EDUZONE: International Peer Reviewed/Refereed Multidisciplinary Journal (EIPRMJ), ISSN: 2319-5045
Volume 13, Issue 2, July-December, 2024, Available online at: www.eduzonejournal.com

Software Reliability Growth Models Determining Release Time and Testing Effort using Exponentiated-Gompertz Method

Podduturu Sharanya[1], Dr. Deepak Sharma[2]

[1]Research Scholar in Computer Science and Engineering Dept, Monad University, Hapur, Pilkhuwa Distt, Uttar Pradesh, India
[2]Professor in Computer Science and Engineering Dept, Monad University, Hapur, Pilkhuwa Distt, Uttar Pradesh, India

ABSTRACT

One important consideration for the software product is quality. Every stage of the software development process was done with the utmost attention to produce a high-quality output. The development process often incorporates a number of quantitative and qualitative methods. The final software product's features ought to meet every requirement. A crucial component that measures the likelihood that a software program will function before it genuinely fails to carry out its intended purpose is reliability. Software testing is a crucial stage that required enormous amounts of resources. This testing step required more than half of the budget, which is why it was conducted in a controlled setting. The critical point at which software product testing was terminated is thought to be the software product release time, and and it might be released onto the market, in which case the software product ought to be dependable and of high quality. In order to calculate the software release time, we used the exponentiated-gompertz function as the testing effort function in this paper's investigation of the idea of software testing effort dependent software reliability growth models. As a result, three genuine time datasets were used to compute the testing effort dependent models that were built. Least squares estimation is used to estimate parameters, and measures such as Mean Squared Error (MSE) and Absolute Error (AE) are used to compare models. The performance of the suggested testing effort dependant model outperformed that of the other models.

Index Terms: Mean Square Error, Absolute, Software Testing, Software Cost, Software Reliability, and Testing Effort.

INTRODUCTION

The development and modernization of industries has increased the greater extent of modern devices, whether the hospitals, business companies, telecommunication and aeronautics. These modern gadgets and computing devises need always software thus software should have more quality and reliability. Software industries struggling from past few decades to prepare and design a quality and reliable software as the development process its self is fuzzy and complex in nature. Now a days industries are adapting new techniques and metrics to achieve the intended quality and reliability. Software reliability was defined as probability of software product could able to work for certain period of time before it could actually fail. Usually, software development activities are amalgamated with several tasks.

Software testing is considered to be the most important phase among all the phases where heavy resources were consumed. Software testing phase usually done in systematic environment by considering all resource and time aspects. Past few years several authors have proposed different types of Software reliability growth models under Nonhomogeneous Poisson process models, where each model have its own pros and cons. [1] Designed a Nonhomogeneous Poisson process stochastic model where failure intensity is decreases exponentially with time. [2] they assumed that the model follows the failure detection rate which varies with time. [3] has proposed an S shaped model inflection model by incorporating inflection parameter in their model. Usually most of the software reliability growth models comes under either. exponential or S- shaped family depending on the kind of assumption they made during the model design. [4-6] Some authors have proposed software reliability models based on the relation between failure intensity and its test coverage, they grouped these models under test coverage models. [7-9] In recent years there has been considerable interest was shown in release time determination of the software, several models were proposed under this assumption. [10-18] Several studies were carried out to investigate the inclusion of testing effort into the software reliability growth models. Testing effort is defined as number of test cases used for testing, number of people allotted for testing and fault determination time. Much of the work was carried out by considering the various testing effort related functions in Software reliability growth models, however there were still some models unable to fit to the data in all circumstances. So, in this study we have proposed a new [19, 20] exponentiated-gompertz distribution as testing effort into the software reliability growth model. The reason for using proposed testing effort function is, basically failure datasets have either exponential or S- shaped distribution characteristic. In this paper we have investigated functional characteristic of software reliability growth model by considering Exponentiated-Gompertz testing effort function.

These Nonhomogeneous Poisson process software reliability growth models usually have the stochastic in nature [21]. These models have failure intensity function which was represented by (t). N(t) represents the number of failures in the software at a time t and m(t) represents the mean or expectation. This mean value is computed through by the following equation. The relation between mean and intensity of failure data was given by E[N(t)]=m(t) and the relation between mean and failure intensity is given by

$$m(t) = \int_0^t \lambda(t)dt$$

N(t) have the probability mass function which was given by (1)

$$Pr\{N(t) = n\} = \frac{(m(t))^n \times e^{-m(t)}}{n!}$$

$$n = 0,1,2.. \infty \quad (2)$$

This paper is organized as follows; Section 2 describes the various testing efforts related to Nonhomogeneous Poisson process models. Section 3 brief over view of Software reliability growth models and their properties. Section 4 describes the various failure datasets were used for study. Section 5 provide the detailed information on performance metrics used for evaluation of models. Section 6 describes the total review on results and discussion and section 7 conclusion.

RELATED WORKS

The Poisson model [22] decreases the fault rate in geometric progression. Exponential, normal, gamma and Weibull functions are also used to design the SRGM. The SRGMS are abstractly unsuitable [23] for various proprietary data sets cross study comparisons and leads to an inadequate usage of statistical testing results. For NHPP, the Exponential Weibull testing effort function (TEF) [24] is used for designing the inflection S-shaped SRGMs. It is found to be flexible with imperfect debugging. For Noisy input, the analysis of sensitivity [25] is required to measure the conclusion stability in terms of noise levels concerned. The genetic programming based on symbolic regression [26] is found to be efficient for SRGM design. A hierarchical quality model [27, 28] based SRGM is automatically calculates the metric values and their correlation with various quality profiles. The distance-based approach (DBA) [29] based SRGM is evaluated for selecting the optimal parameters and yields the ranking. It identifies the importance of criteria for the application. The Japanese software development system utilizes the Gompertz curve [30] for residual faults estimation. In SRGM, multiple change points are crucial for detection of environment changes [31]. These are known for efficiently handling both the imperfect and ideal debugging conditions. In SRGM, application of Queuing modes is used for describing the fault detection rate and correction procedure. For this, the extension of infinite server queuing models [32, 33] is used based on multiple change points. The Component Based Software Development (CBSD) [34, 35] is considered as building blocks for SRGM. The CBSD primarily focuses on selection of appropriate user requirements. Some of the soft errors can be minimized by using minimum redundancy concept of critical data [36, 37]. In [38] Anjum et.al, presented a ranking based weighted criteria for reliability assessment where the failure data sets are ranked accordingly.

METHODOLOGY

The present paper proposes a novel approach for quality assessment for software growth model. Contribution of Paper discussed in section A and Motivation behind this Work is discussed in section B.

Contribution of Paper

Non-Homogeneous Poisson Process Software Reliability Growth Models with Generalized Gompertz TEF

Software reliability growth model with Generalized Gompertz testing effort was given by the following assumptions [11-13, 14-17].

- The fault removal process follows the Nonhomogeneous Poisson process models (NHPP)
- The software system is subjected to failure at random time caused by fault remaining in the system.
- The mean time number of faults detected in the time interval (t, t+Δt) by the current test effort is proportional for the mean number of remaining faults in the system.
- The proportionality is constant over the time.
- Consumption curve of testing effort is modeled by a Generalized Gompertz TEF.
- Each time a failure occurs, the fault that caused it is immediately removed and no new faults are introduced.
- We can describe the mathematical expression of a testing-effort based on following

$$(t) \times \frac{1}{dt} = (a - (t)) \quad (3)$$

$\omega(t)$

In the equation (3) parameters 'a' represents total number of faults before the testing begins and 'b' represents the failure detection rate. Solving the equation (3) assuming m(0)=0 at t=0 we get the required mean value function as

$$(t) = (1 - e^{-(W(t)-W(0))}) \quad (4)$$

From the equation (5) Generalized Gompertz testing function produces the value zero when t=0. Now substitute the equation (5) into the equation (4) to the required software reliability growth model with proposed testing effort function.

$$(t) = a\,(1 - e^{-(\phi\ 1\ ))}) \quad (5)$$

From the equation (4) failure intensity is computed by

$$(t) = (t)^{-W(t)} \quad (6)$$

Now substitute the equation (7) into the equation (6) we get

$$(t) = ab\,(\phi\,(1 - e^{-(e^{\eta t}-1)})^{\delta\delta-1}\,\delta\delta\mu\eta e^{-\mu e^{\eta t}+\eta t+\mu})\,e^{-(\phi\,(1-e\ ))} \quad (7)$$

Motivation behind this Work

Various testing efforts used related to non-homogeneous poisson process software reliability growth models.

Since in the era of software reliability growth models sundry authors have proposed different software testing effort functions. Conventionally testing effort plays a consequential role in software reliability growth model, and can be expressed in terms of number of testing persons, number of test cases and time required to find the number of faults during testing.

Exponential, Rayleigh curve and Weibull Curve [11,13,17]: These testing effort functions were proposed by Yamada in his research papers.

$$WW_e(t) = (1 - e^{-rt}) \quad (8)$$

$$WW\ (t) = (1 - e^{-rt^2}) \quad (9)$$

$$WW_{wwe}\ (t) = (1 - e^{-rt^m}) \quad (10)$$

Equation (3), (4) and (5) represents the testing effort Curves cognate to exponential Rayleigh and Weibull. $WW_e\ (t)$, $WW_R\ (t)$, and $WW_{wwe}\ (t)$ represents the cumulative testing efforts of exponential, Rayleigh and Weibull Exponential curve describes the process deteriorating monotonically. Whereas Rayleigh curve shows the different nature as compared to exponential unlike exponential curve it first increases and after reaches to certain level it declines. Weibull curve shows the different variations depending on the value of m, and curve reaches to maximum at m=3.

Logistic curve [16]:

Logistic curve has many advantages in several applications, because the characteristic of authentic time datasets has very near sodality among themselves.

$$WW\ (t) = \frac{A}{L\ (1+\beta e^{-rrt})} \quad (11)$$

$W_L(t)$ represents the logistic curve cumulative testing effort. In the equation (11) β represents the inflection parameter.

Log-Logistic Curve [21]:

This curve was used by the authors to capture the increasing and decreasing phenomenon in failure intensity function. $W_{LL}(t)$ represents the cumulative log-logistic testing effort function.

$$W_{LL}(t) = N \frac{(\theta t)^k}{(1+(\theta t)^k)} \quad (12)$$

Exponentiated-Gompertz(Generalized Gompertz) Distribution[19,20]:

This distribution was proposed by A. El-Gohary, Ahmad Alshamrani, Adel Naif Al-Otaibi. Main advantage of this testing effort function is shows various failure intensity increase, decreasing, constant and bath tub depending on the shape parameter. The Generalized Gompertz cumulative testing effort is given by

$$W_{EG}(t) = \mu > 0, \eta > 0 \text{ and } \delta\delta > 0 \quad (13)$$

$W_{EG}(t)$ represents the cumulative testing effort function, δ represents the shape parameter.

$$W_{EG}(t) = \int_0^t \omega(t)dt \quad (14)$$

(t) represents current testing effort expenditure which was represented by the equation (15)

$$(t) = \phi\,(1 - e^{-(e^{\eta t}-1)^{\delta\delta}-1}) \quad (15)$$

This generalized Gompertz exhibits various special cases in the forms of distributions depending on the values of parameters.

- Generalized exponential distribution with parameter (μ,δ) can obtained by making parameter η to approaches to zero.
- Generalized Gompertz with parameters (μ,η) can be obtained by assigning the zero to parameter δ.
- One parameter μ exponential model can be obtained by making parameter η to approaches to zero and assign the parameter δ to 1.

Failure Datasets used in this Research

- DS1[3]: This dataset was drawn from the work carried out by Ohba 1984, where data belongs to the PL/1 database application software consisting of approximately 1,317,000 lines of code. A total of nineteen and 47.65 CPU hours were spend to capture 328 software errors are removed. The test was continued further in order to recover further errors, and they have recovered round 358 errors.
- DS2[15]: The second dataset was cited from the research papers published y Musa 1987 and Musa 1999. Name of the system was T1 and called Rome air development project. The size of the software approximately 21700 lines of code around 21 weeks and 25.3 CPU hours was spent to discover around 136 faults were reported. A total of 93 CPU hours spends during software testing.
- DS3: Third dataset was belonging to the research paper published day Tomha [39]. In this cumulative number of software bugs discovered 86 and 22 days spend during testing.

Model Performance Metrics and Analysis

Parameter Estimation Methods [40,14]

Here we used least square parameter estimation to estimate the ,δ,η, and μ parameters of the model. These parameters are estimated for the dataset which is in the form $(x_i, W_i) < x_1, x_2, x_3 \dots x_n > and < W_1, W_2, W_3 \dots W_n >$ and $(x_i, m_i) < x_1, x_2, x_3 \dots x_n > and < m_1, m_2, m_3 \dots m_n >$; here x_i represents the time of testing ,W_i represents the cumulative testing effort, m_i represents the cumulative errors in the given datasets and 'n' represents the number of error terms in the datasets. Following is the expression in which we can derive the residual sum of square

$$(\phi, \mu, \eta, \delta\delta) = \sum^n (W - W(t))^2 \quad (16)$$

$=1 \qquad i \qquad i$

Equation (16) can be expressed in another form by substituting the equation (8).

$$S(\phi, \mu, \eta, \delta\delta) = \sum^{\eta t}_{i=1} (W - (\phi(1 - e^{-\mu(e^{\delta\delta\, i} - 1)}))^2 \qquad (17)$$

To estimate the parameters, we simply partially differentiate the equation (17) with respect to each parameter and equating the resultant equation to zero, which was shown in the equation (18).

$$\frac{\partial S(\phi,\mu,\eta,\delta\delta)}{\partial \phi} = 0 \qquad (18)$$

In this paper we used python-based program for scipy library with curve fit function for parameter estimation.

Performance Analysis and Evaluation Metrics [16,18]

Performance and evaluation metrics are important for analysis and comparative study of various models with proposed model. The following are the metrics were used in this paper for comparative and performance measure.

MSE: (mean square error): This metric is useful to measure the difference between actual and estimated values of cumulative errors.

$$MSE = \sum^{n}_{i=1} \left(\frac{((t_i) - y_i)^2}{k}\right) \qquad (19)$$

Lower value of MSE indicates fewer fitting errors, and excellent goodness of fitting.

Prediction error: It measure the difference between actual and estimated number of errors at any instant of time.

$$PE_i = (m_i - y_i) \qquad (20)$$

Average value of prediction error known as Bias.

$$Bias = \sum^{n}_{i=1} \frac{(m_i - y_i)}{n} \qquad (21)$$

Lower the value of Bias indicates better goodness of fit.

Variation: The standard deviation of Bias and prediction error known as Variation. Lower the value of Variation indicates better goodness of fit.

$$Variation = \frac{\sum(PE_i - Bia)^2_{i=1}}{n-1} \qquad (22)$$

Root Mean Square Error Prediction Error: It measures how close is out predicted value with estimated value. Lowe the value indicates the better goodness of fit.

$$RMSPE = (Bias)^2 + (Variation)^2 \qquad (23)$$

Lower the value of RMSPE indicates better goodness of fit.

AE (Accuracy of estimation): It is defined as the ratio of difference in actual total number of errors and total number of errors estimated at the end of the testing by the estimated total number of errors at the end of testing. Here M_a is the total number of errors estimated and 'a' is actual number of errors detected at the end of the testing.

EXPERIMENTAL RESULTS AND DISCUSSION

$$AE = \frac{M_a - a}{M_a} \qquad (24)$$

Figure (1) shows the fitting of testing effort to the actual dataset 1. By using least square parameter estimation was used to estimate the parameters of our proposed testing effort model and estimated model parameters are ϕ = 61.32, μ = 0.2585, η = 0.0982, $and\ \delta\delta$ = 0.907. Figure 1 shows the proposed testing effort curve and actual testing effort. As from the figure 1 shows that out proposed curve fits well and completely captures the real testing effort.

Figure (2) shows the fitting of testing effort to the actual dataset 2. By using least square parameter estimation was used to estimate the parameters of our proposed testing effort model and estimated model parameters are ϕ = 25.88, μ = 0.00395, η = 0.0328, $and\ \delta\delta$ = 1.408. Figure 2 shows the proposed testing effort curve and actual testing effort. As from the figure 1 shows that out proposed curve fits well and completely captures the real testing effort.

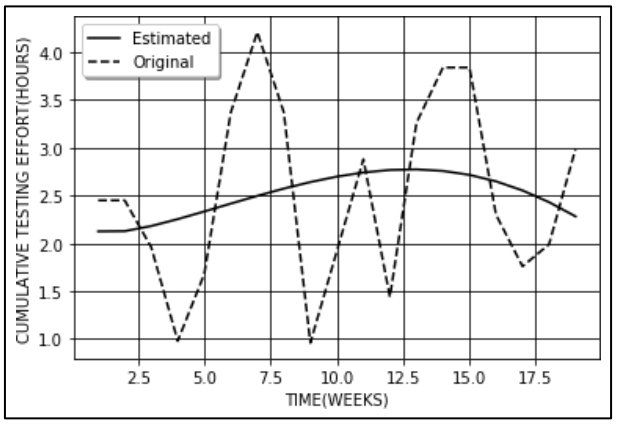

Fig.1. Estimated Generalized Gompertz Testing Effort for Dataset1.

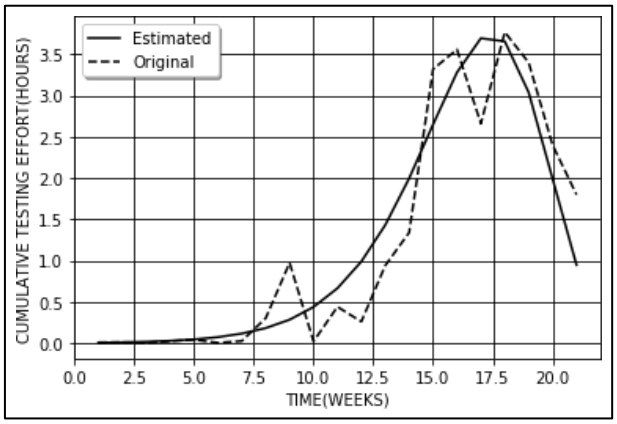

Fig.2. Estimated Generalized Gompertz Testing Effort for Dataset2.

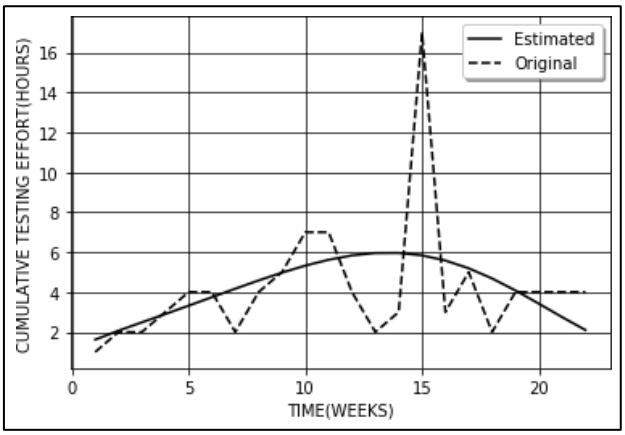

Fig.3. Estimated Generalized Gompertz Testing Effort for Dataset3.

Figure (3) shows the fitting of testing effort to the actual dataset 3. By using least square parameter estimation was used

to estimate the parameters of our proposed testing effort model and estimated model parameters are ϕ = 96.67, μ = 0.2055, η = 0.1269, *and* $\delta\delta$ = 1.202 Figure 3 shows the proposed testing effort curve and actual testing effort. As from the figure 1 shows that out proposed curve fits well and completely captures the real testing effort.

Table 1 describes different testing effort functions and their relative performance metrics for the dataset 1.

Figure 4 shows the relative error curve for the dataset 1, which specifies how far our proposed model follows the actual failure phenomenon.

Table 2 describes various models and their respective parameters and performance metrics like AE and MSE for the dataset 1. The software reliability growth model with proposed testing effort has a=565.1, b=0.0196, AE=57.84 and MSE=103.06. From the Table 2 model with proposed testing effort had less AE and MSE values compared with other models.

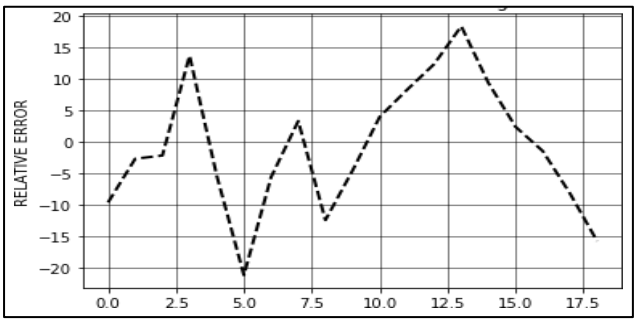

Fig.4. Relative Error Graph SRGM with Generalized Gompertz Testing Effort for Dataset1.

Table 1. Performance Analysis of Proposed Testing Effort Function and other Functions for Dataset1

TEF	Bias	Variation	RMS-PE	'E (at the ending of the testing
Exponentiated-Gompertz	0.0326	0.93	0.90	0.10
Exponentiated Weibull	0.0446	0.95	0.92	-0.09
Burr type X	0.038	0.95	0.93	-0.03
Yamada exponential	-0.39	1.37	1.42	1.27
Yamada Rayleigh	0.83	2.17	2.27	2.50
Yamada Weibull	0.03	0.96	0.93	-0.38
Huang logistic	-0.10	1.31	1.28	1.05

Table 2. Performance Analysis of Proposed Software Reliability Model and other Models for Dataset1

MODEL	a	b	AE(%)	MSE
SRGM with eq. proposed TEF eq.13	565.1	0.0196	57.84	103.06
SRGM with Exponentiated Weibull [14]	565.4	0.0196	57.93	113.10
SRGM with [14] Burr type X	565.7	0.0196	69.15	123.67
Yamada Exponential model with exponential curve [41]	828.25	0.0118	131.4	140.7
Yamada Rayleigh Model with with Rayleigh curve [41]	459.1	0.0273	28.23	268.4
Yamada Weibull Model with Weibull curve [42]	565.35	0.0196	57.91	122.1
Huang Logistic model [16]	394.08	0.0427	10.06	118.6
Ohba exponential model [3]	4555.4	0.0267	27.09	206.9
Infection S shaped model [3]	389.1	0.0935	8.69	133.5
Delayed S shaped model [2]	374.05	0.1976	4.48	168.7
G-O Model [1]	760	0.0322	112.2	139.8
Delayed S shaped model with Rayleigh [16]	333.14	0.1004	6.93	798.5

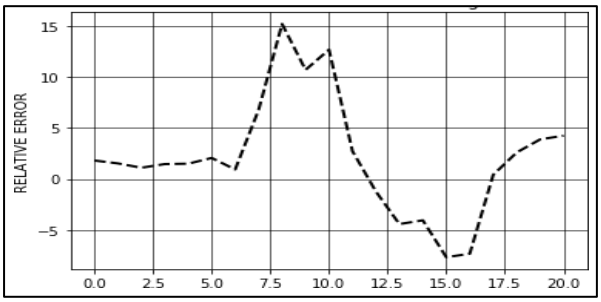

Fig.5. Relative Error Graph SRGM with Generalized Gompertz Testing Effort for Dataset2.

Table 3 describes different testing effort functions and their relative performance metrics for the dataset 2.

Table 4 describes various models and their respective parameters and performance metrics like AE and MSE for the dataset 2. The software reliability growth model with proposed testing effort has a=134.23, b=0.153, AE=28.60 and MSE=48.20. From the Table 2 model with proposed testing effort had less AE and MSE values compared with other models.

Figure 5 shows the relative error curve for the dataset 2, which specifies how far our proposed model follows the actual failure phenomenon.

Table 3. Performance Analysis of Proposed Testing Effort Function and other Functions for Dataset2

TEF	Bias	Variation	RMS-PE	PE (at the ending of the testing
Exponentiated-Gompertz	0.038	0.358	0.351	0.171
Exponentiated Weibull	0.07	0.387	0.384	0.127
Burr type X	0.155	0.50	0.512	-0.47
Yamada exponential	-16.53	6.35	17.71	-13.29
Yamada Rayleigh	-1.149	3.46	3.64	6.03
Huang logistic	0.055	0.35	0.35	-0.10

Table 4. Performance Analysis of Proposed Software Reliability Model and other Models for Dataset2

MODEL	a	b	AE(%)	MSE
SRGM with eq. proposed TEF eq . 13	134.23	0.153	28.60	48.20
SRGM with [14] Exponentiated Weibull	133.87	0.154	28.79	78.55
SRGM with Burr type X[15]	134.11	0.152	28.67	123.7
Yamada Rayleigh Model with [41] with Rayleigh curve	866.94	0.00962	25.11	89.24
Huang Logistic model [16]	138.02	0.1451	26.58	62.41
Ohba exponential model[3]	137.2	0.156	27.12	3019.6
Infection S shaped model[3]	159.11	0.0765	15.36	118.3
Delayed S shaped model[2]	237.2	0.0963	26.16	245.24
G-O Model[1]	142.32	0.1246	24.29	2438.3

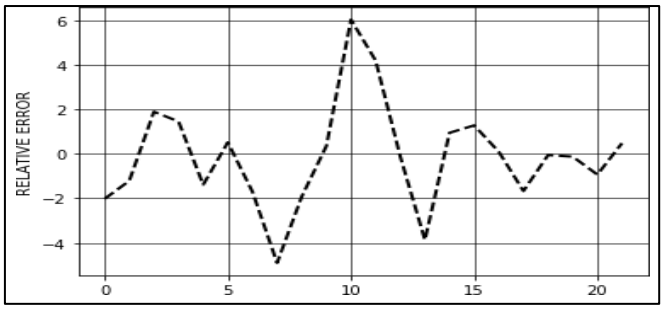

Fig.6. Relative Error Graph SRGM with Generalized Gompertz Testing Effort for Dataset3.

Table 5 describes different testing effort functions and their relative performance metrics for the dataset 3.

Table 5. Performance Analysis of Proposed Testing Effort Function and other Functions for Dataset3

TEF	Bias	Variation	RMS-PE	E (at the ending of the testing
Exponentiated-Gompertz	-0.008	2.179	2.129	1.273
Exponentiated Weibull	0.058	2.212	2.156	1.288
Burr type X	0.155	2.35	2.35	-0.457
Yamada Weibull	0.182	2.37	2.37	-0.44
Yamada Rayleigh	0.324	2.38	2.40	0.15
Huang logistic	-0.122	2.28	2.28	0.19

Table 6 describes various models and their respective parameters and performance metrics like AE and MSE for the dataset 3. The software reliability growth model with proposed testing effort has a=94.89, b=0.025 and MSE=5.42. From the Table 2 model with proposed testing effort had less AE and MSE values compared with other models.

Table 6. Performance Analysis of Proposed Software Reliability Model and other Models for dataset3

MODEL	a	r	MSE
SRGM with eq. 13 proposed TEF	94.89	0.025	5.42
SRGM with Exponentiated Weibull[14]	94.8	0.025	5.70
SRGM with eq Burr type X [14]	94.61	0.0254	6.55
Yamada Weibull Model with Weibull curve[42]	87.03	0.0345	7.77
Huang Logistic model[16]	88.89	0.0390	25.2
Delayed S shaped model [2]	88.65	0.028	6.31
G-O Model [1]	137.1	0.0515	25.3

Figure 6 shows the relative error curve for the dataset 3, which specifies how far our proposed model follows the actual failure phenomenon

Optimal Release Time Policy

Identifying the exact time of release of software product from the software industry is an important issue. [7-9, 15] Many authors have focused on this issue. Determining the exact time of release of software product can greatly impact on the software cost and its quality. If the software product released in a short time can save time and cost but we have to compromise with reliability; if the product released after a long and through testing, the product can have quality but it increases the cost of testing. Release time of the software product depends on two factors, one reliability and another cost. So, we need to determine the time of delivery of software product such that the product can have high reliability and cost of expenditure of testing is minimized.

Software Reliability

Software reliability is defined as probability of product should function correctly in a given environment before it fails.

$$R = e^{-[m(t+\Delta t)-m(t)]} \qquad (25)$$

In the equation (25) R represents the reliability of software product. Figure 7 represents the estimated reliability of software product from the proposed model. In the equation 25 m(t) is mean value of proposed software reliability model. Δt represents the fraction of time period, in our case we assumed the value 0.2.

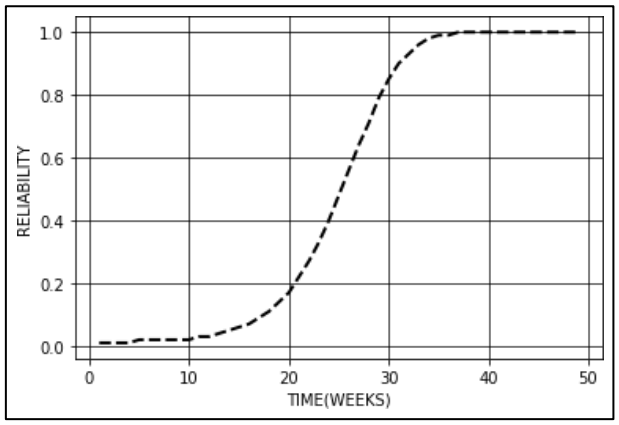

Fig.7. Estimated Reliability Curve for the Dataset1.

Software Cost

Computing the software cost during the software development was is an important task, several authors have concentrated much on this issue. Cost of the software product development can be an integration of different cost's, one before testing phase, one during the testing and cost related to after delivery. Let's assume C_1 is the cost incur for finding the errors pre testing phase, C_2 is cost to detect the errors during the testing phase and C_3 is cost incur to find errors after the testing phase.

$$COST(t_i) = C_1 m(t_i) + C_2 [m(T) - m(t_i)] + C_3 \int_0^{t_i} w(t_i)dt \quad (26)$$

The term COST in the equation (26) represents the total cost incurs during product development. T represents the constant usually we assume maximum time for this quantity, in our paper we assign the value 100 to this term.

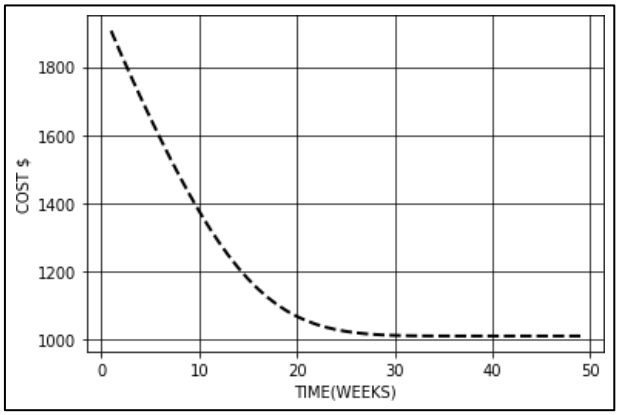

Fig.8. Estimated Cost Curve for the Dataset1.

From the equation (26) we computed the cost of development by assigning C_1=1$, C_2=5$ and C_3=10$ for the dataset-1. Figure 8 shows the graph of computed cost for the proposed software reliability growth model. Table 7 shows the computed cost and reliability of proposed software reliability model.

Release Time Determination

From the equation (25) and (26) we have computed reliability and cost of software product. Software product can be released into market either time at which cost of product become optimal or time at which reliability of software become standard accepted level. In this paper standard acceptance of software reliability level is 0.95 which can be obtained at T_R= 38.5 weeks and optimal cost COST=1120$ was obtained at T_C=40 weeks.

$$T_0 = \{ , T_C \} \quad (27)$$

From the equation (27) we can extract the optimal time T_0 which satisfies the our both conditions. In our case the optimal release time T_0=38.5 weeks.

Table 7. Cost and Reliability for the Dataset 1

TIME	RELIABILITY	COST	TIME	RELIABILITY	COST	TIME	RELIABILITY	COST
11	0.03	1456.36	21	0.18	1179.84	31	0.66	1124.67
12	0.03	1413.67	22	0.21	1167.97	32	0.72	1123.53
13	0.04	1374.48	23	0.25	1158.11	33	0.77	1122.67
14	0.05	1338.8	24	0.29	1150.02	34	0.81	1122.03
15	0.06	1306.61	25	0.33	1143.45	35	0.85	1121.56
16	0.07	1277.83	26	0.38	1138.18	36	0.89	1121.22
17	0.08	1252.35	27	0.44	1134	37	0.92	1120.97
18	0.1	1230.01	28	0.49	1130.71	38	0.94	1120.8
19	0.12	1210.62	29	0.55	1128.15	39	0.96	1120.68
20	0.15	1193.97	30	0.61	1126.18	40	0.97	1120.6

CONCLUSIONS

Using the recently developed Exponentiated-Gompertz testing effort, we have examined the testing effort dependent software reliability growth model in this research. The Exponentiated-Gompertz testing effort function can be used to fit any kind of dataset and is linked to different failure rate features. The software reliability growth model provides an excellent match and accurately represents the real testing effort when combined with the proposed testing effort function. By comparing the suggested model with earlier models using performance indicators, it was determined that the proposed testing effort model produced satisfactory results. The software product's release schedule is computed in this paper based on many characteristics, including cost and dependability.

REFERENCES

[1]. A. L. Goel and K. Okumoto, “Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures,” IEEE Trans. Rel., vol. R-28, no. 3, pp. 206–211, Aug. 1979, doi: 10.1109/TR.1979.5220566.

[2]. S. Yamada and S. Osaki, “Software Reliability Growth Modeling: Models and Applications,” IIEEE Trans. Software Eng., vol. SE-11, no. 12, pp. 1431–1437, Dec. 1985, doi: 10.1109/TSE.1985.232179.

[3]. M. Ohba, “Inflection S-Shaped Software Reliability Growth Model,” in Stochastic Models in Reliability Theory, vol. 235, S. Osaki and Y. Hatoyama, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, pp. 144–162. doi: 10.1007/978-3-642- 45587-2_10.

[4]. B. Ciciani and A. Pasquini, “Software Reliability Models and Test Coverage,” in Safe Comp 96, E. Schoitsch, Ed. London: Springer London, 1997, pp. 105–114. doi: 10.1007/978-1-4471-0937-2_9.

[5]. X. Cai and M. R. Lyu, “Software Reliability Modeling with Test Coverage: Experimentation and Measurement with A Fault- Tolerant Software Project,” in the 18th IEEE International Symposium on Software Reliability (ISSRE ’07), Trollhattan, Sweden, Nov. 2007, pp. 17–26. doi: 10.1109/ISSRE.2007.17.

[6]. Q. Li and H. Pham, “A testing-coverage software reliability model considering fault removal efficiency and error generation,” PLoS ONE, vol. 12, no. 7, p. e0181524, Jul. 2017, doi: 10.1371/journal.pone.0181524.

[7]. S. Yamada, H. Narihisa, and S. Osaki, “Optimum release policies for a software system with a scheduled software delivery time,” International Journal of Systems Science, vol. 15, no. 8, pp. 905–914, Aug. 1984, doi: 10.1080/00207728408926610.

[8]. Chin-Yu Huang, Sy-Yen Luo, and M. R. Lyu, “Optimal software release policy based on cost and reliability with testing efficiency,” in Proceedings. Twenty-Third Annual International Computer Software and Applications Conference (Cat. No.99CB37032), Phoenix, AZ, USA, 1999, pp. 468–473. doi: 10.1109/CMPSAC.1999.814328.

[9]. P. K. Kapur and R. B. Garg, “Cost–reliability optimum release policies for a software system under penalty cost,” International Journal of Systems Science, vol. 20, no. 12, pp. 2547–2562, Dec. 1989, doi: 10.1080/00207728908910332.

[10]. L. Fiondella and S. S. Gokhale, “Software Reliability Models Incorporating Testing Effort,” OPSEARCH, vol. 45, no. 4, pp. 351–368, Dec. 2008, doi: 10.1007/BF03398825.

[11]. S. Yamada, H. Ohtera, and H. Narihisa, “Software Reliability Growth Models with Testing-Effort,” IEEE Trans. Rel., vol. 35, no. 1, pp. 19–23, 1986, doi: 10.1109/TR.1986.4335332.

[12]. N. Ahmad and Md. Zafar Imam, "Software Reliability Growth Models with Log-logistic Testing-Effort Function: A Comparative Study," IJCA, vol. 75, no. 12, pp. 8–11, Aug. 2013, doi: 10.5120/13161-0818.
[13]. S. Yamada and H. Ohtera, "Software reliability growth models for testing-effort control," European Journal of Operational Research, vol. 46, no. 3, pp. 343–349, Jun. 1990, doi: 10.1016/0377-2217(90)90009-Z.
[14]. M. U. Bokhari and N. Ahmad, "SOFTWARE RELIABILITY GROWTH MODELING FOR EXPONENTIATED WEIBULL
[15]. FUNCTION WITH ACTUAL SOFTWARE FAILURES DATA," in Innovative Applications of Information Technology for the Developing World, Kathmandu, Nepal, Jul. 2007, pp. 390–395. doi: 10.1142/9781860948534_0062.
[16]. N. Ahmad, M. G. M. Khan, S. M. K. Quadri, and M. Kumar, "Modelling and analysis of software reliability with Burr type X testing-effort and release-time determination," Jnl of Modelling in Management, vol. 4, no. 1, pp. 28–54, Mar. 2009, doi: 10.1108/17465660910943748.
[17]. C.-Y. Huang, S.-Y. Kuo, and M. R. Lyu, "An Assessment of Testing-Effort Dependent Software Reliability Growth Models," IEEE Trans. Rel., vol. 56, no. 2, pp. 198–211, Jun. 2007, doi: 10.1109/TR.2007.895301.
[18]. S. Yamada, J. Hishitani, and S. Osaki, "A software reliability growth model for test-effort management," in [1991] Proceedings the Fifteenth Annual International Computer Software & Applications Conference, Tokyo, Japan, 1991, pp. 585–590. doi: 10.1109/CMPSAC.1991.170243.
[19]. K. Pillai and V. S. Sukumaran Nair, "A model for software development effort and cost estimation," IIEEE Trans. Software Eng., vol. 23, no. 8, pp. 485–497, Aug. 1997, doi: 10.1109/32.624305.
[20]. A. El-Gohary, A. Alshamrani, and A. N. Al-Otaibi, "The generalized Gompertz distribution," Applied Mathematical Modelling, vol. 37, no. 1–2, pp. 13–24, Jan. 2013, doi: 10.1016/j.apm.2011.05.017.
[21]. H. H. Abu-Zinadah, "Goodness-of-Fit Tests for the Exponentiated Gompertz Distribution," vol. 2, no. 4, p. 9, 2014.
[22]. S. S. Gokhale and K. S. Trivedi, "Log-logistic software reliability growth model," in Proceedings Third IEEE International High-Assurance Systems Engineering Symposium (Cat. No.98EX231), Washington, DC, USA, 1998, pp. 34–41. doi: 10.1109/HASE.1998.731593.
[23]. Kapur P.K., Pham Hoang A., Gupta, Jha P.C., "Software Reliability, Assessment, with, OR Applications," ISBN: 978-0- 85729-203-2 (Print) 978-0-85729-204-9 (Online)
[24]. Stefan Lessmann, Swantje Pietsch, Baesens Bart, Mues Christophe, "Benchmarking, Classification Models, for Software, Defect Prediction- A Proposed. Framework, and Novel Findings", IEEE Transactions on Software Engineering, Volume 34,
[25]. Issue 4, July to Aug 2008, Page(s): 485 – 496, DOI: 10.1109/TSE.2008.35
[26]. N. Ahmad, Khan M.G.M., L.S. Rafi, "A study of testing effort dependent inflection S - shaped SRGMs with imperfect debugging", International Journal of Quality and Reliability, Management, 2010, Volume27, Issue 1, pages: 89–110, DOI: https://doi.org/10.1108/02656711011009335
[27]. Liebchen Gernot A, Shepperd Martin, "Data sets and Data Quality in Software Engineering", PROMISE '08Proceeding of the fourth international workshop on Predictor models, in software engg. (PROMISE'08), 2008, ACM, New York, USA, pages: 39- 44, DOI: http://dx.doi.org/10.1145/1370788.137079
[28]. Afzal Wasif and Torkar Richard, "Review - On the application, of genetic, Programming, for software, engineering predictive modeling", Expert Syst. Appl. 38, 9 (Sept 2011), 11984 to 11997, DOI=http://dx.doi.org/10.1016/j.eswa.2011.03.041.
[29]. Samoladas Ioannis, Gousios Georgios, Spinellis Diomidis, Stamelos Ioannis, "The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation", Vol. 275, IFIP series: The International, Federation for, Information Processing, pp 237-248.
[30]. Md. Nasar, Prashant Johri, Udayan Chanda, "Software Testing Resource Allocation and Release Time Problem: A Review",
[31]. International Journal of Modern Education and Computer Science, vol.6, no.2, pp.48-55, 2014.
[32]. Sharma Kapil,Garg Rakesh , Nagpal C.K.,Garg R.K., "Selection of Optimal SRGMs Using a Distance. Based Approach," IEEE Transactions on Reliability (Volume: 59, Issue:2, June 2010), Page(s): 266 – 276.
[33]. Ohishi Koji, Okamura Hiroyuki, and Dohi Tadashi. 2009. Gompertz software reliability model: Estimation, algorithm and empirical, validation, J. Syst. Softw. 82, March 2009, 535-543. DOI=http://dx.doi.org/10.1016/j.jss.2008.11.840.
[34]. Huang Chin Yu,Lyu Michael R., "Estimation & Analysis of Some Generalized Multiple, Change, Point Software Reliability Models," IEEE Transactions on Reliability (Volume: 60, Issue: 2, June 2011), Page(s): 498 – 514.
[35]. Huang Chin Yu, Hung Tsui-Ying, "Software, reliability, analysis and assessment using queuing. models with multiple. change points", Vol. 60, Issue 7, October 2010, Pages 2015–2030.
[36]. Li X., Xie M., Ng S.H., "Sensitivity analysis of release time of software reliability models incorporating testing effort with multiple change-points", Applied Mathematical Modeling, Volume 34, Issue 11, November 2010, Pages 3560-3570, DOI: https://doi.org/10.1016/j.apm.2010.03.006
[37]. Kalaimagal Sivamuni and Srinivasan Rengaramanujam, 2008, A retrospective on software component quality

models, ACM SIGSOFT Engg. Notes, Volume 33, Issue 6 (October 2008), Pages 1 to 10, DOI: http://dx.doi.org/10.1145/1449603.1449611.

[38]. Chandra MouliVenkata Srinivas Akana, C. Divakar, Ch. Satyanarayana, "Design of Generalized Weighted Laplacian Based Quality Assessment for Software Reliability Growth Models", International Journal of Information Technology and Computer Science, Vol.10, No.5, pp.48-54, 2018.

[39]. Saeid A. Keshtgar, Bahman B. Arasteh,"Enhancing Software Reliability against Soft-Error using Minimum Redundancy on Critical Data", International Journal of Computer Network and Information Security, Vol.9, No.5, pp.21-30, 2017.

[40]. Javaid Iqbal,"Analysis of Some Software Reliability Growth Models with Learning Effects", International Journal of Mathematical Sciences and Computing, Vol.2, No.3, pp.58-70, 2016.

[41]. Mohd. Anjum, Md. Asraful Haque, Nesar Ahmad,"Analysis and Ranking of Software Reliability Models Based on Weighted Criteria Value", International Journal of Information Technology and Computer Science, vol.5, no.2, pp.1-14, 2013.

[42]. Y TOHMA, JACOBY, MURATA, YAMAMOTO "Hyper- Geometric Distribution Model to Estimate the Number of Residual Software Faults" IIEEE Trans. Eng.1989.

[43]. H. Pham, System Software Reliability. 2006. Accessed: Jun. 14, 2021. [Online]. Available: https://doi.org/10.1007/1-84628- 295-0

[44]. S. Yamada, and H.Ohtera, "Software reliability growth models for testing-effort control Elsevier Science Publishers B.V. (North-Holland) 1990.

[45]. S. Yamada, J. Hishitani, and S. Osaki "Software-Reliability Growth with a Weibull Test-Effort: A Model & Application" IIEEE Trans. Reliability Eng., vol. 42, no. 1, Mar. 1993, doi: 10.1109/32.624305.