International Journal of Scientific & Engineering Research, Volume 4, Issue 6, June 2013
ISSN 2229-5518

Study on Two Dimensional Three stage Software Reliability Growth Model

Rashmi Upadhyay, Prashant Johri

Abstract— Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment. Software release time is a crucial decision to make and has been extensively studied in the literature. A number of models have been proposed in last two decades to discuss the software release time problem. From the earlier studies, it has been observed that more and more faults are detected as testing grows and as additional (testing) effort are used. In this paper, we reviewed the existing NHPP models for fault removal and proposed three stage software reliability growth model for minimizing the faults using Cobb Douglas production function for accumulating the effect of testing time and effort. The total fault removal phenomenon is the superposition of the three processes. The model has been validated, evaluated and compared with other existing NHPP models by applying them on actual failure data sets cited from real software development projects. The results show that the proposed model provides improved goodness of fit for software failure data and estimate results in SPSS and forecasting method is also proposed for the models.

Index Terms— Forecasting, Non homogeneous Poisson Process (NHPP), Software Reliability, Software Reliability Growth Model (SRGM), Statistical Package for Social Sciences(SPSS), Testing Effort.

1 INTRODUCTION

Software plays a critical role not only in scientific and business related enterprises, but also in daily life where it helps to run devices such as phones, cars, and television sets etc. Although advances have been made towards the production of defect free software, but this can be a costly, time consuming process. To effectively manage their budgets, managers require accurate information about the software development process and how the software reliability grows during this process. It is anticipated that software is made more reliable with the effect of this process, and can be modeled through the use of Software Reliability Growth Models (SRGMs).

SRGMs are applicable to the later stages of testing process. They can provide very useful information about how to improve reliability. Some important metrics, such as the number of initial faults, failure intensity, reliability within a specific time period, number of remaining faults, mean time between failures (MTBF), and mean time to failure (MTTF), can be easily determined through SRGMs. However, choosing a good model that can be used to explain the current and past failure behavior most adequately is very important. From our study, we find that many authors considered a Non-homogeneous Poisson Process (NHPP) as a stochastic process to describe the fault process. Ideally, these models provide a means of characterizing the development process, and enable software reliability practitioners to make predictions about the expected future reliability of software under development. Software development is a time consuming, costly process with high quality targets.

- *Rashmi Upadhyay is currently pursuing M.Tech Computer Science and Engineering, School of computing science and engineering, Galgotias University,Uttar Pradesh,India.E-mail: rurashmiupadhyay1@mail.com*
- *Prashant Johri is currently working with School of computing science and engineering, Galgotias University,Uttar Pradesh,India E-mail: johri.prashant@gmail.com*

In this paper, We develop a model for two dimensional software reliability modeling and its failure process, which is a relatively unexplored area in the field of software reliability. Development of the software process for a firm is dependent on three major assets: time, effort, and tangible resources. All assets are limited, and precious. The software development process consists of requirements and design, coding, testing, and maintenance phases. Though in building quality software each phase plays its role, it is the testing phase which is always marked as the most crucial phase of the development process. In the last phase of software development process, the testing is carried out to detect and fix software faults introduced by human work, prior to its release for the operational use. The software faults that cannot be detected and fixed remain in the released software system after the testing phase. Thus, if a software failure occurs during the operational phase, then a computer system stops working and it may cause serious damage to our daily life. If the length of software testing is long, we can remove many software faults in the system and its reliability increases. However, it leads to increase the testing cost and to delay software delivery. In addition to modeling the software fault detection process, The total fault removal process is the superposition of the two NHPP. The main assumption of this model is that the simple faults removed immediately on their detection (the time between the fault detection and removal is negligible), while the hard faults need more cause analysis and consequently require more time to be removed. Kapur et al [2, 3] proposed an SRGM with three types of fault. For each type, the FRR per remaining faults is assumed to be time independent. The first type is modeled by an Exponentional model of Goel and Okumoto [1]. The second type is modeled by Delayed S-shaped model of Yamada et al. [5]. The third type is modeled by three stage Erlang model proposed by Khoshogoftaar [4]. Later they extended their model to cater for more types of faults [3]. In their model autors ignored the role of the learning process during

the testing phase by not accounting for the experience gained with the progress of software testing. The proposed model in this paper considers learning of the fault removal team by incorporating time dependent logistic learning function during the removal phase as it is expected the learning will grow with time. To cater the need of high exactitude of software reliability we require a software reliability growth model which supply not only the testing time but also the testing effort. For this we develop a two dimensional software reliability growth model integrating the joint effect of testing time and testing effort on the number of faults removed in the software.

The Proposed three stage model is validated with using two datasets DS-II and MUSA-413 and compares with other two models i.e. Goel Okumoto model and yamda model and with comparison table models has been proposed. The model considers that software system consists of a finite number of components and takes into account different types of errors, the time lag between the failure and fault removal processes. We have also presented numerical examples based on real data set obtained from real software development project. The results of the proposed model are very encouraging when compared with other model developed under similar environment

The paper is organized as follows. In Section 2, Literarture review is given. In Section 3, Software reliability modeling wit basic and proposed model framework is presented. The parameter estimation results and data validation of the formulated model is shown In Section 4, In Section 5 model evaluation is presented. Finally, conclusions and scope for future research are penned in Section 6.

2 Literature Review

According to the ANSI definition [6], "Software reliability is the probability of failure-free software operation for a specified period of time in a specified environment". An SRGM provides a mathematical relationship between time span of testing or using the software and the cumulative number of faults detected. It is used to assess the reliability of the software during testing and operational phases

The reliability of any software relies on testing phase. The testing phase focuses on detecting and correcting the faults in the software. It has been observed that more and more faults are detected as testing grows and as additional (testing) effort are used. Testing phase of software development life cycle increases the reliability of software using the tools of Software Reliability Engineering.The software reliability growth model (SRGM) describes the relationship between the cumulative number of detected software errors and the time span of software testing.

The software faults are categorized into three types according to the number of stages needed to remove them. The fault is classified as simple if the time delay between failure observation, fault isolation and fault removal is negligible. For the simple faults, the fault removal phenomenon is modeled by the exponential model of Goel and Okumoto [1] with the logistic learning function. If there is a time delay the fault is classified as a hard fault. For the hard faults, the fault removal phenomenon is modeled similar to Delayed S-shaped model of Yamada et al. [7] with the logistic learning function during the removal phase as it is expected the learning will grow with time. If the removal of a fault after it is isolation involves even a greater time delay, it is classified as a complex fault. For the complex faults, the fault removal phenomenon is modeled by three-stage Erlang model proposed by Khoshogoftaar [6] with the logistic learning function. Models due to Bittanti *et al.* [13], and Kapur & Garg [11] are examples of flexible models. Later, Kapur *et al.* [12] proposed an SRGM with three types of faults. The first type was modeled by an Exponential model. The second type was modeled by a delayed S-shaped model. The third type is modeled by a three-stage Erlang model. The total removal phenomenon is again modeled by the superposition of the three SRGMs.Pham *et al.* [19] proposed a general imperfect software debugging model with an S shaped fault-detection rate. Recently, Kapur *et al.* [17] proposed a unified modeling framework for developing software reliability growth models under imperfect debugging. A number of models have been derived using this framework. Recently, Inoue *et al.* [10] also proposed a two-dimensional software reliability growth model with a change-point that described an actual phenomenon being related to the software reliability growth process.

Kapur *et al.* [9] also proposed a two dimensional modeling framework which was applied in determining optimal allocation of testing time and resources simultaneously to a modular software system. In 1991, Kapur and Garg [16] formulated release policies incorporating the effect of testing resource expenditure for an exponential SRGM under the added assumption that testing resource curves are described by either exponential, Rayleigh, or Weibull curves. Huang and Lyu [2] proposed an SRGM with a generalized testing effort function, and studied optimal release policies based on cost and reliability considering testing effort and efficiency. In 2007, Kapur *et al.* [14] proposed an SRGM with two types of imperfect debugging, and determined the optimal release time of the software

In this paper, we develop a three stage model which incorporates the combined effect of testing time and effort to remove the faults lying dormant in the software.The model developed is based on the Cobb Douglas production function [18]. Recently, two dimensional software reliability models have been developed to appraise the software quantitatively. For solving software reliability problem the software engineer innovating a two dimensional model at the place of one dimensional since in one dimensional analysis the object variable is dependent on one basic variable although the object takes on many different roles based upon its dependence on various other factors. Two dimensional models are used to capture the joint effect of testing time and testing effort on the number of faults removed in the software. In economics, the Cobb-Douglas functional form of production functions is extensively used to personify the relationship of an output to inputs.

3 Software Reliability Modelling

3.1 Model Development

The proposed model of this paper is based on categorizing the

faults in the software according to their removal complexity. The removal complexity is assumed to be proportional to the amount of time required to remove the fault. This complexity / time lag is represented by the number of stages required to remove the fault after the failure observation / fault isolation (with delay between the stages). This approach allows the proposed model to be flexible and simple.

The model assumes that the testing phase consists of three processes namely, failure observation, fault isolation and fault removal.

Basic Model Assumptions

1. Failure observation / fault removal phenomenon is modeled by NHPP [2].
2. Software is subject to failures during execution caused by faults remaining in the software
3. The faults existing in the software are of three types: simple, hard, and complex faults.
4. Each time a failure is observed, an immediate (delayed) effort takes place to decide the cause of the failure and to remove it.
5. During the fault isolation / removal, no new fault is introduced into the software.
6. Failure rate of the software is equally affected by faults remaining in the software.

The time dependent behavior of fault removal process is explained by a Software Reliability Growth Model (SRGM). In literature, several SRGMs have been proposed to measure the reliability during the testing phase. Most of these can be categorized under Non Homogeneous Poisson Process (NHPP) models. The assumption that governs these models is, 'the software failure occurs at random times during testing caused by faults lying dormant in software.' And, for modeling the software fault detection phenomenon, counting process $\{N(t);t \geq 0\}$ is defined which represents the cumulative number of software faults detected by testing time t. The SRGM based on NHPP is formulated as:

$$Pr\{N(t)=n\} = \frac{m(t)^n \cdot \exp(-m(t))}{n!}, \quad n = 0,1,2,...$$

Where m (t) is the mean value function of the counting process N (t).

The model proposed in this paper incorporates the effect of these two vital factors simultaneously using a Cobb Douglas production function. The Cobb-Douglas function presents a simplified outlook of the economy in which production output is obtained by the amount of labor occupied, and the amount of capital invested. While there are many factors influencing economic performance, their model demonstrated remarkable accuracy. The mathematical form of the production function is specified as

$$Y = AL^{v}K^{1-v}$$

Where, Y is the total production (the monetary value of all goods produced in a year), L is the labor input, K is the capital input, A is total factor productivity, and v is the elasticity of labor. The value of v is constant, and is determined by available technology.

To cater the combined effect of testing time and effort, we use the Cobb-Douglas production function [18] of the form

$\tau \cong s^r t^{1-r} \qquad 0 \leq r \leq 1$

Let $\{N(s,u), s \geq 0, u \geq 0\}$ be a two-dimensional stochastic process representing the cumulative number of software failures by time s and testing coverage u. A two-dimensional NHPP with a mean value function m (s, u) is formulated as:-

$$\Pr(N(s,u)=n) = \frac{(m(s,u))^n}{n!}\exp((-m(s,u)) \quad , \text{n=0,1,2…}$$

Notations:

a Initial number of faults.
b Fault detection rate per remaining fault.
l Logistic learning factor.
t Testing time.
s Effort.
r Resource elasticity to testing time.
m(s,t) Cumulative number of faults removed by time t and with using effort s.

3.2 Modelling framework

3.2.1 GO Model Modelling Framework

GO model were used the combined effect of time and effort by reperesenting the rate of change of the cumulative number of faults is given as

$$m_r'(\tau) = \frac{b}{1+l*\exp(-b\tau)}(a - m_r(\tau)) \qquad (1)$$

Integrating the above equation with initial condition $m(\tau = 0=0$, and using (1), we get

$$m(s,t) = \frac{a(1-\exp(-b\tau))}{1+l*\exp(-b\tau)}$$

Replace $\tau \cong s^r t^{1-r}$ and we get this equation.

$$m(s,t) = \frac{a(1-\exp(-bs^r t^{1-r}))}{1+l*\exp(-bs^r t^{1-r})} \qquad (2)$$

3.2.2 Two Stage model Modelling framework

In Two stage model,we combined effect of time and effort by reperesenting the rate of change of the cumulative number of faults is given as

$$m_f'(\tau) = b(a - m_f(\tau)) \qquad (3)$$

$$m_r'(\tau) = \frac{b}{1+l*\exp(-b\tau)}(m_f(\tau) - m_r(\tau)) \qquad (4)$$

Integrating the above equation with initial condition $m(\tau = 0=0$, and using (3), we get

$$m(s,t) = \frac{a(1-\exp(-b\tau)(1+b\tau)}{1+l*\exp(-b\tau)}$$

Replace $\tau \cong s^r t^{1-r}$ and we get this equation.

$$m(s,t) = \frac{a(1-\exp(-bs^r t^{1-r})(1+bs^r t^{1-r})}{1+l*\exp(-bs^r t^{1-r})} \quad (5)$$

3.3 Proposed model Modelling framework

The combined effect of testing time and effort, we use the Cobb-Douglas production function.

Under the above assumptions, the differential equation representing the rate of change of the cumulative number of faults detected w.r.t. to the combined effect of time and effort is given as

$$m_f'(\tau) = b(a - m_f(\tau)) \quad (6)$$

$$m_i'(\tau) = b(m_f(\tau) - m_i(\tau)) \quad (7)$$

$$m_r'(\tau) = \frac{b}{1+l*\exp(-b\tau)}(m_i(\tau) - m_r(\tau)) \quad (8)$$

Integrating the above equation with initial condition $m(\tau = 0=0$, and using (6), we get

$$m(s,t) = \frac{a(1-\exp(-b\tau)(1+b\tau+b^2\tau^2)}{1+l*\exp(-b\tau)}$$

Replace $\tau \cong s^r t^{1-r}$ and we get this equation.
Replace $\tau \cong s^r t^{1-r}$ and we get this equation.

$$m(s,t) = \frac{a(1-\exp(-bs^r t^{1-r})(1+bs^r t^{1-r}+b^2(s^r t^{1-r})^2))}{1+l*\exp(-bs^r t^{1-r})}$$

(9)

In the above model, if ,$l = 0$ the above model reduces to a 2-dimensional exponential model. For other values, it is Sshaped. If , $r = 0$ then it becomes a flexible time dependent SRGM proposed by Kapur et al [15].

4 ESTIMATION PARAMETERS

The SRGMs only become useful if it is possible to estimate their parameters. To validate the model described in Section 3.1, parameter estimation on actual software failure data is done.

In our study, we have used the Statistical Package for Social Sciences (SPSS). For the estimation of the parameters of the proposed model, nonlinear regression (NLR) modules of SPSS have been used to estimate the unknown parameters of the proposed model given in equation (5). Since the data set used is given in the form of pairs (t_i,xi,si)(i=1,2,…,k), where x_i is the cumulative number of faults removed 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 and s_i is the cumulative execution time. The SPSS modules use the iterative estimation algorithms namely sequential quadratic programming and Levenberg-Marquardt method to find the least square estimates of the parameters.

4.1. Model validation

To check the validity of the proposed model to describe the software reliability growth, it has been tested on two Data Sets DS-II and Musa-413 .

The data set DS-II have 1301 cumulative faults and and working for 35 weeks and use 1846.92 cumulative CPU hours for remove these faults and the data set MUSA-413 have 136 cumulative faults and and working for 21 weeks and use 25.26327 cumulative CPU hours for remove these faults.

In this paper we assume that the effort data set were better goodness of fit in three stage model as compared with GO model and 2 stage model. Parameters of SRGM were estimated using SPSS.

TABLE 1
PARAMETER ESTIMATION DS-II

	a	b	r	l
Basic				
GO	1350	.010	.0623	1.9
2stage	1350	.066	.209	1.9
Proposed				
3 satge	1403.395	.166	0.020	1.0

TABLE 2
PARAMETER ESTIMATION MUSA-413

	a	b	r	l
Basic				
GO	135.00	0.224	0.399	4.872
2stage	140.00	0.258	0.178	9.874
Proposed				
3 satge	175.46	0.203	0.128	1.0

The parameter estimation results are given in the table 1 and

table 2 respectively for DS-I and MUSA 413.
The performance of an SRGM is judged by its ability to fit the past software failure / fault removal data (goodness of fit) and to predict satisfactorily the forecasted method for the validated data for analysis of forecasting.
In DS – II and MUSA – 413 the failure curves are exponential but again the proposed SRGM gives very good fits proving its flexibility. The plot of actual values and the estimated values of fault removal and faults remaining for the two data sets are depicted in figures 1 and 2 respectively.

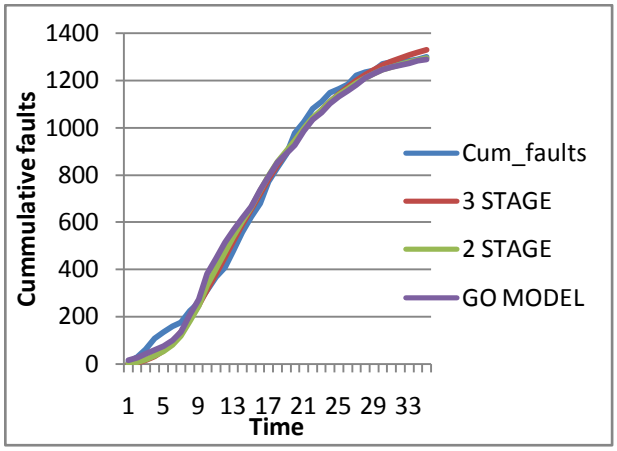

Fig. 1. for DS-II parameter estimation graph

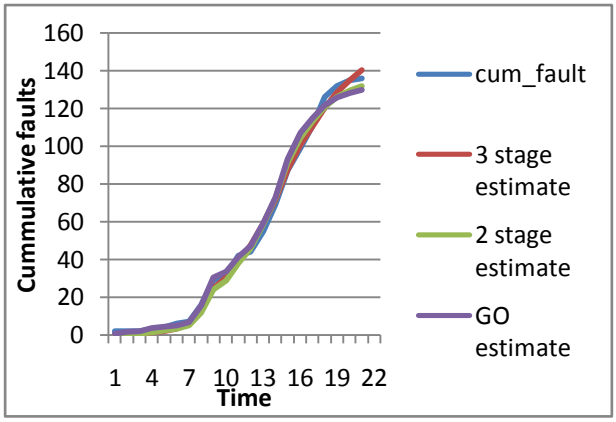

Fig. 2. for MUSA-413 parameter estimation graph

5 Model Evaluation

The performance of SRGM is judged by their ability to fit the past software failure / fault data (goodness of fit) and to predict the forecasted method .
Therefore, we use two types of comparison criteria:
i)The Goodness of Fit Criterion.
ii)Forecasted method.

5.1 The Goodness of Fit Criterion

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

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

Where k is the number of observations. Lower value of MSE indicates less fitting error, thus better goodness of fit.

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

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

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

3. Coefficient of ADJ R^2 :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:

$$\text{ADJ } R^2 = 1 - \frac{(residual \quad SS)/(n-k)}{(corrected \quad SS)/(n-d)}$$

Where n is the number of weeks of a software working for removing the faults ,k is the number of independents variable in a data set and d is the number of dependent variable in a data set. It also ranges in value from 0 to 1. Small values indicate that the model does not fit the data well.

4.Sum of residual square error (SSE): This coefficients is a residual Sum of Squares(SS). Smaller the value of SSE is indicates the better goodness fit of data.

In other words, we evaluate the performance of the models using M*SE* ,SSE , R^2, and ADJ R2 metrics. The smaller the value of the metric the better the model fits relative to other models run on the same Data Set.

The proposed model is compared with the other models. The results are produced in table–. The proposed model is better for the given data sets under the above mentioned criteria.

TABLE 3
COMPARISON TABLE FOR DS-II

	R^2	ADJ R^2	SSE	MSE
Basic				
GO	.991	0.9075	66905.605	2158
2Stage	.993	0.9267	52989.934	1709
Proposed				
3 stage	.995	0.9439	40522.503	1307

TABLE 4
COMPARISON TABLE FOR MUSA-413

	R^2	ADJ R^2	SSE	MSE
Basic				
GO	.996	0.9957	223.792	13.164
2Stage	.995	0.9945	282.618	16.625
Proposed				
3 stage	.997	0.9969	155.120	9.125

5.2 Forecasted method

The forecast error is the difference between the actual value and the forecast value for the corresponding period.

$E_t = Y_t - F_t$

where E is the forecast error at period t, Y is the actual value at period t, and F is the forecast for period t.

i)Mean Absolute Deviation (MAD):

$$MAD = \frac{\sum_{t=1}^{N} |E_t|}{N}$$

ii)Root Mean squared error (RMSE):

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N} |E_t|}{N}}$$

we evaluate the performance of the models using MSE ,SSE , R^2, and ADJ R2 metrics for fitted data and and evaluate the performance of the models using MAD, RMSE of 20% data which is forecasted. The smaller the value of the metric the better the model fits relative to other models run on the same Data Set.

TABLE 5
AFTER FORECASTING THE PARAMETER ESTIMATION TABLE OF DS-II

	a	b	r	l
Basic				
GO	1400	.012	.592	2.9
2stage	1516	.050	.230	1.0
Proposed				
3 satge	1425.0	.164	0.020	1.0

TABLE 6
COMPARISON TABLE OF DS-II AFTER FORECASTING

Models	Fitted data(80%)				Forecasted(20%)	
	R2	ADJ R^2	SSE	MSE	MAD	RMSE
Basic						
GO	.992	0.9918	38161.196	1526	32.72	32.37
2Stage	.992	0.9915	40208.502	1675	72.87	72.11
Proposed						
3 stage	.993	0.9929	35381.914	1474	27.41	31.30

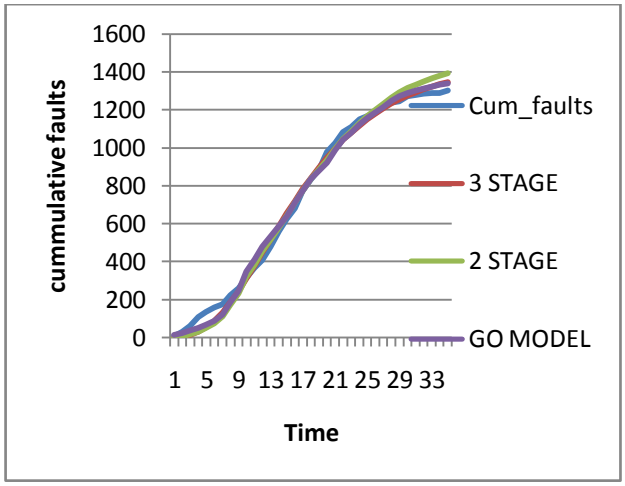

Fig. 3. Forecasted graph for DS-II

TABLE 7
AFTER FORECASTING THE PARAMETER ESTIMATION TABLE OF MUSA-413

	a	b	r	l
Basic				
GO	108.00	0.294	0.502	3.712
2stage	120.00	0.243	0.312	1.626
Proposed				
3 satge	180.00	0.195	0.112	1.0

TABLE 8
COMPARISON TABLE OF MUSA-413 AFTER FORECASTING

Models	Fitted data(80%)				Forecasted(20%)	
	R2	ADJ R^2	SSE	MSE	MAD	RMSE
Basic						
GO	.995	0.9946	105.313	8.101	26.8187	27.6611
2Stage	.994	0.9937	129.173	9.936	16.7001	17.5016
Proposed						
3Stage	.996	0.9958	83.076	6.390	3.80352	4.36485

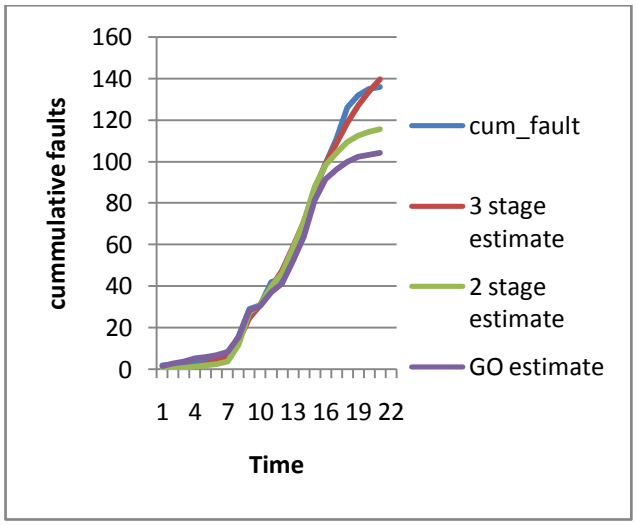

Fig. 4. Forecasted graph for MUSA-413.

6 CONCLUSION AND FULTURE SCOPE

We have developed a three stage software reliability model. This model uniquely takes into account for removing the faults in three stage i.e. fault detection, fault isolation, and fault removal. Further, the model has been developed under the assumption that the fault removal process is governed not only by testing time but also by the effort expended. A relatively unexplored area of software reliability is investigated with this development structure. The proposed three stage model is applied to a real data sets. We have also presented numerical examples based on real data set obtained from real software development project the results are also formulated with forecsted method Results clearly show that the three stage model provides better goodnees of fit as well as it is also good for forecasted method.

In future , the result estimated using SPSS is optimize and solved using a GA which minimizes the expected software cost subject to removing a minimum desired proportion of faults in the proposed model.

REFERENCES

[1] Goel AL, Okumoto K. Time dependent error detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability* **1979**; R-28(3): 206-211.

[2] Kapur PK and R.B. Garg (1992), A software reliability growth model for an error removal phenomenon, Software Engineering Journal 7, 291-294.

[3] Kapur P.K., R.B. Garg and S. Kumar (1999), Contributions to Hardware and Software Reliability, World Scientific, Singapore.

[4] Khoshogoftaar TM and Woodcock TG (1991), "Software Reliability Model Selection: A case study", Proceedings of the international symposium on software reliability Engineering, pp.183-191.

[5] Yamada S, Ohba M, and Osaki S. (1984) S-shaped software reliability growth models and their applications, IEEE Transactions on Reliability, 1984; R-33: 169-175

[6] Lyu M.R. (1996), "Handbook of Software Reliability Engineering ", Mcgraw Hill.

[7] Yamada S, Ohba M, and Osaki S. S-Shaped Software Reliability Growth Models and Their Applications. *IEEE Transactions on Reliability*, **1984**; R-33: 169-175.

[8] S. Inoue and S. Yamada, "Two-dimensional software reliability measurement technologies," in *In the proceedings of IEEE IEEM*, 2009.

[9] P. K. Kapur, A. G. Aggarwal, and G. Kaur, "Simultaneous allocation of testing time and resources for a modular software," *International Journal of Systems Assurance Engineering and Management*, vol. 1, no. 4, pp. 351–361, 2010.

[10] S. Inoue, K. Fukuma, and S. Yamada, "Two-dimensional change-point modeling for software reliability assessment," *International Journal of Reliability*, vol. 17, no. 6, pp. 531–542, December 2010, Quality and Safety Engineering.

[11] P. K. Kapur and R. B. Garg, "A software reliability growth model for an error removal phenomenon," *Software Engineering Journal*, vol. 7, no. 4, pp. 291–294, 1992.

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

[13] S. Bittanti, P. Bolzern, E. Pedrotti, and R. Scattolini, "A flexible modeling approach for software reliability growth," in *Software Reliability*

Modelling and Identification, G. Goos and J. Harmanis, Eds. Berlin, Germany: Springer-Verlag, 1988, pp. 101–140.

[14] P. K. Kapur, H. Pham, A. Gupta, and P. C. Jha, *Software Reliability Assessment with OR Applications*. London, U.K.: Springer, 2011.

[15] P. K. Kapur, R. B. Garg, A. G. Aggarwal, and A. Tandon, "Two dimensional flexible software reliability growth model and related release policy," in *Proceedings of the 4th National Conference; INDIACom-2010*, February 25–26, 2010.

[16] P. K. Kapurand and R. B. Garg, "Optimal release policies for software systems with testing effort," *International J. System Science*, vol. 22, no. 9, pp. 1563–1571, 1991.

[17] P. K. Kapur, H. Pham, S. Anand, and K. Yadav, "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.

[18] P. K. Kapur, H. Pham, *Fellow, IEEE*, Anu G. Aggarwal, and Gurjeet Kaur, "Two Dimensional Multi-Release Software Reliability Modeling and Optimal Release Planning", IEEE TRANSACTIONS ON RELIABILITY, VOL. 61, NO. 3, SEPTEMBER 2012.

[19] H. Pham, L. Normann, and X. Zhang, "A General imperfect software debugging model with S-shaped fault detection rate," *IEEE Trans. Reliability*, vol. 48, no. 2, pp. 169–175, 1999.