Cost Optimization of SRGM using Genetic Algorithm

Manish Saraswat
Department of MCA,
Geetanjali Institute of
Technical Studies, Udaipur

ABSTRACT
The applications of computer systems have been increased immensely during the last few decades and the system reliability is major concern which is depends upon reliability of software and hardware components. Software testing is quality assurance process which confirms that the product is error free and reliable. The reliability of software is major quality attribute which ensure failure free operations and maintainability, therefore reliability assessment is necessary. Software reliability control the optimal release time and cost of software development. In this paper various fault detection and removal strategies are discussed to increase the reliability. A software reliability growth model with imperfect debugging based on Non-homogenous Position Process (NHPP) model is incorporated. The reliability estimation is based on testing and operational reliability of systems. The various numerical parameters are examined and results are presented with the GA tool of MATLAB for optimal release policy based on cost and reliability criterion.

General Terms
Reliability, Optimization, Genetic Algorithm.

Keywords
Reliability growth, Non-homogeneous Poisson process Genetic Algorithm, Optimal release policy.

1. INTRODUCTION
Now a day computer systems are used in all aspects of our life, such as entertainment, education impartment, medical diagnosis, aviation control, command and control of nuclear reactions, robot control and so on. The size and complexity of software is also increasing, especially the software which are used in real time systems are highly complicated, because these are concurrent, intercommunicated based on trimming constraints, cross-plat-formed, heterogeneous, non-deterministic and consists of large number of modules which are developed by several methods, tools and techniques.

Software failure can cause a serious problem even fatal in critical application and loss of business, therefore a effective software testing is necessary. Testing is one of the most effective and frequently used quality assurance technique which is carried throughout the software development process. It ensures software correctness, completeness and quality and reliability of system.

Software reliability engineering is focused on developing and maintaining techniques and models used for software reliability assessment measure attributes and parameters quantitatively. In order to assess the reliability of software systems various models have been introduced by researchers during last four decades, which may include parameters like software architecture, testing techniques, software operations and software failure manifestation mechanisms. The software reliability models try to record behaviour of software during the failure with respect to time. There are three reliability modelling approaches i.e error seeing and tagging, data domain and time domain.

The time domain approach is most popular and in this software reliability modelling is to perform curve filling of observed time based failure data by the pre- specified model formulae, so that model can be parameterized with statistical techniques such as least square or maximum like hood methods. The reliability models based on various assumptions like:

a) The reliability in operational environment is to be measured is same as testing environment

b) If the failure occurs, the fault which responsible for failure must remove immediately.

c) The fault removal process should never introduce a new fault.

d) A number of faults remain in the software and these fault manifests them self to cause failure. The fault may be in statistical source in form of mathematical formulae. The number of fault and failure rate reduces when testing is carried out, resulting to this growth reliability software are often called software reliability growth models [SRGM].

Software development cost and software reliability both are interrelated to each other. If reliability is lower than optimum, the actual cost for developer may be less but it is higher for customer due to excessive repairs and maintenance while reliability is higher than optimum, the cost and release time of product is increases, due to greater and striker requirements for component assemblies, but maintenance cost of product for customer is decreases.

In our investigation testing reliability and operational reliabilities are considered. The testing reliability of the software is the probability that show the product is error free during the testing phase while the operational reliability represent the probability that the product is error free during the operation phase or execution time. Our present investigation is focused on to develop the imperfect debugging(it the chance of occurring new errors during the development or testing phase) based on non-homogenous Poisson process (NHPP) model to evaluate the software testing reliability as well as software operational reliability. Numerical parameters are examined and results are presented using genetic algorithm for optimal release policy based on cost and reliability criterion.

This paper is organized into following sections

In **Section 2** a brief overview of related work is presented. **Section 3** provides a brief overview of genetic algorithm and Matlab. **Section 4** presents the model along with necessary notations and assumptions and software testing model with testing and operational reliability is also analyzed here.

Section 5 examine the results and S**ection 6** have concluding remarks and direction for future research work

2. RELATED WORK

In software development life cycle (SDLC) testing and debugging activities are performed throughout the development process. Imperfect debugging is the phenomenon which is based on probability of new fault new fault or error insertion during the testing or development phases of software. Several authors [cf Shyur, 2003; Chang and Lie, 2009; Ahmad et al., 2010; Aggarwal et al., 2010; Rafi et al., 2010; Ahmad et al. 2011; Aktekin et al., 2013; Jain et al., 2012; Peng et al., 2014] have studied the concept of imperfect debugging in SRGM. Jain et al. [2013] discussed a warranty cost model with imperfect debugging in SRGM.

The SRGM first proposed by Jolinsky and Moranda in 1972 and till date various SRGMs have been proposed some of them are exponential failure time class models[lyu 1991] , weibull & gama[Pham, 2000] function time class models[Huang,2003], infinite failure category models failure[Xie, 1996] and Beysian model[Huang et al., 2003] and so on. A SRGM with imperfect debugging and fault removal efficient model was discussed by Purnaith et al. [2012]. Satyaprasad et al. [2010] presented genetic algorithm base imperfect debugging in SRGM. Kapur et al. [2012] discussed two dimensional SRGM by incorporating the debugging time lag function. Jain et al. [2014] discussed imperfect debugging SRGM with testing effort function.

The concepts of testing and operational reliability are major research area during the last three decades. The various researchers [cf. Farr and Simith, 1988; Yamada et al.,1992; Ferdous et al.,1995; Frank et al.,1998; Jain and Priya, 2005; Kapur and Bardhan, 2008; Bashir et al., 2008] worked on reliability assessment of software.

Genetic algorithms are search algorithms which are conceptually based on the methods, which the living organism adopts to survive in their living environment. The genetic algorithms are heuristic in nature, their performance and output efficiency can vary across the multiple runs. In genetic algorithms the tests are interpreted as optimization problem [Goldberg G, 1989]. A few articles and models are successfully implemented by researchers [cf. Minhora and Tohma, 1995; Painton and Campbell, 1995; Young et al., 1999; Levitin and Lisnianski, 2001; Dai et al., 2010; Prasad et al., 2010; Agarwal et al., 2012].

Software releasing time is very important factor and it very difficult to developers to decide when testing is performed perfectly and when releases the product in mark after reliability assessment and on what minimum cost development. The software release time determination is very important parameter which is discussed by various researchers on time to time by applying release polices and models Okumoto and Goel [1980] first discussed the optimal release policy with respect to cost and benefit point of view. Cost optimization problem is also discussed by various researchers in their publications [cf Yamada, 1994; Kimura et al., 1999; Jain and Priya, 2002]. Huang et al. [2005] discussed the optimal release time for software with respect to cost testing effort and test efficiency

Rafi et al. [2010] has discussed optimal release policy of logistics exponential testing effort function later Chaterjee et al. [2012] studied optimal release policy on SRGM with imperfect debugging and change point. Rana et al. [2012] discussed software release time and cost optimization using genetic algorithm based approach. Quadri et al. [2011] and Srivastava et al. [2012] also explained releasing time of software.

3. GENETIC ALGORITHMS

The Genetic Algorithm is based on principles of natural selection and natural genetics. Goldberg in 1989 was pioneer to applying genetic algorithm in area of search and optimization and machine learning process. It is directed random search technique, used to find global optimization solution in a complex and multidimensional Now a day's these are using variety of applications like test case generation, prioritization, scheduling, designing neural network and combinatorial optimization. The main principle of genetic algorithm is to optimize the function evolutions by the continuous changing of population of individual solutions.

The genetic algorithm relies on three basic operators' reproduction, crossover and mutation, which are described as:-

Reproduction

This operator determines how the individuals should be selected for reproduction from the existing population. The selection of individuals is based on their calculated fitness value. An individual which contribute more copies in solution has more probability of being selected in next generation. There are following two methods of individual selection:

(i) Rowlett wheel

In this method the individuals are selected by statistical approach. The individuals showing higher fitness on wheel get more copies in creation of new population while the individuals showing poor fitness are rejected.

(ii) Binary Tournament

In this method two individuals are selected randomly and an individual having more fitness is selected as one parent to construct new offspring. The probability of individual selection can be calculated as

P1= fitness

$$P1 = \frac{\text{fitness}}{\sum_{i=0} \text{len (population)} \text{ fitness}}$$

Crossover

During the reproduction a complete new population of offspring is constructed, after this crossover operator is applied to them to generate two new members for next generation. The crossover operator selects the two parents or strings having more fitness value and swaps them at appropriate points to create two new offspring for the next generation of population.

Mutation

This operator in genetic algorithm introduces a random alteration in the value of a string position. Therefore the mutation operator creates a new population by changing the string position in a single chromosome. The mutation operator mutates the gene in a chromosome by one or more string values. In a binary code parameter there is simply changing of 1 to 0 and vice versa. The mutation operator ensures a degree of diversity in the population and also prevents the saturation or stagnation in optimal point.

The overall functionality of Genetic Algorithm is done as described below:

Step 1: initialize the population randomly on the basis of program and attributes

Step 2: Calculate the fitness of individual chromosome and select chromosomes which have higher fitness for construction of next generation offspring.

Step 3: Apply the Crossover operator with high probability at appropriate points to exchange bit values or information between two chromosomes to create two new offspring for next generation.

Step 4: Apply the Mutation operator with low probability to mutate or change the bit values in one chromosome.

Step 5: Re-calculate the fitness of new generated population and insert the new population (set new population as current population).

Step 6: If termination criteria is met, stop the search otherwise go to step-2

Steps for Genetic Algorithm Implementation

3.1 MATLAB Overview

The MATLAB(Matrix Laboratory) is the scientific software which was written by Dr. Cleve Moler for Mathworks Inc. to easy access of matrix product, The MATLAB provides various tools which automate the various problem solution steps by facilitating inbuilt functions and parameters. For genetic algorithm a tool 'Genetic Algorithm and Direct Search tool Box' is provided which extends the optimization capabilities by providing the useful functions for solution of optimization based problems. The 'Genetic Algorithm and Direct search tool box' include the routines which are helpful to searching of best fitness function using genetic algorithm, simulated annealing with direct search approach. The tool box also facilitates:

- To specify constraints for optimization problem
- Population size for problem
- Number specification of elite children
- Fitness function Migration among the sub-populations

These options can be customaries by providing user-defined functions and represent the problem in a variety of data formats for example using of variables that are either mix-integer or complex. Tools also facilitate to set stopping criteria, stalling fitness limit or number of generations and to victories the fitness function to improve the execution speed.

4. MODEL DESCRIPTION

This section describes Software Reliability Growth Model (SRGM) by incorporating impact debugging for a failures. Following of notations are used to construct the model:

R_{te} ($\Delta t/t$) : Testing Reliability

R_{op} ($\Delta t/t$) : Operational Reliability

R_0 : Reliability Requirement

m(t) : Mean value function

$\lambda(t)$: Failure intensity function

$a_j(t)$: Time dependant fault contents function for j^{th} type error, j=1,2,3

a_j : Expected number of errors to be eventually detected

b_j : Error detection rate for j^{th} type error, j=1,2,3

β_j : Fault introduction rate for j^{th} type error, j=1,2,3

p_j : Content proportion of j^{th} type error, j=1,2,3

C(T) : cost function of software system at time T

R(T) : Reliability function of software system at time T

Following of assumptions are considered for software reliability growth modelling:

(i) When the errors are detected and removed from the software system, there may be possibility for new errors insertion during the fault removal process.

(ii) The probability of finding an error in a program is proportional to mean number of remaining faults in the remaining system.

(iii) The probability of introducing a new error is constant

(iv)The fault removal processes are controlled by non-homogenous Poisson Process (NHPP)

Based on the assumptions, we have following differential equation to governing of SRGM:

$$\frac{dm(t)}{dt} = b_j (a_j(t) - m(t)), \quad (1)$$

$$\frac{da_j(t)}{dt} = \beta_j \frac{dm(t)}{dt} \quad (2)$$

Equation (1) defines the fault observation and now we solve the equation under the initial conditions as:

$a_j(0) = ap_j$, m(0)=0, so obtain,

$$m(t) = \frac{ap_j}{1-\beta_j}\left[1 - e^{-b_j(1-\beta_j)t}\right], \quad (3)$$

and

$$a_j(t) = \frac{ap_j}{1-\beta_j}[1 - \beta_j \exp(-b_j(1-\beta_j)t \quad (4)$$

In this study our main purpose is to evaluate the total number of faults removed, therefore the mean value function is considered as follows:

$$m(t) = \sum_{j=1}^{3} m_j(t)$$

In this paper our main purpose is to evaluate the total number of faults removed, therefore the mean value function is considered as follows:

$$m(t) = \sum_{j=1}^{3} m_j(t)$$

Software Release Time on basis of Reliability Criteria

The decision about software releasing time is very crucial for developers because it depends upon completion of testing and achieving reliability levels. In this section we will discuss software release policy on the basis of reliability criteria. A software release time depends upon to proper judgment of reliability levels, software should be reached on optimal level of reliability before its releasing in market for real life applications.

In this paper we have considered testing and reliability constraints, both of constraints are used analyze and to understand the problem of optimal software release time. The testing and operational reliability can be expressed by following equations:

$$R_{te} = (\Delta t/t) = \exp[m(t) - m(m+\Delta t)]$$

$$R_{op} = (\Delta t/t) = \exp[\lambda(t)\,\Delta t)]$$

Release policy of Software based on Cost Criteria

Let C (T) is the total cost function incurred on the testing of software system. The cost function can be explained as:

$C(T) = C_1 m(T) + C_2 [m(T_l) - m(T)] + C_3 (T)$

Where C_1 is correction cost, error during the testing phase and C_2 is the correction cost, during the operational phase; ($C_2 > C_1$), C_3 is the cost of testing per unit testing effort expenditures and T_l is software life cycle length.

Release policy of Software based on Cost- Reliability Criteria

This section describes and suggests the optimal release policy for the software in terms of cost and reliability criterion. The ideal situations state that software should achieves optimal reliability level (R_0) in minimum cost. Our fundamental objective is to exactly determine optimal release time that minimizes the total cost to acquire the highest level of reliability (R_0).The cost optimization problem for proposed software reliability estimation model is formulated as:

Minimize C (T)

Subject to $R_{te}(\Delta t/t) \geq R_0$,

$$R_{op}(\Delta t/t) \geq R_0$$

Where $C_2 > C_1 > 0$, $C_3 > 0$; $\Delta t > 0$, $0 < R_0 < 1$.

Let T_{Rte} and T_{Rop} denotes the optimal release times satisfying equation for testing and operational reliability respectively

5. RESULT ANALYSIS

In this section suggested policies, numerical values and assumptions are examined by implementing them in MATLAB. For solution purpose default parameters are set as;

C_1=300, C_2=900, C_3=400, a=100, p_1=0.0173, p_2=0.3420, p_3=0.6407, β_1=.1, β_2=.2, β_3=.5, b_1=0.01, b_2=0.02, b_3=0.03. On the basis of these numerical parameters, various results are obtained and graphs have been plotted against numerical values.

The graphical representation of cost and reliability function with respect to time is explained. The figure 1(i-vi) represents the results for cost C(T) with respect to time(T) for the various values of b_1, b_2 ,b_3 i.e fault detection rates and β_1, β_2,β_3 i.e faults introduction rates. The plots concludes that cost C(T) decreases to minimum level after that it gradually increases by increasing the values of b_1, b_2 , β_1, β_2, β_3 but in case of increment in value b_3, the cost C(T) is increases, this is due to fact that mere cost is involved in removal of fault.

The figure 3 and Figure 2 explain effects of different parameters on operational reliability (R_{op}) and testing reliability (R_{te}).The figure 2(i-vi) shows that R(T) constantly increases with increasing values of testing time, it gradually increases for the increasing values of b_1,b_2,b_3,β_1,β_2. The plot also shows that when there is increment in value of β_3, there is decreasing trend of reliability. The figure 3 (i-vi) shows that the R(T) grows rapidly with respect to incremental time of testing. Same trend is seen for the increased values of b_1, b_2, b_3, β_1, β_2, and β_3.

Optimal Release Time Estimation using Genetic Algorithm

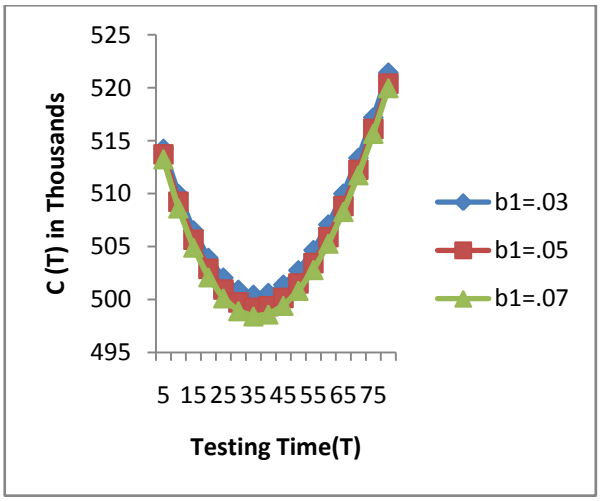
In this section we have tried to calculate the optimal release time of software by applying proposed model and genetic algorithm.

To estimate the optimal release time and corresponding cost, we have used the default parameters values as: C_1=300, C_2=900, C_3=400, a=100, p_1=0.0173, p_2=0.3420, p_3=0.6407, β_1=.1, β_2=.2, β_3=.5, b_1=0.01, b_2=0.02, b_3=0.03 for the solving optimization problem formulated in above stated equation. The table 1 describes various parameter of genetic algorithm to provide the optimal solution:

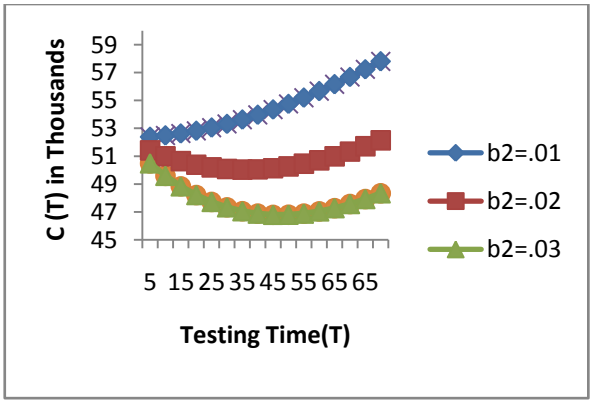
Table 1.1: Genetic Algorithm Parameters

S.No	Parameter name	Parameter value
1	Population Size	100
2	Population Type	Double Vector
3	No. Of Generations	30
4	Selection Method	Tournament
5	Cross Over Probability	0.8
6	Mutation Probability	0.1

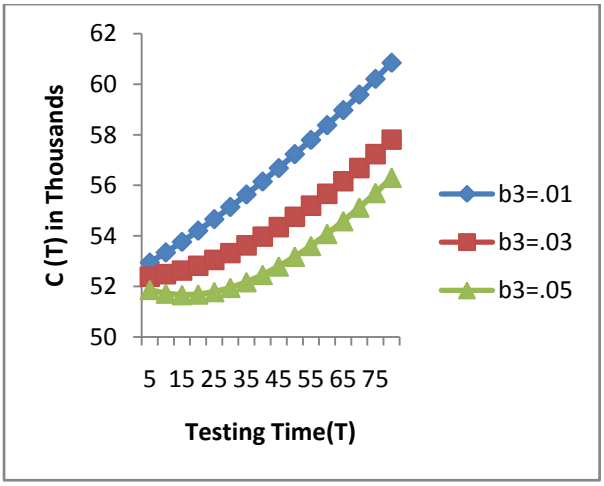
During the experiment implementation we have observed that genetic algorithm is effective, it is useful to solving software reliability growth by facilitating exact optimal release time. The optimal release time and corresponding cost obtained by genetic algorithm for the discussed illustration and parameter values is **T*= 14.231 and Cost C(T)= $ 4465364.34.**



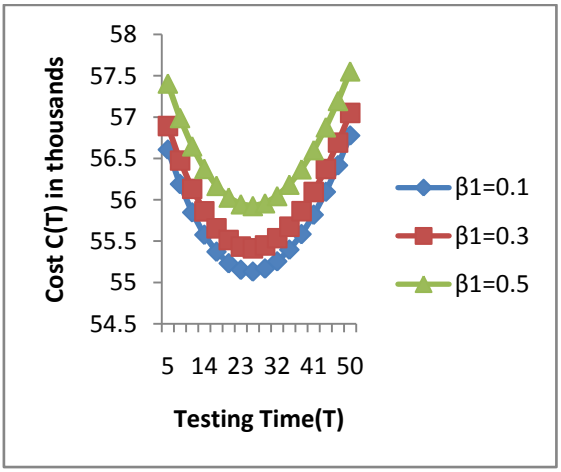
1(i)



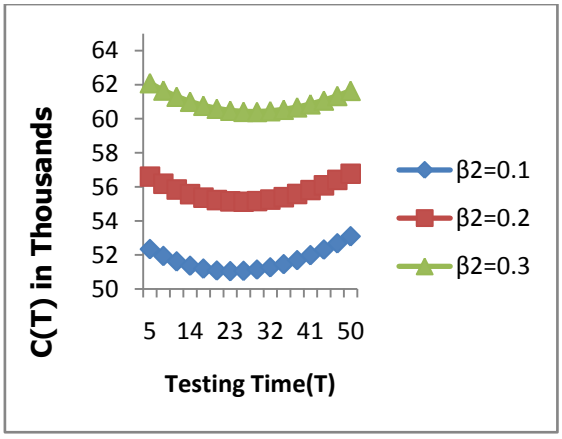
1(ii)



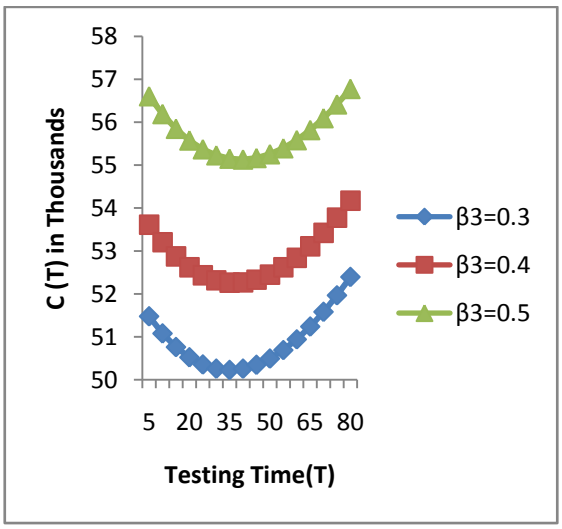
1(iii)



1(iv)

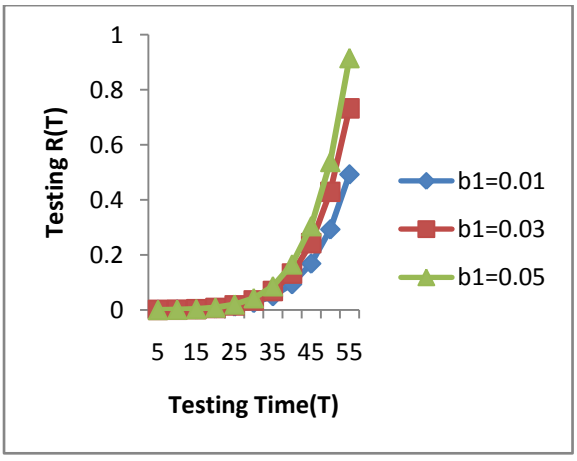


1(v)

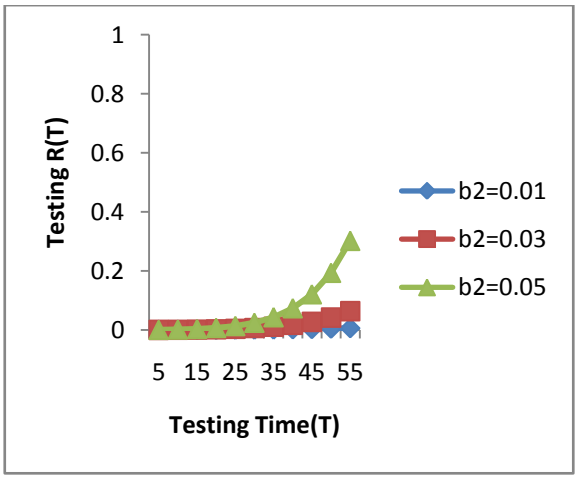


1(vi)

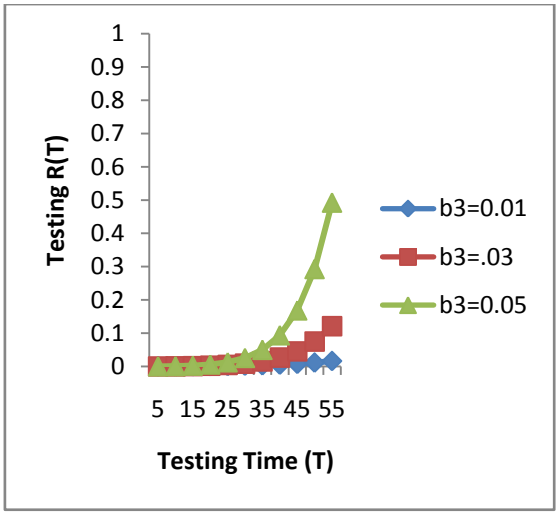
Fig: 1 C(T) vs T for Varying parameters (i) b1 (ii) b2 (iii)b3 (iv)β1 (v)β2 (vi)β3



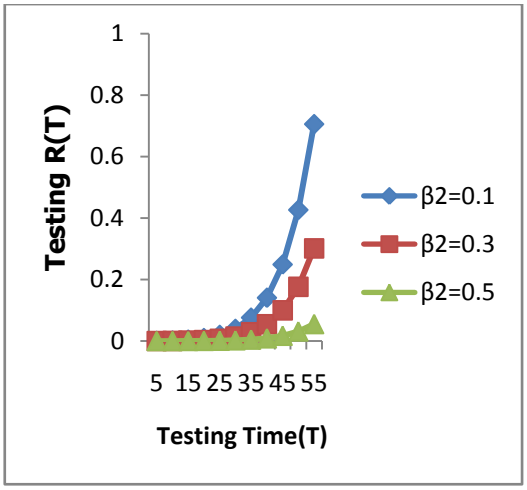
2(i)



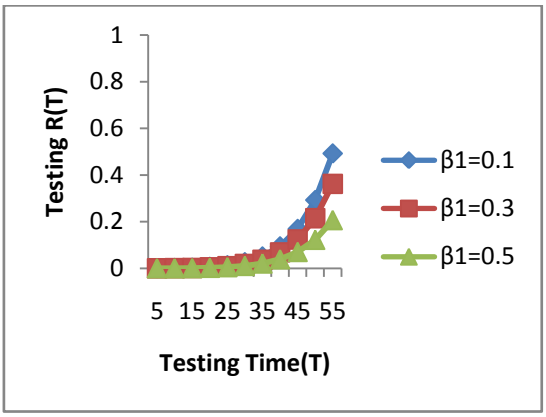
2(ii)



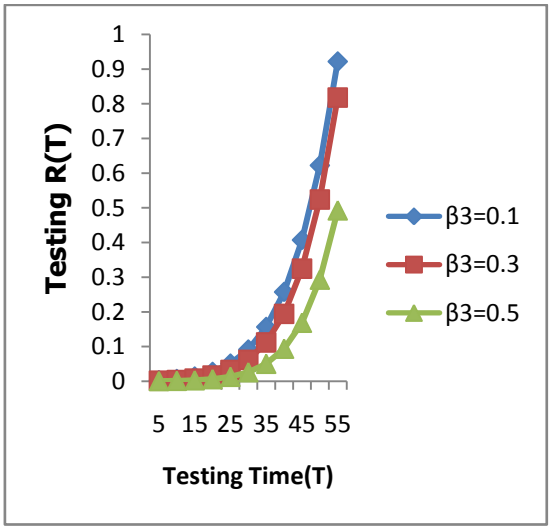
2(iii)



2(v)

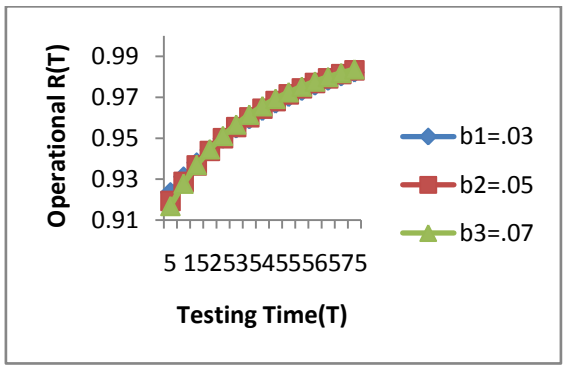


2(iv)

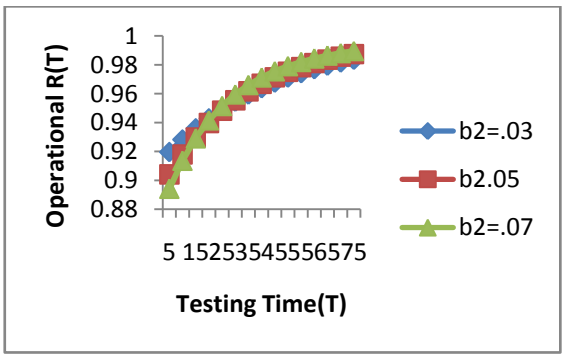


2(vi)

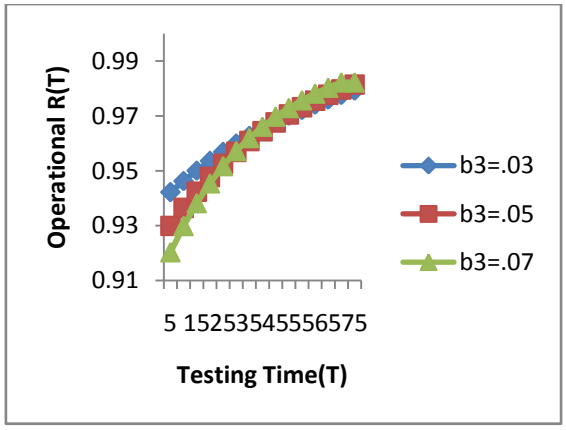
Fig:2 Testing Reliability R(T) vs T for Varying parameters (i) b1 (ii) b2 (iii)b3 (iv)β1 (v)β2 (vi)β3



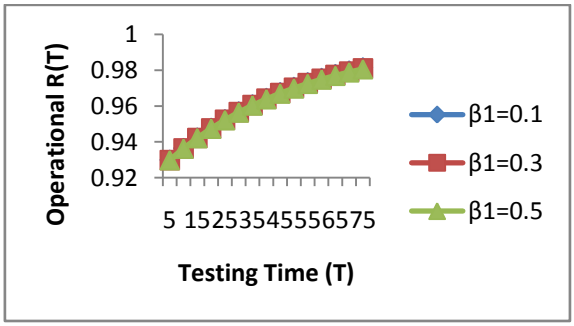
3 (i)



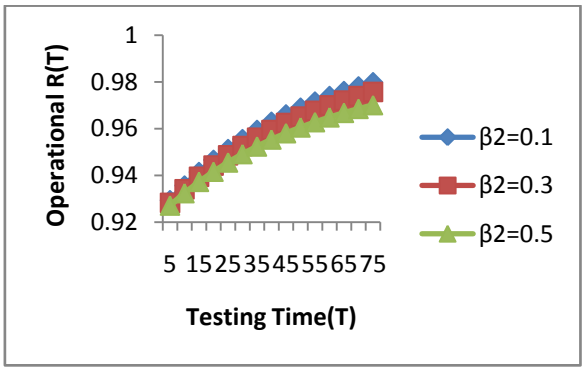
3(ii)



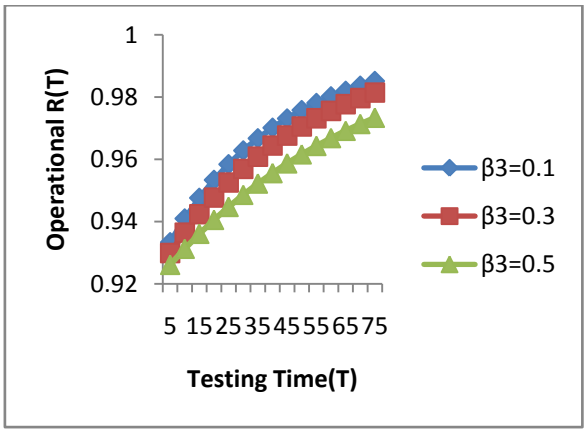
3(iii)



3(iv)



3(v)



3(vi)

Fig: 3 Operational Reliability R(T) vs T for Varying parameters (i) b1 (ii) b2 (iii)b3 (iv)β1 (v)β2 (vi)β3

6. CONCLUSIONS

In this paper we have studied Software Reliability Growth Model (SRGM) with imperfect debugging based on Non-homogenous Position Process (NHPP) and proposed a quite general and suitable model for modeling the reliability growth for many real time and embedded software systems. In this paper testing and operational reliability are considered, both of reliability criterion are helpful determine reliability constraints and to minimize the cost of software development. In this paper we have used genetic algorithm tool to calculate total expected cost of software development and optimal release time.

7. REFERENCES

[1] Frankl et.al.[1998], 'Evaluating testing methods by delivered reliability', *IEEE Transactions on Software Engineering*, 24(8), page(s) 586-601, doi: 10.1109/32.707695.

[2] Kapur and Bardhan [2006], 'Statistical Models in Software Reliability and Operations' Research, Springer Handbook of Engineering Statistics 2006, pp 477-496 Print ISBN 978-1-85233-806-0.

[3] Bashir et. al. [2008], 'Reliability and Validity of Qualitative and Operational Research Paradigm' Pakistan Journal of Statistics and Operation Research, Vol. 4. No. 1, Jan 2008, pp35-45.

[4] Tevfik and Toros[2013], 'Imperfect debugging in software reliability: A Bayesian approach', Elsevier in its journal European Journal of Operational Research Volume (Year):2013.
[4]Peng et al.[2014], *'Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction'*, Reliability Engineering & System Safety, Volume 126 . pp. 37-43. ISSN 0951-83203

[5] Agarwal et al.[2010], 'Optimal testing resource allocation for modular software considering imperfect debugging and change point using genetic algorithm', DOI:10.1109/ICRESH,2010

[6] Rafi and Akthar[2010], 'Imperfect Debugging SRGM with Software Module Testing and Resource Allocation Dependent Release Policy' International Journal of Computer applications, © 2010.

[7] Lyu [1996], 'Handbook of Software Reliability Engineering', IEEE Computer Society Press and McGraw-Hill, 1996.

[8] Pham[2000], *Software Reliability*, Springer, Singapore,2000.

[9] Xie[1991], *Software Reliability Modelling*, World Scientific Publishing Company, 1991

[10] Huang et al.[2003], 'A Unified Scheme of Some Non-Homogeneous Poisson Process Models for Software Reliability Estimation', *IEEE Transactions on Software Engineering*, vol. 29, no. 3, March 2003, pp. 261-269.

[11] Prasad et al.[2012], 'SRGM with Imperfect Debugging by Genetic Algorithms', International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 8, August 2012 ISSN: 2277 128X

[12] Rana and Bilandi[2013], 'A Genetic Based Intelligent Approach to Estimate Software Release Using Agile'*MIS Review* Vol. 18, No. 2, March (2013), pp. 19-50, 2013.

[13] Quadri et al.[2011], 'Optimal Software Release Policy Approach Using Test Point Analysis and Module Prioritization', Global Journal of Computer Science and Technology Volume 11 Issue 2 Version 1.0 February 2011,online ISSN: 0975-4172 & Print ISSN: 0975-4350.

[14] Agarwal et al.[2012], 'Genetic Algorithm Based optimal testing effort allocation problem for modular software, BVICAMs, International Journal of Information Technology ,vol. 4,No.1,pp-17.

[15] Boland and Chuiv[2007], 'Optimal Times for software release when the repair is imperfect', Statistics and probability Letters,Vol.77, No. 12,pp1176-1184.

[16] Chang and Liu[2009], 'A Generalized JM model with applications to imperfect', Vol 33, No.9, pp3578-3588.

[17] Chartterjee et al.[2012], 'Effect of change point and imperfect debugging in software reliability and its optimal release policy' Mathematical and Computer Modelling, of Dynamic Systems, Vol. 18, No. 5,pp 539-551

[18] Jain and Priya[2005] 'Software reliability issues under operational and testing constraints', Asia-Pacific Journal of Operational Research,Vol.22,No.1 pp 33-49.

[19] Jain et al.[2012], 'Software reliability growth model(SRGM) with imperfect debugging, fault reduction and multiple change-point, International Journal of Mathematics in operation research.

[20] Okumoto and Goel[1980], 'Optimum release time for software system based on reliability and cost criteria', The journal of system and software, Vol.14, No. 1,pp 315-318.

[21] Pham [1996], 'software cost model with imperfect debugging, random life cycle and penalty cost, International journal of system science, Vol 27, No 12 pp 453-463.

[22] Prasad et al. [2010], 'SRGM with imperfect debugging by genetic algorithm', International Journal of software engineering and applications, Vol. 1, No 2,pp 66-79.

[23] Minhora and Tohma[1995], 'Parameter estimation of hyper geometric distribution software reliability growth model by genetic algorithms', proceedings of sixth international journal of software Engineering.

[24] Painton and Cambell[1995], 'Genetic Algorithm in optimization of system reliability', IEEE transactions on reliability, Vol 44,No. 2, pp172-178.

[25] Farr and Smith[1988], 'A tool for statistical modelling and estimation of reliability functions for software', Journal of system software, Vol. 8 No. 1, pp 47-55.

[26] Goldberg DE[1989], 'Genetic Algorithms in search, optimization and Machine Learning', Addition-Wesley.

[27] jain et al.[2013], 'Prediction of reliability growth and warranty cost of software with fault reduction, imperfect debugging and multiple change point, international', journal of operation research.