Imperfect-Debugging SRGM with Software Module Testing and Resource Allocation Dependent Release Policy

Shaik.Mohammmad rafi
Assoc. Professor
Dept. of Computer science and engineering
Affiliated to J.N.T University.
Kakinada. INDIA.

Shaheda Akthar
Assoc. Professor
Dept. of Computer science and engineering
Affiliated to J.N.T University
Kakinada. INDIA

ABSTRACT
Testing is one of the important phase in software development. Main Purpose of testing is to identify the number of errors present in the software. In the history of software development several testing techniques and methods are used in finding out the errors. Module testing is one of the sophisticated testing technique. Software release problem is the one of oldest of problem in which managers could find a time at which testing is be to stopped such that released software should have more quality. During the testing many resources are consumed; every manager's intension is to find efficient method of allocating resources during software module testing such that it saves time and resource. In this paper we have combined the software release problem with resource allocation with software reliability growth model with imperfect-debugging phenomenon. Experiments are conducted on datasets. The results show our proposed model fits better than other.

General Terms
Software reliability, software testing, software release time, resource allocation

Keywords
Non-homogeneous passion process, Software reliability growth model, resource allocation, imperfect-debugging,

I. INTRODUCTION
Software development consists of four phases like Analysis, design, coding and testing. Among these testing is considered as the most important phase in which all bugs related to software are identified. Many testing techniques are implemented these days. Unit testing, integration testing, and module test. Testing intended in using to find errors and improve the product reliability. Reliability of a software product is defined as working conditions of the software before it met with first error in a given environmental conditions. If the test is more effective it could find more number of errors in short span of time and the same time if a test is not feasible if it is unable to find the errors. In the past few decades several papers are published in the context of software reliability growth models [13,15]. Recently some models proposed testing-effort dependent software reliability growth models [5, 8,11,12].

Software reliability growth models differentiated based on considering the time and number of faults [13]. The former one is called time dependent models and later one is failure dependent. An effort better described by number of persons involved in testing, number of test cases and time allocated for testing. Module test is one of the important test in which all modules related to software product is considered for testing [5]. During module testing, integration testing, and system testing many of the resources are consumed; like number of persons, number of test cases and number of hours spent [4,5].

Several papers are proposed in area of software development cost [2,3,8]. All proposed models derived their cost based on the reliability. But in reality several factors can effect the cost the software. The COCOMO [14] had considered the several factors in that reliability is one factor. Some papers considered the cost function is non linear function of failure rate [1,2,,3]. Similarly some had proposed the software reliability growth models by considering the cost and release time policy [2,3]. Kubat and Berman proposed a cost allocation models based on satisfying budget and development cost [1,16]. Yamada, Goel and Okumoto had proposed cost model based on amount of testing effort send during the software testing [1,2,3,8,11]. Generally complex software consists of several modules. Resources allocation in such kind of software products is a challenging issue. When complex software put under test each module consumes about 30-40% of test resources [16]. So a manager has to decide how he can allocate these testing resources effectively. Recent many papers are presented in efficient resource allocation [4,6,7,10,17]. It is great importance to use the optimal release policies based on resource allocation. Such that the efficient resource allocation can impact the software release time. Nishiwaki , Yamada and Ichimori [10] had proposed release policies on a resource allocation with SRGM by considering that testing is perfect in nature; but in reality by removing one fault can produces another fault or there is a chance that fault removal process in not in perfect. In this paper we use SRGM with imperfect debugging phenomenon.

In this paper we proposed an optimal release policy based on resource allocation with imperfect-debugging SRGM. Section-II describes NHPP software reliability growth model with imperfect-debugging environment. Section-III describes optimal cost estimation through resource allocation. Section-IV shows all the experimental calculated values.

2. SOFTWARE RELIABILITY GROWTH MODELING

2.1 NHPP software reliability growth model with logistic-exponential TEF imperfect-debugging environment.

An SRGM with a logistic-exponential TEF is formulated based on the following assumptions [5,8,11,12]

1) The fault removal process follows the NHPP.
2) The software system is subjected to failure at random times caused by faults in the system.

3) The mean number of faults detected in the time interval $(t,t+\Delta t)$ by the current testing effort is proportional to the mean number of remaining faults in the system.
4) The consumption of testing effort is modeled by a logistic-exponential TEF [8,9].
5) Each time a failure occurs, the fault that cause it is removed, it is possible to introduce new errors.

$$\frac{dm(t)}{dt}\times\frac{1}{w(t)}=r(t)\times[n(t)-a(t)] \qquad (1)$$

$n(t)$ is described as the sum of expected number of initial faults. We assume that

$$n(t)=a+\beta\times m(t) \qquad (2)$$

Solving above two equations (1) and (2) at conditions m(0)=0 and r(0)=r , we get the mean value function

$$m(t)=\frac{a}{(1-\beta)}\times(1-\exp[-r(1-\beta)W^*(t)]) \qquad (3)$$

Now the number of remaining faults

$$m_{remaining}(t)=n(t)-m(t)=a\times(\exp[-r(1-\beta)W^*(t)]) \qquad (4)$$

2.2. Reliability evaluation

Generally a software reliability growth model provides the measure the reliability during the testing of software. Reliability is defined as "failure free software over a period of time in a given environmental conditions. Reliability mathematically represented as [3,11,12]

$$R(t)=R(t+\delta t)=\exp(-(m(t+\delta t)-m(t))) \qquad (5)$$

Another measure of software reliability at time t is defined as the ratio of cumulative number of error detected to total of initial errors in the software.[11]

$$R(t)=\frac{m(t)}{a} \qquad (6)$$

From above two equations (5) or (6) we can calculate the reliability of the software at ant time t.

2.3. Parameter estimation

Estimating the model parameters from real datasets involves the method of MLE. Now suppose the parameters a, r, and β are determined for n observed data pairs.

Then likely hood parameters for a, r, and β in the NHPP model with m(t) in equation (3) is given by

$$L\equiv P_r\{N(t_1)=m_1,N(t_1)=m_1......N(t_n)=m_n\}$$
$$\prod_{k=1}^{n}\frac{\{m(t_k)-m(t_{k-1})\}}{(m_k-m_{k-1})}\times\exp[-(m(t_k)-m(t_{k-1}))] \qquad (7)$$

Where m0=0 for t0=0 taking the natural logarithm of the likelihood function in eq (7) we have

$$\ln L=\sum_{k=1}^{n}(m_k-m_{k-1})\ln[m(t_k)-m(t_{k-1})]$$
$$-\sum_{k=1}^{n}((m(t_k)-m(t_{k-1}))-\sum_{k=1}^{n}\ln[(m_k)-m_{k-1})!]. \qquad (8)$$

Now from (3)

$$m(t_k)-m(t_{k-1})=\frac{a}{(1-\beta)}\times\{(\exp[-r(1-\beta)W(t_{k-1})]) \qquad (9)$$
$$-(\exp[-r(1-\beta)W(t_k)])\}$$

$$\sum_{k=1}^{n}[m(t_k)-m(t_{k-1})]=m(t_n)=\frac{a}{(1-\beta)}\times(1-\exp[-r(1-\beta)W(t_n)]) \qquad (10)$$

$$\ln L=\sum_{k=1}^{n}(m_k-m_{k-1})\ln[\frac{a}{(1-\beta)}\times\{(\exp[-r(1-\beta)W(t_{k-1})]) \qquad (11)$$
$$-(\exp[-r(1-\beta)W(t_k)])\}]-\{\frac{a}{(1-\beta)}\times(1-\exp[-r(1-\beta)W(t_n)]\}$$

$$\frac{\partial\ln L}{\partial a}=\frac{\sum_{k=1}^{n}(m_k-m_{k-1})}{a}-\frac{1}{(1-\beta)}(1-\exp(-r(1-\beta)W(t_n)))=0 \qquad (12)$$

$$\therefore a=\frac{m_n(1-\beta)}{(1-\exp(-r(1-\beta)W(t_n)))} \qquad (13)$$

Same way calculate parameters r and β.

3 RESOURCE ALLOCATION PROBLEM

3.1. Problem description

Consider software consists of N number of modules which differ from their size, complexity, the kind of function they perform. Each module is tested individually for removing the errors. There fore using a SRGM with logistic-exponential TEF with imperfect-debugging environment is more suitable for the problem. Parameters for the each module is either are estimated from LSE or MLE. Here every manager should be capable to decide how to allocate the software testing resources to each module to reduce the total cost and achieves maximum reliability. And also to calculate the optimal release of the software by allocating resources efficiently.

3.2. Modeling the mean Value function

Using the SRGM with testing effort is given by eq(3) to model the number of faults removed by the time t, the mean value function of fault removal process for the i th module is given by

$$m_i(t)=\frac{a_i}{(1-\beta_i)}\times(1-\exp[r_i(1-\beta_i)W_i(t)]) \qquad i=1,2,\ldots N \qquad (14)$$

Let qi be the testing effort spend on the i th module during the testing time T; the mean value function is given by

$$m_i(q_i)=\frac{a_i}{(1-\beta_i)}\times(1-\exp[r_i(1-\beta_i)q_i])....i=1,2...N \qquad (15)$$

3.3. Modeling the cost functions

Cost is a one important factor in analyzing the release of the software. Cost of testing before release and the costs of fixing the errors before and after release are counted as software cost factors [ ]. The total cost of testing effort expenditure is given by

$$C(T)=C_1m(T)+C_2[m(T_{LC})-m(T)]+C_3W(T) \qquad (16)$$

Where C(T) is the total cost of the software and

C_1	= cost of fixing an error during testing.
C_2	= cost of fixing an error during operation where $C_2>C_1$
C_3	= cost of testing per unit time
T_{LC}	= software life cycle length
T	= software release time; amount of testing time.

New software cost function for ith module is given by

$$z_i=C_{1m}\frac{a_i}{(1-\beta_i)}\times(1-\exp[-r_i(1-\beta_i)q_i]) \qquad (17)$$
$$+C_{2m}a_i\exp[-r_i(1-\beta_i)q_i]+C_{3m}q_i$$

C_{1m}	= cost of correcting an error during module testing
C_{2m}	= cost of correcting an undetected errors during the module testing
C_{3m}	= cost of module testing for unit testing effort expenditure

Total cost of the all modules

$$Z = TC = \sum_{i=1}^{N} z_i = C_{1m}\sum_{i=1}^{N}\frac{a_i}{(1-\beta_i)}\times(1-\exp[-r_i(1-\beta_i)q_i]) \quad (18)$$

$$+C_{2m}\sum_{i=1}^{N}a_i\exp[-r_i(1-\beta_i)q_i]+C_{3m}\sum_{i=1}^{N}q_i$$

Now from the equation (11)

$$\text{Minimize } C_{1m}\sum_{i=1}^{N}\frac{a_i}{(1-\beta_i)}\times(1-\exp[-r_i(1-\beta_i)q_i]) \quad (19)$$

$$+C_{2m}\sum_{i=1}^{N}a_i\exp[-r_i(1-\beta_i)q_i]+C_{3m}\sum_{i=1}^{N}q_i$$

$$\text{Subjected to } \sum_{i=1}^{N}q_i \le W, \quad q_i \ge 0 \quad (20)$$

Use solve equation (12) and (13) we use the Lagrange multiplier method can be applied. Non linear kuhn-Tucker is the most important result of optimization. Associating the multiplier λ with Eq. (12) and E.q (13) we get the following equation

$$L(q_1,q_2,q_3.....q_N,\lambda) = C_{1m}\sum_{i=1}^{N}\frac{a_i}{(1-\beta_i)}\times(1-\exp[-r_i(1-\beta_i)q_i]) \quad (21)$$

$$+C_{2m}\sum_{i=1}^{N}a_i\exp[-r_i(1-\beta_i)q_i]+C_{3m}\sum_{i=1}^{N}q_i+\lambda(\sum_{i=1}^{N}q_i-W)$$

Now differentiating the above equation with respect to q_i

We get

$$\frac{\partial L}{\partial q_i} = C_{1m}\sum_{i=1}^{N}\times\frac{a_i}{(1-\beta_i)}(r_i(1-\beta_i)\exp[-r_i(1-\beta_i)q_i]+C_{2m}\times\sum_{i=1}^{N}(-a_i r_i(1-\beta_i)) \quad (22)$$
$$\times\exp[-r_i(1-\beta_i)q_i]+C_3+\lambda=0$$

Simplifying the above equation we get

$$q_i = \frac{-\{\sum_{i=1}^{N}\ln a_i r_i((1-\beta_i)C_2-C_1)-\ln(\lambda+C_3)\}}{r_i(1-\beta_i)} \quad (23)$$

$$\text{Let } A_i = a_i r_i[(1-\beta_i)C_2-C_1] \quad (24)$$

$$\text{Now } q_i = \frac{-\{\ln A_i-\ln(\lambda+C_3)\}}{r_i(1-\beta_i)} \quad (25)$$

$$\frac{\partial L}{\partial \lambda} = (\sum_{i=1}^{N}q_i - W) = 0 \quad (26)$$

$$\text{Now } \frac{-\{\sum_{i=1}^{N}\ln A_i-\ln(\lambda+C_3)\}}{r_i(1-\beta_i)} - W = 0 \quad (27)$$

$$\ln(C_3+\lambda) = \frac{\sum_{i=1}^{N}\frac{1}{r_i(1-\beta_i)}\ln A_i - W}{\sum_{i=1}^{N}\frac{1}{r_i(1-\beta_i)}} \quad (28)$$

$$\lambda = -C_3+\exp\left[\frac{\sum_{i=1}^{N}\frac{1}{r_i(1-\beta_i)}\ln A_i - Q}{\sum_{i=1}^{N}\frac{1}{r_i(1-\beta_i)}}\right] \quad (29)$$

If suppose λ=0 then Ak > C3 ≥ Ak+1 then

$$q_i = \max\left[0, \frac{-\{\ln A_i - \ln(C_3)\}}{r_i(1-\beta_i)}\right] \quad (30)$$

From the Lagrange multipliers

$$\lambda^* \in \{0, \lambda_1, \lambda_2, \lambda_3.......\lambda_N\} \quad (31)$$

Then optimal value of

$$q_i^* = \max\left[0, \frac{-\{\ln A_i - \ln(\lambda^*+C_3)\}}{r_i(1-\beta_i)}\right] i=1,2,3,..N \quad (32)$$

4. NUMERICAL EXAMPLES

It is assumed that parameters ai , ri , and βi are already been calculated. Total effort send during the project 1000 person/hours, C1=2, C2=10 and C3=0.5. Optimal value of λ*=0.5337.

TABLE-1
SUMMERY OF VALUES OF ALL PARAMETERS AND ALLOCATED TESTING RESOURCE EXPENDITURE q_i^*

S.No	a_i	r_i	β_i	q_i	z_i
1	68	0.00567	0.1	317.59	384.65
2	14	0.02361	0.23	69.37	84.45
3	6	0.04736	0.14	30.85	38.56
4	55	0.004652	0.45	202.25	382.85
5	15	0.01536	0.12	79.66	105.13
6	41	0.005151	0.4	148.56	308.87
7	21	0.008844	0.1	111.84	160.39
8	9	0.01614	0.38	13.36	82.71
9	22	0.004551	0.27	26.52	220.53
10	11	0.004754	0.25	0	135.97

(W=1000, C_1=2, C_2=10 and C_3=0.5)

The total cost of the software testing $Z = \sum_{i=1}^{N} z_i$ =1904.1. As We can see that if we use the perfect debugging the cost is around 1854.8. This increase in the cost due to the imperfect debugging some of faults are not completely removed which increases the expenditure and leads to increase in the total cost.

TABLE-2
SUMMERY OF VALUES OF ALL PARAMETERS AND ALLOCATED TESTING RESOURCE EXPENDITURE q_i^*

S.No	a_i	r_i	β_i	q_i	z_i
1	68	0.00567	0.1	368.43	387.56
2	14	0.02361	0.23	83.65	85.36
3	6	0.04736	0.14	37.22	38.84
4	55	0.004652	0.45	303.65	368.97
5	15	0.01536	0.12	98.86	105.71
6	41	0.005151	0.4	232.51	297.27
7	21	0.008844	0.1	144.44	161.05
8	9	0.01614	0.38	39.28	75.83
9	22	0.004551	0.27	76.34	206.51
10	11	0.004754	0.25	100.63	113.67

(W=1500, C_1=2, C_2=10 and C_3=0.5)

From the Table-2 total cost of the software testing

$$Z = \sum_{i=1}^{N} z_i = 1840.77.$$

TABLE-3
SUMMERY OF VALUES OF ALL PARAMETERS AND ALLOCATED TESTING RESOURCE EXPENDITURE q_i^*

S.No	a_i	r_i	β_i	q_i	z_i
1	68	0.00567	0.1	430.04	398.51
2	14	0.02361	0.23	100.94	88.80
3	6	0.04736	0.14	44.94	40.18
4	55	0.004652	0.45	426.52	383.75
5	15	0.01536	0.12	122.11	109.44
6	41	0.005151	0.4	334.22	307.75
7	21	0.008844	0.1	183.93	166.99
8	9	0.01614	0.38	70.70	76.35
9	22	0.004551	0.27	170.97	210.31
10	11	0.004754	0.25	0	109.13

(W=2000, C_1=2, C_2=10 and C_3=0.5)

From the Table-3 total cost of the software testing

$$Z = \sum_{i=1}^{N} z_i = 1891.21.$$

5. CONCLUSIONS

In this paper we have investigated how an imperfect-debugging can influence the resource allocation and its release policy. By allocating the testing effort efficiently to each module we can optimize the cost of testing. In this we used a imperfect debugging SRGM based on non homogeneous Poisson process. The model describes the time dependent fault detection and testing resource expenditure spent during the testing. It was observed that total cost (Imperfect-debugging) 1904.1 has been higher than (perfect-debugging cost) 1854.8.

7. ACKNOWLEDGMENT

Authors like to thanks to all authors mentioned in the references.

8. REFERENCES

[1] Koch , H.S and Kubat, P. : “Optimal Release Time for Computer Software “, *IEEE Trans. Software Eng*. Pp.323-327, Vol SE-9, No. 3, (May 1983).

[2] Okumato, K and Goel, A.L:”Optimal Release Time for Software System Based on Reliability and Cost Criteria “,J.Systems and Software pp .315-318. Vol. 1 (1980).

[3] Yamada, S and Osaki, S: “Cost-reliability Optimal Release Policies for Software Systems” *IEEE Trans. , Reliability* , pp.422-424, Vol. R-34. No.5 (Dec 1985)

[4] Ohtera , H and Yamada , S: “Optimal Allocation and Control Problem for Software Testing-resources” *IEEE Trans., Reliability,* pp.171-176 , Vol. R-39 No-2 (Jun.1990).

[5] Yamada S. and Ohtera, H and Narihisa , H :” software Reliability Growth model with Testing –effort” *IEEE Trans. Reliability* , pp. 19-23 ,Vol. ,R-35, No.1 (Apr.1986).

[6] Lyu, M.R., S. Rangarajan, and A. P. A. van Moorsel. *Optimal Allocation of Test Resources for Software Reliability Growth Modeling in Software Development*, *IEEE Trans. on Reliability* 2002; 51(2):183-192.

[7] Huang, C.Y., J. H. Lo., S. Y. Kuo and M. R. *Optimal Allocation of Testing-Resource considering Cost, Reliability, and Testing-Effort*, Dependable Computing, 2004. Proceedings. 10th *IEEE Pacific Rim International Symposium* on 3-5 March 2004:103-112.

[8] Rafi, S.K., K.Nageshwara Rao, and Shaheda Akthar. Software reliability growth model with logistic-exponential TEF and Analysis of software release policy, *International Journal on Computer Science and Engineering* 2010;2(2): 387-399.

[9] Lan, Y. and L. Leemis. The Logistic-Exponential Survival Distribution, Naval Research Logistics (NRL) 2005; 55(3): 252-264.

[10] Nishiwaki, M., S. Yamada, and T. Ichimori. T*esting-resource Allocation Policies based on an Optimal Software Release Problem*, Mathematica Japonica 1996; 43(1):91-97

[11] 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.

[12] Yamada, S, Hishitani, J and **S.** Osaki, "Software Reliability Growth Model with Weibull Testing Effort: A Model and Application," ***IEEE Trans. on Reliability,*** Vol. R-42, pp.100-105. 1993.

[13] **Goel,** A. L., and Okumoto, K., Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures, ***IEEE Trans. Reli. 20,206-2110979).***

[14] Czuchra W. Optimizing budget spending for software implementation and testing. Comput Oper Res 1999;26:731–47.

[15] Yamada S and S. Osaki, “Software Reliability Growth Modeling: Models and Applications,” *IEEE Trans. Software Eng.*, vol. 11, pp. 1,431-1,437, 1985.

[16] Berman, O. and N. Ashrafi. *Optimization Models for Reliability of Modular Software Systems*, IEEE Trans. on Software Engineering 1993; 19(11):1119-1123.

[17] Yamada, S., T. Ichimori, and M. Nishiwaki. *Optimal Allocation Policies for Testing Resource Based on a Software Reliability Growth Model,*International Journal of Mathematical and Computer Modelling 1995; 22: 295-301.