EDICIONES Universidad Cooperativa de Colombia

Ingeniería Solidaria
ISSN (en línea): 2357-6014
Vol. 14, No. 25 [special issue], May 2018
Artículo de investigación / Research article
doi: https://doi.org/10.16925/.v14i0.2229

2-Dimensional Multi-Release Software Reliability Modeling considering Fault Reduction Factor under Imperfect Debugging

Diseño de fiabilidad bidimensional del *software* de múltiples lanzamientos con base en el factor de reducción de fallas en la depuración imperfecta

Sameer Anand[1]✉, Vibha Verma[2], Anu Gupta Aggarwal[3]

[1] *University of Delhi, Delhi, India*
[2] *University of Delhi, Delhi, India*
[3] *University of Delhi, Delhi, India*

✉ College of Business Studies, psp Area IV, Dr. K.N. Katju Marg, Sector 16, Rohini, New Delhi, Delhi 110089, India. Email: sameeranand@sscbsdu.ac.in

Received on: October 23th, 2017 **Accepted on:** February 22th, 2018 **Available online:** May 1st, 2018

How to cite this article: S. Anand, V. Verma and A. Gupta-Aggarwal, "2-Dimensional Multi-Release Software Reliability Modeling considering Fault Reduction Factor under imperfect debugging", *Ing. Sol.*, vol. 14, no. 25 (Special issue), pp. 12, May 2018. doi: https://doi.org/10.16925/.v14i0.2229

Abstract

Introduction: The present research was conducted at the University of Delhi, India in 2017.

Methods: We develop a software reliability growth model to assess the reliability of software products released in multiple versions under limited availability of resources and time. The Fault Reduction Factor (FRF) is considered to be constant in imperfect debugging environments while the rate of fault removal is given by Delayed S-Shaped model.

Results: The proposed model has been validated on a real life four-release dataset by carrying out goodness of fit analysis. Laplace trend analysis was also conducted to judge the trend exhibited by data with respect to change in the system's reliability.

Conclusions: A number of comparison criteria have been calculated to evaluate the performance of the proposed model relative to only time-based multi-release Software Reliability Growth Model (SRGM).

Originality: In general, the number of faults removed is not the same as the number of failures experienced in given time intervals, so the inclusion of FRF in the model makes it better and more realistic. A paradigm shift has been observed in software development from single release to multi release platform.

Limitations: The proposed model can be used by software developers to take decisions regarding the release time for different versions, by either minimizing the development cost or maximizing the reliability and determining the warranty policies.

Keywords: 2-dimensional software reliability modelling, fault reduction factor, imperfect debugging, multi-release, trend analysis.

Research article

doi: https://doi.org/10.16925/.v14i0.2229

Diseño de fiabilidad bidimensional del *software* de múltiples lanzamientos con base en el factor de reducción de fallas en la depuración imperfecta

Resumen

Introducción: la presente investigación se realizó en la Universidad de Delhi, India en 2017.

Métodos: desarrollamos un modelo de crecimiento de confiabilidad de *software* para evaluar la confiabilidad de los productos de *software* lanzados en múltiples versiones bajo disponibilidad limitada de recursos y tiempo. El factor de reducción de fallas (FRF) se considera una constante en entornos de depuración imperfecta, mientras que la tasa de eliminación de fallas está dada por el modelo de forma retardada en S.

Resultados: se valida el modelo propuesto en un conjunto de datos de cuatro lanzamientos de la vida real mediante un análisis de bondad de ajuste. También se aplicó el análisis de tendencia de Laplace para juzgar la tendencia que presentan los datos con respecto al cambio en la confiabilidad del sistema.

Conclusiones: se calculó una serie de criterios de comparación para evaluar el rendimiento del modelo propuesto en relación con el modelo de crecimiento de confiabilidad del *software* (SRGM) de múltiples lanzamientos basado únicamente en el tiempo.

Originalidad: en general, el número de fallas eliminadas no es el mismo que el número de fallas experimentadas en intervalos de tiempo determinados, por lo que la inclusión de FRF en el modelo lo mejora y lo hace más realista. Se ha observado un cambio de paradigma en el desarrollo de *software*, que pasa de un lanzamiento único a una plataforma múltiples lanzamientos.

Limitaciones: los desarrolladores de *software* pueden emplear el modelo propuesto para tomar decisiones con respecto al tiempo de lanzar diferentes versiones, ya sea minimizando el costo de desarrollo o maximizando la confiabilidad y determinando las políticas de la garantía.

Palabras clave: diseño de confiabilidad de *software* bidimensional, factor de reducción de fallas, depuración imperfecta, múltiples lanzamientos, análisis de tendencias.

Design de fiabilidade bidimensional do *software* de múltiplos lançamentos tendo em conta o fator de redução de falhas na depuração imperfeita

Resumo

Introdução: esta pesquisa foi realizada na Universidade de Deli, na Índia, em 2017.

Métodos: desenvolvemos um modelo de crescimento de confiabilidade de software para avaliar a confiabilidade dos produtos de software lançados em múltiplas versões sob disponibilidade limitada de recursos e tempo. O fator de redução de falhas (FRF) é considerado uma constante em contextos de depuração imperfeita, enquanto a taxa de eliminação de falhas é dada pelo modelo de forma retardada em S.

Resultados: o modelo proposto é avaliado em um conjunto de dados de quatro lançamentos da vida real mediante uma análise de bondade de ajuste. Também foi utilizada a análise de tendência de Laplace para avaliar a tendência apresentada pelos dados com respeito à mudança na confiabilidade do sistema.

Conclusões: uma série de critérios de comparação foi calculada para avaliar o rendimento do modelo proposto em relação com o modelo de crescimento de confiabilidade do *software* (SRGM) de múltiplos lançamentos baseado unicamente no tempo.

Originalidade: em geral, o número de falhas eliminadas não é o mesmo que o número de falhas existentes em intervalos de tempo determinados, sendo assim, a inclusão do FRF no modelo o torna melhor e mais realista. Foi observada uma mudança de paradigma no desenvolvimento de *software*, que passa de um lançamento único a uma plataforma de múltiplos lançamentos.

Limitações: o modelo proposto pode ser utilizado pelos desenvolvedores de software para tomar decisões com respeito ao tempo de lançar diferentes versões, seja para minimizar o custo de desenvolvimento ou maximizar a confiabilidade e determinar as políticas de garantia.

Palavras-chave: análise de tendências, depuração imperfeita, design de confiabilidade de software bidimensional, fator de redução de falhas, múltiplos lançamentos.

1. Introduction

Quality assessment of a software product before its introduction in the market has become an essential task. The software development teams use SRGM to assess the reliability growth of software systems during testing and operational phases. SRGM are used to estimate the optimal time period for testing, number of faults dormant in the system, failure intensity, development cost, etc. Organizations like ANSI/AIAA, IBM, Motorola, Hewlett Packard, CISCO, among others, apply and recommend SRGM during testing and operational phases [1].

In the last few decades a number of NHPP based software reliability growth models have been proposed in literature (Goel and Okumoto [2, pp. 3-5]; Ohba [3, pp. 3-5]; Yamada et al. [4]; Tamura and Yamada [5]) based on different sets of assumption, e.g., perfect debugging, change point, delayed fault removal process, error generation, etc. In most of the real life software development projects, the debugging process may not be perfect because of the code's complexity, team inefficiency, huge size of software, etc., and it may lead to the generation of errors or slowdown in fault removal rate. If debugging involves error generation, then there may be an increase in the number of residual faults of the software after the debugging effort. On the other hand, if there is a slowdown in the fault detection rate (FDR) observed, a fault, upon its detection, is removed in multiple numbers of efforts but without changing the residual fault content of the software.

Another noteworthy phenomenon that may be observed in software development environment is the difference between the number of faults detected and number of faults removed during the given length of time. Musa [6, p. 5], [7, p. 5], [8, p. 5] modeled this difference by Fault Reduction Factor (FRF) which was defined as the total number of faults removed in proportion to the total number of failures experienced during testing. Musa represented FRF by a constant (Musa's basic execution time model) which can be negative, between 0 and 1 or greater than 1. According to Musa, the value of FRF depends on a number of environmental factors such as severity of faults, the type of testing techniques/tools employed, skills of the man-power involved etc.

In extant software reliability literature, FRF has been incorporated in multi-release reliability growth modeling. Rapid developments in hardware and conductive economic conditions have forced the firms to continuously innovate and upgrade their products so as to maintain their market presence and beat competition. Software companies update their products by extending the functionalities of their current product along with the new features. Some of the major reasons for the multiple versions of a software system have been highlighted in Fig. 1.

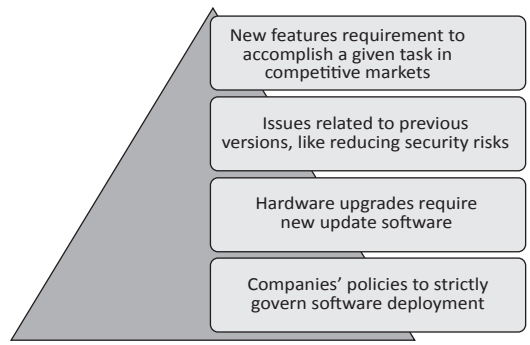

Fig. 1. Need for multiple versions of software product
Reference: the authors

Though new releases are made to keep pace with development and dynamics of customer needs, it may also result in spite of the failure rate and fault content of the software system. Therefore, the quality maintenance of a multi-release software is a critical task for the developers as they have to accomplish such development under time and resources constraints. In this paper we propose an FRF based two-dimensional Delayed S-Shaped SRGM for a multi-release software system under imperfect debugging. We have validated the proposed model on Wood's 2-D Multi-Release dataset collected for Tandem computers. Laplace [9] trend analysis on this data has been carried out to analyse if the model is appropriate to fit the data. Parameter estimation using non-linear least square estimation was done in SPSS.

This article is organized as follows: Section 1 provides an overview of the related work. Section 2 offers a detailed description of the proposed model. In section 3 the reliability trend of dataset is tested using Laplace trend analysis. The parameter estimation and model validation is found in Section 4. Finally, section 5 offers conclusions.

2. Related Work

SRGMs: Goel-Okumoto Model [2, p. 1] is one of the initial SRGMs which considered the exponential nature of the failure process and the homogeneity of the faults. Few years later Ohba [3, p. 1] proposed an inflexion S-shaped model under the assumption of two types of faults, namely independent and dependent faults. In the same year, Yamada et al. [10, pp. 4-5] developed the model to represent time delay between fault removal and failure observation.

Yamada's Delayed S-Shaped Model for single Release: This model depicts S-Shaped curves for fault removal implying improvement in the testing efficiency/learning of the team. It was presented by Yamada [10, p. 5] under the assumption that there is a time difference between fault detection and correction. At first the rate of failure increases, but there is a decrease in the failure rate later suggesting a learning phenomenon. The mean values function (MVF) for this model is:

$$m(t) = a(1 - (1 + bt)e^{-bt} \quad (1)$$

where a is the number of faults present in the software initially, b is the rate of fault removal.

Multi-Release SRGM: Release of software systems in multiple versions has been necessary to respond to the complexity of real life applications, rapid technological advancements, risk associated with the development of a new product and constraint on availability of manpower and delivery time. Releasing software with different versions helps to retain and attract potential customers. The coding done for new features development and enhancement of existing features leads to the generation of faults. Kapur et al. [11] developed SRGM for a software product with four releases to incorporate the upgrades made during each release.

The mathematical formula of MVF for each release is given as:

$$m_1(t) = a_1 F_1 \quad 0 \le t \le t_1 \quad (2)$$

$$m_2(t) = a_2 F_2(t - t_1) + a_1(1 - F_1(t_1)) \quad F_2(t - t_1)\, t_1 \le t < t_2 \quad (3)$$

$$m_3(t) = a_3 F_3(t - t_2) + a_2(1 - F_2(t_2))F_3(t - t_3) + a_1(1 - F_1(t_1))(1 - F_2(t_2))F_3(t - t_2)\, t_2 \le t \le t_3 \quad (4)$$

$$m_4(t) = a_4 F_4(t - t_3) + a_3(1 - F_3(t_3))F_4(t - t_3) + a_2(1 - F_2(t_2))(1 - F_3(t_3))F_4(t - t_3) + a_1(1 - F_1(t_1))(1 - F_2(t_2))(1 - F_3(t_3))F_4(t - t_3)\, t_3 \le t \le t_4 \quad (5)$$

where a_i is the initial fault content and $F_i(.)$ is the Failure time CDF for the *ith* release.

Two-dimensional SRGM: Most of the SRGMs have been developed considering the software failure process to be dependent on testing time (Goel et. al. [2, pp. 1, 3]; Ohba [3, pp. 1, 3]; Yamada [10, p. 5]) or testing resource consumption (Yamada et al. [12], Kapur et al. [13]). In recent years Ishii and Dohi [14] proposed a 2-D SRGM and validated their model on 2-time scales. Inoue and Yamada [15] proposed a reliability assessment method using 2-D Weibull SRGM. Kapur et al. [16, p. 6] used 2-D Multi-Release framework to release the planning of a software product.

FRF: The concept of FRF is based upon the fact that number of faults removed need not be the same as number of failures experienced in a given interval of time. Musa [6, p. 2], [7, p. 2], [8, p. 2] introduced the concept of FRF to distinguish between fault and failure. According to him FRF can be expressed as:

$$\text{FRF} = \text{Number of faults removed/ Number of failures experienced} \quad (6)$$

Under Musa's perspective, "FRF is usually positive and less than one but it can be negative or zero. It can be greater than one in the case of finding the fault that produced a particular failure that resulted in removal of other faults as well."

Hsu et al. [17] proposed SRGM by considering time variable FRF that depicts increasing, decreasing and constant patterns. The general model is defined as:

$$\frac{dm(t)}{dt} = b(t)(a - m(t)) \text{ where } b(t) = bf(t) \quad (7)$$

MVF for different patterns depicted by FRF due to the effect of environmental factors are:

a) Constant pattern

$$f(t) = f \quad 0 < f \leq 1 \text{ and } m(t) = a(1 - e^{-fbt}) \tag{8}$$

b) Increasing pattern

$$f(t) = 1 - (1 - f_0)e^{-bt} \quad \text{and}$$
$$m(t) = a\left(1 - e^{-b\left(\frac{(f_0 - 1)(1 - e^{-bt})}{b}\right)}\right) \tag{9}$$

c) Decreasing pattern

$$f(t) = f_0 e^{-bt} \text{ and } m(t) = a\left(1 - e^{-b\left(\frac{f_0(1 - e^{-bt})}{b}\right)}\right) \tag{10}$$

where and represent FRF and FDR respectively

Later on Pachauri et al. [18] described FRF as inflexion S-Shaped function for a multi-release software system. Chatterjee et al. [19] formulated the Weibull-type FRF under perfect and imperfect debugging process. Aggrawal et al. [20] proposed a multi release growth model with exponentiated Weibull FRF and change point under imperfect debugging.

Imperfect debugging: It was first proposed by Goel [21]. He extended Jelinski and Moranda Model [22] by assuming that error may be generated during the faults removal process. Kapur et al. [23] proposed a unified framework for SRGM under two types of imperfect debugging. Jain et al. [24] integrated the concept of FRF and imperfect debugging in exponential SRGM. Chatterjee et al. [25] proposed an SRGM under imperfect debugging and FRF as a Weibull function. The objective of our paper is to present a general model that can deal with all these factors simultaneously, so that the modeling done becomes more realistic and helps to fit data more accurately.

3. Proposed Modeling Framework

The reliability growth model proposed in this paper is two-dimensional in nature; it includes testing time and amount of testing resources consumed. Cobb-Douglas production function (Fig. 2) is used to represent the combined effect of time and resources on the fault removal process [16, p. 5].

The Cobb-Douglas function is mathematically given as:

$$O = CI_1^e t_2^{1-e} \tag{11}$$

where O = The output produced; $I_1 I_2$ is the amount of two inputs used; e = first input's elasticity and C is the productivity factor.

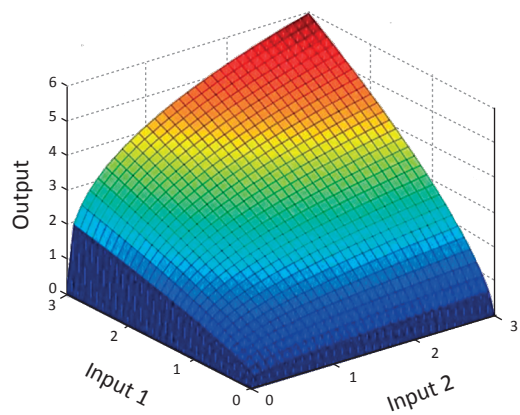

Fig . 2. Cobb-Douglas production function with two inputs
Reference: the authors

The proposed model is based on the following assumptions:

1. Fault removal process follows NHPP.
2. Each and every fault remaining in the software affects the failure rate.
3. There is a finite number of faults in the system.
4. Fault removal process is represented by Delayed S-Shaped model.
5. A fault is removed after its detection but it may lead to the generation of more faults.
6. There is a difference between the number of faults removed and the number of faults detected in a given interval of time.

7. The joint effect of time and resources on the expected number of faults removed is represented by Cobb-Douglas function given as follows:

$$\psi \cong t^{e} r^{1-e} \quad 0 \le e \le 1 \tag{12}$$

where t represents length of time spent on testing and r represents the amount of testing resource spent during testing.

Using the above mentioned assumptions we obtain the following differential equation for the change in cumulative number of faults removed:

$$m^{'}(\psi) = f\frac{b^{2}\psi}{1+b\psi}(a + lm(\psi) - m(\psi)) \tag{13}$$

where l is the rate of error generation, f is fault reduction factor, b is fault detection rate and $m(\psi)$ is the expected number of faults removed after spending t time and r amount of resources.

Solving the equation (13) using initial conditions $m(\psi) = 0$ $at\psi = 0$ we get,

$$m(\psi) = \frac{a}{1-l}\left(1 - (1+b\psi)^{f(1-l)} e^{-bf(1-l)\psi}\right) \tag{14}$$

Using (12) and (14) we get:

$$m(t,r) = \frac{a}{1-l}\left(1 - \left(1+bt^{e}r^{1-e}\right)^{f(1-l)} e^{-bf(1-l)t^{e}r^{1-e}}\right) \tag{15}$$

This can also be written as

$$m(t,r) = \frac{a}{1-l}F(t,r) \tag{16}$$

where

$$F(t,r) = \left(1 - \left(1+bt^{e}r^{1-e}\right)^{f(1-l)} e^{-bf(1-l)t^{e}r^{1-e}}\right)$$

Now we model the fault removal process for a multi-release software.

First Release: At the time of the first release, a product is introduced in the market for the first time. Before entering the market, the development team test the software rigorously so as to remove maximum possible faults from the software system. But due to market competition, which bounds the team with time and cost constraints, it becomes almost impossible for the team to remove all the faults before the first release. The faults which remain undetected even after testing are taken care off in the next version. Mathematically the faults removal for the first release may be represented as:

$$m_{1}(\psi) = a_{1}^{*}F_{1}(\psi) \quad 0 \le \psi < \psi_{1} \tag{17}$$

where $a_{1}^{*} = \frac{a_{1}}{1-l_{1}}$ and $F_{1}(\psi) = \left(1 - (1+b_{1}\psi)^{f_{1}(1-l_{1})} e^{-b_{1}f_{1}(1-l_{1})\psi}\right)$ (ψ is as given by expression (12))

Second Release: After releasing the first version, firms need to develop newer versions so as to remain competitive in the market. If a software firm does not do so, then in spite of a great response for the first version, the product may become obsolete due to technological advancements and the firm may lose its market share. The second version is launched with some additional features and enhanced functionalities. During its testing efforts are made to detect and remove the faults due to new features as well as the left over faults of the first release. The number of cumulative faults removed during this release is given as:

$$m_{2}(\psi) = (a_{2}^{*} + (a_{1}^{*} - m_{1}(\psi_{1})))F_{2}(\psi - \psi_{1}) \quad \psi_{1} \le \psi \le \psi_{2} \tag{18}$$

where $a_{2}^{*} = \frac{a_{2}}{1-l_{2}}$ and $F_{2}(\psi) = \left(1 - (1+b_{2}\psi)^{f_{2}(1-l_{2})} e^{-b_{2}f_{2}(1-l_{2})\psi}\right)$

Future Releases: In the same manner we can express the MVF for next n releases.

$$m_3(\psi) = (a_3^* + (a_2^* - m_2(\psi_2)))F_3(\psi - \psi_2) \quad \psi_2 \le \psi \le \psi_3 \tag{19}$$

$$m_4(\psi) = (a_4^* + (a_3^* - m_3(\psi_3)))F_4(\psi - \psi_3) \quad \psi_3 \le \psi \le \psi_4 \tag{20}$$

$$m_n(\psi) = (a_n^* + (a_{n-1}^* - m_{n-1}(\psi_{n-1})))F_n(\psi - \psi_{n-1}) \quad \psi_{n-1} \le \psi \le \psi_n \tag{21}$$

4. Trend Analysis

In this section we will analyse the reliability trend of data set using Laplace Trend analysis. The aim is to find if the developed SRGM is appropriate for the dataset. Trend basically refers to either increase or decrease in reliability as the testing progresses. Several methods like the graphical approach by Asher et al. [26] or the analytical method used by Kanoun [27] exist for performing trend analysis. But Laplace's trend test proposed by Asher et al. [28], Cox et al. [29], and extended by Kanoun [27] is widely used for the determination of trends. The method involves computing the Laplace factor for each time unit to observe the trend with testing increasing time. The Laplace factor is given as:

$$h(\psi) = \frac{\sum_{i=1}^{\psi}(i-1)n(i) - \frac{\psi-1}{2}\sum_{i}^{\psi} n(i)}{\sqrt{\frac{\psi^2-1}{12}\sum_{i=1}^{\psi} n(i)}} \tag{22}$$

where $n(i)$ is the failure during $\psi_i(t_i^{th}$ time unit and r_i^{th} resource consumption).

The interpretation of Laplace factor is shown in Table 1. We have calculated the Laplace factor for each of the releases and plotted this against time to observe the trend. Fig. 3 shows Laplace trend analysis plots and Table 2 shows the Laplace factor for each release at the end of the testing period. All the releases show either stability or growth in reliability i.e. due to rigorous testing and fault removal the reliability is increasing and becomes stable. This shows the human learning phenomenon and is captured by S-Shaped curves. Hence the data is accurate for the proposed SRGM, its' failure curve is S-Shaped. We have also done a graphical representation of reliability growth for each of the releases (Fig. 3). Therefore, it is appropriate to use Delayed S-Shaped Model for representing the fault removal process.

Table 1. Laplace factor Interpretation

Laplace Factor Value	Interpretation
Negative	Reliability growth
Positive	Reliability decrease
Between -2 and 2	Stable reliability

Reference: the authors

5. Model Validation

A SRGM is important only if it can be validated on actual failure data of software and its parameters

Table 2. Laplace factor for each release and its trend

Release	Testing Time	Resource Consumption	Laplace factor	Reliability Trend
1	20	10000	-3.96947	Growth
2	19	10272	-2.34446	Growth
3	12	5053	-1.64044	Stable
4	19	11305	-1.38162	Stable

Reference: the authors

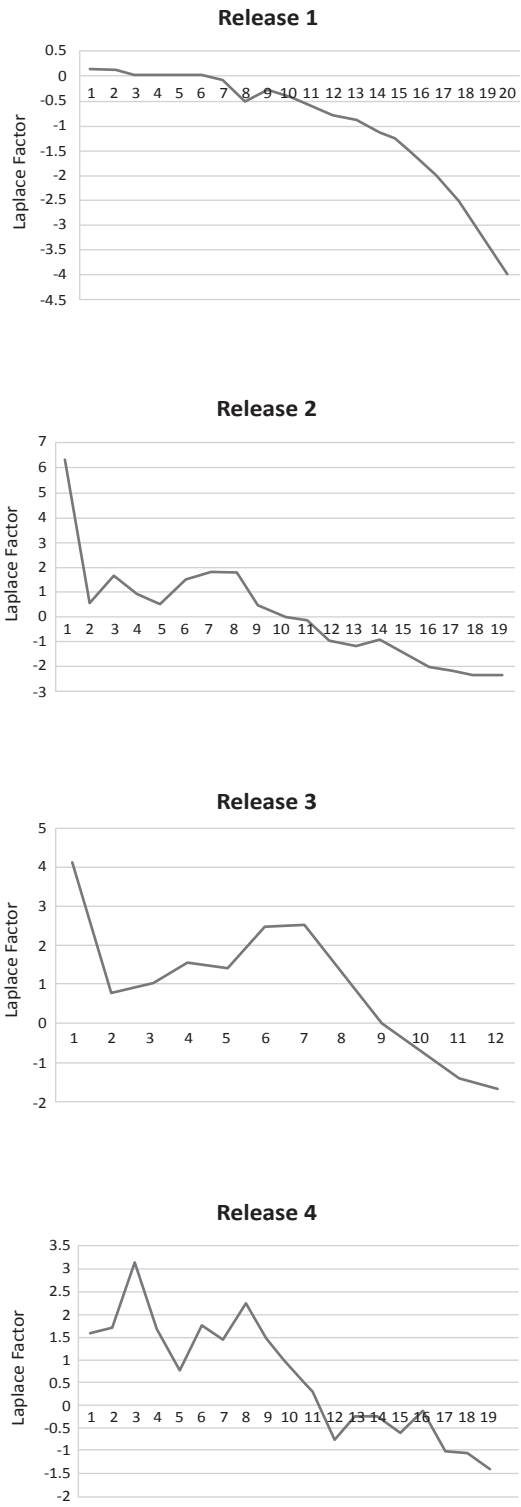

Fig . 3. Reliability Trend for Four Release Dataset
Reference: the authors

can be estimated with considerable accuracy and goodness of fit. The proposed model is validated on Tandem computers dataset in which the faults for four releases were recorded with respect to time and resources spent. The dataset shows that during the first release, 100 faults were removed when tested for 20 weeks using 10 000 units of resources consumption. 120 faults were observed in second release after 19 weeks of testing and the application of 10 272 unit's resources while the third release was subjected to testing for 12 weeks using 5 053 units of resources and 61 faults were reported. Similarly, in the fourth release 11 305 unit resources were applied for 19 weeks resulting in 42 bugs.

The parameters have been estimated using the NLLS (Non-Linear Least Square) in SPSS (Statistical Package for Social Sciences). The values of the parameters estimated for the first release were used for parameter estimation of the second release. The faults that remained undetected during testing of the first release were added to the fault content of the second release. In the same manner we have estimated the parameters for subsequent releases. The estimated parameter values are shown in Table 3.

Table 3. Parameter estimates

Release	a	b	l	e	f
1	106.901	0.159	0.173	0.248	0.006
2	120.002	0.213	0.257	0.247	0.003
3	60.989	0.069	0.021	0.355	0.021
4	39.786	0.067	0.021	0.589	0.090

Reference: the authors

The surface curves for estimated values for number of faults removed corresponding to number of faults removed during different releases are shown given in Fig. 4 (x-axis represents time and y-axis represent resource consumption).

To verify the relative advantage provided by the 2-dimensional modeling scheme over only time based models we estimated the parameters again by substituting $e = 0$ in the expression (12). Table 4 shows the values of performance criteria for 1-D (only time dependent) model and 2-D (both time and resource dependent) model respectively. This

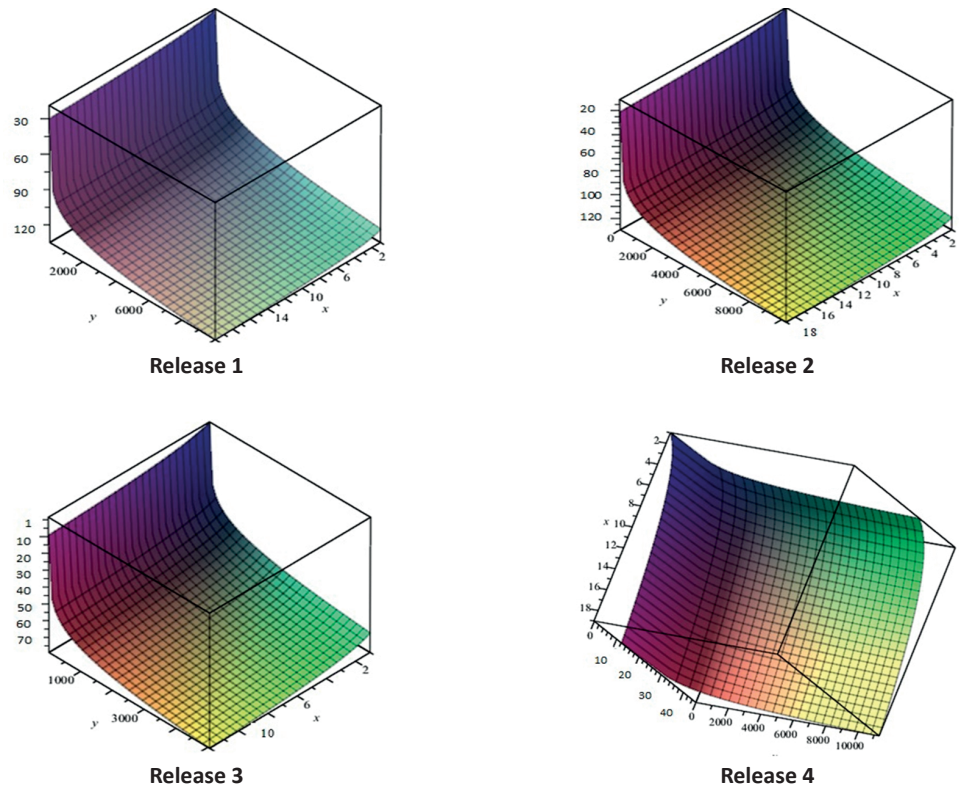

Fig. 4. Estimated values for $m(t, r)$ different releases
Reference: the authors

table includes the values of comparison criteria namely mean square error (MSE), Predictive Ratio Risk (PRR), Predictive power (PP) and Coefficient of determination (R^2). For the first three criteria the dataset is said to fit the model better if the values are lower. But it is not the case of R^2. Higher values of R^2 indicate a better fit of the model while smaller values imply the failure of model in explaining the variations in the data or a poor fit. Since MSE is sensitive to the outliers in the dataset, we have also used some other performance indicators like Mean Absolute percentage error (MAPE) and Mean Absolute Scaled error (MASE) to evaluate model performance. For these criteria smaller values also represent a better fit of the data.

Boxplot representation of dataset for the model will clear the preference of the proposed model for estimating the number of faults removed in a software system. Boxplot divides the data into quartiles and helps understand the spread as well as concentration of data. Apart from this we can easily identify the outliers i.e. the data values that are far away from other values. We plotted boxplot for the absolute residuals obtained after predicting faults using the proposed model. (Fig. 5)

Wider boxplot (Release 3) represents the spread in the data with larger extreme values, i.e. difference in observed and predicted faults is more than the compact ones (Release 1, 2, 4); these represent the concentration of data, i.e. the observed and predicted values are closer to each other. Spread and compactness of boxplot shows the distance of residual from the mean residual value. Higher value of residual implies that there are more mismatches between the estimated and fault predicted using the proposed model and vice-versa. Right-skewed (Release 2 and 3) implies data is concentrated at the lower end while Left- skewed (Release 1 and 4) boxplot concentration of data is at the upper end. Concentration of residual at lower end implies that most of the predicted values are near to the estimated bugs. We can also observe outliers in release 1, 2 and 4. So we can conclude that the model fits best for release 1, 2 and 4 but less for the release 3.

Table 4. Comparison using performance criteria

Release	Model	R^2	MSE	PRR	PP	MAPE	MASE
1	Time-Dependent Model	0.977	18.654	45.039	16.106	21.417	0.639
	Proposed Model	0.989	8.89	9.914	6.424	8.247	0.486
2	Time-Dependent Model	0.991	12.345	25.533	10.428	17.508	0.456
	Proposed Model	0.996	5.412	5.159	3.829	6.391	0.298
3	Time-Dependent Model	0.980	8.498	6.826	4.903	17.268	0.525
	Proposed Model	0.996	1.749	1.720	1.291	7.138	0.180
4	Time-Dependent Model	0.990	1.421	2.679	2.761	11.208	0.493
	Proposed Model	0.991	1.294	2.211	2.604	11.181	0.492

Reference: the authors

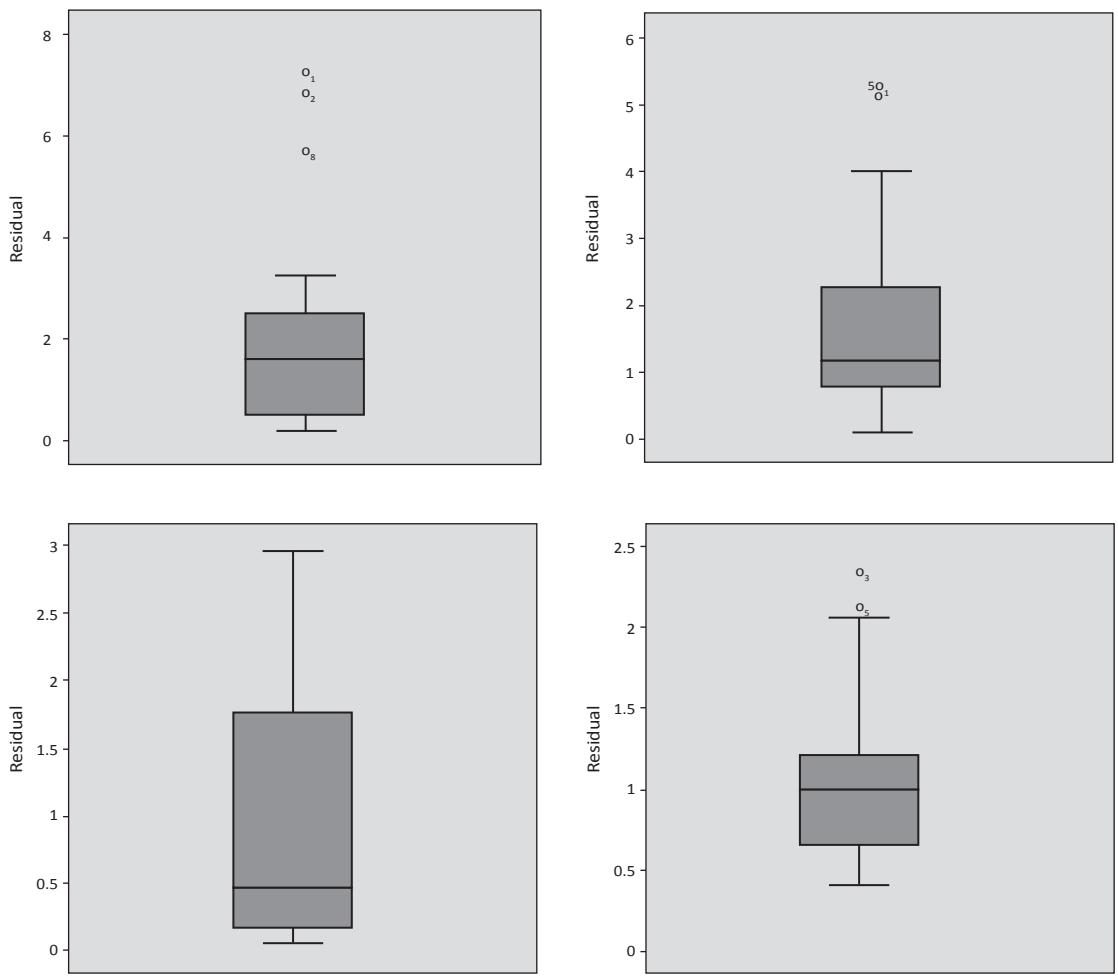

Fig. 5. Boxplot of absolute residuals using Least Square Estimation
Reference: the authors

6. Conclusions

In this paper we have proposed a 2-Dimensional modeling scheme based on Fault Reduction Factor. FRF is assumed to be a constant while fault removal rate is represented by Delayed S-Shaped model under imperfect environment. The proposed scheme is meant for software systems with multiple versions. A numerical study has been carried out to validate the model with respect to a four-release dataset. Comparisons have been drawn between the proposed two-dimensional model and only time-based model in terms of number of criteria, namely R^2, MSE, PRR, PP, MAPE and MASE. The proposed model may be used by software product planners to decide about optimal time to launch different versions or to decide about warranty policies. The work presented in this paper may be extended to determine optimal maintenance policies for a software system.

References

[1] S. Yamada, *Software Reliability Modelling: Fundamentals and applications*, Japan: Springer, 2014.

[2] A. L. Goel and K. Okumoto, "Time Dependent Error Detection Rate Model for Software Reliability and Other Performance Measures", *IEEE Transactions on Reliability*, vol. 28, no. 3, pp. 206-211, 1979. doi: https:///doi.org/10.1109/TR.1979.5220566

[3] M. Obha, "Software Reliability Analysis Models", *IBM Journal of Research and Development,* vol. 28, no. 4, pp. 428-443, 1984. doi: https://doi.org/10.1147/rd.284.0428

[4] S. Yamada, K. Tokuno, and S. Osaki, "Imperfect debugging models with fault introduction rate for software reliability assessment", *International Journal of Systems Science,* vol. 23, no. 12, pp. 2241-2252, 1992. doi: https://doi.org/10.1080/00207729208949452

[5] Y. Tamura and S. Yamada, "A flexible stochastic differential equation model in distributed development environment", *European Journal of Operational Research*, vol. 1, no. 168, pp. 143.152, 2006. doi: https://doi.org/10.1016/j.ejor.2004.04.034

[6] J. D. Musa, "A theory of software reliability and its application", *IEEE Transactions on Software Engineering*, vol. SE-1, no. 3, pp. 312-327, 1975.

[7] J. D. Musa, A. Iannino, and K. Okumoto, *Software Reliability: Measurement, Prediction, Application,* Texas: McGraw-Hill, 1987.

[8] J. D. Musa, *Software Reliability Engineering: More Reliable Software, Faster and Cheaper*, 2nd ed., Bloomington: Authorhouse, 2004.

[9] A. Wood, "Predicting Software Reliability", *IEEE Computers*, vol. 29, no. 11, pp. 69-77, 1996. doi: https://doi.org/10.1109/2.544240

[10] M. Ohba and S. Yamada, "S-shaped Software Reliability Growth Model", *Proceedings of the 4th International Conference on Reliability and Maintainability,* pp. 430-436, 1984. doi: https://doi.org/10.1109/TR.1983.5221735

[11] P. K. Kapur, A. Tandon, and G. Kaur, "Multi Upgradation Software Reliability model", in *IEEE Proceedings of 2nd International Conference on Reliability, Safety and Hazard,* December 14-16, pp. 478-474, 2010. doi: https://doi.org/10.1109/ICRESH.2010.5779595

[12] S. Yamada, H. Ohtera, and H. Narihisa, "Software Reliability growth models with testing-effort", *IEEE Transactions on Reliability*, R-35, pp-19-23, 1986. doi: https://doi.org/10.1109/TR.1986.4335332

[13] P. K. Kapur, A. G. Aggarwal and G. Kaur, "Testing Resource dependent Flexible reliability growth model for software with multiple releases", in *International Conference on Development and Applications is Statistics in Emerging Areas of Science and Technology (ICDASEAST 2010),* Jammu, India, 8-10 Dec 2010.

[14] T. Ishii and T. Dohi, "Two-Dimensional Software Reliability Models and Their Application," *Proceedings 12th Pacific Rim International Symposium Dependable Computing,* pp. 3-10, 2006. doi: https://doi.org/10.1109/PRDC.2006.64

[15] S. Inoue and S. Yamada, "Two-Dimensional Software Reliability Measurement Technologies", in *Proceedings of IEEE*, 2009. doi: https://doi.org/10.1109/IEEM.2009.5373378

[16] P. K. Kapur, H. Pham, A. G. Aggarwal, and G. Kaur, "Two Dimensional Multi-Release Software Reliability Modeling and Optimal Release Planning", IEEE Transactions on Reliability, vol. 61, no. 3, pp. 758-768, 2012. doi: https://doi.org/10.1109/TR.2012.2207531

[17] C. J. Hsu, C. Y. Huang, and J. R. Chang, "Enhancing software reliability modeling and prediction through the introduction of time variable fault reduction factor", *Applied Mathematical Modelling*, vol. 35, no. 1, pp. 506-521, 2011. doi: https://doi.org/10.1016/j.apm.2010.07.017

[18] B. Pachauri, J. Dhar, and A. Kumar, "Incorporating inflection S-shaped fault reduction factor to enhance software reliability growth", *Applied Mathematical Modeling*, vol. 39, no. 5, pp. 1463–1469, 2014. doi: https://doi.org/10.1016/j.apm.2014.08.006

[19] S. Chatterjee and A. Shukla, "Modelling and analysis of software fault detection and correction process through Weibull type FRF, Change point and Imperfect Debugging", *Arab Journal of Science and Engineering,* vol. 41, pp. 5009-5025, 2016. doi: https://doi.org/10.1007/s13369-016-2189-0

[20] A. G. Aggarwal, V. Dhaka, and N. Nijhawan, "Reliability analysis for multi-release open-source software systems with change point and exponentiated Weibull fault reduction factor", *Life cycle Reliability safety engineering,* vol. 6, no. 1, pp. 3-14, 2017. doi: https://doi.org/10.1007/s41872-017-0001-0

[21] A. L. Goel, "Software reliability models: assumptions, limitations and applicability" *IEEE Transactions on Software Engineering*, vol. 11, no. 12, pp. 1411-1423, 1985. doi: https://doi.org/10.1109/TSE.1985.232177

[22] Z. Jelinski and P. Moranda, "Software reliability research", *Statistical Computer Performance Evaluation*, New York: Academic Press, 1972, pp. 465-484. doi: https://doi.org/10.1016/B978-0-12-266950-7.50028-1

[23] 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 Trans. Reliability*, vol. 60, no. 1, pp. 331-340, 2011. doi: https://doi.org/10.1109/TR.2010.2103590

[24] M. Jain, T. Manjula, and T. R. Gulati, "Software Reliability Growth Model (SRGM) with Imperfect Debugging, Fault Reduction Factor and Multiple Change-Point", in *Proceedings of the International Conference on SocProS, AISC 131,* pp. 1027-1037, 2012. doi: https://doi.org/10.1007/978-81-322-0491-6_95

[25] S. Chatterjee and J. B. Singh, "A NHPP based software reliability model and optimal release policy with Logistic-Exponential test coverage under imperfect debugging", *International Journal on System Assurance Engineering and Management*, vol. 5, no. 3, pp. 399-406, 2014. doi: https://doi.org/10.1007/s13198-013-0181-6

[26] H. Ascher, "Regression Analysis of Repairable Systems Reliability", *Electronic Systems Effectiveness and Life Cycle Costing,* vol. 3, pp. 119-130, 1983. doi: https://doi.org/10.1007/978-3-642-82014-4_8

[27] K. Kanoun, "Software dependability growth characterization, modeling and evaluation," Doctorate es-Sciences dissertation, Institute National Polytechnique de Toulouse, France, LAAS Reo. 89-320, Sept. 1989.

[28] H. Ascher and H. Feingold, "Application of Laplace's test to repairable system reliability," in *Proceedings of 1st International conference on Reliability and Maintainability,* Paris, France, June 19-23, pp. 219-225, 1978.

[29] D. R. Cox and P. A. Lewis, *The Statistical Analysis of Series of Events*, London: Chapman & Hall, 1978.