A Generalized Software Fault Classification Model

OMAR SHATNAWI[1] and P.K. KAPUR[2]
[1]Department of Computer Science, Al al-Bayt University, Mafraq 25113, JORDAN
[2]Department of Operational Research, University of Delhi, Delhi 110007, INDIA
dromali@lycos.com

Abstract: - Most non-homogenous Poisson process (NHPP) based software reliability growth models (SRGMs) presented in the literature assume that the faults in the software are of the same type. However, this assumption is not truly representative of reality. It has been observed that the software contains different types of faults and each fault requires different strategies and different amount of testing-effort to remove it. If this assumption is not taken into account, the SRGM may give misleading results. This paper proposes a generalized model based on classification the faults in the software system according to their removal complexity. The removal complexity is proportional to the amount of testing-effort required to remove the fault. The testing-effort expenditures are represented by the number of stages required to remove the fault after the failure observation / fault isolation (with time delay between the stages). Therefore, it explicitly takes into account the faults of different severity and can capture variability in the growth curves depending on the environment it is being used and at the same time it has the capability to reduce either to exponential or S-shaped growth curves. Such modelling approach is very much suited for object-oriented programming and distributed development environments. Actual software reliability data have been used to demonstrate the proposed generalized model.

Key-Words: - Software engineering, Software testing, Software reliability, NHPP, SRGM, Fault severity.

1 Introduction

Software reliability modelling is very important due to the fact that it is not possible to produce fault-free software. The faults in the software occur due to human imperfection. These faults manifest themselves in terms of failures when the software is run. Testing phase in the software development process aims at detecting and removing these faults and making the software more reliable. Thus it is very important to evaluate software reliability during testing phase, based on software fault data analysis. Models concerned with the relationship between time span of testing and the cumulative numbers of faults removed through software testing are called software reliability growth models (SRGMs). Based on non-homogeneous Poisson process (NHPP), several SRGMs have been developed because the models can be easily applied in actual software development.

In the software reliability literature, most researchers assume a constant fault removal rate per fault in deriving their NHPP based SRGMs [1-4]. That is, they assume that all faults have equal probability of being removed during the software testing process, and the rate remains constant. In reality, the fault removal rate strongly depends on the skill of test teams, program size and software testability [5-8]. Through real data experiments and analyzes on several software development projects, it has been observed that the fault removal rate has three possible trends as time progresses: increasing, decreasing or constants. It decreases when the software has been used and tested repeatedly, showing reliability growth. It can also increase if the testing techniques or requirements are changed, or new faults are introduced due to new software features or imperfect debugging. The learning-process of test-team has also been studied. The learning is closely related to the changes in the efficiency of testing during a testing phase. The idea is that in organizations that have advanced software-process, testers might be allowed to improve dynamically their testing process as they learn more about the product. This could result in a fault removal rate which increases monotonically over the testing period. Learning usually manifests itself as a changing fault detection rate. To capture the learning-process, the researchers adopted a time-dependent fault removal rate [9-11].

The rest of this paper is organized as follows: Section 2 revisits the software fault classification model and extends it to formulate the proposed generalized software fault classification model. Sections 3 and 4 present the method used for parameter estimation, and the criteria used for validation and selection respectively. Section 5 and

6 provide the applications of the proposed model to actual software reliability data sets cited from real software development projects and model selection respectively. This paper concludes in Section 7.

2 Software Reliability Modelling

2.1 Model Development

Most SRGMs developed in the literature, assume that the faults in the software are of the same type. This assumption implies that the fault removal rate per remaining fault is independent of the testing time and the fault removal intensity is linearly proportional to the remaining faults. Such assumption helps to simplify the problem of modelling and provides to a certain extent plausible results. However, this assumption is not truly representative of reality. It has been observed that the software contains different types of faults and each fault requires different strategies and different amount of testing-effort to remove it. If this assumption is not taken into account, the SRGM may give misleading results. To address this issue several SRGMs have been developed in the literature. Ohba proposed the hyper exponential model, assuming that software consists of different modules [12]. Each module has its own characteristics and thus the faults detected in a particular module have their own peculiarities. Therefore, the fault removal rate for each module is not the same. He suggested that the fault removal process for each module can be modelled separately and the total fault removal phenomenon is the sum of the fault removal process of all the modules. Yamada et al. proposed a modified exponential model assuming that the software contains two types of faults namely, simple and hard [13]. Both faults are modelled separately and accordingly the fault removal process is the sum of the two models. Kareer et al. proposed a model with two types of faults [14]. For each type, the fault removal rate per remaining faults is assumed to be time dependent. Each fault type is modelled by the Delayed S-shaped model [5]. The total removal phenomenon is modelled by the superposition of the two models. Kimura et al. proposed the exponential S-shaped model [15]. This model again assumes that the software contains two types of faults namely, simple and hard. The removal of simple faults is described by the exponential model [4] while the removal of hard faults is described by the Delayed S-shaped model [5]. The total fault removal process is again the superposition of the two models. Kapur et al. proposed a model with three types of faults [16], modelled by the exponential model [4], the Delayed S-shaped model [5] and the three-stage Erlang model [17]. For each type, the fault removal rate is assumed to be constant. In all these developed models, the authors ignore the role of the learning-process during the testing phase by not accounting for the experience gained with the progress of testing.

To address these issues, we have recently proposed a software fault classification model based on the assumption that the testing phase consists of three processes namely, failure observation, fault isolation and fault removal [18]. In this model, software faults are classified into three types namely, simple, hard and complex according to the amount of testing-effort needed to remove them. The time delay between the failure observation and the subsequent fault removal is assumed to represent the severity of the faults. The model also incorporates a logistic learning function during the removal phase as it is expected the learning-process will grow with time. The total fault removal phenomenon is the superposition of the three processes. The model had been validated and compared with the well-established NHPP models including exponential [4], delayed S-shaped [5] and inflection S-shaped [6-8] based on actual software reliability data. The results were found to be fairly encouraging in terms of goodness-of-fit, predictive validity, and reliability evaluation measures.

In this paper, the software fault classification model with three types of faults [18] is revisited and further extended to represent a software system with more than three types of faults. A method is adopted to select the number of faults type, which can be included in the proposed generalized software fault classification model, so that the model fits and predicts the fault removal process satisfactorily [2].

2.2 Model Assumptions

1. Failure occurrence, fault detection, or fault removal phenomena follow an NHPP with mean value function $m_{GE\text{-}n}(t)$.
2. Faults are classified depending on their severity and accordingly efforts are needed to detect and remove them.
3. Fault removal process is prefect.
4. Each time a failure occurs, an immediate (delayed) effort takes place to decide the cause of the failure in order to remove it. The

time delay between the failure observation and its subsequent fault removal is assumed to represent the severity of the faults. The more severe the fault, more the time delay.

5. Fault removal complexity is proportional to the amount of testing-effort the number remove the fault. The testing-effort expenditures are represented by the number of stages required to remove the fault after failure observation / fault detection (with time delay between the stages)
6. Fault removal rate of the first fault type is proportionality constant and for the remaining faults types is a logistic function as it is expected the learning-process will grow with time.

2.3 Model Notations

- a Initial fault-content of the software.
- i Type of fault (i=1,2,3,…,n).
- a_i Initial content of fault-type i ($\sum_{i=1}^{n} a_i = a$).
- b_i Proportionality constant represents failure rate / fault isolation(s) / fault removal rate per fault for fault-type i.
- $b_i(t)$ Logistic learning-process function, i.e., fault removal rate per fault for fault-type i.
- $m_{ii}(t)$ Mean number of fault removed of type i in i number of processes (stages) by time t.
- β Constant parameter in the logistic learning function.

2.4 Model Formulation

Assuming that the software contains 'n' different types of faults and on each type of fault a different strategy is required to remove the cause of failures due to that fault. We assume that for a type i(i=1,2,3,..,n) fault, i different processes (stages) are required to detect/remove the cause of the failure.

The time-dependent fault removal rate per fault for i(i=2,3,..,n) fault-type, which is a nondecreasing S-shape curve and capture the learning-process of the software testers given as

$$b_i(t) = \frac{b_i}{1+\beta e^{-b_i t}} \tag{1}$$

The expected number of faults removed in (t, $t+\Delta t$) is proportional to the number of faults remaining to be removed. Accordingly we may write the following differential equations:

For i=1

$$\frac{d}{dt} m_{11}(t) = b_1\left(a_1 - m_{11}(t)\right) \tag{2}$$

For i=2

$$\begin{aligned} \frac{d}{dt} m_{21}(t) &= b_2\left(a_2 - m_{21}(t)\right) \\ \frac{d}{dt} m_{22}(t) &= b_2(t)\left(m_{21}(t) - m_{22}(t)\right) \end{aligned} \tag{3}$$

For i=3

$$\begin{aligned} \frac{d}{dt} m_{31}(t) &= b_3\left(a_3 - m_{31}(t)\right) \\ \frac{d}{dt} m_{32}(t) &= b_3\left(m_{31}(t) - m_{32}(t)\right) \\ \frac{d}{dt} m_{33}(t) &= b_3(t)\left(m_{32}(t) - m_{33}(t)\right) \end{aligned} \tag{4}$$

Solving the differential equations given in (2), (3), and (4) under the initial conditions $m_{11}(t=0)=0$, $m_{22}(t=0)=0$, and $m_{33}(t=0)=0$ respectively we get

$$\begin{aligned} m_1(t) \equiv m_{11}(t) &= a_1\left(1 - e^{-b_1 t}\right) \\ m_2(t) \equiv m_{22}(t) &= a_2 \frac{1-(1+b_2 t)e^{-b_2 t}}{1+\beta e^{-b_2 t}} \\ m_3(t) \equiv m_{33}(t) &= a_3 \frac{1-\left(1+b_3 t+\frac{b_3^2 t^2}{2}\right)e^{-b_3 t}}{1+\beta e^{-b_3 t}} \end{aligned} \tag{5}$$

The software fault classification model is the superposition of the three NHPP with mean value functions given in (5). Thus, the mean value function of the superposed NHPP is

$$\begin{aligned} m_{GE-3}(t) &= \sum_{i=1}^{3} m_i(t) \\ &= a_1\left(1 - e^{-b_1 t}\right) + \sum_{i=2}^{3} a_i \frac{1-\left(\sum_{j=0}^{i-1} \frac{(b_i t)^j}{j!}\right)e^{-b_i t}}{1+\beta e^{-b_i t}} \end{aligned} \tag{6}$$

The removal rate per faults for the first three types of faults is given as

$$\begin{aligned} d_1(t) &= \frac{\frac{d}{dt} m_1(t)}{a_1 - m_1(t)} = b_1 \\ d_2(t) &= \frac{\frac{d}{dt} m_2(t)}{a_2 - m_2(t)} = \frac{b_2(1+\beta+b_2 t) - b_2(1+\beta e^{-b_2 t})}{(1+\beta e^{-b_2 t})(1+\beta+b_2 t)} \\ d_3(t) &= \frac{\frac{d}{dt} m_3(t)}{a_3 - m_3(t)} = \frac{b_3(1+\beta+b_3 t+\frac{b_3^2 t^2}{2}) - b_3(1+\beta e^{-b_3 t})(1+b_3 t)}{(1+\beta e^{-b_3 t})(1+\beta+b_3 t+\frac{b_3^2 t^2}{2})} \end{aligned} \tag{7}$$

We observe that $d_1(t)$ is constant with respect to time 't' while $d_2(t)$ 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, $m_2(t)$ and $m_3(t)$ fault growth curves behave similar to the $m_1(t)$ 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 to b_1.

Substituting $b_3=b_2=b_1$ in the right hand side of equation (6), yields to $d_3(t)<d_2(t)<b_1$, which is in accordance with the severity of the faults [2,19]. Assuming $b_3=b_2=b_1=b$(say), then equation (6) can be written as

$$m_{GE-3}(t)=\sum_{i=1}^{3}m_i(t)$$
$$=a_1\left(1-e^{-bt}\right)+\sum_{i=2}^{3}a_i\frac{1-\left(\sum_{j=0}^{i-1}\frac{(bt)^j}{j!}\right)e^{-bt}}{1+\beta e^{-bt}} \quad (8)$$

The improvement in the performance of the software fault classification model was attributed to the inclusion of three types of faults [18]. This observation and results suggest that the inclusion of more fault type in the software fault classification model may further improve its performance.

The procedure followed in equations (2), (3), and (4) can be extended to formulate the proposed generalized software fault classification model with 'n' types of faults.

Similarly for $i=n$, we have

$$\frac{d}{dt}m_{n1}(t)=b_n\left(a_n-m_{n1}(t)\right)$$
$$\frac{d}{dt}m_{n2}(t)=b_n\left(m_{n1}(t)-m_{n2}(t)\right)$$
$$\frac{d}{dt}m_{n3}(t)=b_n\left(m_{n2}(t)-m_{n3}(t)\right)$$
$$\vdots$$
$$\frac{d}{dt}m_{nn}(t)=b_n(t)\left(m_{nn-1}(t)-m_{nn}(t)\right) \quad (9)$$

Note that the first subscript stands for the type of fault and the second subscript stands for the number of processes (stages) required to remove the fault after it failure occurrence / fault detection and is dependent on the type of the fault, i.e., if the fault is of the type k, then it will be removed in k stages because of its complexity.

Solving the differential equations (2), (3), (4), and (9) respectively we get

$$m_1(t)\equiv m_{11}(t)=a_1\left(1-e^{-b_1t}\right)$$
$$m_2(t)\equiv m_{22}(t)=a_2\frac{1-(1+b_2t)e^{-b_2t}}{1+\beta e^{-b_2t}}$$
$$m_3(t)\equiv m_{33}(t)=a_3\frac{1-\left(1+b_3t+\frac{b_3^2t^2}{2}\right)e^{-b_3t}}{1+\beta e^{-b_3t}}$$
$$\vdots$$
$$m_n(t)\equiv m_{nn}(t)=a_n\frac{1-\left(\sum_{j=0}^{i-1}\frac{(b_it)^j}{j!}\right)e^{-b_it}}{1+\beta e^{-b_3t}} \quad (10)$$

The proposed generalized software fault classification model is the superposition of all the NHPP with mean value functions given in (10). Thus, the mean value function of the superposed NHPP is

$$m_{GE-n}(t)=\sum_{i=1}^{n}m_i(t)$$
$$=a_1\left(1-e^{-bt}\right)+\sum_{i=2}^{n}a_i\frac{1-\left(\sum_{j=0}^{i-1}\frac{(b_it)^j}{j!}\right)e^{-b_it}}{1+\beta e^{-b_it}} \quad (11)$$

The analysis followed in equation (7) can again be applied. Therefore, assume $b_n=\ldots=b_3=b_2=b_1=b$. Accordingly, equation (11) can be rewritten as

$$m_{GE-n}(t)=\sum_{i=1}^{n}m_i(t)$$
$$=a_1\left(1-e^{-bt}\right)+\sum_{i=2}^{n}a_i\frac{1-\left(\sum_{j=0}^{i-1}\frac{(bt)^j}{j!}\right)e^{-bt}}{1+\beta e^{-bt}} \quad (12)$$

The exponential, delayed S-shaped, inflection S-shaped, exponential S-shaped, generalized Erlang, and software fault classification models given in [4,5,6-8,15,2,18] respectively, can be obtained from proposed generalized software fault classification model given in equation (12). Thus high lighting its applicability and flexibility. Here, we test the proposed generalized software fault classification model with '2', '3', '4', and '5' types of faults. These models are referred to $m_{GF\text{-}2}(t)$, $m_{GF\text{-}3}(t)$, $m_{GF\text{-}4}(t)$, and $m_{GF\text{-}5}(t)$ respectively.

3 Parameter Estimation Technique

The maximum likelihood estimation (MLE) method is used to estimate the parameters of the proposed generalized software fault classification model given in equation (12).

Since all the data sets used are given in the form of pairs $(t_i,x_i)(i=1,2,...,k)$, where x_i is the cumulative number of faults detected by time t_i $(0<t_1<t_2<\ldots<t_k)$ and t_i is the accumulated time spent to remove x_i faults. The Likelihood function 'L' for the unknown parameters with the mean value function is given as

$$L\left(Parameter|(t_i,x_i)\right)=\prod_{i=1}^{k}\frac{\left(m(t_i)-m(t_{i-1})\right)^{x_i-x_{i-1}}}{(x_i-x_{i-1})!}e^{-(m(t_i)-m(t_{i-1}))} \quad (13)$$

The MLE of the SRGM parameters can be obtained to by maximizing equation (13) with respect to the constraints: ($a_i\geq 0$, $0<b<1$, $\beta\geq 0$).

4 Model Validation

To check the validity of the proposed generalized software fault classification model to describe the software reliability growth, it has been tested on four software reliability data sets obtained from real software development projects.

The first data set (DS-I) had been collected during 20 weeks of testing (10,000 CPU hours were utilized) one of four major releases of the software products at Tandem Computers Company, Los Anglos (CA), 100 faults were detected during the period [3].

The second data set (DS-II) had been collected during 111 days of testing a real time monitor and control software system consist of about 200 modules, on average has 1 KLOC written in FORTRAN, 481 faults were detected during the period [3].

The third data set (DS-III) had been collected during 12 months of testing a command, control and communication software system of size 1317K LOC written by CENTRAN and ALC, 2657 faults were detected during the period [20].

The fourth data set (DS-IV) had been collected during 35 months of testing a defense, ground based radar software system of size 124K LOC written by Jovial, 1301 faults were detected during the period [20].

The performance of an SRGM judged by its ability to fit the past software fault data (goodness-of-fit) and to predict satisfactorily the future behavior of the fault removal process from present and past data behaviour (predictive validity) [2,21].

4.1 The Goodness-of-Fit Criteria

4.1.1 The Sum of Squared Error (SSE)

This metrics measures the distance of a model estimate value from the actual data, and defined as

$$SSE=\sum_{i=1}^{k}\left(\hat{m}_{GE-n}(t_i)-x_i\right)^2 \quad (14)$$

where k is the number of observations, $\hat{m}_{GE-n}(t_i)$ is the estimated cumulative number of faults by time t_i obtained from the fitted mean value function and x_i is the total number of faults removed by time t_i.

Lower value of SSE indicates less fitting error, thus better goodness-of-fit.

4.1.2 The Akaike Information Criterion (AIC)

This metrics measures the ability of a model to maximize the likelihood function that is directly related to the degree of freedom during fitting, and defined as

$$AIC=-2\times\log\left(Max.\ of\ \ L\right)+2\times N \quad (15)$$

where 'N' is the number of the parameters used in the model and 'L' likelihood function.

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

4.1.3 Coefficient of Multiple Determination (R^2)

This metric measures the percentage of the total variation about the mean accounted for by the fitted curve. It ranges in value from 0 to. We define this coefficient 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

$$R^2=1-\frac{residual\ \ SS}{corrected\ \ SS} \quad (16)$$

Small values indicate that the model does not fit the data well.

4.2 The Predictive Validity Criterion

Predictive validity is defined as the capability of the SRGM to determine the fault removal phenomena from present and past fault removal data. This capability is significant only when the fault removal phenomena are changing [21].

Suppose that we have found x_k faults by the testing end time t_k. The fault removal data up to time $t_j(\leq t_k)$, i.e., the software reliability data is truncated to time t_j are used to estimate the parameters of the SRGM. The number of faults removals by time t_k can be predicted by the SRGM and compared to the reported fault removal at this time, i.e., x_k.

We can check the predictive validity by tabulating the relative prediction error (RPE) values against the testing progress ratio (t_j/t_k) in percent for software reliability data.

The relative prediction error (RPE) is defined as

$$RPE = \frac{\left(\hat{m}_{GE-n}(t_k) - x_k\right)}{x_k} \quad (17)$$

where $\hat{m}_{GE-n}(t_i)$ is the predicted cumulative number of faults by time t_k.

The RPE will approach zero as t_j approaches t_k. If the RPE values are positive/negative, the model tends to overestimate/underestimate the future fault removal phenomena. Numbers closer to zero imply more accurate prediction. The relative error is said to be acceptable if it is within ±10 percent [2].

4.3 Model Selection Criteria

The software reliability data and the comparison criteria used earlier are applied to check the performance of the proposed generalized software fault classification model. The data are truncated into different proportions and used to estimate the parameters of the models under comparison. For each truncation, one value SSE and RPE are obtained [2]. For a given model, these values may vary from one point to another**.**

To measure the performance of the models under comparison all over the truncations, the following criteria are used

- Average Sum of Squared Error (ASSE)

$$ASSE = \sum_{i=1}^{j} \frac{SSE_i}{j} \quad (18)$$

where 'j' is the number of truncations and SSE_i is the SSE of the i^{th} data set truncation.

- Average Relative Prediction Error (ARPE)

$$ARPE = \sum_{i=1}^{j} \frac{|RPE_i|}{j} \quad (19)$$

where $|RPE_i|$ is the absolute value of the RPE of the i^{th} data set truncation.

5 Data Analyses and Model Comparisons

The software reliability data and the comparison criteria in terms of goodness-of-fit and predictive validity defined earlier in Section 4 are applied to check the performance of the proposed generalized software fault classification model defined in equation (12) with 'n' types of faults. As the number of fault types may tend to be large, modelling each fault type individually is not practically possible. However, if the faults classified into a smaller number of groups, where each group contains the faults of common characteristics, the number of faults can be reduced. Therefore, we have tested this proposed generalized model for only five types of faults. That is, the models under comparison are $m_{GF\text{-}2}(t)$, $m_{GF\text{-}3}(t)$, $m_{GF\text{-}4}(t)$, and $m_{GF\text{-}5}(t)$ that estimates the presence of '2', '3', '4', and '5' types of faults respectively.

5.1 First Software Development Project

The results of the parameter estimation and the goodness-of-fit metrics in terms of SSE, AIC, and R^2 of the models under comparison are given in Tables 1 and 2 respectively. On applying $m_{GF\text{-}2}(t)$, the model reveals the presence of two types of faults where the majority are of type '2'. Accordingly, the fault removal phenomena are described by a combination of types '1' and '2'. On applying $m_{GF\text{-}3}(t)$, the model reveals the presence of two types of faults where the majority are of type '3'. Accordingly, the fault removal phenomena are described by a combination of types '1' and '3'. On applying $m_{GF\text{-}4}(t)$, the model reveals the presence of three types of faults where the majority are of type '4'. Accordingly, the fault removal phenomena are described by a combination of types '1', '3', and '4'. On applying $m_{GF\text{-}5}(t)$, the model reveals the presence of two types of faults where the majority are of type '5'. Accordingly, the fault removal phenomena are described by a combination of types '1' and '5'. The duration of test justifies the estimation of high complexity fault-type (which required more testing-effort to be removed). It is clear from Table 2 that the results of the goodness-of-fit metrics are significantly improving with the introduction of more types of faults.

Table 1. Parameter Estimation Results (DS-I)

Models under Comparison	Parameter Estimation						
	a_1	a_2	a_3	a_4	a_5	b	β
$m_{GF\text{-}2}(t)$	39	62	—	—	—	0.410	33.7
$m_{GF\text{-}3}(t)$	40	0	62	—	—	0.409	20.6
$m_{GF\text{-}4}(t)$	39	0	12	52	—	0.441	13.4
$m_{GF\text{-}5}(t)$	35	0	0	0	67	0.526	2.50

Table 2. Goodness-of-Fit Results (DS-I)

Models under Comparison	Comparison Criteria		
	SSE	AIC	R^2
$m_{GF\text{-}2}(t)$	35.68	79.86	.998
$m_{GF\text{-}3}(t)$	34.10	80.03	.998
$m_{GF\text{-}4}(t)$	33.66	82.25	.998
$m_{GF\text{-}5}(t)$	30.33	79.43	.998

The results of the SSE and RPE of the models under comparison for different truncations of the data set are given Tables 3 and 4 respectively. It is observed that the values vary from one truncation to another. Overall the values are significantly improving with the introduction of more types of faults. It is observed that the values vary from one truncation to another. It is clear from Table 4 that the values overestimates the fault removal process except when the testing progress ratio is about 80% (on applying $m_{GF\text{-}2}(t)$), and 70% and 60% (on applying $m_{GF\text{-}2}(t)$ and $m_{GF\text{-}3}(t)$) it underestimates the fault detection process.

Table 3. SSE Metric Results (DS-I)

Models under Comparison	Testing Progress Ratio			
	60%	70%	80%	90%
$m_{GF\text{-}2}(t)$	146.4	63.7	36.5	35.8
$m_{GF\text{-}3}(t)$	77.5	42.7	38.0	34.8
$m_{GF\text{-}4}(t)$	34.4	44.4	39.3	35.8
$m_{GF\text{-}5}(t)$	39.4	36.8	38.0	32.6

Table 4. RPE% Metric Results (DS-I)

Models under Comparison	Testing Progress Ratio			
	60%	70%	80%	90%
$m_{GF\text{-}2}(t)$	-4.46	-2.18	-0.10	0.49
$m_{GF\text{-}3}(t)$	-2.39	-.730	2.53	1.07
$m_{GF\text{-}4}(t)$	1.38	2.73	2.27	1.79
$m_{GF\text{-}5}(t)$	2.60	2.41	2.53	1.90

5.2 Second Software Development Project

The results of the parameter estimation and the goodness-of-fit metrics in terms of SSE, AIC, and R^2 of the models under comparison are given in Tables 5 and 6 respectively. On applying $m_{GF\text{-}2}(t)$, the model reveals the presence of two types of faults where the majority are of type '2'. Accordingly, the fault removal phenomena are described by a combination of types '1' and '2'. On applying $m_{GF\text{-}3}(t)$, $m_{GF\text{-}4}(t)$ and $m_{GF\text{-}5}(t)$, they models reduce to $m_{GF\text{-}2}(t)$. The duration of test justifies the estimation of low complexity fault-type. It is clear from Table 6 that the results are not improving with the introduction of more than two types of faults.

Table 5. Parameter Estimation Results (DS-II)

Models under Comparison	Parameter Estimation						
	a_1	a_2	a_3	a_4	a_5	b	β
$m_{GF\text{-}2}(t)$	133	350	—	—	—	0.080	6.11
$m_{GF\text{-}3}(t)$	133	350	0	—	—	0.080	6.11
$m_{GF\text{-}4}(t)$	133	350	0	0	—	0.080	6.11
$m_{GF\text{-}5}(t)$	133	350	0	0	0	0.080	6.01

Table 6. Goodness-of-Fit Results (DS-II)

Models under Comparison	Comparison Criteria		
	SSE	AIC	R^2
$m_{GF\text{-}2}(t)$	32793	659	0.987
$m_{GF\text{-}3}(t)$	32793	659	0.987
$m_{GF\text{-}4}(t)$	32793	659	0.987
$m_{GF\text{-}5}(t)$	32793	659	0.987

The results of the SSE and RPE of the models under comparison for different truncations of the data set are given Table 7 and 8 respectively. It is observed that the values do not vary from one truncation to another. It is clear from Table 8 that the values overestimate the fault removal process.

Table 7. SSE Metric Results (DS-II)

Models under Comparison	Testing Progress Ratio			
	60%	70%	80%	90%
$m_{GF\text{-}2}(t)$	168134	58234	34364	32857
$m_{GF\text{-}3}(t)$	168134	58234	34364	32857
$m_{GF\text{-}4}(t)$	168134	58234	34364	32857
$m_{GF\text{-}5}(t)$	168134	58234	34364	32857

Table 8. RPE% Metric Results (DS-II)

Models under Comparison	Testing Progress Ratio			
	60%	70%	80%	90%
$m_{GF\text{-}2}(t)$	11.23	6.88	1.65	0.50
$m_{GF\text{-}3}(t)$	11.23	6.88	1.65	0.50
$m_{GF\text{-}4}(t)$	11.23	6.88	1.65	0.50
$m_{GF\text{-}5}(t)$	11.23	6.88	1.65	0.50

5.3 Third Software Development Project

The results of the parameter estimation and the goodness-of-fit metrics in terms of SSE, AIC, and R^2 of the models under comparison are given in Tables 9 and 10 respectively. On applying $m_{GF\text{-}2}(t)$, the model reduces to $m_{GF\text{-}1}(t)$. On applying $m_{GF\text{-}3}(t)$, the model reveals the presence of three types of faults where the majority are of type '1'. Accordingly, the fault removal phenomena are described by a combination of types '1', '2', and '3'. On applying $m_{GF\text{-}4}(t)$, the model reveals the presence of three types of faults where the majority are of type '4'. Accordingly, the fault removal phenomena are described by a combination of types '1', '2', and '4'. On applying $m_{GF\text{-}5}(t)$, the model reveals the presence of two types of faults where the majority are of type '5'. Accordingly, the fault removal phenomena are described by a combination of types '1', '2', and '5'. The duration of test justifies the estimation of high complexity fault-type. It is clear from Table 10 that the results are significantly improving with the introduction of more types of faults.

Table 9. Parameter Estimation Results (DS-III)

Models under Comparison	Parameter Estimation						
	a_1	a_2	a_3	a_4	a_5	b	β
$m_{GF\text{-}2}(t)$	3330	0	—	—	—	0.136	0
$m_{GF\text{-}3}(t)$	1683	97	1351	—	—	0.306	0
$m_{GF\text{-}4}(t)$	1168	668	0	1176	—	0.421	0
$m_{GF\text{-}5}(t)$	786	868	0	0	1260	0.573	0

Table 10. Goodness-of-Fit Results (DS-III)

Models under Comparison	Comparison Criteria		
	SSE	AIC	R^2
$m_{GF\text{-}2}(t)$	11742	170	0.998
$m_{GF\text{-}3}(t)$	9530	172	0.998
$m_{GF\text{-}4}(t)$	8610	164	0.999
$m_{GF\text{-}5}(t)$	6242	152	0.999

The results of the SSE and RPE of the models under comparison for different truncations of the data set are given Tables 11 and 12 respectively. It is observed that the values vary from one truncation to another. The values of $m_{GF\text{-}5}(t)$) are the lowest except for truncation 70%. Overall the values are significantly improving with the introduction of more types of faults. The values of the models under comparison overestimates the fault removal process except when the testing progress ratio is about 70% (on applying $m_{GF\text{-}2}(t)$) and 60% (except for $m_{GF\text{-}5}(t)$) it underestimates the fault removal process.

Table 11. SSE Metric Results (DS-III)

Models under Comparison	Testing Progress Ratio			
	60%	70%	80%	90%
$m_{GF\text{-}2}(t)$	83698	26919	11869	12050
$m_{GF\text{-}3}(t)$	27866	11091	17249	12924
$m_{GF\text{-}4}(t)$	61868	16123	23870	10654
$m_{GF\text{-}5}(t)$	13586	12595	10601	7325

Table 12: RPE% Metric Results (DS-III)

Models under Comparison	Testing Progress Ratio			
	60%	70%	80%	90%
$m_{GF\text{-}2}(t)$	-0.051	-0.019	0.010	0.012
$m_{GF\text{-}3}(t)$	-0.010	0.020	0.035	0.027
$m_{GF\text{-}4}(t)$	-0.027	0.034	0.046	0.022
$m_{GF\text{-}5}(t)$	0.021	0.030	0.026	0.016

5.4 Fourth Software Development Project

The results of the parameter estimation and the goodness-of-fit metrics in terms of SSE, AIC, and R^2 of the models under comparison are given in Tables 13 and 14 respectively. On applying $m_{GF\text{-}2}(t)$, the model reveals the presence of two types of faults where the majority are of type '2'. Accordingly, the fault removal phenomena are described by a combination of types '1' and '2'. On applying $m_{GF\text{-}3}(t)$, the model reduces to $m_{GF\text{-}2}(t)$. On applying $m_{GF\text{-}4}(t)$, the model reveals the presence of three types of faults where the majority are of type '2'. Accordingly, the fault removal phenomena are described by a combination of types '1', '2', and '3'. On applying $m_{GF\text{-}5}(t)$, the model reduces to $m_{GF\text{-}4}(t)$. The duration of test justifies the estimation of high complexity fault-type. It is clear from Table 14 that the results are improving with the introduction of more types of faults.

Table 13. Parameter Estimation Results (DS-IV)

Models under Comparison	Parameter Estimation						
	a_1	a_2	a_3	a_4	a_5	b	B
$m_{GF\text{-}2}(t)$	132	1184	—	—	—	0.224	29.32
$m_{GF\text{-}3}(t)$	132	1184	0	—	—	0.224	29.32
$m_{GF\text{-}4}(t)$	136	934	248	0	—	0.225	27.29
$m_{GF\text{-}5}(t)$	136	934	248	0	0	0.225	27.29

Table 14. Goodness-of-Fit Results (DS-IV)

Models under Comparison	Comparison Criteria		
	SSE	AIC	R^2
$m_{GF\text{-}2}(t)$	3413	344	1.000
$m_{GF\text{-}3}(t)$	3413	344	1.000
$m_{GF\text{-}4}(t)$	3609	347	1.000
$m_{GF\text{-}5}(t)$	3609	347	1.000

The results of the SSE and RPE of the models under comparison for different truncations of the data set are given Tables 15 and 16 respectively. It Overall the values are significantly improving with the introduction of more types of faults. It is clear from Table 16 that the values overestimates the fault removal process except when the testing progress ratio is about 70%, 80%, and 90% (on applying $m_{GF\text{-}2}(t)$) and $m_{GF\text{-}3}(t)$) it underestimates the fault removal process.

Table 15: SSE Metric Results (DS-IV)

Models under Comparison	Testing Progress Ratio			
	60%	70%	80%	90%
$m_{GF\text{-}2}(t)$	73257	3491	3493	3415
$m_{GF\text{-}3}(t)$	73257	3491	3493	3415
$m_{GF\text{-}4}(t)$	10078	3524	3659	4218
$m_{GF\text{-}5}(t)$	10078	3524	3659	4218

Table 16: RPE% Metric Results (DS-IV)

Models under Comparison	Testing Progress Ratio			
	60%	70%	80%	90%
$m_{GF\text{-}2}(t)$	2.09	-0.15	-0.52	-0.12
$m_{GF\text{-}3}(t)$	2.09	-0.15	-0.52	-0.12
$m_{GF\text{-}4}(t)$	0.51	-0.06	-0.68	0.38
$m_{GF\text{-}5}(t)$	0.51	-0.06	-0.68	0.38

6 Model Selection

As discussed earlier in Section 5, the proposed generalized software fault classification model was able provide a good fit and predictive validity for all software reliability data sets. It is observed that for each data set one version of the proposed generalized model with specific combination of faults types is selected. The $m_{GF\text{-}5}(t)$ can be selected to model DS-I and DS-III, while $m_{GF\text{-}2}(t)$ is selected to model DS-II and DS-IV.

The problem is that the model selection is not a one step process. In some cases, there was a contradiction between the goodness-of-fit and the predictive validity. Also, we have observed that using the proposed generalized model with a higher number of faults improves the goodness-of-fit and the predictive validity in some cases (as in DS-I and DS-III) while it does not help in other case (as in DS-II and DS-IV).

In this section, we select the best generalized software fault classification model with optimum (minimum) number of faults types. The model selection methodology is simple. We have to select from the models under comparisons, the one that provides the minimum values of ASSE and ARPE.

Table 17 shows that the values of the ASSE are decreasing with the increase number of faults types while ARPE are increasing with the increase number of faults types except for $m_{GF\text{-}3}(t)$. Since the ARPE of the proposed generalized software fault classification model with five types of faults (i.e., $m_{GF\text{-}5}(t)$) is within the acceptable level, it can be selected for the first software development project. The selection of this model does not necessarily mean that there are five types of faults in the software, but it means that the software system contains faults of type 'five' (i.e., highly complex faults, see Table 1) thus highlighting its flexibility.

Table 17: Overall Performance Results (DS-I)

Models under Comparison	Overall Performance Criteria	
	ASSE	ARPE%
$m_{GF\text{-}2}(t)$	70.61	1.81
$m_{GF\text{-}3}(t)$	48.24	1.69
$m_{GF\text{-}4}(t)$	38.44	2.04
$m_{GF\text{-}5}(t)$	36.71	2.36

Table 18 shows that the values of ASSE and ARPE remain the same in spite of increasing the number of types of faults. Thus, the proposed generalized software fault classification model with two types of faults (i.e., $m_{GF\text{-}2}(t)$) can be selected for the second software development project.

Table 18: Overall Performance Results (DS-II)

Models under Comparison	Overall Performance Criteria	
	ASSE	ARPE%
$m_{GF\text{-}2}(t)$	73397	6.06
$m_{GF\text{-}3}(t)$	73397	6.06
$m_{GF\text{-}4}(t)$	73397	6.06
$m_{GF\text{-}5}(t)$	73397	6.06

Table 19 shows that the values of ASSE are decreasing with the increase number of faults types while ARPE values are very close to each other. It is clear that the proposed generalized software fault classification model with five types of faults (i.e., $m_{GF\text{-}5}(t)$) can be selected for the third software development project.

Table 19: Overall Performance Results (DS-III)

Models under Comparison	Overall Performance Criteria	
	ASSE	ARPE%
$m_{GF\text{-}2}(t)$	33634	0.023
$m_{GF\text{-}3}(t)$	17282	0.023
$m_{GF\text{-}4}(t)$	28129	0.032
$m_{GF\text{-}5}(t)$	11027	0.023

Table 20 shows that the values of ASSE and ARPE are decreasing with the increase number of faults types. It is clear that the proposed generalized software fault classification model with four types of faults (i.e., $m_{GF\text{-}4}(t)$) can be selected for the fourth software development project.

Table 20: Overall Performance Results (DS-IV)

Models under Comparison	Overall Performance Criteria	
	ASSE	ARPE%
$m_{GF\text{-}2}(t)$	13401	0.32
$m_{GF\text{-}3}(t)$	13401	0.32
$m_{GF\text{-}4}(t)$	5370	0.04
$m_{GF\text{-}5}(t)$	5370	0.04

7 Conclusion

The unified modelling approach adopted allows the proposed generalized software fault classification model to be flexible and simple. The flexibility is achieved by modelling the removal of each fault type by a specific growth curve according to the fault removal complexity. The objective of the unified modelling approach is to provide an efficient, reliable, and widely applicable model. Finally, we believe that the approach followed in this paper will help to a great extent in limiting the choice of finding the suitable model for a particular testing environment.

References:

[1] Xie M, *Software Reliability Modelling*, World Scientific, 1991.

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

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

[4] 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.

[5] 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.

[6] Ohba M, Inflection S-shaped Software Reliability Growth Model, In: Osaki S and Hatoyama Y (eds.) *Stochastic Models in Reliability Theory*, Springer-Verlag, Berlin, 1984.

[7] Bittanti S, Blozern P, Pedrotti E, Pozzi M and Scattolini A, A Flexible Modelling Approach in Software Reliability Growth, In Goos G and Hartmanis I (eds.), *Software Reliability Modeling and Identification*, Springer-Verlag: Berlin, 1988, pp. 101-140.

[8] Kapur PK and Garg RB, A Software Reliability Growth Model for Error Detection Phenomenon. *Software Engineering Journal,* Vol. 7, No. 4, 1992, pp. 291-294.

[9] Kuo S, Huang H and Lyu MR, Framework for Modelling Software Reliability using various Testing-Effort and Fault-Detection Rates, *IEEE Transactions on Reliability*, Vol. 50, No. 3, 2001, pp. 310-320.

[10] 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, 1999, pp. 169-175.

[11] Kapur PK, Gupta Anu, Shatnawi Omar and Yadavalli VSS, A Discrete NHPP Model for Software Reliability Growth with Imperfect Fault Debugging and Fault Generation, *International Journal of Performability Engineering*, Vol. 2, No. 4, 2006, pp. 321-334.

[12] Ohba M, Software Reliability Analysis Models, *IBM Journal of Research and Development*, Vol. 28, No. 4, 1984, pp. 428-443.

[13] Yamada S, Osaki S and Narihisa H, Software Reliability Growth Models with Two Types of Errors, *Recherche Opérationnelle / Operations Research* (*RAIRO*), Vol. 19, No. 1, 1985, pp. 87-104.

[14] Kareer N, Kapur PK and Grover PS., An S-shaped Software Reliability Growth Model with Two Types of Errors, *Microelectronics and Reliability*, Vol. 30, No. 6, 1990, pp. 1085-1090.

[15] Kimura M, Yamada S and Osaki S, Software Reliability Assessment for an Exponential S-shaped Reliability Growth Phenomenon, *Computers and Mathematics with Applications*, Vol. 24, No. 1/2, 1992, pp. 71-78.

[16] Kapur PK, Younes S and Agarawal S, Generalized Erlang Software Reliability Growth Model, *ASOR Bulletin*, Vol. 14, No. 1, 1995, pp. 5-11.

[17] Khoshogoftaar TM, Nonhomogenous Poisson Process for Software Reliability Growth, *Proceedings of the International Conference on Computational Statistics* (*COMPSTAT*), Copenhagen, Denmark, 1988, pp. 13-14.

[18] Kapur PK, Shatnawi Omar and Yadavalli VSS, A Software Fault Classification Model. *South African Computer Journal*, Issue 33, 2004, pp. 1-9.

[19] Kapur PK, Bardhan AK and Shatnawi Omar, Why Software Reliability Growth Modelling should Define Errors of Different Severity, *Journal of the Indian Statistical Association*, Vol. 40, No. 2, 2002, pp. 119-143.

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

[21] Musa JD, Iannino A and Okumoto K, *Software Reliability*, McGraw-Hill, 1987.

[22] 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.

[23] 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.

[24] Junhong G, Hongwei L and Xiaozong Y, A Software Reliability Growth Model Based on Time Series Nonlinear Analysis, *WSEAS Transactions on Signal Process*, Vol. 1, No. 1, 2005, pp. 47-54.