Modelling Software Fault debugging Complexity under Imperfect Debugging Environment

Omar Shatnawi
Al al-Bayt University
Jordan
dromali@lycos.com

Abstract: *The fault debugging progress is influenced by various factors all of which may not be deterministic in nature such as the debugging effort, debugging efficiency and debuggers skill, and debugging methods and strategies. In order to address these realistic factors that influencing the debugging process we propose an integrated no homogeneous Poisson process (NHPP) based software reliability model. The integrated modelling approach incorporates the effect of imperfect fault debugging environment, fault debugging complexity and learning debuggers' phenomenon. The debugging phase is assumed to be composed of three processes namely, fault detection, fault isolation and fault removal. The software faults are categorized into three types namely, simple, hard and complex according to their debugging complexity.As the debugging progresses, the fault removal rate changes to capture learning process of the debuggers. In order to relax the ideal debugging environment, two types of imperfect debugging phenomena are incorporated. Incorporating the imperfect fault debugging phenomena in software reliability modelling is very important to the reliability measurement as it is related to the efficiency of the debugging team. Accordingly, the total debugging process is the superposition of the three debugging activities processes. Such modeling approach can capture the variability in the software reliability growth curve due to debugging complexity of the faults depending on the debugging environment which enables the management to plan and control their debugging activities to tackle each type of fault. Actual test datasets cited from real software development projectshave been used to demonstrate the proposed model.*

Keywords: *Software reliability engineering, software testing and debugging, non-homogenous Poisson process, imperfect debugging, fault debugging complexity.*

1. Introduction

Computers are being widely used for a variety of applications in our daily life. With the rapid advancement in the technology, the cost of computer hardware has been steadily declining while on the contrary the cost of computer software is increasing. The production of computer software is seen to be the most prominent industry today. Therefore, it is of utmost importance to develop high quality software systems. Software reliability is one of the most important characteristics of software product quality. Its measurement and management technologies during the software product life-cycle are essential to produce and maintain reliable software systems [20].
Observing the fault debugging phenomenon, software quality in terms of its reliability can be measured. Software reliability models based on the NHPP have been quite successful tools in practical software reliability engineering. These models consider the debugging process as a counting process characterized by its main value function. Software reliability can be estimated once the main value function is determined. Model parameters are usually estimated using either maximum likelihood estimate (MLE) or least-square estimate methods [5,7,13,20,21].
Several software reliability models have been developed in the literature to monitor and control the debugging process of the software systems [2,6,10,11,12,21]. During debugging phase it has been observed that the relationship between the debugging time and the corresponding number of faults removed is either exponential or S-shaped.
This paper is organized as follows. Section 2 presents the model development and formulation for the proposed modelling approach. Section 3 provides the technique used for data analyses. Section 4 provides goodness of fit criteria used for validation and evaluation purpose. The goodness of fit of the proposed model is compared with the exponential model [2], delayed S-shaped model [21], Inflection S-shaped mode [11], Erlang model [5], and software fault classification model [8] in section 5. We conclude this paper in Section 6.

2. Software Reliability Modelling

Most of NHPP based software reliability models were proposed under the assumption that similar effort and strategy is required for removing each of the faults. Such assumption helps to simplify the problem of modeling and provides to a certain extent plausible results. However, this assumption is not truly representative of reality. Different faults may require a different amount of efforts and strategy for their removal from the system. To incorporate this phenomenon, faults are categorized into different types. Yamada *et al*. [22] proposed a modified exponential model assuming that there are two types of faults in the software. Later, Kapur*et.al* [5] proposed the Erlang model by categorizing the faults encountered into three types namely: simple, hard and complex. It is assumed that the time delay between the failure observation and its subsequent removal represent the complexity of faults. It has been assumed in these models that the fault removal rate remains constant over the entire debugging period. Due to the complexity of the software system, the debugging team may not be able to remove the faults at the same rate. As the debugging progresses, the fault removal rate changes. Learning usually manifests itself as a changing fault removal rate. To capture the learning-process, Kapur*et al.*[8] proposed a software fault classification model that integrated the effect of learning-process phenomenon of the debugging team in the Erlang model. Recently Shatnawi and Kapur[10] further extended the software fault classification to count for finite number of faults. However, in these models debugging process is assumed to be ideal. The assumption may not hold true in many situations.Due to the complexity of the software system and the incomplete understanding of the software requirements, specifications and structure, the debugging team may not be able to remove the fault perfectly and the original fault may remain. This phenomenon is known as imperfect fault debugging[6,7,14,15].

In order to relax the ideal debugging assumption, we integrate the effect of imperfect fault debugging on reliability growth of software based on the assumption of the classification model [8].Such type of integrated modeling approach is very much suited for object-oriented and distributed systems development environments [7,16,17].

The following are the basic assumptions in developing and formulation the proposed modelling approach:

1. Debugging process follows an NHPP with mean value function $m\ (t)$.
2. Software is subject to failures during execution caused by the remaining faults.
3. The faults existing in the software are of three types: simple, hard and complex. They are distinguished by the amount of effort needed to remove them and modelled by 1-stage, 2-stage and 3-stage removal processes respectively.
4. Each time a fault detected, an immediate (delayed) effort takes place to decide the cause of the failure in order to remove it. The time delay between the fault detection and its subsequent fault removal is assumed to represent the debugging complexity of the faults.
5. The debugging process is imperfect.
6. Fault removal rate of the simple fault is proportionality constant, whereas for hard and complex is a logistic function as it is expected the learning-process will grow with time.
7. The expected number of faults removed in$(t, t+\Delta t)$ is proportional to the number of faults remaining to be removed.

The following notations are used for the mathematical development and formulation purpose:

a_i	Fault-content of type$i\,(\sum_{i=1}^{3} a_i = a)$, where$a$is the total fault-content.
b_i	Proportionality constant represents the fault detection/isolation rate per fault of type *i*.
$b_i(t)$	Logistic learning function represents the fault removal rate per fault of type *i*.
$m_{id}(t)$	Mean number of fault detected of type *i*by *t*.
$m_{ii}(t)$	Mean number of fault isolated of type *i*by *t*.
$m_{ir}(t)$	Mean number of fault removed of type *i*by *t*.
β_i	Inflection factor of debug personnelin the fault removal rate per fault of type *i*.
p_i	Probability of perfect debugging, i.e., debugging efficiencyof type *i*.

2.1 Modelling Approach Development

In this section we revisited software classification model [8] that can be applied for reliability estimation for a software project expected to contain three different types of faults. The model categorise faults of different complexity depending on their debugging activities.Recall that the time delay between the fault detection and subsequent fault removal represents the complexity of the faults. Therefore, the model assuming that the software contains three types: simple, hard and complex.

A simple fault debugging is modelled as a 1-stage process as follows:

$$\frac{\partial}{\partial t} m_{1r}(t) = b_1(a_1 - m_{1r}(t)) \qquad (1)$$

The one-stage process as modelled above describes the fault detection, fault isolation and fault removal processes. Solving the differential equation (1) under the boundary condition $m_{1r}(t=0)=0$, we get

$$m_{1r}(t) = a_1(1 - e^{-b_1 t}) \qquad (2)$$

The harder types of faults are assumed to take more effort. In other words it also means that the debugging team personnel have to spend more time to analyze the detected faults and consequently need more effort to remove them. Debugging process for such faults is modelled as 2-stage process as follows:

$$\frac{\partial}{\partial t} m_{2d}(t) = b_2(a_2 - m_{2d}(t))$$

$$\frac{\partial}{\partial t} m_{2r}(t) = b_2(t)(m_{2d}(t) - m_{2r}(t))$$

Where $b_2(t) = \frac{b_2}{1+\beta_2 e^{-b_2 t}}$(3)

The first stage of the two-stage process as modelled above describes the fault detection and the fault isolation process. The second stage describes the fault removal process. Solving, the system of differential equations given in (3) under the boundary conditions$m_{2d}(t = 0) = 0$ and $m2rt=0=0$respectively, we get

$$m_{2r}(t) = \frac{a_2(1-(1+b_2 t)e^{-b_2 t})}{1+\beta_2 e^{-b_2 t}} \qquad (4)$$

The complex fault debugging process is modelled as a 3-stage process,

$$\frac{\partial}{\partial t} m_{2d}(t) = b_3(a_3 - m_{2d}(t))$$

$$\frac{\partial}{\partial t} m_{3i}(t) = b_3(m_{2d}(t) - m_{3i}(t))$$

$$\frac{\partial}{\partial t} m_{3r}(t) = b_3(t)(m_{3i}(t) - m_{3r}(t))$$

Where $b_3(t) = \frac{b_3}{1+\beta_3 e^{-b_3 t}}$(5)

The first stage of the three-stage process as modelled above describes the fault detection. The second stage describes the fault isolation process. The third stage describes the fault removal process. Solving the above system of differential equations given in (5) under the boundary conditions $m_{3d}(t = 0) = 0$, $m_{3i}(t = 0) = 0$ and $m_{3r}(t = 0) = 0$respectively, we get

$$m_{3r}(t) = \frac{a_3\left(1-\left(1+b_3 t+b_3^2\frac{t^2}{2}\right)e^{-b_3 t}\right)}{1+\beta_3 e^{-b_3 t}} \qquad (6)$$

The software fault classification model is the superposition of the three NHPP with mean value functions givens in equations (2), (4) and (6) as

$$m(t) = \sum_{i=1}^{3} m_{ir}(t)$$
$$= a_1(1 - e^{-b_1 t}) + \frac{a_2(1 - (1 + b_2 t)e^{-b_2 t})}{1 + \beta_2 e^{-b_2 t}} + \frac{a_3\left(1 - \left(1 + b_3 t + b_3^2 \frac{t^2}{2}\right) e^{-b_3 t}\right)}{1 + \beta_3 e^{-b_3 t}}$$

(7)

To model the debugging phenomenon of each type of fault directly in single stage considering the delay of the removal process. The fault removal rate per fault '$d_i(t)$' for $i(i = 1,2,3)$ fault-type, can be obtained as

$$d_1(t) = \frac{\frac{\partial}{\partial t} m_{1t}}{a_1 - m_{1t}} = b_1$$

$$d_2(t) = \frac{\frac{\partial}{\partial t} m_{2t}}{a_2 - m_{2(n)}} = \frac{b_2\left((1 + \beta_2 + b_2 t) - (1 + \beta_2 e^{-b_2 t})\right)}{(1 + \beta_2 + b_2 t)(1 + \beta_2 e^{-b_2 t})}$$

$$d_3(t) = \frac{\frac{\partial}{\partial t} m_{3t}}{a_3 - m_{3(n)}} = \frac{b_3\left(\left(1 + \beta_3 + b_3 t + b_3^2 \frac{t^2}{2}\right) - (1 + b_3 t)(1 + \beta_3 e^{-b_3 t})\right)}{\left(1 + \beta_3 + b_3 t + b_3^2 \frac{t^2}{2}\right)(1 + \beta_3 e^{-b_3 t})}$$

(8)

Note that $d_2(t)$ and $d_3(t)$ increase monotonically with time t and tend to constants b_2 and b_3 respectively as $t \to \infty$. Thus, in the steady state, hard and complex faults growth curves behave similar to the simple fault growth curve and hence there is no loss of generality in assuming the steady state rates b_2 and b_3 to be equal tob_1. After substituting $b_3 = b_2 = b_1$in the right hand side of equation (8), one can see that $d_1(t) > d_2(t) > d_3(t)$, which is in accordance with the complexity of the faults [4,5,7,16].

Hence there is no loss of generality in assuming the steady state $b_3 = b_2 = b_1 = b$ and $\beta_3 = \beta_3 = \beta_1 = \beta$ (say). Then we may write equation (7) as follows

$$m(t) = a_1(1 - e^{-bt}) + \frac{a_2(1-(1+bt)e^{-bt})}{1+\beta e^{-bt}} + \frac{a_3\left(1-\left(1+bt+b^2\frac{t^2}{2}\right)e^{-bt}\right)}{1+\beta e^{-bt}} \qquad (9)$$

2.2 Modelling Approach Formulation

In order to relax the perfect debugging assumption of the software classification model [8] and given in (9), we introduce the possibility of imperfect fault debugging phenomenon. Incorporating the imperfect fault debugging phenomenon in software reliability modelling can be of immense help to the reliability assessment as it is related to the efficiency of the debugging team. However, the debugging team may not be able to remove the fault perfectly and the original fault may remain leading to a phenomenon known as imperfect fault debugging. Accordingly the debugging phenomenon can be described for the three types of faults $i(i = 1,2,3)$with respect to time in a single stage as follows:

$$\frac{\partial}{\partial t} m_i(t) = p_i d_i(t)(a_i - m_i(t))(10)$$

Solving the above system of difference equations (10), with respect to equation (8) using the probability generating function with the boundary conditions $m_1(t = 0) = m_2(t = 0) = m_3(t = 0) = 0$, respectively, one can get

$$m_1(t) = a_1(1 - e^{-p_1 b_1 t})$$

$$m_2(t) = a_2\left(1 - \left(\frac{(1 + \beta_2 + b_2 t)e^{-b_2 t}}{1 + \beta_2 e^{-b_2 t}}\right)^{p_2}\right)$$

$$m_3(t) = a_3\left(1 - \left(\frac{\left(1 + \beta_3 + b_3 t + b_3^2 \frac{t^2}{2}\right) e^{-b_3 t}}{1 + \beta_3 e^{-b_3 t}}\right)^{p_3}\right)$$

(11)

The proposed model is the superposition of all the NHPP with mean value functions given in (11) and is given as

$$m(t) = \sum_{i=1}^{3} m_{i(n)} = a_1\left(1 - e^{-p_1 b_1 t}\right) + a_2\left(1 - \left(\frac{(1+\beta_2+b_2 t)e^{-b_2 t}}{1+\beta_2 e^{-b_2 t}}\right)^{p_2}\right) + a_3\left(1 - \left(\frac{\left(1+\beta_3+b_3 t+b_3^2\frac{t^2}{2}\right)e^{-b_3 t}}{1+\beta_3 e^{-b_3 t}}\right)^{p_3}\right) \quad (12)$$

The analysis of the fault removal per remaining followed earlier with respect to equation (8) can be applied here. Therefore, there is no loss of generality in assuming the steady state$b_3 = b_2 = b_1 = b$,$p_3 = p_2 = p_1 = p$, and $\beta_3 = \beta_2 = \beta$ (say). Then we may write equation (12) as follows

$$m(t) = a_1\left(1 - e^{-pbt}\right) + a_2\left(1 - \left(\frac{(1+\beta+bt)e^{-bt}}{1+\beta e^{-bt}}\right)^{p}\right) + a_3\left(1 - \left(\frac{\left(1+\beta+bt+b^2\frac{t^2}{2}\right)e^{-bt}}{1+\beta e^{-bt}}\right)^{p}\right) \quad (13)$$

The above proposed model integrates the effect of three types of fault debugging complexity and incorporates the learning phenomenon of the debugger under imperfect fault debugging environment.It should be pointed out here that the proposed model given in equation (13) is more general than that of the classification model [8] and given in equation (9) since it includes the effect of two types of imperfect fault debugging, and has the models due to [7, 14, 21] as special cases. Such modeling approach provides an integrated common platform for not only software reliability growth models with perfect fault debugging but also for imperfect fault debugging.

3. Parameter Estimation Technique

Parameters estimation is of primary concern in software reliability measurement. The maximum likelihood estimation (MLE) method is employed to estimate the unknown parameters of the models under comparison. Since all data sets used are given in the form of pairs$(n_i - x_i)(i = 1,2,\dots,f)$, where $x_i$the cumulative number of faults is detected by n_i test cases$(0 < n_1 < n < \cdots < n)$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 $m(t)$is given as

$$L\big(parameters\big|(n, x_i)\big) = \prod_{i=1}^{k} \frac{\left(m(t_i) - m(t_i - 1)\right)^{x_i - x_{i-1}}}{(x_i - x_{i-1})!} e^{-(m(t_i) - m(t_i - 1))} \quad (16)$$

Taking natural logarithm of (16) we get

$$\ln L = \sum_{i=1}^{k} (x_i - x_{i-1}) \ln\left(m(t_i) - m(t_i - 1)\right) - \left(m(t_i) - m(t_i - 1)\right) \sum_{i=1}^{k} \ln(x_i - x_{i-1}) \quad (17)$$

The MLE of the unknown parameters can be obtained by maximizing the likelihood function subject to the parameters constraints.

For faster and accurate calculations, the statistical package for social sciences (SPSS) based on the nonlinear regression technique has been utilized for the estimation of the parameters of the proposed models and the models under comparison. Non-linear regression is a technique of finding a nonlinear model of the relationship between the dependent variable and a set of independent variables. Unlike traditional linear regression, which is restricted to estimating linear models, non-linear regression can estimate models with arbitrary relationships between independent and dependent variables.

4. Data Analysis Technique

Before applying any software reliability model to a set of fault detection data it is advisable to determine whether the test data does, in fact, exhibit reliability growth. If a set of test data does not exhibit increasing reliability as debugging progresses, there is no point in attempting to estimate and forecast the system's reliability. Since the proposed model is a fault count model, the test may only be applied to data in which the test intervals are of equal length. Therefore, we divided the computer test runs $(0, f]$ into$k$ units of time of equal length. The trend test that is commonly carried out is [3, 15]:

- *Laplace Test.* This test is superior from an optimality point of view and is recommended for use when the NHPP assumption is made. In terms of $n_i(\, i = 1,2,3,\dots,f)$ the number of fault detected during unit of time i, the expression of the Laplace factor is

$$u_f = \frac{\sum_{i=1}^{f}(i-1)n_i - \frac{f-1}{2}\sum_{i=1}^{f} n_i}{\sqrt{\frac{f^2-1}{12}\sum_{i=1}^{f} n_i}} \quad (18)$$

In the context of reliability growth, negative values indicate decreasing failure intensity and thus a reliability increase, positive values suggest increasing failure intensity and thus a reliability decrease, and values oscillating between -2 and +2 indicate stable reliability. Laplace test provides information about reliabilitytrend evolution. However, in order to evaluate the reliability quantitatively, software reliability models should be employed.

5. Model Validation & Comparison Criteria

To check the validity of the proposed software reliability model, and to make a fair comparison with other well-established existing models, we apply three test datasets collected from real software development projects to evaluate the goodness of fit of the models under comparison. The criteria adopted for the purpose are:

- *Coefficient of Multiple Determinations* (S_{quared}). 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,
  $$R_{squared} = 1 - \frac{residual\ \ SS}{corrected\ \ SS} (19)$$
  $R_{squared}$ 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.
- *The mean square fitting error (MSE).* The models under comparison are used to simulate the test data, the difference between the expected values, $\widehat{m}_{n_i}$ and the observed data x_i is measured by MSE as follows:
  $$MSE = \frac{1}{f}\sum_{i=1}^{f}(\widehat{m}_i(t) - x_i)^2 \ (20)$$
  where$f$ is the number of observations.
- *Bias.* The difference between the observation and prediction of number of faults at any instant of time i is known as PE_i (prediction error). The average of PE_s is known as bias.

$$Bias = \frac{1}{f}\sum_{i=1}^{f} PE_i \qquad (21)$$

$$PE_i = Actual(observed)_i - Predicted(estimated)_i$$

- *Variation.* The standard deviation of prediction error is known as variation.
  $$Variation = \sqrt{\frac{1}{f-1}\sum_{i=1}^{f}(PE_i - Bias)^2}(22)$$
- *Root Mean Square Prediction Error (RMSPE).* It is a measure of closeness with which a model predicts the observation.
  $$RMSPE = \sqrt{(Bias^2 + Variation^2)}(23)$$

In other words, we evaluate the performance of the models under comparison using $R_{squared}$, MSE, Bias, Variation, and RMSPE metrics. For $R_{squared}$, the larger the metric value the better while for MSE, Bias, Variation, and RMSPE, the smaller the metric value the better the model fits relative to other models run on the same test dataset.

6. Data Analyses and Model Comparison

This section presents the result of goodness-of-fit comparisons. The performance of the proposed model is compared to those of: the exponential model [2], delayed S-shaped model [21], Inflection S-shaped mode [11], Erlang model [5], and software fault classification model [8].We employ five goodness of fit criteria: $R_{squared}$, MSE, bias, variation, and RMSPEas criteria of the model comparison in this section.

5.1 First Software Development Project

The first software test data had been collected during 20 weeks, spending 10,000 CPU hours of testing one of four major releases of the software products at Tandem Computers Company, Los Anglos (CA), 100 faults were detected during the period [18].
Figure 1.1 traces the Laplace trend test. The values of the trend test are completely negative from beginning. However, in period from 3rd week till 9th week we see some fluctuations but this fluctuation doesn't affect the reliability much and reliability has growing behaviour and after 9th week, the behaviour becomes stable which means that reliability grew monotonically.

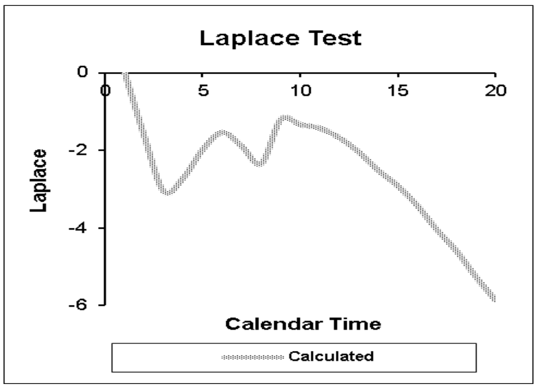

Figure 1.1 Laplace test data trend

The unknown parameters of the models under comparison have been estimated using the regression module of SPSS. The values of estimated parameters have been tabulated below in Table 1.1. The estimation results of the proposed model depict an ideal debugging environment, i.e., that the debugging efficiency parameter 'p' is 100%. Although it seems unrealistic but the probability of perfect debugging fault parameter comes out to be one denying the presence of imperfect fault debugging, which again makes the results of proposed model equivalent to the of model due to [8]. Hence any of these models can be chosen for further analysis and represent the debugging process, the choice can be subjective of the decision maker. The value 33.05 for the shape parameter 'β_i' implies the s-shaped of the actual test data due to the fact that only a few faults are removed at the beginning and faults rapidly become removed. It is estimated that a total of 105 faults are observed in the 20 weeks and all of them were removed successfully in the same debugging period.

Table 1.1. Parameter estimation results

Models under Comparison	Parameter Estimation						
	a	***a₁***	***a₂***	***a₃***	***b***	***p***	***β***
Model due to [2]	134	—	—	—	1.46E-04	—	—
Model due to [21]	102	—	—	—	5.07E-04	—	0
Model due to [11]	134	—	—	—	1.46E-04	—	0
Model due to [5]	122	62	0	60	3.64E-04	—	—
Model due to [8]	105	48	57	0	6.38E-04	—	33.05
Proposed	105	48	57	0	6.38E-04	1	33.05

– indicates the parameter is not part of the corresponding model

Table 1.2. Comparison criteria results

Models under Comparison	Comparison Criteria				
	R squared	***MSE***	***Bias***	***Variation***	***RMSPE***
Model due to [2]	0.9905	7.61	0.448	2.792	2.828
Model due to [21]	0.9493	41.20	1.826	6.314	6.573
Model due to [11]	0.9905	7.61	0.448	2.792	2.828
Model due to [5]	0.9930	5.66	0.367	2.411	2.439
Model due to [8]	0.9974	2.09	0.073	1.481	1.483
Proposed	0.9974	2.09	0.073	1.481	1.483

Table 1.2 above provides the values of the goodness of fit metrics obtained by the models under comparison. It clearly shows that the proposed model is superior. The fitting of the proposed model is illustrated graphically in Figure 1.2. The fitting is excellent.

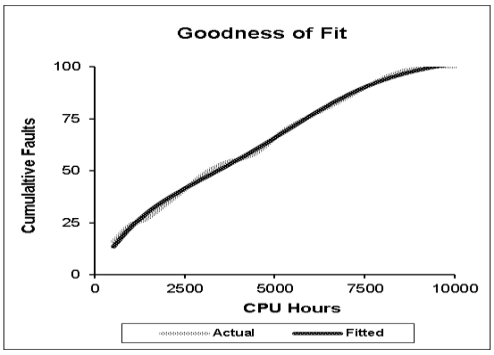

Figure 1. 2 Software reliability growth curves

5.2 Second Software Development Project

The second software test data had been collected during 21 weeks, spending 25.3 CPU hours of testing a real time command and control application (systems T1) of size 21.7K object instructions, 136 faults were detected during the period [10].

Figure 2.1 traces the Laplace trend test. The values of the trend test are oscillating between -2 and +2 indicate stable reliability. However, from 8th week the values are completely positive with some fluctuations but this fluctuation doesn't affect the reliability much and reliability has decay behaviour. As the duration of the decrease seems long, attention must be paid to it. Those situations in which reliability continues to decrease can point to problems in the software [3].

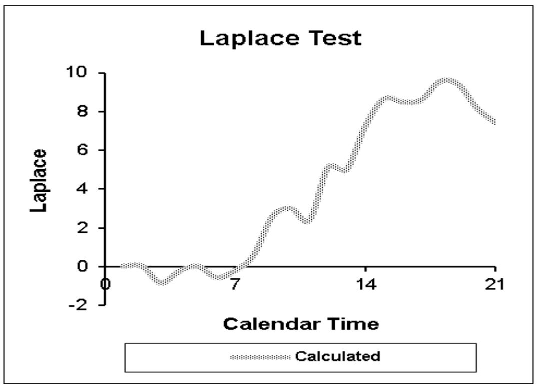

Figure 2.1 Laplace test data trend

The unknown parameters of the models under comparison have been estimated using the regression module of SPSS. The values of estimated parameters have been tabulated below in Table 2.1. The estimation results of the proposed model depict an imperfect fault debugging environment, i.e., that the debugging efficiency parameter 'p' is 35.7%. Therefore, imperfect fault debugging does not change the content of faults in the software. The value 490.51 for the shape parameter 'β_i' implies the S-shaped of the actual test data. It is estimated that a total of 147 faults are observed in the 21 weeks and all of them were removed successfully in the same debugging period.

Table 1.1. Parameter estimation results

Models under Comparison	Parameter Estimation						
	a	***a₁***	***a₂***	***a₃***	***b***	***p***	***β***
Model due to [2]	150	—	—	—	0.086	—	—
Model due to [21]	128	—	—	—	0.288	—	—
Model due to [11]	150	—	—	—	0.086	—	0
Model due to [5]	153	101	52	0	0.140	—	—
Model due to [8]	151	91	60	0	0.176	—	17.20
Proposed	147	104	43	0	0.426	0.357	490.5

– indicates the parameter is not part of the corresponding model

Table 1.2. Comparison criteria results

Models under Comparison	Comparison Criteria				
	R squared	***MSE***	***Bias***	***Variation***	***RMSPE***
Model due to [2]	0.99239	18.03	1.174	4.182	4.343
Model due to [21]	0.96595	80.77	3.502	8.481	9.176
Model due to [11]	0.99239	18.03	1.174	4.182	4.343
Model due to [5]	0.99372	14.90	0.923	3.840	3.950

Model due to [8]	0.99495	11.97	0.511	3.507	3.544
Proposed	0.99545	10.79	0.527	3.322	3.363

Table 2.2 above provides the values of the goodness of fit metrics obtained by the models under comparison. It clearly shows that the proposed model is superior to the other models under comparison. Except for Bias metric the fault classification model [8] got lower value. The Fitting of the proposed model is illustrated graphically in Figure 2.2. The fitting is excellent.
Based on our data analyses and model comparisons. We may conclude that incorporating imperfect fault debugging phenomenon yields better results.

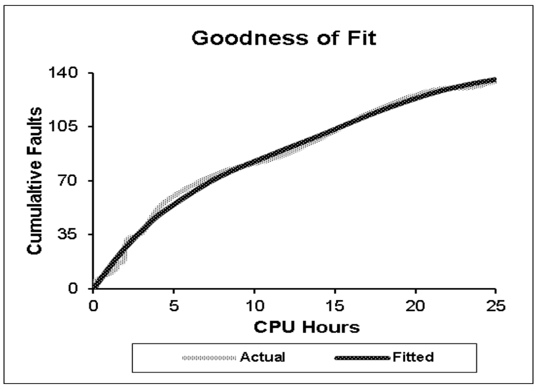

Figure 2. 2 Software reliability growth curves.

7. Conclusion

Categorization the faults in the software system into three types according to their debugging complexity where each type is modelled by a different reliability growth curve helps in capturing variability in the growth curves depending on the debugging environment and at the same time it has the capability to reduce either to exponential or S-shaped growth curves.It is observed that the incorporation of imperfect debugging increases the credibility of the model. To increase the model credibility, two types of imperfect debugging have been incorporated. The results were fairly encouraging when compared with other model developed under similar environment. The results can be viewed through the numerical illustrations shown in Tables and Figures obtained after the estimation performed on three actual test datasets cited from real software development projects.Such type of integrated modeling approach is very much suited for object-oriented and distributed systems development environments. Another point worth mentioning about the modelling approach is its inbuilt flexibility. It can describe an imperfect debugging phenomenon as exponential or S-shaped growth curve. At the same time, it can describe the situation where the imperfect debugging phenomenon does not exist as seen in the first software project development.

Since 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. We feel that the proposed modeling approach is a step in that direction. The extension of the modeling approach incorporating more types of faults is an ongoing challenge that stimulates us to continue working on this direction.

References

[1] Edris K., Shatnawi O., “The Pham Nordmann Zhang (PNZ) software reliability model revisited,” *Proc. 10th IASTED International Conference on Software Engineering*, Innsbruck, Austria, pp. 205-212, 2011.

[2] Goel A.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.

[3] Kanoun K., Kaaniche M., and Laprie J-C., “Qualitative and Quantitative Reliability Assessment,” *IEEE Software*, vol. 14, pp. 77-87, 1997.

[4] Kapur P.K., Bardhan A.K., and Shatnawi O., “Why Software Reliability Growth Modelling should define Errors of Different Severity,” *Journal of the Indian Statistical Association*, vol. 40, no. 2, pp. 119-142, 2002.

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

[6] Kapur P.K., Pham H., Anand S., and Yadav K., “A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation,” *IEEE Transactions on Reliability*, vol. 60, no. 1, pp. 331–340, 2011.

[7] Kapur P.K., Pham H., Gupta A., Jha P.C., *Software Reliability Assessment with OR Applications*, Springer, 2011.

[8] Kapur P.K., Shatnawi O., and Yadavalli V.S.S., “A Software Fault Classification Model,” *South African Computer Journal*, vol. 33, pp. 1-9, 2004.

[9] Misra P.N., “Software Reliability Analysis,” *IBM System Journal*, vol. 22, no. 3, pp. 262-270, 1983.

[10] Musa J.D., Iannino A., and Okumoto K., Software Reliability: Measurement, Prediction, Application, McGraw-Hill, 1978.

[11] Ohba M., "Software Reliability Analysis Models," *IBM Journal of Research and Development*, vol.28, no. 4, pp. 428-443, 1984.

[12] 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. 2, pp. 169-175, 1999.

[13] Pham H., *Software reliability*. Springer, 2000.

[14] Shatnawi O., "Discrete Time NHPP Models for Software Reliability Growth Phenomenon," *International Arab Journal of Information Technology*, vol. 6, No. 2, pp. 124-131, 2009.

[15] Shatnawi O., "Measuring Commercial Software Operational Reliability: An Interdisciplinary Modelling Approach," *EksploatacjaiNiezawodność - Maintenance and Reliability*, vol. 16, no. 4, pp. 585–594, 2014.

[16] Shatnawi O., and Kapur P.K., "A Generalized Software Fault Classification," *WSEAS Transactions on Computers*, vol. 7, no. 9, pp. 1375–1384, 2008.

[17] Shatnawi O., "Testing-effort dependent software reliability model for distributed systems," *International Journal of Distributed Systems and Technologies*, vol. 4, no. 2, pp. 1-14, 2013.

[18] Wood A., "Predicting Software Reliability," *IEEE Computers*, vol. 29, issue 11,pp 69-77, 1996.

[19] Xie M., *Software reliability modelling*. World Scientific, 1991.

[20] Yamada S., Software *Reliability Modeling: Fundamentals and Applications*, Springer, 2014.

[21] Yamada S., Ohba M., and Osaki S., "S-shaped Reliability Growth Modelling for Software Error Detection," *IEEE Transactions on Reliability;*vol. 32, no. 5, pp. 475-478, 1983.

[22] Yamada S., Osaki S., and Narihisa H., "Software Reliability Growth Models with Two Types of Errors," *RechercheOperationnelle / Operations Research (RAIRO)*, vol. 19, no. 1, pp. 87-104, 1985.

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 the deputy dean of the prince Hussein bin Abdullah College for Information Technology at Al al-Bayt University, Jordan. His research interests are in software reliability engineering, with an emphasis on improving software reliability and dependability. He has co-edited a special issue on "Reliability and Optimization, June 2014" of the International Journal of Systems Assurance Engineering and Management (Springer). He has been conferred an award by the Society for Reliability Engineering, Quality, and Operations Management (SREQOM) at the 17th ICQRIT'2015 for promising contributions in Software Reliability Engineering.