set	Parameter Estimation		
		a	b_i	b_f
Proposed Basic	DS-I	111	.0717	.1581
	DS-II	59	.0167	.1550

The fitting of the proposed model to both DS-I and DS-II are graphically illustrated in Figures 1 and 2 respectively. It is clearly seen from both the figures that the proposed model fits both DS-I and DS-II excellently. This highlights it is flexibility.

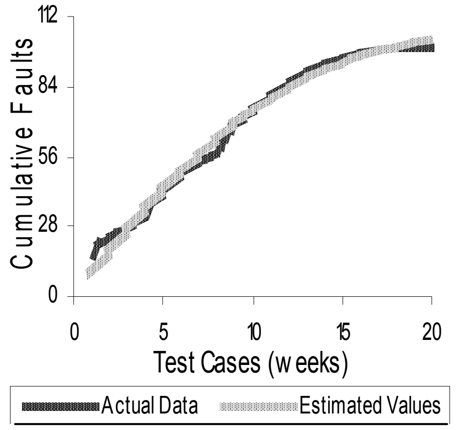

Figure 1. Goodness of fit (DS-I).

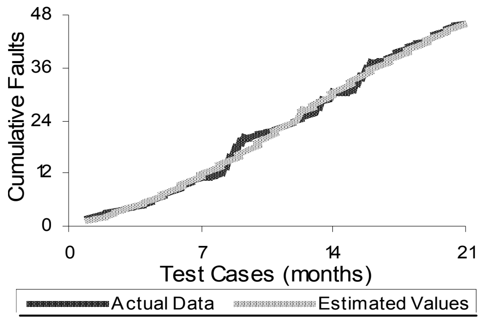

Figure 2. Goodness of fit (DS-II).

Comparison of the proposed model and well-documented discrete time NHPP based SRGMs in terms of goodness of fit is given in Tables 2 and 3 for DS-I and DS-II respectively. Note that during the estimation process of models under comparison it is observed that the exponential model [17] fails to give any plausible result as it over estimates the fault-content and no estimates were obtained for DS-II. It is clearly seen from both the Tables 2 and 3 that the proposed model is the best among the models under comparison in terms of SSE, AIC, and R^2 metrics values, which is very encouraging.

Table 2. Goodness of fit for DS-I.

Models under Comparison	**Parameter Estimation**		**Comparison Criteria**		
	a	*b*	**SSE**	**AIC**	**R^2**
Exponential [17]	130	.0798	232	92	.9857
Delayed S-shaped [6]	106	.2165	357	99	.9781
Proposed Basic	See Table 1.		180	90	.9890

Table 3. Goodness of fit for DS-II.

Models under Comparison	**Parameter Estimation**		**Comparison Criteria**		
	a	*b*	**SSE**	**AIC**	**R^2**
Exponential [17]	*	*	*	*	*
Delayed S-shaped [6]	84	.0831	28	79	.9938
Proposed Basic	See Table 1.		25	77	.9944

Hence, the proposed basic discrete time model fits better than existing models on both DS-I and DS-II.

5.1.2. Predictive Validity Analysis

DS-I and DS-II are truncated into different proportions and used to estimate the parameters of the proposed basic discrete time model. For each truncation, one value of RPE ratio is obtained.

Figures 3 and 4 graphically illustrate the results of the predictive validity. It is observed that the predictive validity of the proposed model varies from one truncation to another. The RPE ratio of the proposed model overestimates the fault removal process in DS-I and DS-II except when the testing progress ratio is about 55% and 50% it underestimates the process in DS-II.

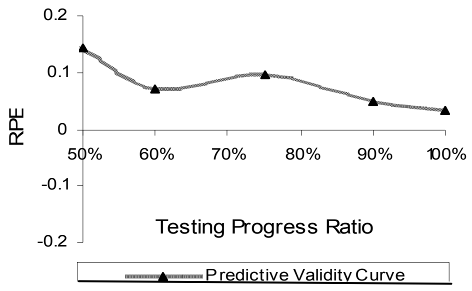

Figure 3. Predictive validity (DS-I).

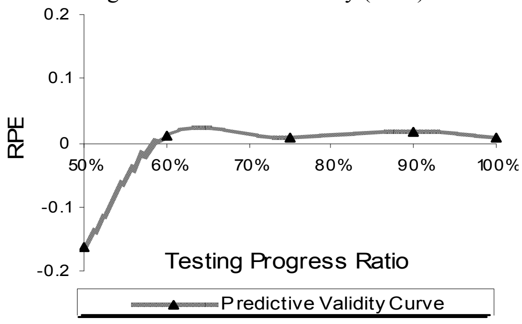

Figure 4. Predictive validity (DS-II).

It is clearly seen from both Figures 3 and 4 that 55% of the total test time is sufficient to predict the future behaviour of the fault removal process reasonably for DS-I and DS-II.

5.2. For Extended Discrete Time Model

5.2.1. Goodness of Fit Analysis

Using MLE method, the estimated values of the proposed extended discrete time model parameters for DS-III and DS-IV are given in Table 4. According to the estimated values of the initial and final fault detection rates (b_i and b_f), the skill of the test-team does improve with time in DS-III and in DS-IV does not. That is why the fault detection process resembles an S-shaped growth curve in DS-III and an exponential curve in DS-IV. According to the estimated values of the fault introduction rate parameter (α) the fault detection process (i.e., the debugging process) in DS-III is perfect and no fault introduced during debugging, whereas in DS-IV was not.

Table 4. Parameters estimations for DS-III & -IV.

Model	**Data Set**	**Parameter Estimation**				
		a	b_i	b_f	*p*	α
Proposed Extended	DS-III	1352	.0087	.1832	.9922	0
	DS-IV	156	.1525	.0005	.9965	.0036

The fitting of the proposed model to both DS-III and DS-IV are graphically illustrated in Figures 5 and 6. It may be noticed that the relationship between the cumulative number of faults and the number of test cases vary from purely exponential to highly S-shaped. It is clearly seen from both the figures that the proposed model fits both the DS-III and DS-IV excellently. This highlights it is flexibility.

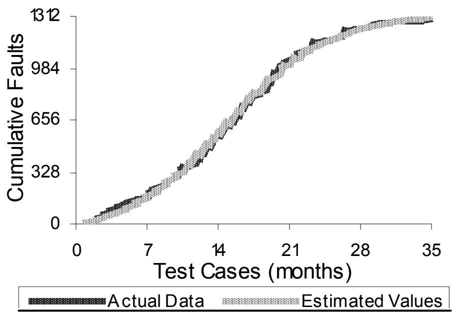

Figure 5. Goodness of fit for DS-III.

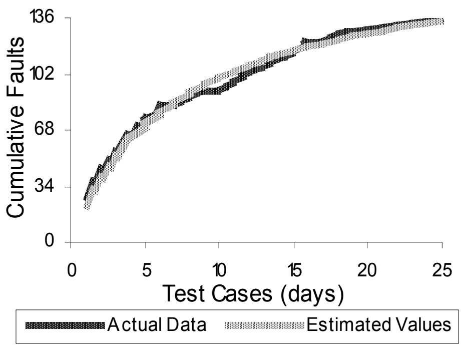

Figure 6. Goodness of fit (DS-IV).

Comparison of the proposed model and well-documented discrete time SRGM based on NHPP in terms of goodness of fit is given in Tables 5 and 6 for DS-III and DS-IV respectively. Note that during the estimation process of models under comparison it is observed that the exponential model [17] fails to give any plausible result as it over estimates the fault-content (a) and no estimates were obtained for DS-III. It is clearly seen from both the Tables 5 and 6 that the proposed model is the best among the models under comparison in terms of SSE, AIC, and R^2 metrics values, which is very encouraging. Hence, the proposed extended discrete time model fits better than existing models on both DS-III and DS-IV.

Table 5. Goodness of fit for DS-III.

Models under Comparison	**Parameter Estimation**		**Comparison Criteria**		
	a	***b***	**SSE**	**AIC**	**R^2**
Exponential [17]	*	*	*	*	*
Delayed S-shaped [6]	1735	.0814	107324	518	.9856
Proposed Extended	See Table 5.		7133	342	.9990

Table 6. Goodness of fit for DS-III.

Models under Comparison	**Parameter Estimation**		**Comparison Criteria**		
	a	***b***	**SSE**	**AIC**	**R^2**
Exponential [17]	136	.1291	766	119	.9664
Delayed S-shaped [6]	126	.2763	2426	176	.8936
Proposed Extended	See Table 5.		306	116	.9866

5.2.2. Predictive Validity Analysis

DS-III and DS-IV are truncated into different proportions and used to estimate the parameters of the proposed extended discrete time model. For each truncation, one value of RPE ratio is obtained.

Figures 7 and 8 graphically illustrate the results of the predictive validity. It is observed that the predictive validity of the proposed model varies from one truncation to another. The RPE ratio of the proposed model overestimates the fault removal process in DS-III and DS-IV except when the testing progress ratio is about 65% and 60% it underestimates the process in DS-IV.

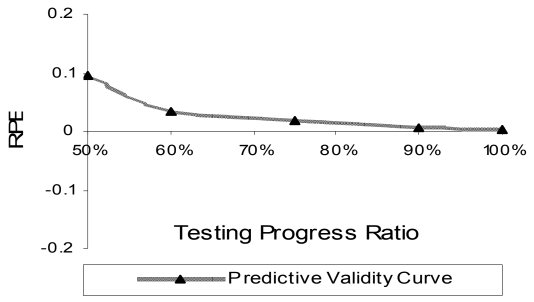

Figure 7. Predictive validity (DS-III).

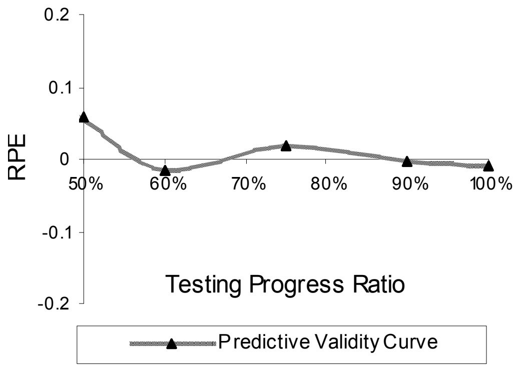

Figure 8. Predictive validity (DS-IV).

It is clearly seen from both Figures 7 and 8 that 50% of the total test time is sufficient to predict the future behaviour of the fault removal process reasonably for DS-III and DS-IV.

6. Conclusion

In this paper, newly developed discrete time SRGM based on NHPP to describe a variety of reliability growth and the increased skill (efficiency) of the testing team or a modification of the testing strategy during testing phase, are proposed.

The proposed discrete time models have been validated and evaluated on actual software reliability data cited from real software development projects and compared with existing discrete time NHPP based models. The results are encouraging in terms of goodness of fit and predictive validity due to their applicability and flexibility. Hence, we conclude that the two proposed discrete time models not only fit the past well but also predict the future reasonably well.

Acknowledgements

I take this opportunity to thank Prof. P. K. Kapur, Dr. A. K Bardhan, and Dr. P. C. Jha of Delhi University for their support every moment I sought. The suggestion, comments, and criticisms of these people have greatly improved this manuscript.

References

[1] Bittanti S., Bolzern P., Pedrotti E., Pozzi M., and Scattolini A., *Software Reliability Modeling and Identification*, Springer-Verlag, USA, 1988.

[2] Brooks D. and Motley W., "Analysis of Discrete Software Reliability Models," *Technical Report*, New York, 1980.

[3] Goel L., "Software Reliability Models: Assumptions, Limitations and Applicability," *IEEE Transactions on Software Engineering*, vol. 11, no. 12, pp. 1411-1423, 1985.

[4] Goel L. and Okumoto K., "Time Dependent Error Detection Rate Model for Software Reliability and Other Performance Measures," *IEEE Transactions on Reliability*, vol. 28, no. 3, pp. 206-211, 1979.

[5] Jelinski Z. and Moranda B., *Statistical Computer Performance Evaluation*, Academic Press, New York, 1972.

[6] Kapur K., Garg B., and Kumar S., *Contributions to Hardware and Software Reliability*, World Scientific, New York, 1999.

[7] Kapur K., Shatnawi O., and Yadavalli S., "A Software Fault Classification Model," *South African Computer Journal*, vol. 33, no. 33, pp. 1-9, 2004.

[8] Kapur K., Singh O., Shatnawi O., and Gupta A., "A Discrete Nonhomogeneous Poisson Process Model for Software Reliability Growth with Imperfect Debugging and Fault Generation," *International Journal of Performability Engineering*, vol. 2, no. 4, pp. 351-368, 2006.

[9] Khoshogoftaar T. and Woodcock G., "Software Reliability Model Selection: A Case Study," *in Proceedings of the International Symposium on Software Reliability Engineering*, pp. 183-191, USA, 1991.

[10] Kuo S., Huang H., and Lyu R., "Framework for Modelling Software Reliability Using Various Testing-Effort and Fault-Detection Rates," *IEEE Transactions on Reliability*, vol. 50, no. 3, pp. 310-320, 2001.

[11] Musa D., Iannino A., and Okumoto K., *Software Reliability*, McGraw-Hill, New York, 1987.

[12] Ohba M., "Software Reliability Analysis Models", *IBM Journal of Research and Development*, vol. 28, no. 1, pp. 428-443, 1984.

[13] Pham H., *Software Reliability*, Springer-Verlag, USA, 2000.

[14] Pham H., Nordmann L., and Zhang X., "A General Imperfect Software-Debugging Model with S-shaped Fault Detection Rate," *IEEE Transactions on Reliability*, vol. 48, no. 3, pp. 169-175, 1999.

[15] Shatnawi O., "Modelling Software Fault Dependency Using Lag Function," *Al Manarah Journal for Research and Studies*, vol. 15, no. 6, pp. 261-300, 2007.

[16] Yamada S., Ohba M., and Osaki S., "S-shaped Software Reliability Growth Models and Their Applications," *IEEE Transactions on Reliability*, vol. 33, no. 1, pp. 169-175, 1984.

[17] Yamada S. and Osaki S., "Discrete Software Reliability Growth Models," *Applied Stochastic Models and Data Analysis*, vol. 1, no.1, pp. 65-77, 1985.

[18] Xie M., *Software Reliability Modelling*, World Scientific, New York, 1991.

Omar Shatnawi received his PhD, in computer science and his MSc in operational research from University of delhi in 2004 and 1999, respectively. Currently, he is head of Department of Information Systems at al-Bayt University. His research interests are in software engineering, with an emphasis on improving software reliability and dependability.