International Journal of Computer Applications (0975 – 8887)
Volume 7– No.11, October 2010

Software Reliability Growth Model with Gompertz TEF and Optimal Release Time Determination by Improving the Test Efficiency

Shaik. Mohammad Rafi
Assoc.Professor Department of computer science
Sri Mittapalli Institute of Technology for Women

Shaheda Akthar
Assoc.professor Dept of computer science
Sri Mittapalli College of engineering

ABSTRACT
Software reliability growth models were used since long time to access the quality of the software which was developed. Past few decades several papers describes reliability growth phenomenon. As the time progress, the number of errors detection and correction also increases. A Large effort is required in testing to increases the rate of detection and correction of error to increase the reliability of the software. Generally a Testing-effort is better described by number of persons involved; number of test cases used and calendar time. When the software is lagging by schedule time then there is need of automated testing tools to cop up with lagging. Use of automated tools can increase the testing efficiency to a greater extent. This paper we proposed a software reliability growth model which incorporates the Gompertz testing-effort function and an analysis is made on optimal release. Experiments are performed on two real datasets. Parameters are estimated. The results show our model is better fit than other.

KEYWORDS:
Delayed S-shaped models, imperfect debugging model, non homogeneous Poisson process, Software reliability growth model, and testing-effort.

ACRONYM

NHPP	: Non Homogeneous Poisson Process
SRGM	: Software Reliability Growth Model
MVF	: Mean Value Function
MLE	: Maximum Likelihood Estimation
TEF	: Testing Effort Function
LOC	: Lines of Code
MSE	: Mean Square fitting Error

NOTATIONS

m (t)	: expected mean number of faults detected in time (0,t]
λ (t)	: failure intensity for m(t)
n (t)	: fault content function
m_d (t)	: Cumulative number of faults detected up to t.
m_r (t)	: Cumulative number of faults isolated up to t.
W (t)	: Cumulative testing effort consumption at time t.
W*(t)	: W (t)-W (0)
A	: expected number of initial faults
r (t)	: failure detection rate function
r	: constant fault detection rate function.
r_1	: constant fault detection rate in the Delayed S-shaped model with Gompertz TEF
r_2	: constant fault isolated rate in the Delayed S-shaped model with Gompertz TEF

1. INTRODUCTION
Software is ruling this world past few years. Communication, business and any other area where there is need of software. Every customer needs a more efficient and error free software. Generally software is developed by humans, so there is change that error may propagate through it. Reliability is considered to be one of the primary important factors for software industry. Many papers are presented in this context. Reliability of software defined as the probability that the software will work before it struck with an error in the given conditional environment. Several authors described the behavior of the software reliability in terms of different failure rates. Describing the complete software resting in terms of mathematical equations are called reliability growth model. People like Goel and Okumato, Yamada and Musa proposed different reliability growth models [1, 21, 22, 23]. During the software testing the failure rate shows different characteristic and cannot be predicted its behavior. The software reliability growth models describe the behavior of software testing process. During the development of software many resources were consumed. The consumption curve of testing resource over the testing period [17] can be thought of as a testing effort curve. The test effort [11, 12, 22, 23] can be described by the man power spent during the test phase, number of CPU hours and the number of executed test cases and so on. In several papers describes the effect of [3, 7, 8, 11, 12, 19, 22, 23] testing effort in the software reliability growth model. Generally software testing effort can be described by Rayleigh, Weibull, exponential and logistic curve [8, 11, 12, 22, 23]. Testing is conducted either manually or incorporating the automated tools [8, 9]. Manual testing is a time consuming process, but development of the software time bound process. As the time progress more and more resources are being consumed. Manual testing can leads to delay in the progress of testing. By incorporating the new automated testing into testing can improve the performance by certain extent [8, 9]. These automated testing tools work efficiently, by tracking the more and more errors but it; increases the cost of adopting new automated tool.

The rest of the paper is organized as section 2 describes the testing-effort function. Section 3 proposed new reliability growth models based on Gompertz TEF. Section 4 describes the model evaluation criterion. 5 model performance analysis. Section 6 Optimal release policy based on reliability and cost. Section 7 Numerical examples. And section 8 conclusions.

2. TESTING-EFFORT FUNCTIONS
In general software testing effort can be defined as the amount of effort spends during the software testing. Testing-effort can be described by following curves. Plenty of curves are proposed in literature to express the testing-effort [3, 5, 7, 14, 22, 23]

a) Exponential curve[22]:
Cumulative testing effort can described in (0,t]:

$$W(t) = \alpha \left(1 - e^{-\beta t}\right) \quad (1)$$

Current testing-effort

$$w(t) = \alpha \beta e^{-\beta t} \quad (2)$$

Where α is the total amount of testing expenditure and β is the consumption rate of the testing-effort

b) Rayleigh Curve [22, 23]: Cumulative testing-effort is described in (0, t]: Rayleigh curve is used by Yamada (1989) to describe the testing effort. Rayleigh curve increases to the maximum peak and decreases gradually [Huang 2007].The Rayleigh distribution is a Weibull one with the shape factor set to two.

Cumulative testing-effort

$$W(t) = \alpha \left(1 - e^{-\beta t^2}\right) \quad (3)$$

Current testing-effort

$$w(t) = 2\,\alpha\,\beta\,t\,e^{-\beta t^2} \quad (4)$$

β is a scale parameter represents the consumption rate of the testing-effort.

c) Weibull Curve [22, 23]: Cumulative testing-effort is described in (0, t]: Weibull curve is very flexible curve to model software testing-effort in (0,t](Yamada 1986) : Weibull curve is flexible curve to model the reliability of the given system. Based on its nature it can take variety of forms based on the shape parameter. When m=2 its shows the Rayleigh curve and m=1 it describes the property of exponential curve.

Cumulative testing-effort

$$W(t) = \alpha \left(1 - e^{-\beta t^m}\right) \quad (5)$$

Current testing-effort

$$w(t) = \alpha\,\beta\,t^{m-1}\,m\,e^{-\beta t^m} \quad (6)$$

Where m is a shape parameter and β is a scale parameter

d) Logistic Curve [5, 7, 12]: Cumulative testing-effort is described in (0, t] (Huang 2002): logistic curve has been used as the growth curve. It is an S shaped curve, describing the first decreasing and then increasing phenomenon. The shape of the logistic distribution is similar to normal distribution.

Cumulative testing-effort

$$W(t) = \frac{\alpha}{1 + A\,e^{-\beta t}} \quad (7)$$

Current testing-effort

$$w(t) = \frac{\alpha A \beta e^{-\beta t}}{\left(1 + A e^{-\beta t}\right)^2} \quad (8)$$

e) Log-Logistic curve [3] :

The log-logistic distribution is the probability distribution of a random variable whose logarithm has a logistic distribution. It is similar in shape to the log-normal distribution but has heavier tails.

Cumulative testing-effort

$$W(t) = \frac{\alpha}{1 + \left(\frac{t}{\lambda}\right)^{-\beta}} \quad (9)$$

Current testing-effort

$$w(t) = \frac{\alpha \left(\frac{t}{\lambda}\right)^{-\beta} \beta}{\left(1 + \left(\frac{t}{\lambda}\right)^{-\beta}\right)^2 t} \quad (10)$$

W (t) cumulative testing-effort function and w (t) is current testing-effort function in (0,t]

'α' is total testing effort expenditure ,λ > 0 scale parameter and β >0 shape parameter.

f) Gompertz Curve: generally the testing-effort consumption is slow at the beginning of the test phase; all the members of the testing team should be familiar with the testing process and its internal details. One all the team members are familiar with testing consumption of testing effort increases. This unusual nature gives the testing-effort to derive the S shaped. Gompertz Curve has been used for many years for fitting to statistical data [4, 20].

The Gompertz Cumulative Testing-effort in (0, t] is given by

$$W(t) = \alpha\,e^{-\beta e^{-ct}} \quad (11)$$

Current testing-effort in time (0, t] is $w(t) = \alpha\,\beta\,c\,e^{-ct - \beta e^{-ct}}$

$$W(t) = \int_0^t w(t)\,dt$$

The current testing-effort reaches its maximum value at

$$t_{max} = \frac{\ln(\beta)}{c} \quad (12)$$

3.SOFTWARE RELIABILITY GROWTH MODEL AND TESTING EFFORT FUNCTIONS

3.1) SRGM WITH GOMPERTZ TESTING-EFFORT FUNCTION

The following assumptions are made for software reliability growth modeling [2, 5, 7, 8, 12, 19, 22, 23]

The fault removal process follows the Non-Homogeneous Poisson process (NHPP)

The software system is subjected to failure at random time caused by fault remaining in the system.

(i) 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.

(ii) Consumption curve of testing effort is modeled by a Gompertz TEF.

(iii) 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

$$\frac{dm(t)}{dt} \times \frac{1}{w(t)} = r \times [a - m(t)] \quad (13)$$

Now the equation 13 has been solved under boundary conditions m(0)=0 and r(t)= r (0< r <1).

$$m(t) = a \times \left(1 - e^{-r \times (W(t) - W(0))}\right) \quad (14)$$

Substituting W (t) from eq. (11), we get

$$m(t) = a\left(1 - e^{-r\left(\alpha e^{-\beta e^{-ct}} - \alpha e^{-\beta}\right)}\right) \quad (15)$$

In general failure intensity function is given by

$$\lambda(t) = a \times r \times w(t) \times e^{-r \times (W(t) - W(0))} \quad (16)$$

Now failure intensity for proposed model is given by

$$\lambda(t) = a\,r\,\alpha\,\beta\,c\,e^{-ct - \beta e^{-ct} - r\alpha e^{-\beta e^{-ct}} + r\alpha e^{-\beta}} \quad (17)$$

$$\text{And } m(t) = \int_0^t \lambda(t)\,dt \quad (18)$$

The number of faults remaining in the system is

$$a - m(t) = m_{remaining}(t) = a\,e^{-r(W(t) - W(0))} \quad (19)$$

the number of faults remains in the systems after infinite amount of time is given by

$$m(\infty) = a\,e^{-r\alpha} \quad (20)$$

3.2. YAMADA DELAYED S-SHAPED MODEL WITH GOMPERTZ TESTING-EFFORT FUNCTION

The delayed 'S' shaped model originally proposed by Yamada [25] and it is different from NHPP by considering that software testing not only of error detection but error isolation. And the cumulative errors detected follow the S-shaped curve. This behavior is indeed initial phases testers are familiar with type of errors and residual faults become more difficult to uncover [6, 16, 17].

From the above steps 3 (A) described we will get a relationship between m(t) and w(t). For extended Yamada S-shaped software reliability model. The extended S-shaped model [Yamada 1983] is modeled by [11, 25]

$$\frac{dm_d(t)}{dt} \times \frac{1}{w(t)} = r_1 \times \left[a - m_d(t)\right] \quad (21)$$

And

$$\frac{dm_r(t)}{dt} \times \frac{1}{w(t)} = r_2 \times \left[a - m_r(t)\right] \quad (22)$$

We assume $r_2 \neq r_1$ solving 2 and 3 boundary conditions $\mathbf{m_d}(t)=0$, we have

$$m_d(t) = a \times \left(1 - e^{\left[-r_1 \times W*(t)\right]}\right)$$

And

$$m_r(t) = a \times \left[1 - \frac{\left(r_1 \times e^{\left[-r_2 \times W*(t)\right]} - r_2 \times e^{\left[-r_1 \times W*(t)\right]}\right)}{r1 - r2}\right] \quad (23)$$

At this stage we assume $r_2 \approx r_1 \approx r$, then using 'L' Hospitals rule the Delayed S-shaped model with TEF is given by

$$m(t) \cong m_r(t) = a \times \left(1 - \left(1 + r \times W*(t)\right) \times e^{\left[-r \times W*(t)\right]}\right) \quad (24)$$

The failure intensity function for Delayed S- shaped model with TEF is given by

$$\lambda(t) = a \times r^2 \times w(t) \times W*(t) \times e^{\left[-r \times W*(t)\right]} \quad (25)$$

4) EVALUATION CRITERIA

4.1) THE GOODNESS OF FIT TECHNIQUE FOR RELIABILITY GROWTH MODEL

Here we used MSE [11, 18, 21] which gives real measure of the difference between actual and predicted values. The MSE defined as

$$MSE = \sum_{i=1}^{k} \frac{\left[m(t_i) - m_i\right]^2}{k} \quad (26)$$

A smaller MSE indicate a smaller fitting error and better performance.

a) Coefficient of multiple determinations (R^2) [18] which measures the percentage of total variation about mean accounted for the fitted model and tells us how well a curve fits the data. It is frequently employed to compare model and access which model provides the best fit to the data. The best model is that which proves higher R^2. That is closer to 1.

b) The predictive Validity Criterion

The capability of the model to predict failure behavior from present & past failure behavior is called predictive validity. This approach, which was proposed by (J.Dmusa 1987], can be represented by computing RE for a data set

$$RE = \frac{(m(t_q) - q)}{q} \quad (27)$$

c) In order to check the performance of the Gompertz testing-effort and make a comparisons criteria for our SSE criteria: SSE can be calculated as: [18]

$$SSE = \sum_{i=1}^{n} \left[y_i - m(t_i)\right]^2 \quad (28)$$

Where y_i is total number of failures observed at a time t_i according to the actual data and $m(t_i)$ is the estimated cumulative number of failures at a time t_i for i=1,2,…..,n.

4.2) EVALUATION OF EFFORT FUNCTION [11]

$$PE_i = Actual(observed)_i - Predicted(estimated)_i \quad (29)$$

$$Bias = \sum_{i=1}^{n} \frac{PE_i}{n} \quad (30)$$

$$Variation = \sqrt{\sum_{i=1}^{n} \frac{\left(PE_i - Bias\right)^2}{n-1}} \quad (31)$$

$$MRE = \frac{\left|M_{estimated} - M_{actual}\right|}{M_{actual}} \quad (32)$$

5) MODEL PERFORMANCE ANALYSIS

5.1) DS1: the first set of actual data is from the study by Ohba(1984)[16].the system is PL/1 data base application software , consisting of approximately 1,317,000lines of code .During nineteen weeks of experiments, 47.65 CPU hours were consumed and about 328 software errors are removed. Fitting the model to the actual data means by estimating the model parameter from actual failure data. Here we used the LSE (non-linear least square estimation) to estimate the parameters [13]. Calculations are given in appendix A All parameters of other distribution are estimated through MLE. The unknown parameters of Gompertz TEF are α=70.55(CPU hours), β=3.304, c=0.1109 and the curve reaches its maximum value at t_{max}=10.77 weeks. Correspondingly the estimated parameters of Logistic TEF are N=54.84(CPU hours), A=13.03 and b=0.2263/week and Rayleigh TEF N=49.32 and b=0.00684/week. Fig.1 plots the comparison between observed failure data and the data estimated by Gompertz TEF, Logistic TEF and Rayleigh TEF. The PE, Bias, Variation, MRE and RMS-PE for Gompertz, Logistic and Rayleigh are listed in Table I. From the TABLE I we can see that Gompertz TEF has lower PE, Bias, Variation, MRE and RMS-PE than Logistic and Rayleigh TEF. We can say that our proposed model fits better than the other one. In the table II we have listed estimated values of SRGM with different testing-efforts. We also give the values of SSE, R^2, and MSE. We observed that our proposed model has smallest MSE and SSE value when compared with other models. The 95% confidence limits for the all models are given in the Table III. All the calculations can found in the appendix. Fig .3 shows the RE curves for the different selected models.

TABLE I
COMPARISION RESULT FOR DIFFERENT TEF APPLIED TO DS1

TEF	Bias	Variation	MRE	RMS-PE
Gompertz	-0.0348	1.0198	0.009619	1.019
Logistic	-0.098262	1.306677	0.022246	1.302977

Rayleigh	0.830337	2.169314	0.052676	2.004112

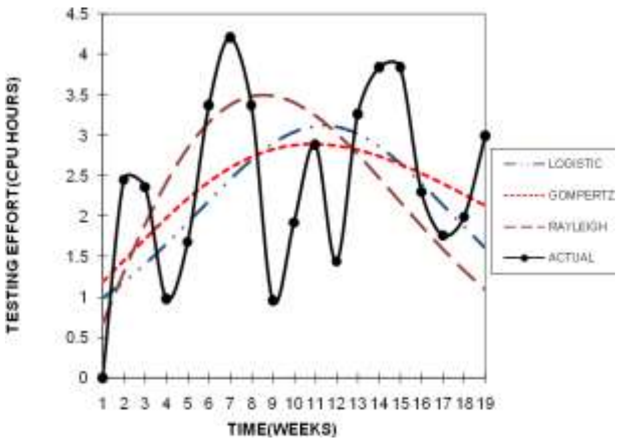


FIG 1. OBSERVED/ESTIMATED GOMPERTZ, LOGISTIC AND RAYLEIGH TEF FOR DS1.

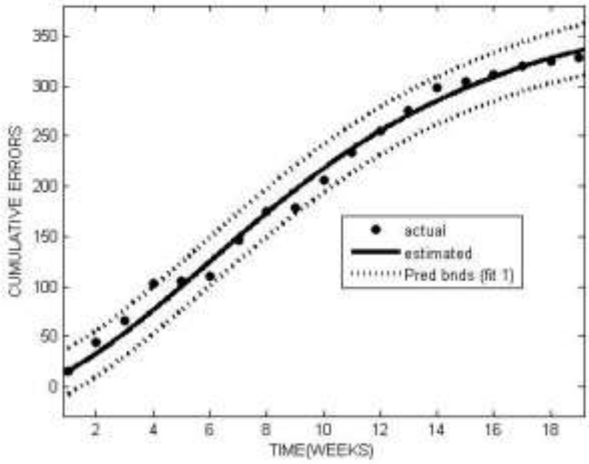


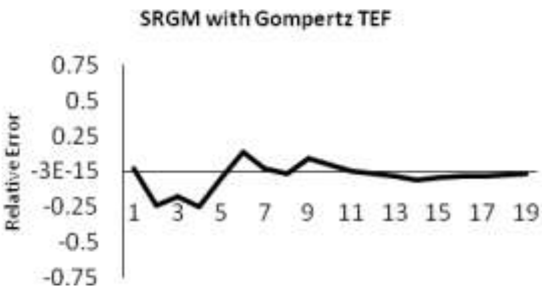
FIG 2. CUMULATIVE ERRORS FOR SRGM WITH GOMPERTZ FOR DS1

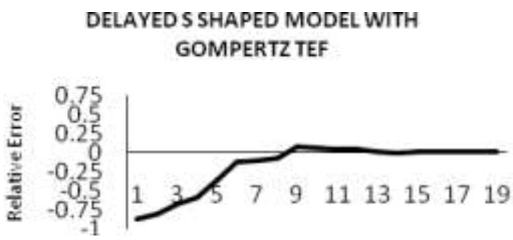
Table II
ESTIMATED PARAMETER VALUES AND MODEL COMPARISION FOR DS1

Models	a	r	SSE	R^2	MSE
SRGM with Gompertz TEF	437.3	0.03251	1980	0.9899	116.42
Delayed S shaped model with Gompertz TEF	330.9	0.1138	8754	09554	514.83
SRGM with Logistic TEF	395.6	0.04164	2167	0.989	127.46
Delayed S shaped model with Logistic TEF	319.3	0.1339	11060	0.9436	650.25
SRGM with Rayleigh TEF	459.1	0.02734	5100	0.974	299.98
Delayed S shaped model with Rayleigh TEF	333.2	0.1004	15170	0.9226	892.2
G-O model	760.5	0.03227	2656	0.9865	156.2
Yamada Delayed S shaped model	374.1	0.1977	3205	0.9837	188.51

Table III
95% CONFIDENCE LIMIT FOR DIFFERENT SELECTED MODELS (DS1)

Models	a		r	
	Lower	Upper	Lower	Upper
SRGM with Gompertz TEF	385.1	489.5	0.02585	0.03917
SRGM with Logistic TEF	358	433.2	0.03399	0.04928
SRGM with Rayleigh TEF	348.6	569.6	0.01651	0.03817
Yamada Delayed S shaped Model with Gompertz TEF	300.8	361	0.09423	0.1334
Yamada Delayed S shaped Model with Logistic TEF	291	347.5	0.1088	0.1589
Yamada Delayed S shaped Model with Rayleigh TEF	288.7	377.7	0.07507	0.1258
G-O model	465.4	1056	0.01646	0.04808
Yamada Delayed S shaped model	343.7	404.4	0.1748	0.2205





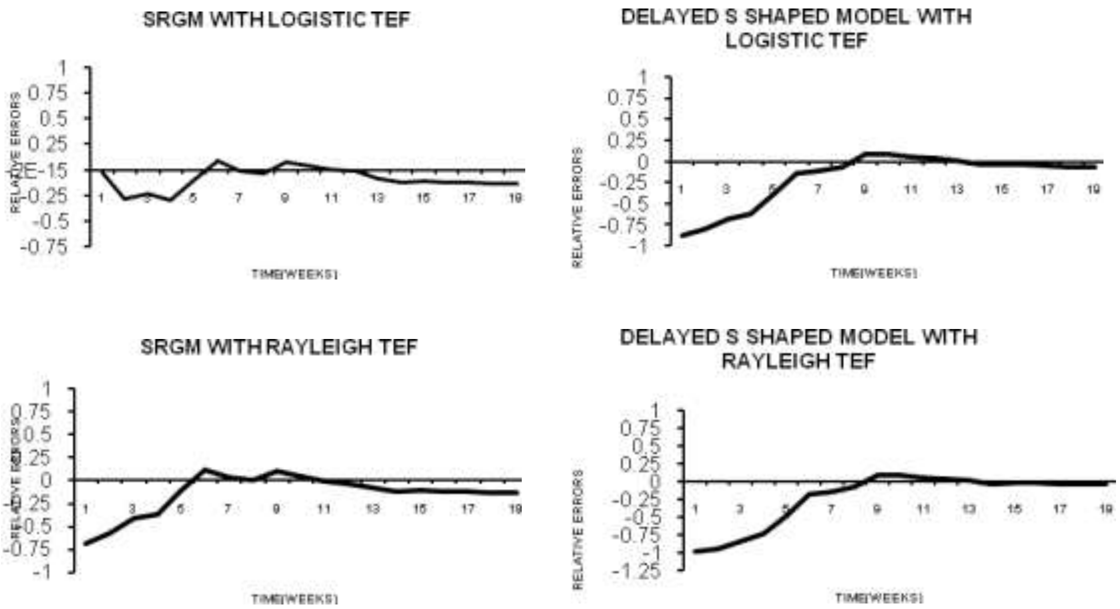


FIG.3 RE CURVES OF SELECTED MODELS COMPARED WITH ACTUAL FAILURE DATA (DS1)

5.2) DS2 [2]: the dataset used here presented by wood from a subset of products for four separate software releases at Tandem computer company. Wood Reported that the specific products & releases are not identified and the test data has been suitably transformed in order to avoid confidentiality issue. Here we use release 1 for illustrations. Over the course of 20 weeks, 10000 CPU hours are consumed and 100 software faults are removed. Similarly the least square estimates [13] of the parameters for Gompertz TEF in the case of DS2 are α=11090(CPU hours), β=3.446,c=0.1663 and the curve reaches its maximum value at t_{max}=7.44 weeks. Correspondingly the estimated parameters of Logistic TEF are N=9974(CPU hours), A=13.22 and b=0.2881/week and Rayleigh TEFN=9669 and b=0.009472/week. The computed Bias, Variation, MRE , and RMS-PE for Gompertz TEF, Logistic TEF and Rayleigh TEF are listed in the table IV ,fig 5 graphically illustrate the comparisons between the observed failure data, and the data estimated by the Gompertz TEF, Logistic TEF and Rayleigh TEF. From the figure 5 we can observe the Gompertz curve covers the maximum points like other TEFs. Now from the table V we can conclude our TEF better fit than other. Their 95% confidence bounds are given in the table VI. From the above we can see that SRGM with Gompertz TEF have less MSE than other models

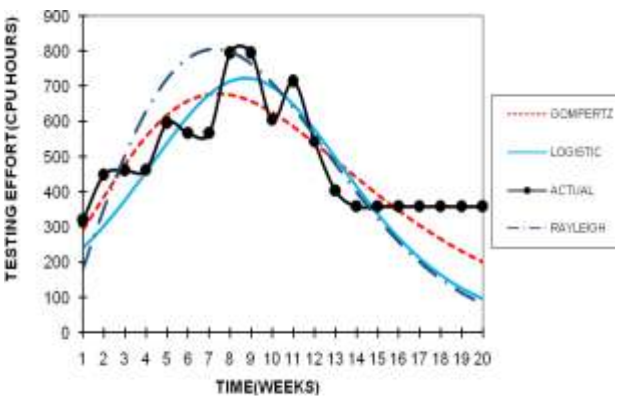


FIG 4. OBSERVED/ESTIMATED GOMPERTZ, LOGISTIC AND RAYLEIGH TEF FOR DS2.

Table IV
COMPARISION RESULT FOR DIFFERENT TEF APPLIE D TO DS2

TEF	Bias	Variation	MRE	RMS-PE
Gompertz	-1.284	104.7	0.020	104.3
Logistic	-19.345	198.44	0.026	197.5
Rayleigh	121.61	322	0.055	298.23

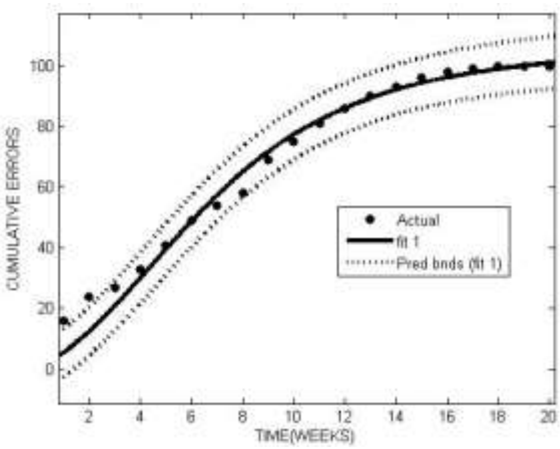


FIG 5. CUMULATIVE AND RESIDUAL ERROR FOR SRGM WITH GOMPERTZ TEF FOR DS2

Table V

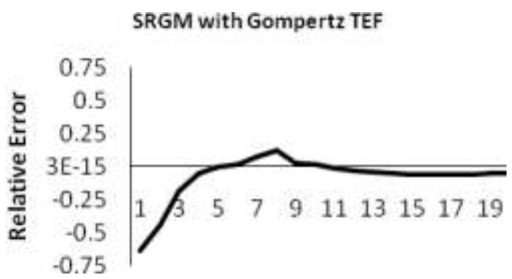
ESTIMATED PARAMETER VALUES AND MODEL COMPARISION FOR DS2

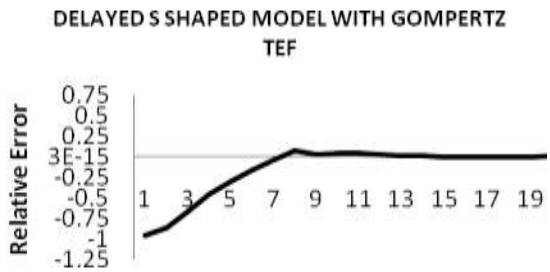
Models	a	r	SSE	R^2	MSE
SRGM with Gompertz TEF	122.4	0.0001841	376.1	0.9769	20.89
Delayed S shaped model with Gompertz TEF	100.3	0.0005645	1314	0.9192	72.98
SRGM with Logistic TEF	112.3	0.0002399	433.1	0.9734	24.06
Delayed S shaped model with Logistic TEF	96.88	0.0006853	1577	0.903	87.61
SRGM with Rayleigh TEF	120.9	0.0001791	792.5	0.9513	44.03
Delayed S shaped model with Rayleigh TEF	99.4	0.0005434	1930	0.8813	107.1

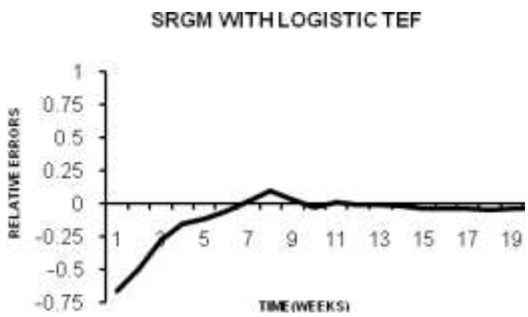
Table VI

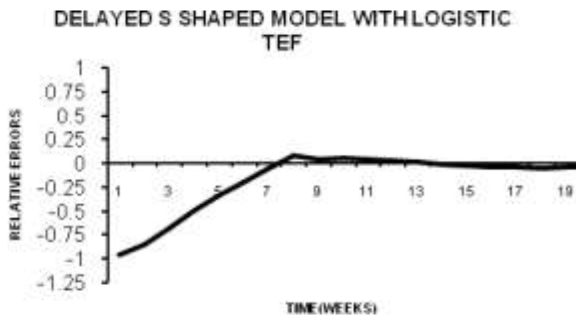
95% CONFIDENCE LIMIT FOR DIFFERENT SELECTED MODELS (DS2)

Models	a		r	
	Lower	Upper	Lower	Upper
SRGM with Gompertz TEF	107.6	137.3	0.0001395	0.0002286
SRGM with Logistic TEF	101.4	123.1	0.000186	0.0002938
SRGM with Rayleigh TEF	98.4	143	0.0001122	0.0002461
Yamada Delayed S shaped Model with Gompertz TEF	92.68	107.8	0.0004685	0.0006604
Yamada Delayed S shaped Model with Logistic TEF	88.64	105.1	0.0005346	0.0008359
Yamada Delayed S shaped Model with Rayleigh TEF	88.24	110.6	0.0003991	0.0006877









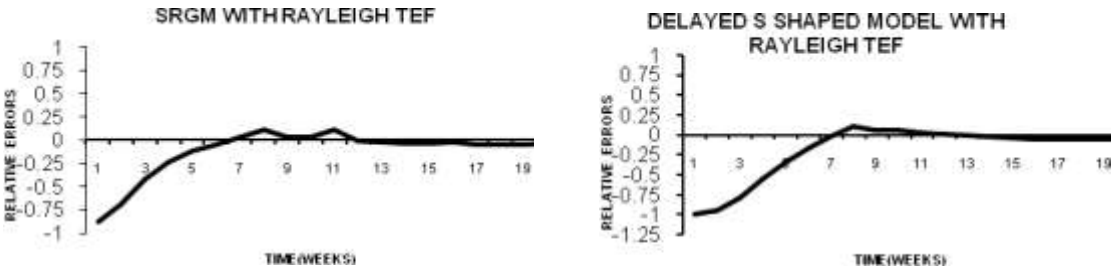


FIG.6 RE CURVES OF SELECTED MODELS COMPARED WITH ACTUAL FAILURE DATA (DS2)

6) OPTIMAL SOFTWARE RELEASE POLICY

6.1) OPTIMAL RELEASE POLICY BASED ON COST

One of the major challenges for software industry is to know, how much of test should be conducted, what is its reliability and when the software has be released into the market [15, 26]. The total cost of the software is summation of cost of correcting the errors before and after the release of the software. C_1 cost of correcting an error during the testing, C_2 cost of correcting an during operation and C_3 cost of testing per unit testing expenditure ($C_2 > C_1$)

$$C1(T) = C_1\, m(T) + C_2\left[m(T_{LC}) - m(T)\right] + C_3\left(\int_0^T w(t)\,dt\right) \qquad (33)$$

C1(T) is the total cost of the testing.

Differentiate the eq.() with respect to T then the optimize the solution to get the required solution

$$\frac{d}{dT}C1(T) = C_1\left(\frac{d}{dT}m(T)\right) - C_2\left(\frac{d}{dT}m(T)\right) + C_3\, w(T) \qquad (34)$$

From above equation $\frac{d}{dT}C1(T) = 0$, $\frac{d}{dT}m(T) = \lambda(T)$ and $\lambda(T) = a\, r\, w(T)\, e^{-r\,[W(T) - W(0)]}$

The minimum value of C1(T) is found by observing the two cases $\frac{\lambda(T)}{w(T)}$ at T=0.

1) if $\frac{\lambda(0)}{w(0)}$ =a.r $\le \frac{C_3}{C_2 - C_1}$ then $\frac{\lambda(T)}{w(T)} \le \frac{C_3}{C_2 - C_1}$ for 0< T<T_{LC}. It can be obtained that $\frac{dC(T)}{dT}$>0 for 0 <T < T_{LC} and the minimum value of C(T) can found at T=0.

2) if $\frac{\lambda(0)}{w(0)}$=a.r >$\frac{C_3}{C_2 - C_1}$ > $\frac{\lambda(T)}{w(T)}$=a.r.$e^{-r\alpha}$, there can be found a finite and unique real number

$$T_0 = -\frac{\ln\left(-\frac{\ln\left(-\frac{-r\,\alpha + \ln\left(-\frac{C3}{a\,r\,(-C2 + C1)}\right)e^{b}}{r\,\alpha}\right) - b}{b}\right)}{c} \qquad (35)$$

6.2) RELEASE TIME BASED ON RELIABILITY

Generally software release problem associated with the reliability of a software system. Here in this first we discuss the optimal time based on reliability criterion. If we know software has reached its maximum reliability for a particular time. By that we can decide right time for the software to be delivered out. Goel and Okumoto [1] first dealed with the release problem considering the software cost-benefit. The conditional reliability function after the last failure occurs at time t is obtained by

$$R(t+\Delta t/t) = \exp(-[m(t+\Delta t/t) - m(t)])$$

$$= \exp(-m(\Delta t) \times \exp(-r \times W*(t))) \qquad (36)$$

Taking the logarithm on both sides of the above equation and rearrange the above equation we obtain

$$\ln R = -m(\Delta t) \times \exp(-r \times W*(t)) \qquad (37)$$

$$\text{Thus } W*(t) = \frac{1}{r}\left[\ln m(\Delta t) - \ln\ln\left(\frac{1}{R}\right)\right] \qquad (38)$$

Another way of defining the reliability based on another model the ratio of cumulative number of error detected and initial number of errors at a given time is given by [8. 9]

R (T) =m (T)/a (39)

We can the above equation and get the unique T_1 which satisfying the above equation R (T_1) =R_0.

6.3) SOFTWARE RELEASE TIME BASED ON COST AND EFFICIENCY

Automated testing tools are useful in facilitating speedup the testing process [8, 9, 10]. Complexity of software can increase the time to test the software, it is often seen the allotted time for testing of software can exceed its required schedule time. When the situation like that arises, we adopt a new automated testing tool to increase the efficiency of the system. The new adopted automated testing tools not only speedup the testing process; it increases the efficiency of the testing by certain extent. The total cost of the testing will increase by adopting the new automated testing tools. P is described as fractions of extra errors found during the software testing phase.

The overall cost of software is rearranged to

$$C2(T) = C_0(T) + C_1\,(1+P)\,m(T) + C_2\left[m(T_{LC}) - (1+P)\,m(T)\right] + C_3\left(\int_0^T w(t)\,dt\right) \qquad (40)$$

From above C_0(T) is cost of adopting the new automated testing tools into testing phase. P is defined as the number of addition faults that has to be detected during the [8, 9, 10] testing. As the P value increases it increases the total cost of the software. C_0(T) cost may not be constant during the testing, it all depends on the nature of the testing tool used in the testing. The cost of C_0(T) is increase with testing time. In order to minimize the cost C2(T) the following relation holds between C2(T) and C1(T).

C1 (T) - C2 (T) $\ge$ 0 (41)

From eq.(33) and eq.(39) to satisfy the above equation

$$C_1\, m(T) + C_2\left[m(T_{LC}) - m(T)\right] - C_0(T) - C_1\,(1+P)\,m(T) - C_2\left[m(T_{LC}) - (1+P)\,m(T)\right] \ge 0 \qquad (42)$$

From the above equation

C_0(T)≤ P × m(T) ×(C_2-C_1)[8, 9] (43)

There are several possibilities of C_0(T) which satisfies the cost of adopting the automated testing tools during the testing phase[8,9].

a) C_0(T) is constant: in this the cost of the automated testing tools remains constant, due to engaging same type of tool in different instant of time in testing.

b) C_0(T) is proportional to test expenditure : in this an additional automated cost is added by introducing different automated tools like fixing patches, upgrading, and maintenance support into testing.

c) $C_0(T)$ is exponentially related to the test expenditure: for a large data base and certain complex software they need some extra sophisticated automated tools. By introducing these tools in different time interval during the testing phase, increases the cost of testing. A large tools require large cost, by that it increases the total cost. As the testing is progress the cost of adopting the new testing tool increases exponentially.

Theorem 1: Assume $C_0(T) = C_0$ (constant), $C_0 > 0$, $C_1 > 0$, $C_2 > 0$, $C_3 > 0$, and $C_2 > C_1$; then we have

I) $\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_s) - W(0)\right]} > C_3$,

and $\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_{LC}) - W(0)\right]} < C_3$

there exist a unique solution

$$T_0 = -\frac{\ln\left(\dfrac{\ln\left(-\dfrac{r\,\alpha}{-r\,\alpha + \ln\left(-\dfrac{C_3}{a\,r\,(-C_2 + C_1)\,(1+P)}\right) e^{\beta}}\right) + \beta}{\beta}\right)}{c}$$

then the optimal release time $T^* = T_0$.

II) If $\left(C_2\,(1+p) - C_1\,(1+p)\right) a\, r\, e^{-r\left[W(T_s) - W(0)\right]} < C_3$ then the optimal release time $T^* = T_s$.

III) If $\left(C_2\,(1+p) - C_1\,(1+p)\right) a\, r\, e^{-r\left[W(T_s) - W(0)\right]} > C_3$ then the optimal release time $T^* = T_{LC}$.

Proof: taking the derivative to the equation (39) and substituting the testing effort into the equation we get the equation

$$C_1\,(1+P)\,a\,r\,e^{-r\left[W(T) - W(0)\right]} - C_2\,(1+P)\,a\,r\,e^{-r\left[W(T) - W(0)\right]} + C_3 = 0 \quad (44)$$

$\left(C_2\,(1+p) - C_1\,(1+p)\right) a\, r\, e^{-r\left[W(T_s) - W(0)\right]} \leq C_3$

then $T_s < T < T_{LC}$ therefore software release time $T^* = T_s$.

$\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_s) - W(0)\right]} > C_3$, then $T_s < T < T_{LC}$ there exist unique optimal release time $T^* = T_{LC}$.

Theorem 2: Assume $C_0(T) = C_{01} + C_0 \times \int_{T_s}^{T} w(t)\, dt$, $C_{01} > 0$, $C_0 > 0$, $C_1 > 0$, $C_2 > 0$, $C_3 > 0$, and $C_2 > C_1$; then we have

Case1)

$\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_s) - W(0)\right]} > C_3 + C_0$,

and

$\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_s) - W(0)\right]} < C_3 + C_0$

there exist unique solution

$$T_0 = -\frac{\ln\left(-\dfrac{\ln\left(-\dfrac{-r\,\alpha + \ln\left(-\dfrac{C_3 + C_0}{a\,r\,(-C_2 + C_1)\,(1+P)}\right) e^{\beta}}{r\,\alpha}\right) - \beta}{\beta}\right)}{c}$$

satisfying

$\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_s) - W(0)\right]} = C_3 + C_0$

optimal release time $T^* = T_0$.

Case2) if $\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_s) - W(0)\right]} < C_3 + C_0$, then $T^* = T_s$.

Case3) if $\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_{LC}) - W(0)\right]} > C_3 + C_0$ then $T^* = T_{LC}$.

Theorem 3: Assume $C_0(T) = C_{01} + C_0 \times \left(\int_{T_s}^{T} w(t)\, dt\right)^k$, $C_{01} > 0$, $C_0 > 0$, $C_1 > 0$, $C_2 > 0$, $C_3 > 0$, and $C_2 > C_1$; then we have

I) if $\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_s) - W(0)\right]} > C_3$, and

$\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_{LC}) - W(0)\right]} - k \times C_0 \times \left(\int_{T_{LC}}^{T_s} w(t)\, dt\right)^{k-1} < C_3$

There exists unique solution satisfying the equation

$\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_{LC}) - W(0)\right]} - k \times C_0 \times \left(\int_{T_{LC}}^{T_s} w(t)\, dt\right)^{k-1} = C_3$

Then the optimal release time $T^* = T_0$.

II) If $\left(C_2\,(1+p) - C_1\,(1+p)\right) a\, r\, e^{-r\left[W(T_s) - W(0)\right]} < C_3$ then the optimal release time $T^* = T_s$.

III)if

$\left(C_2 \times (1+p) - C_1 \times (1+p)\right) \times a \times r \times e^{-r \times \left[W(T_{LC}) - W(0)\right]} - k \times C_0 \times \left(\int_{T_{LC}}^{T_s} w(t)\, dt\right)^{k-1} > C_3$

Then there exist unique solution $T^* = T_{LC}$.

7) NUMERICAL EXAMPLES

7.1) TOTAL COST AND RELIABILITY WITHOUT EFFICIENCY

For the dataset one from its calculated parameters α=70.55(CPU hours), β=3.304, c=0.1109, a=437.3 and r=0.03251 ,C_1=10\$, C_2=40\$, C_3=100\$ and T_{LC}=100 from the equation (35) the cost and reliability of the software are

From above table it observed that optimal software release time is around T^*=19.18 at total cost of 10251.

7.2) COST AND RELIABILITY BASED ON EFFICIENCY

For the dataset one from its calculated parameters α=70.55(CPU hours), β=3.304, c=0.1109, Assume C_{01}=1000\$, C_0=10\$, C_1=10\$, C_2=40\$, C_3=100\$, T_{LC}=100, k=1, T_s=19.

Table VII
Cost and Reliability without efficiency

Time(T)	Reliability R(T)	Total Cost	Time(T)	Reliability R(T)	Total Cost
10	0.1297	11425	18	0.4508	10257
11	0.1548	11123	19	0.4984	10249

12	0.1856	10875	20	0.5441	10255
13	0.2217	10677	21	0.5875	10273
14	0.2625	10524	22	0.6281	10298
15	0.3070	10411	23	0.6656	10329
16	0.3540	10333	24	0.7001	10364
17	0.4024	10283	25	0.7315	10400

Table VIII
Cost and Reliability with efficiency based on the cost
Function $C_0(T) = 1000 + 10\left(\int_{19}^{T} w(t)\,dt\right)$

P	Time T*	Cost C(T*)	Reliability R(T*)	P	Time T	Cost C(T*)	Reliability R(T*)
0.01	19.01	11381	0.7733	0.10	19.1	10475	0.8438
0.02	19.02	11281	0.7811	0.11	19.11	10375	0.8517
0.03	19.03	11180	0.7890	0.12	19.12	10274	0.8595
0.04	19.04	11080	0.7968	0.13	19.13	10173	0.8674
0.05	19.05	10979	0.8046	0.14	19.14	10072	0.8752
0.06	19.06	10878	0.8124	0.15	19.15	9971	0.8831
0.07	19.07	10778	0.8203	0.16	19.16	9870	0.8909
0.08	19.08	10677	0.8281	0.17	19.17	9769	0.8988
0.09	19..09	10576	0.8360	0.18	19.18	9667	0.9067

It is observed that the from eq.(33) optimal time T^*=19.18 and the cost $C1(T^*)$=10251 ; whereas from the eq.(40) the cost of the software is 9667 at P=0.18 and T^*=19.18 and its reliability has been increased from 0.51 to 0.9067. From this we can conclude that C1(T) > C2(T).

From above table we observed that as the value of P increases the optimal time increases and total cost decreases. Increases in P means we will find more and more errors during the testing. It also describes the efficiency of the software testing.

8. CONCLUSION

In this paper an analysis is made on software reliability growth model with Gompertz TEF. Our model fairly fit to the data, but Gompertz Curve is little optimistic in nature. It reaches to its peak value very quickly. If we neglect this phenomenon this model gives the realistic value in software. It is also seen that proposed Gompertz TEF in SRGM can fit for any kind of software failure data. By incorporating both TEF and test efficiency we can reduce the total testing cost and increase in the reliability.

Appendix -A

$$\frac{dm_d(t)}{dt} \times \frac{1}{w(t)} = r1 \times \left[a - m_d(t)\right] \qquad \textbf{(45)}$$

$$m_d(t) = a \times \left(1 - e^{-r \times W(t)}\right) \qquad \textbf{(46)}$$

$$\frac{dm_r(t)}{dt} \times \frac{1}{w(t)} = r2 \times \left(m_d(t) - m_r(t)\right) \qquad \textbf{(47)}$$

$$\frac{dm_r(t)}{dt} + r2 \times w(t) \times m_r(t) = r2 \times w(t) \times m_d(t) \qquad \textbf{(48)}$$

I.F of above equation $e^{\int_0^t r2 \times w(t)\,dt}$

$$m_r(t) \times e^{r2 \times W(t)} = \int_0^t r2 \times w(t) \times m_d(t) \times e^{r2 \times W(t)}\,dt \qquad \textbf{(49)}$$

solving above equation substituting $m_d(t)$

$$m_r(t) = a\left(1 - \left(\frac{\left(r1 \times e^{-r2 \times W(t)} - r2 \times e^{-r1 \times W(t)}\right)}{r1 - r2}\right)\right) \qquad (50)$$

Above equation approaches to infinity so we apply the L' Hospitals Rule by letting

$$f(r2) = \left(r1 \times e^{-r2 \times W(t)} - r2 \times e^{-r1 \times W(t)}\right) \qquad (51)$$

$$g(r2) = r1 - r2 \qquad \textbf{(52)}$$

$$\lim_{r2 \to r1} \frac{f(r2)}{g(r2)} = \lim_{r2 \to r1} \frac{(f(r2) - f(r1))}{(g(r2) - g(r1))} \qquad (53)$$

$$f'(r1) = -r1 \times W(t) \times e^{[-r1 \times W(t)]} - e^{[-r1 \times W(t)]} \qquad (54)$$

$$g'(r1) = -1$$

And $$\frac{f'(r1)}{g'(r1)} = (1 + r1 \times W(t)) \times e^{[-r1 \times W(t)]} \qquad (55)$$

Appendix -B

Using the estimated parameters α, β, and c above, we estimate the reliability growth parameters a and r in (14). Suppose that the data on the cumulative number of detected errors y_k in a given time interval $(0, t_k]$ (k = 1, 2,..., n) are observed. Then, the joint probability mass function, i.e. the likelihood function for the observed data, is given by

$$L \equiv \left\{Pr\left(N(t_1) = y_1, N(t_2) = y_2 \,.....\, N(t_n)\right) = y_n\right\}$$

$$\prod_{k=1}^{n} \frac{\left[m(t_k) - m(t_{k-1})\right]^{y_k - y_{k-1}}}{\left(y_k - y_{k-1}\right)} \times \exp\left(-m(t_n)\right) \qquad (56)$$

From eq :14 $\frac{\partial}{\partial a}\ln(L) = 0$

$$0 = \sum_{k=1}^{n} \frac{y_k - y_{k-1}}{a} - 1 + \left(1 + r\,t_n\right) e^{-r\,W(t_n)} \qquad (57)$$

$$\frac{\partial}{\partial r}\ln(L) = 0$$

$$a = \frac{y_n}{1 - \left(1 + r\,W(t_n)\right) e^{-r\,W(t_n)}} \qquad (58)$$

$$a \times \left(W\left(t_n\right)\right)^2 \times \exp\left(-r \times W\left(t_n\right)\right) = \sum_{k=1}^{n}\left(y_k - y_{k-1}\right) \times \left\{\left(\left(W\left(t_k\right)\right)^2 \times \exp\left(-r \times W\left(t_k\right)\right) - \left(W\left(t_{k-1}\right)\right)^2 \times \exp\left(-r \times W\left(t_{k-1}\right)\right)\right) \Big/ \left(\left(1 + r \times W\left(t_{k-1}\right)\right) \times \exp\left(-r \times W\left(t_{k-1}\right)\right) - \left(1 + r \times W\left(t_k\right)\right) \times \exp\left(-r \times W\left(t_k\right)\right)\right)\right\} \quad (59)$$

REFERENCES

[1] A.L. Goel and K. Okumoto, A time dependent error detection rate model for a large scale software system, *Proc. 3rd USA-Japan Computer Conference*, pp. 3540, San Francisco, CA (1978).

[2] A.Wood, Predicting software reliability, IEEE computers 11 (1996) 69–77.

[3] Bokhari, M.U. and Ahmad, N. (2006), "Analysis of a software reliability growth models: the case of log-logistic test-effort function", in Proceedings of the 17th International Conference on Modelling and Simulation (MS'2006), Montreal, Canada, pp. 540-545.

[4] Charles P. Winsor "the Gompertz Curve As Growth curve" proceedings of National Academy of Sciences, January 15, 1932.

[5] C.-Y. Huang, S.-Y. Kuo, J.Y. Chen, Analysis of a software reliability growth model with logistic testing effort function proceeding of Eighth International Symposium on Software Reliability Engineering, 1997, pp. 378–388.

[6] Goel, A.L., "Software reliability models: Assumptions, limitations, and applicability", *IEEE Transactions on Software Engineering* SE-11 (1985) 1411-1423.

[7] Huang, C.Y. and Kuo, S.Y. (2002), "Analysis of incorporating logistic testing-effort function into software reliability modeling", IEEE Transactions on Reliability, Vol. 51 No. 3, pp. 261-70.

[8] Huang, C.Y., Lyu M.R "Optimal Release time for Software systems Considering Cost, Testing-effort and Test efficiency" IEEE Transaction on reliability VOL 54 No , December 2005.

[9] Huang, C.Y., Kuo, S.Y. and Lyu, M.R. (1999), "Optimal software release policy based on cost, reliability and testing efficiency", in Proceedings of the 23rd IEEE Annual International

[10] Huang, C.Y., Kuo, S.Y. and Lyu, M.R. (2000), "Effort-index based software reliability growth models and performance assessment", in Proceedings of the 24th IEEE Annual International Computer Software and Applications Conference (COMPSAC'2000), pp. 454-9.

[11] Huang, Lyu and Kuo "An Assesment of testing effort dependent software reliability Growth model". IEEE transactions on Reliability Vol 56, No: 2, June 2007

[12] Huang and S. Y. Kuo, "Analysis and assessment of incorporating logistic testing effort function into software reliability modeling," *IEEE Trans. Reliability*, vol. 51, no. 3, pp. 261–270, Sept. 2002.

[13] Jong –Wuu Wu ,WenLiang Hung Chih Hui Tsai "Estimation of parameter of the Gompertz distribution using the least square method" 2003 Elsevier.

[14] Kapur, P.K. and Younes, S. (1994), "Modeling an imperfect debugging phenomenon with testing effort", in Proceedings of 5th International Symposium on Software Reliability Engineering (ISSRE'1994), pp. 178-83.

[15] K. Pillai and V. S. Sukumaran Nair, "A model for software development effort and cost estimation," *IEEE Trans. Software Engineering*, vol. 23, no. 8, August 1997.

[16] M. Ohba, Software reliability analysis models, IBM J. Res. Dev. 28 (1984) 428–443.

[17] M.R. Lyu, Handbook of Software Reliability Engineering, Mcgraw Hill, 1996.

[18] Pham, H. (2000), Software Reliability, Springer-Verlag,NewYork,NY.

[19] Quadri, S.M.K., Ahmad, N., Peer, M.A. and Kumar, M. (2006), "Nonhomogeneous Poisson process software reliability growth model with generalized exponential testing effort function", RAU Journal of Research, Vol. 16 Nos 1-2, pp. 159-63.

[20] R.D. Berger "comparision of the Gompertz and Logistic Equations to Describe plant Disease progress" Ecology and Epidemiology 13 Aug 1980.

[21] Xie, M. (1991), Software Reliability Modeling, World Scientific Publication, Singapore.

[22] Yamada, H. Ohtera and R. Narihisa, "Software Reliability Growth Models with Testing-Effort," IEEE Trans. Reliability, Vol. R-35, pp. 19-23 (1986).

[23] Yamada, H. Ohtera, Software reliability growth model for testing effort control, Eur. J. Oper. Res. 46 (1990) 343–349.

[24] Yarnada, S.Osalci, "Software reliability growth modeling: models and applications", *IEEE Trans. Software Engineering*, vol.l I, no.12, p.1431-1437, December 1985.

[25] Yamada, S., Ohba, M., Osaki, S., 1983. S-shaped reliability growth modeling for software error detection. IEEE Trans. Reliab. 12, 475–484.

[26] Yamada, S. and Osaki, S. (1985b), "Cost-reliability optimal release policies for software systems", IEEE Transactions on Reliability, Vol. R-34 No. 5, pp. 422-4.