ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

# Generalized Framework with Different Severity of Faults for Modelling Software Reliability Growth during Testing

[I]Mohammad Altaf Dar, [II]D N Gowsami, [III]Anshu Chaturvedi
[I,II]School of Studies in Computer Science & Applications, Jiwaji University, Gwalior (M.P.), India
[III]Madhav Institute of Technology & Science, Gwalior (M.P.) India

**Abstract**
*A commonly used approach for measuring software reliability is by using an analytical model. A lot of models immerge in the literature, but none of them is universal to all the failure data set. In this paper Software Reliability Growth Model (SRGM) is developed which incorporates that the software system consists not only the single type of fault, but contain several types of faults.The faults in the system are considered into three types according to the amount of testing effort needed to remove them. The proposed model is able to fit a variety of reliability growth curves and has been validated on actual software test data sets and its performance has been compared with well-documented SRGMs reported intheliterature. The comparison criteria used in this study are MSE and SSE*

**Keywords**
*Software Reliability Growth Models, Faults, Failures and Testing Phase,*

## I. Introduction

Software plays a vital role in our society. Even though households, many of the gadgets and the automobiles are software controlled. The society has developed an extraordinary dependence on software. With the advent of the computer age, computers, as well as the software running on them, are playing a vital role in our daily lives. There are already numerous instances where failures of computer controlled systems have led to massive loss of human lives and money. Software's impact on our society and culture continues to be profound, As its importance grows, the software community continually attempts to develop technologies that will make it easier, faster, and less expensive to build high-quality computer programs. For all this we need to maintain the quality of software. So far as the quality of software is concerned, one of the primary ingredients of the software is software reliability.

### Reliability

Reliability denoted by R (t) is the probability that a system will be successful in the intervals from 0 to time t.
R (t) = P (T > t) t >= 0;
Where, T is random variable, denoting time to failure or failure time.

### Unreliability

Unreliability denoted f (t) is the probability that a given system will fail by time t.
i.e.
f (t) = P(T <= t) t >= 0;
The reliability of any system depends on the correctness of the system design, the correctness of the mapping of the system design to implementation, and the reliability of the components making up the system. As far as the computer system is concerned, it consists of two parts i.e. Hardware and Software, it is better to study their reliability separately.
Now, as far as the Software reliability is concerned, it's hard to achieve because the complexity of the software tends to be high, while any system with high degree of complexity including software will be hard to reach a certain level of reliability. "The probability, or the relative frequency of times, that a given program will work as intended by the user, i.e., without failures, in a specified environment and for a specified duration can be termed as Software reliability" [1].

Software reliability is the function of failure free operations of a computer program in a specified environment within a specified time.
SR= f( ff0, Se, St)
Where
SR – Software reliability.
ff0 – Failure free operations.
Se – Specified environment.
St - Specified time.
Software reliability is not directly related to the function of time.
$SR \neq f(t)$
Therefore, it can be written as follows:
$SyR \approx Ø\{(SR), (HR)\}$
Where
SyR – System reliability.
SR – Software reliability.
HR – Hardware reliability.
Ø – factor Constant.
So the reliability is directly related to the number of failures. A system can operate or it can fail, hence the statement can written in probabilistic form as follows: [2]
1(certainty) = P(R) + P (F)
Rearranging the statements,
P(R) = 1 – P(F)
Therefore, problem of SR is directly proportional to the software complexity.
The aim of the software personnel and software developing industry is to develop the software, which is having reliability above the specified level. A commonly employed approach for measuring software reliability is by utilizing an analytical model whose parameters is generally calculated from available data sets of package failures. A lot of models immerge in the literature, but none of them is universal to all the failure data set.
In this paper Software Reliability Growth Model (SRGM) is developed which came to this idea that the software system consists not only the single type of fault, but contain several types of faults according to the amount of time or effort to remove them. According to this paper the software contains simple, complex and harder type of faults which are modelled according to their types i.e., Type I models the simple type of fault, Type II models the complex type of fault and the Type III models the harder

type of fault. However, each type of fault has a specific impact on the progress of software testing. To consider this problem Ohba [3] proposed a Hyper-exponential SRGM that considers the system contain different types of modules. Each module has its own character and thus fault found, in particular, module have their own strong point. Yamada et al. [4] proposed a modified exponential SRGM considering that the system contains two types of faults. In this model the fault observed in the former stage of the software phase, a great bit of simple errors were eliminated while the hard faults are taken out in the next stage of the testing. Kareer et al. [5] proposed a SRGM considering that the system contains two types of faults. Each type of fault is modelled by S-shaped SRGM [6]. Kimura et al. [3] proposed an exponential, S-shaped SRGM, considering that the system contains two types of faults (simple and hard). The removal of simple faults is described by NHPP exponential model, whereas the removal of hard faults is described by the S - shaped model. The total fault removal process is the superposition of the two NHPP. Kapur at el. [7, 8] proposed an SRGM considering that the system contains three types of faults.Rafi and Akthar [12] developed three SRGMs considering the generalized modified Weibull (GMW), Gompertz and logistic exponential curves as testing-effort functions, respectively, with optimal release policy.

This paper is described in different sections, the section 1givesa general introduction and consists brief review of literature, the section 2 describe about the proposed model development, the section 3 describes about the parameter estimation and comparison criteria of the proposed model and the section 4 contain the conclusion of the paper.

## II. Software Reliability Modelling

A software reliability model is a mathematical expression that specifies the general form of the software failure process as a function of factors such as fault introduction, fault removal and the operational environment.

### 2.1 Model Development

In this paper, an SRGM is developed in which testing phase consists of three different processes, namely failure observation, fault isolation and fault removal. The time delay between the failure, observation and subsequent removal represent the hardness of the defects. The more sever the fault, more the time delay. The fault is simple, if the time delay between failure, observation, fault isolation and fault removal is negligible. This type of fault removal is modelled by the first stage of model [9]. If there is a delay between failure, observation, and fault removal then the fault is considered as a hard type of fault. This type of fault removal is modelled by the second stage of the proposed model. If the removal of fault after isolation involves a greater delay, then this type of fault is considered as a complex type of fault. This type of fault removal is modelled by three stages of the proposed model.

### 2.2 Model Assumption

1. Failure observation, fault removal is modelled by NHPP.
2. Software failure occurs due to the remaining faults in the software
3. The faults in the software are of three types, namely simple, harder and complex faults which are modelled according to their types (I. e., Type I, Type II, and Type III respectively.).
4. Software faults are of three types, which are of different severity. Depends on the severity of faults, testing effort is needed.
5. A fault occurs, an immediate delay takes place to decide the cause of failure and then remove it.
6. All faults are independent or independent and are equally detectable.
7. No new fault is introduced during the fault removal process.
8. The fault removal rate is dependent to the testing effort, by which testing time can increase
9. The fault removed in (t, t+t) is proportional to the expected number of software failures remaining to be removed.

### 2.3. Model Notations

a: Total number of initial faults in the software.
bi:Proportionality constant.
bi(t): Time dependent fault removal rate per fault for type i faults.
i(i=1, 2, 3): Types of faults (Simple, Harder and Complex).

α,p,q
Constant.
mif(t): Mean number of failures caused by type i faults by time t of the software.

### 2.4. Model Formulation

The total detected faults are given by

$m(t) = m_1(t) + m_2(t) + m_3(t)$

--1

Consider $a_1$, $a_2$, and $a_3$ to be simple, complex and hard types of faults in a system and ($a_1+a_2+a_3=a$).

**Type I (Simple type of fault):**

The simple type of fault is modelled as follows:

$$\frac{d}{dt}m_1(t) = b_1(t)(a_1(t)-m_1(t)) \quad \text{--2}$$

$$a_1(t)= a_1 e^{\propto t} \quad \text{--3}$$

$$b_1(t) = b_1 \quad \text{--4}$$

Solving the above differential equation with initial conditions $m_1(0) = 0$, we get

$$m_1(t) = \frac{a_1 b_1}{\propto + b_1}[e^{\propto(t)} - e^{-b_1(t)}] \quad \text{--5}$$

**Type II (Hard type of fault):**

At this point the faults are thought to require more testing effort i.e.,The testing members have to pass more time studying the case of failure and however it requires more time to remove them. This type of fault is modelled as follows:

$$\frac{d}{dt}m_{2f}(t) = b_2(t)(a_2(t)-m_{2f}(t)) \quad \text{--6}$$

$$\frac{d}{dt}m_2(t) = b_2(t)(m_{2f}(t)- m_2(t)) \quad \text{--7}$$

$$a_2(t)= a_2 e^{\propto t} \quad \text{--8}$$

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

$b_2(t) = b_2$ --9

Solving the above differential equations with initial conditions $m_{2f}(0) = 0$ and $m_2(0) = 0$, we get

$m_2(t) = \frac{b_2{}^2 a_2}{(\propto+b_2)^2}[e^{\propto t} - (t(b_2+\propto)+1)e^{-b_2 t}]$ --10

**Type III (Complex type of Fault):**

The complex type of fault is modelled as follows:

$\frac{d}{dt}m_{3f}(t) = b_3(t)(a_3(t)-m_{3f}(t))$ --11

$\frac{d}{dt}m_3(t) = b_3(t)(m_{3f}(t)- m_3(t))$ --12

$a_3(t)= a_3 e^{\propto t}$ --13

$b_3(t) = b_3$ --14

Solving the above differential equations with initial conditions $m_{3f}(0) = 0$, and $m_3(0) = 0$, we get

$m_3(t) = \frac{b_3{}^2 a_3}{(\propto+b_3)^2}[e^{\propto t} - (t(b_3+\propto)+1)e^{-b_3 t}]$ --15

The mean value functions given in equations 5, 10 and 16. Thus the mean value function is as:

$m(t) = m_1(t) + m_2(t) + m_3(t)$ --16

Considering $b_1 = b_2 = b_3 = b$, $a_2 = ap$, $a_3 = aq$, and $a_1 = a(1-(p+q))$ in the above equation, we get the mean value function as:

$m(t) = \frac{a\,b}{\alpha+b}[e^{\alpha(t)}\left(1 - \frac{p+q}{\alpha+b}\right) - (1 - (p + q)\frac{\alpha-bt(\alpha+b)}{\alpha+b})e^{-bt}]$ --17

A dataset used in Kapil Sharma et al.[10] shown in table 2 is used in this study. This dataset shows defects taken from software releases at Tandem Computers Company. In this paper MSE and SSE comparison criteria are applied.

## III. Parameter Estimation

To support the model applicability both the parameter estimation and model validation are the necessary aspects.The mathematical equation of the proposed SRGM is non-linear. Nevertheless, it is hard to discover the answer for a nonlinear model using Least Square Method and requires numerical algorithm to resolve it. To overcome this problem we use Statistical Software Package such as SPSS. SPSS is a statistical package for social sciences. To estimate the parameters of the proposed model, a Least Square method (Non-linear regression method) is used. Non-linear regression method finds the relationship between the dependent and independent variable. Non-linear regression can estimate models with arbitrary relationships between autonomous and dependent variables.

## IV. Comparison Criteria

The operation of the SRGM is calculated by its ability to accommodate the past system failure data set. Here the Goodness of Fit comparison is used.

### 4.1. Goodness of Fit

The term goodness of fit is used in two different contexts, in one context it denotes the question if sample of information comes from a population with a specific distribution. In another context it denotes the question of "How good does mathematical model fit to the data."

### a) Mean Square Fitting Error (MSE)

The model under the comparison is used to simulate the fault data, the difference between the expected values $m(t_i)$, and the observed data $Y_i$ is measured by MSE as follows:

$MSE = \sum_{i=1}^{k}\frac{(m(t_i)-y_i)^2}{k}$ --18

Where "k" is the number of observations. The lower the MSE indicates a less fitting error, thus better goodness of fit.

### b) Sum of Squared Errors (SSE)

The SSE is a mathematical approach to determining the scattering of data points; found by squaring the length between each data point and the line of best fit and then summing all of the squares.The sum of the squared errors, SSE, is defined as follows:

$SSE = \sum_{i=1}^{N}(Yi - \hat{Y}i)^2$ -- 19

Where:

$Yi$ is the actual observations time series

$\hat{Y}i$ is the estimated or forecasted time series

## V. Model Validation

To validate as well to determine the software reliability growth of a proposed model, it has been tried out on testing data collected from Tandem Computers [10]. The proposed model has been compared with the NHPP of Goel Okumoto [11] and Kapur et al. [13]. The result is shown in tables 1 below. After observing the goodness of fit, the values of MSE and SSE are smaller than the values of others Software reliability growth models. Generally the performance of the proposed model is fairly good. The figures of observed values and estimated values of the dataset are depicted in fig. 1-2.

Table 1: Results of proposed model

| Model | Parameter estimation | | | | | Comparison criteria | |
|---|---|---|---|---|---|---|---|
| | a | b | α | p | q | MSE | SSE |
| Proposed model | 42.583 | .054 | 0 | 0 | 0 | 61.07337 | 1221.4673 |
| Goel Okumoto [11] | 69.450 | .057 | - | - | - | 171.43122 | 3428.62443 |
| Kapur et al. [13] | 69.643 | .056 | 0 | 0 | 0 | 172.17645 | 3503.34621 |

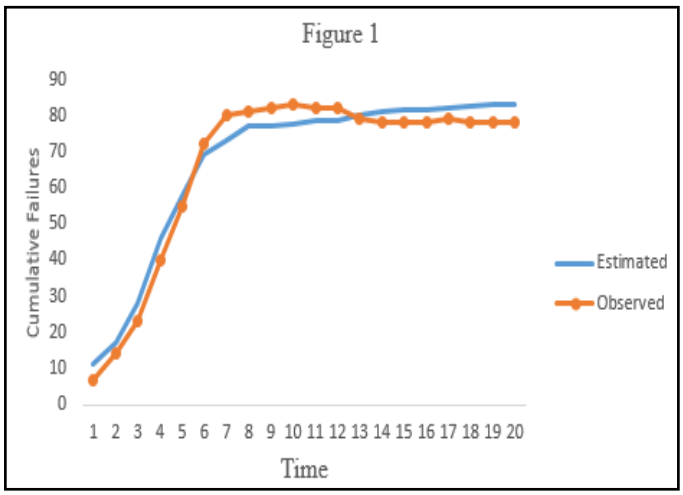


Fig. 1: Goodness-of fit between observed and (Kapur) estimate curve

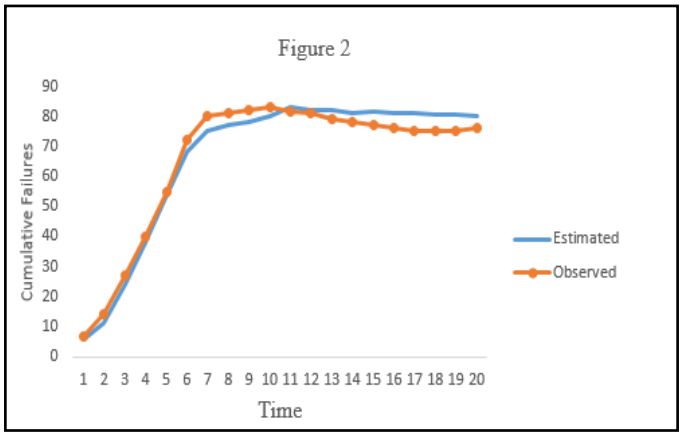


Fig. 2: Goodness-of fit between observed and (Kapur) estimate curve

The results of proposed model shown in the above table are comparatively satisfactorily encouraging as compared to the other SRGMs in the literature.

## VI. Conclusion

In this paper, an SRGM is proposed which assumes the three different processes, namelyfailure, observation, fault isolation and fault removal are considered during the testing Phase. The time delay between the failure, observation and subsequent removal is taken over to represent the severity of the defects.The presences of faults are removed according to the time needed to remove them. The simple type of fault is modelled by Yamada 1984, the complex type of fault is modelled by type II of the proposed model and harder type of faults is modelled by the type IIIof the proposed model.The proposed model is Validated and compared with actual software test data sets and well-documented SRGMs in the literature. The performance of the proposed model was encouraging.

Table 2: Tandem Computer Software Failure

| Weeks | C P U Hours | Defects | Weeks | C P U Hours | Defects |
|---|---|---|---|---|---|
| 1 | 519 | 16 | 11 | 6539 | 81 |
| 2 | 968 | 24 | 12 | 7083 | 86 |
| 3 | 1430 | 27 | 13 | 7487 | 90 |
| 4 | 1893 | 33 | 14 | 7846 | 93 |
| 5 | 2490 | 41 | 15 | 8205 | 96 |
| 6 | 3058 | 49 | 16 | 8564 | 98 |
| 7 | 3625 | 54 | 17 | 8923 | 99 |
| 8 | 4422 | 58 | 18 | 9282 | 100 |
| 9 | 5218 | 69 | 19 | 9641 | 100 |
| 10 | 5823 | 75 | 20 | 10000 | 100 |

## References

[1]. *Musa, John., "A Theory of Software Reliability and Its Application," IEEE Trans. on Soft. Eng., pp 312-327, 1975.*
[2]. *Articles on "System & Software Reliability", by CSE300 of advance software engineering Nov. 2005*
[3]. *Kimura, Yamada, Oaski, "Software Reliability Assessment for an Exponential S-Shaped Reliability Growth Phenomenon." Computer and Mathematics with Application, pp 71- 78, 1992*
[4]. *Yamada, Oask and Narihisa, "A software reliability growth model with two type of errors" R.A.I.R.O, pp 87-104, 1985*
[5]. *Kareer, Kapur, Grover, "S-shaped software reliability growth model with two types of errors." Microelectronics and reliability, pp 1085-1090, 1990.*
[6]. *Yamada, Ohba, Oask, "S-shaped software reliability growth model and their applications" IEEE Trans. On Reliability, pp169-175, 1984.*
[7]. *Kapur, Garg, and Kumar, "Contributions to Hardware and Software Reliability," World Scientific, Singapore, 1999.*
[8]. *Kapur, Younes, and Agarwal," Generalized Erlange Software Reliability Growth Model," ASOR Bulletin, pp 5-11, 1995.*
[9]. *Yamada, Ohba, Oask, "S-shaped software reliability growth model and their applications" IEEE Trans. On Reliability, pp169-175, 1984.*
[10]. *Kapil, Rakesh, Nagpal and Garg, "Selection of Optimal Software Reliability Growth Models Using a Distance Based Approach," IEEE Trans. On Reliability, pp 266-277, 2010.*
[11]. *Goel and Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," IEEE Transactions on Reliability, pp. 206–211, 1979.*
[12]. *Rafi, Akthar, Software reliability growth model with Gompertz TEF and optimal release time determination by improving the test efficiency, Int. J. Comput. Appl. pp. 34–43. 2010*
[13]. *Kapur, Goswami, Shatnawi and Ompal, "A general software reliability growth model with logistic learning functions," Varahmihir Journal of Computer & Information Science, pp 91-104, 2006.*