I.J. Modern Education and Computer Science, 2014, 6, 46-52
Published Online June 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2014.06.06

Dynamic Effort Allocation Problem Using Genetic Algorithm Approach

Md. Nasar
School of Computing Science and Engineering, Galgotias University, Gr. Noida, India
Email: nasar31786@gmail.com

Prashant Johri and Udayan Chanda
School of Computing Science and Engineering, Galgotias University,Gr. Noida, India
Department of Management, Birla Institute of Technology & Science (BITS) Pilani, India
Email: {johri.prashant, udayanchanda}@gmil.com

***Abstract*—**Effort distribution plays a major role in software engineering field. Because the limited price projects are becoming common today, the process of effort estimation becomes crucial, to control the budget agreed upon. In last 10 years, numerous software reliability growth models (SRGM) have been developed but majority of model are under static assumption. The basic goal of this article is to explore an optimal resource allocation plan to minimize the software cost throughout the testing phase and operational phase under dynamic condition using genetic algorithm technique. This article also studies the resource allocation problems optimally for various conditions by investigating the activities of the model parameters and also suggests policies for the optimal release time of the software in market place.

***Index Terms*—**SRGM, optimal control theory, testing effort allocation, genetic algorithm, release time problem.

I. Introduction

Software development phase is complex but it is a major part of any Software Development Life Cycle (SDLC). The challenge will become harder when the process of development is considered in the dynamic environment. To decrease uncertainty in the process, companies often set up different kind of project/software management tools to coordinate with the other modules of the software project. Software will be released to the customers after the testing phase of SDLC. With superior development and testing efforts, superior quality software can be guaranteed. But this may be very time consuming and is unattractive in the dominant modest market conditions. Distribution of financial efforts to a software development project through the testing phase in the dynamic environment will be a critical decision that a software manager has to take. Throughout testing resources such as human effort and computer time are consumed. The error detection and removal process will completely depend upon the behavior and quantity of resources used. Several software reliability growth models (SRGMs) are proposed in the last 10 years to discuss the minimization issue of the testing effort expenditures [1], [2]. Often these models are based on the assumption that the testing effort consumption and testing time follow Rayleigh and exponential distribution. The time dependent performance of the testing effort has been studied by many authors [3-6].

Earlier studies explored that exponential curve can be used if the software testing resources are equally consumed with respect to the testing time otherwise Rayleigh curve. Weibull-type and Logistic functions will also be used to define the testing effort. Another school of thought assumes that the resource consumption can be expressed as an explicit function of the total number of faults deleted and calendar time consumed [7]. More recently, [8] have given a SRGM based on stochastic differential equations in order to study the active position of the open source development assuming that the intensity of software failure fully depends on the time, and the software fault exposure occurrence on the bug tracking system retain an unbalanced state. As discussed, over the last three decades many SRGMs have been proposed to minimize the total testing effort expenditures, but mostly under static environment. However in this paper we have tried to study an optimal resource allocation plan to optimize the testing cost of software throughout the testing phase under dynamic situation. This paper also explores the optimal resource allocation problems for various conditions by examining the behavior of the model parameters and also proposes the related release policy.

The paper uses genetic algorithm (GA) approach to discuss dynamic allocation problem. Optimal control theory is used for problem formulation and analysis of the model properties. The proposed approach is helpful to solve the dynamic nature of the problem which is difficult to solve with ordinary optimization technique.

The paper is subdivided into seven sections. Section two describes the research background related to testing effort allocation and release time problem. Section three describe about the model and its solution. We use optimal control theory and genetic algorithm for solving problem. Section four describes about basic parameter for genetic algorithm. Section five solves a numerical problem for testing effort allocation using genetic algorithm. Section

Copyright © 2013 MECS

six describe about optimal release policy for introducing software product in market for solving this problem we use genetic algorithm. Finally in section seven we conclude our results and findings.

Nomenclature

a	is the initial fault content in the software.
b	is the fault detection rate.
f(t)	is the number of fault removed at time 't'.
m(t)	is the cumulative number of fault detected till time 't' due to the testing effort $w_1(t)$.
T	the planning period.
$C_1(m(t),w_2(t))$	Cost per unit at time 't' for cumulative faults removed m(t) and debugging effort $w_2(t)$.
C_2	is the cost of testing per unit testing efforts.
W	is the total resources utilized during the SDLC at any point of time 't'.

II. Research Background

Software reliability growth model (SRGM) provides a measure to estimate upcoming failure behavior from identified or presumed features of the software. Numerous SRGMs have been given in software reliability field under some set of conventions and testing environment. The proposed SRGM in this article takes into account that the time is dependent upon variation in the testing effort. The testing efforts that direct the pace of testing for almost all kind of software projects are [7]:

(a) Manpower which contains:
- Failure detection professionals.
- Failure rectification professionals.

(b) Computer time.

The vital role of person engaged in testing is to execute test cases and match the test results with preferred specifications. Upon a failure, the error affecting it is identified and then removed by failure rectification personnel. The influence of testing effort has been included in few SRGMs [8-15]. In 1976, Myers planned that software system should be constructed and tested separately in a sequential step. [6], [5] and [16] have recommended that system level testing happened only after the system was fully developed. Recently, [17], however, suggested that software development, system debugging and software testing should be considered as simultaneous activities. [14] Investigated the association between the number of faults deleted with respect to testing effort and/or time. The authors proposed that throughout the testing phase of an SDLC, faults are removed in two stages. First a failure happens and then the fault causing that failure is corrected; hence the testing effort should be spent on two separate processes; failure detection and failure rectification. In their paper, the authors developed an SRGM incorporating time delay not only between the two phases but also through the segregation of resources between them and proposed two alternate methods for controlling the testing effort for achieving the preferred reliability or error detection level. [18] Discussed a development process environment in which system integration will take place when the total number of errors in the software will achieve a definite threshold. [19-20] studied software release policies to minimize total development cost while achieve a definite reliability objective in dynamic environment. [21] Have proposed optimal release policy for module based software.

III. Model Formulation And Solution

We initiate our study by defining a common model with a very little assumption. We are limiting our analysis to the case of a firm that controls its resources for testing and debugging under limited planning horizon. We are also consistent with the plan that the hidden faults in the software are exposed and removed throughout the testing phase, and the total number of faults residual in the software slowly decreases as the testing progresses. Hence, it is rational to assume the following differential equation:

$$f(t) = \frac{d}{dt} m(t) = bw_1(t)[a - m(t)] \quad (1)$$

Now, suppose the software firm wants to minimize the overall expenditure over the fixed scheduling horizon T. Then, the objective function for the company will be specified by;

$$\text{Min} \int_0^T [c_1(t) f(t) + c_2 w_1(t)] dt$$

Subject to

$$f(t) = \frac{d}{dt} m(t) = f(b, w_1(t)) \quad (2)$$

$$\frac{m(t)}{a} \geq m_T \geq m(T) \geq m_d (= am_T)$$

Where
$m(0)=0$ and $w_1(t)+w_2(t)=W$
$(w_1(t); w_2(t))\geq 0$ and $c_1(t) = c_1(m(t),w_2(t))$.

Here, in the above optimal problem $0<m_T<1$, we have considered the desired reliability level to be at least m_a (where m_a is unique). The planning period is [0,T] and the assumption $m(T)\geq m_d$ means that the firm aims at reaching at least the level m_a at the end of the planning period. The planning problem is to find the allocation of resources that minimizes the total expenditure.

To solve the above optimization problem, let $w_1^*(t)$ be an admissible control vector which transfers (m_0,t_0) to a target (m(T),T), where final state m(T) is specified but the final time T is not specified. t_0 and m_0 are the initial time and state, and are both fixed, i.e. $(t_0,m(0)=(0,m_0))$. Assuming that m*(t) is corresponding to $w1^*(t)$, then by Pontryagin Maximum principle, in order for $w1^*(t)$ to be

Copyright © 2013 MECS

optimal, it is essential that there exists a non-zero, continuous vector function $\lambda^*(t)$ and a constant scalar λ_0 such that [22]:

(a) $\lambda^*(t)$ and m*(t) are the explanation of the canonical system

$$\frac{\partial m^*(t)}{\partial t} = \frac{\partial H}{\partial \lambda}(m^*, \lambda^*, w_1^*, t)$$

$$\frac{\partial \lambda^*(t)}{\partial t} = -\frac{\partial H}{\partial m}(m^*, \lambda^*, w_1^*, t)$$

Where the current value of Hamiltonian function with λ_0 =1 must initially be defined, that is

$$H(m(t),w_1(t),\lambda(t),t)= -[c_1(t) f(t) + c_2 w_1(t)] + \lambda(t) f(t) = [-c_1(t) + \lambda(t)] f(t) - c_2 w_1(t) \quad (3)$$

(b) $H(m^*, \lambda^*, w_1^*, t) \geq H(m^*, \lambda, w_1, t)$

(c) All boundary conditions be satisfied.

$$W_1^*(t) \text{ maximizes } [-c_1(t) + \lambda(t)] f(t) - c_2 w_1(t) \forall w_1(t) \geq 0 \quad (4)$$

We can interpret λ(t) as the marginal value of faults at time 't', which will be negative because increasing the number of faults will increase the debugging charge. The physical explanation of the Hamiltonian H can be given as follows:

λ(t) stands for future cost incurred as one more fault is introduced in the system (at time t). Thus the Hamiltonian is the sum of testing cost $c_2w_1(t)$, current cost $c_1(t)f(t)$ and future cost $\lambda(t)f(t)$. Here, H denotes the instantaneous entire cost of the firm at time t.

The followings are two necessary conditions that hold for an optimal solution:

$$\frac{\partial H}{\partial w_1} = 0 \Rightarrow -(c_1 - \lambda) f w_1 - c_1 w_1 f - c_2 = 0$$

Other optimality condition is

$$\frac{\partial^2 H}{\partial w_1^2} \leq 0 \Rightarrow -(c_1 - \lambda) f_{w1w1} - c_{1w1w1} f - 2 c_{1w1} f_{w1} \leq 0$$

Where $c_{1w1} = \frac{\partial c_1}{\partial w_1}$ and $c_{1w1w1} = \frac{\partial^2 c_1}{\partial w_1^2}$

From the above optimality conditions, we will get the following results:

$$\Rightarrow w_1^*(t) = \frac{(\lambda(t) - c_1(t)) b(a - m(t)) - c_2}{c_{1w1(t)}(t) b(a - m(t))} \quad (5)$$

And,

$$w_2^*(t) = w - w_1^*(t)$$

The value of

$$m(t + \Delta t) = m(t) + \Delta t[b w_1 (a - m(t))] \quad (6)$$

And,

$$\lambda(t + \Delta t) = \lambda(t) + \Delta t[-b w_1(t) (c_1 - \lambda(t))] \quad (7)$$

The corresponding Hamiltonian will be given as

$$H(t)= (-c_1(t) + \lambda(t)) b w_1(t)(a - m(t)) - c_2 w_1(t) \quad (8)$$

And the adjoint variable λ(t) can be defined as

$$\frac{d}{dt}\lambda(t) = \lambda^*(t) = -b w_1(t)(c_1 - \lambda(t))] \quad (9)$$

With the transversality condition at $t=T^*$, $H(T^*)= 0$ and $\lambda(T^*)\leq 0$ (=0 if $m^*(T^*)>m_a$)

From (9) and the transversality condition of λ(T) we have

$$\lambda(t) = \lambda(T) + \int_0^T b w_1(t)(c_1 - \lambda(t))\, dt \quad (10)$$

The necessary condition for optimality is $H_{w1}=0$

Now,

$$H_{w1} = (-c_1 + \lambda) x w_1 - c_2 = (\lambda - c_1) b(a - m) - c_2 \quad (11)$$

The above optimization problem is solved by genetic algorithm approach (GA), which is categorized as a powerful computerized exploratory search and optimization method [23-26].

Copyright © 2013 MECS

GA is a popular computerized heuristic method based on the Darwin's evolutionary theory and natural genetics. Genetic Algorithm has been effectively implemented on diverse area of optimization problem like transportation, assignment problems, inventory control, job scheduling and decision-making problems based on real-life situation. The main essential feature of Genetic Algorithm (GA) is to mimic the genetic reproduction and natural evolution procedure artificially. In this process populations undertake continuous change using some genetic operators like selection, inheritance, crossover (also called recombination), and mutation. Genetic algorithm is very beneficial for solving difficult type of problems which are very hard to solve using some other algorithmic techniques. The Genetic algorithm (GA) evolution usually begins with a population of randomly generated individuals to an assumed problem where each and every individual is described using some form of encoding (Binary encoding, permutation encoding, value encoding) as a chromosome. We used binary encoding in our problem. These chromosomes will evaluate their objective function. Based on their objective functions value, chromosomes inside the population will be chosen for reproduction and selected individuals are manipulated by crossover and mutation. Basically, the crossover is used to produce offspring from two selected chromosomes. And the mutation is use for a small alteration to breed offspring. For every new generation of solutions, there is some improvement over the previous one. This method is applied iteratively until satisfies the termination criterion.

To implement the above Genetic Algorithm for the planned model, the following standard mechanisms will be measured;

(i) Genetic algorithm parameters (size of population, rate of mutation and rate of crossover)
(ii) Representation of chromosome
(iii) Population initialization
(iv) Evaluation of objective function
(v) Selection process
(vi) GA operators (Crossover, Mutation and elitism)

IV. GA Parameters

Primarily, we have to decide the different parameters on which the said GA depends. All these parameters include population size (p-size), maximum number of generation (max-gen), crossover rate (p-cros) and mutation rate (p-mute). For choosing population size there is no fixed rule for GA. If the size of population is huge, keeping of records in middle steps of genetic algorithm might raise some difficulties at execution time. If the size of population is very small, few genetic operators will not work appropriately. Regarding the number of maximum generations, there is no fixed sign to consider this value. From the natural genetics it is clear that the crossover rate is always greater than mutation rate. Generally, the crossover rate lies between 0.5 and 0.95 whereas mutation rate lies between 0.06 and 0.02.

Chromosome representation Genetic Algorithm always starts with the initial population of solutions represented as chromosomes. A chromosome comprises of genes where each gene represents a specific feature of the solution.

Initial population for a specified total testing effort W and for fault detected at time t, GA produces the initial population randomly. It will initialize random values within the bounds of every variable. The main role of the population is to hold likely solutions.

Selection In genetic algorithm selection operator plays vital role because selection is the very first operator applied to the population. The main goal of this operator is to choose the above average chromosome based on the fitness value of every chromosome and remove lower average chromosome from the population for the subsequent generation under the principle "survival of the relatively fit". In this Experiment, we have used tournament selection.

However, to solve the problem dynamically, the initial value of $\lambda(0)$ and $f(0)$ must be given first then, from objective function we will find the optimal value of effort and fault detected at time t.

Table 1. Parameters of the GA

Parameter	Value
Population Size	20
Number of Generation	100
Selection Mode	Tournament
Crossover Probability	0.8
Mutation Probability	0.2

Based on the above parameters problem is solved using MATLAB (gatool) version 7.4.0 Global Optimization Toolbox User's Guide [27].

V. Numerical analysis

Assume that $a=100$, $b=0.2$, $w1(0)=0.5$, $\lambda(0)=40$, $f(0)=2$, $C0=1000$, $c2=5000$, $ma=95$

The parameter values of 'a' and 'b' can be estimated for any given dataset outside the normative model by considering equation (1). Though these parameters are mutually correlated with w1(t) and w2(t), due to the interdependence between the two, one effort can be expressed in the form of the other. In this investigation, the main aim is to check the importance of allocation of the testing effort (w1), we use genetic algorithm to allocate testing effort.

Below figure shows the optimal allocation of testing effort with respect to time.

Copyright © 2013 MECS

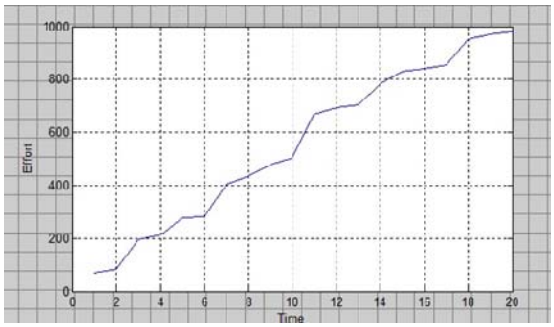

Fig 1: Allocation of testing effort w_1 versus time

VI. Software Release Policy

In this segment we will discuss the optimal time to handover a software product from development stage to operational stage. The literary meaning of the term 'to release' is to specify that the software has met a defined quality level during testing and is ready for mass distribution.

An ideal software release policy addresses the following concerns of a release manager:

- Avoid excessive time to market due to over testing.
- Manage all product associated activities, from idea to retirement
- Optimize product performance and plan upcoming investments
- Manage and increase the software products reliability.
- Balance client wants for reasonable price, delivery on time and a reliable product.
- Determine whether the software product is good enough to release to client, minimizing the risks of releasing software with serious problem.

For a given total fault content in software 'a' we generate initial population to solve this problem. And we used all above parameters.

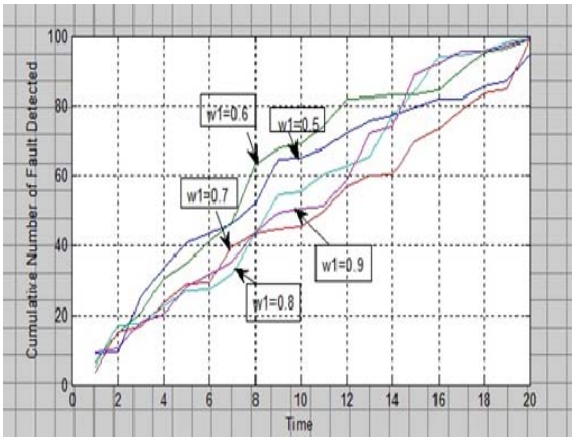

Fig 2: Cumulative number of faults removed versus time.

Table 2. Parameters of the GA for cumulative fault removing

Parameter	Value
Population Size	20
Number of Generations	80
Selection Mode	Tournament
Crossover Probability	0.8
Mutation Probability	0.2

In this study, the main goal is to check the significance of distribution of testing effort (w_1). Hence, its value has been supposed to be constant during the life cycle. During this analysis the value of w_1 is gradually increased keeping the other parameters constant and the rate of fault removal increases quickly.

The proper implementation of release decision depends on whether or not the software will be transferred from its development stage to operational stage. In other words, it is a trade-off between primary release to capture the profit of prior market introduction, and the delay of product release to enhance functionality or to improve quality. Testing and debugging process play a vital role in software release policy. For given total cost of the software, we generate initial population to solve these problems. Figure show the optimal time to release software.

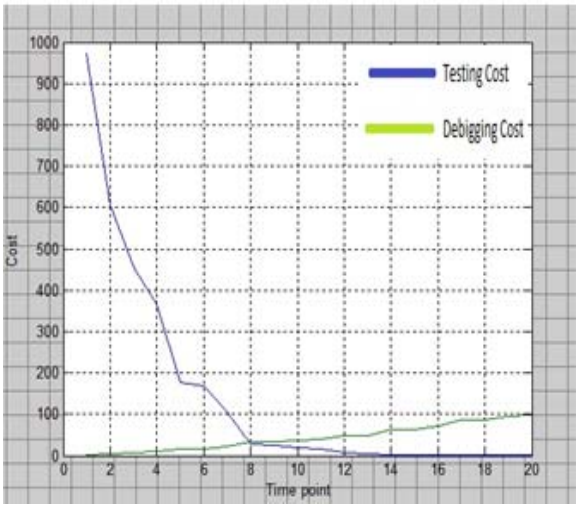

Fig 3: w1=0.9

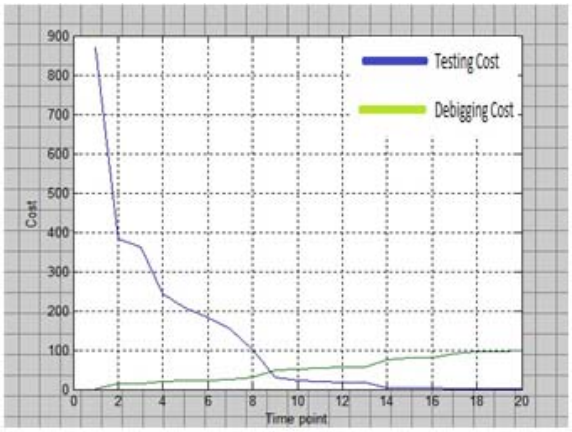

Fig 4: w1=0.93

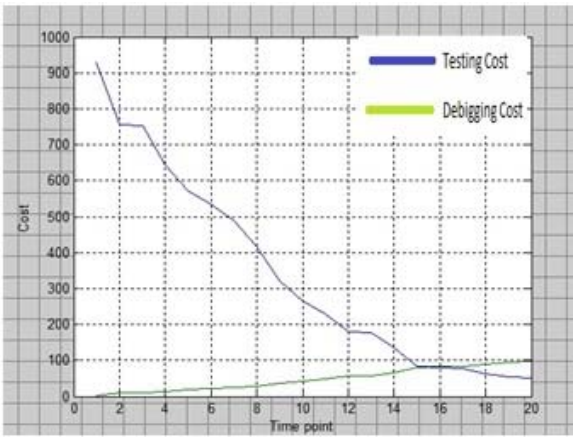

Fig 5: w1=0.95

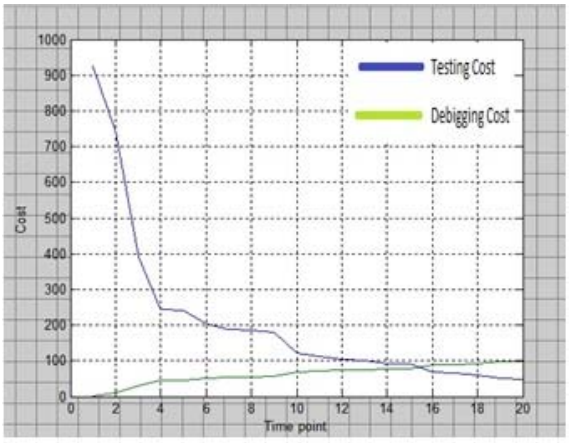

Fig 6: w_1=0.97

From the above investigation, it can be concluded that as the value of w1 decreases, the point of interaction between the cost of fixing and cost of detecting gets delayed. And, at the same time, it takes longer period to intersect them. From the above mathematical exercise it is very clear that the better time to release software is when the interaction point of fixing and detecting cost lies under minimum cost criterion.

VII. Conclusion

In this paper, we propose an alternative foundation for optimal allocation of testing resources using genetic algorithm. Using the control theoretic approach, we recommend that software testing and debugging should be viewed as simultaneous behaviors. During the investigation, we have examined allocating effort in dynamic environment using genetic algorithm. Also, we have examined that due to the experience curve phenomenon, the effort required to fix an error keeps on decreasing with time. At the same time, testing effort keeps on increasing as in the later stages of a planning period it becomes tough to detect faults. So company employs more efforts to detect the remaining faults. Using the proposed methods, we can easily control the consumption rate of testing effort expenses and discover more and more faults in a definite time interval. This means that the developers and testers can devote their time and resources to complete their testing tasks based on well controlled expenditures. This paper also studies the optimal resource allocation problems for various conditions by examining the behavior of the model parameters. A detailed optimization policy based on genetic algorithm is proposed and numerical examples are also exemplified. The release problem is also discussed. The software product's success depends on the time of its introduce time in the marketplace. In such instance, we need to conclude the optimal stopping time for testing. Based on theoretical and empirical study in this paper, we observed that the optimal release time of software is the time point where the total cost incurred due to correction coincides with the cost of detection maintaining the strict reliability constraint. Otherwise, the organization will wait for perfect time.

References

[1] Chatterjee, S., Misra, R.B., and Alam, S.S. 'Joint Effect of Test Effort and Learning Factor on Software Reliability and Optimal Release Policy', International Journal of Systems Science, 1997, 28, 391–396.

[2] Kapur, P.K., Garg, R.B., and Kumar, S. Contributions to Hardware and Software Reliability, Singapore: World Scientific 1999.

[3] Basili, V.R. and Zelkowitz, M.V., 'Analyzing Medium Scale Software Development', in Proceedings of the 3rd International Conference on Software Engineering, 1979, pp. 116–123.

[4] Kapur, P.K., and Garg, R.B. 'Cost Reliability Optimum Release Policy for a Software System with Testing Effort', OPSEARCH, 1990, 27, 109–116.

[5] Hou, R.H., Kuo, S.Y., and Chang, Y.P. 'Optimal Release Times for Software Systems with Scheduled Delivery Time Based on the HGDM', IEEE Transactions of Computer, 1997, 46, 216–221.

[6] Yamada, S., Hishitani, J., and Osaki, S. 'Software Reliability Growth Model with Weibull Testing Effort: A Model and Application', IEEE Transactions on Reliability, 1993, R-42, 100–105.

[7] Musa, J.D., Iannino, A., and Okumoto, K., Software Reliability: Measurement, Prediction, Applications, New York: Mc Graw Hill. 1987.

Copyright © 2013 MECS

[8] Tamura, Y., and Yamada, S. 'Optimisation Analysis for Reliability Assessment Based on Stochastic Differential Equation Modelling for Open Source Software', International Journal of Systems Science, 2009, 40, 429–438.
[9] Myers, G.J. Software Reliability: Principles and Practices, New York: John Wiley & Sons. 1976.
[10] Xie, M. Software Reliability Modeling, Singapore: World Scientific. 1991.
[11] Ichimori, T., Yamada, S. and Nishiwaki, M. 'Optimal Allocation Policies for Testing-resource Based on a Software Reliability Growth Model', in Proceedings of the Australia–Japan Workshop on Stochastic Models in Engineering, Technology and Management, 1993, pp. 182–189.
[12] Huang, C.-Y., Kuo, S.-Y. and Chen, J.Y. 'Analysis of a Software Reliability Growth Model with Logistic Testing Effort Function', in Proceeding of 8th International Symposium on Software Reliability Engineering, 1997, pp. 378–388.
[13] Pillai, K., and Nair, V.S.S. 'A Model for Software Development Effort and Cost Estimation', IEEE Transactions on Software Engineering, 1997, 23, 485–497.
[14] Kapur, P.K., and Bardhan, A.K. 'Testing Effort Control Through Software Reliability Growth Modelling', International Journal of Modelling and Simulation, 2002, 22, 90–96.
[15] Kapur, P.K., Gupta, A., Shatnawi, O.R., and Yadavalli,V.S.S. 'Testing Effort Control Using Flexible Software Reliability Growth Model with Change Point', International Journal of Performability Engineering, 2006, 2, 245–262.
[16] Pham, H., and Zhang., X. 'A Software Cost Model with Warranty and Risk Costs', IEEE Transactions on Computer, 1999, 48, 71–75.
[17] Blackburn, J.D., Scudder, G.D., and Van Wassenhove, L.N. 'Concurrent Software Development', Communications of the ACM, 2000, 43, 200–214.
[18] Chiang, I.R., and Mookerjee, V.S. 'A Fault Threshold Policy to Manage Software Development Projects', Information Systems Research, 2004, 15, 3–19.
[19] Jain, M. and Priya, K,. Optimal policies for software testing time. Journal of Computer Society of India, 2002. 32, 25-30.
[20] Zheng, S. Dynamic release policies for software systems with a reliability constraint. IIE Transactions, 2002. 34, 253-262.
[21] Jain, M. and Gupta, R. Optimal Release Policy of Module-Based Software. Quality Technology and Quantitative Management 2011. Vol. 8, No. 2, pp. 147-165.
[22] Sethi, S.P., and Thompson, G.L., Optimal Control Theory – Applications to Management Science and Economics (2nd ed.), New York: Springer. 2005.
[23] Goldberg, D. E., Genetic Algorithms: in Search Optimization and Machines Learning (New York: Addison-Wesley). 1989.
[24] David, L. Handbook of Genetic Algorithms. New York: Van Nostrand Reinhold. 1991.
[25] Deb K., Optimization for Engineering Design-Algorithms and Examples. Prentice Hall of India,New Delhi. 1995.
[26] Lin, C., Shen, S., Yeh, Y., & Ding, J. Dynamic Optimal Control Policy In Advertising Price and Quality. International Journal of Systems Science, 32, 2. Business Source Premier Database. 2001.
[27] Global Optimization Toolbox User's Guide R2012a The MathWorks, Inc.
[28] Chang, Y.-C. 'A Sequential Software Release Policy', Annals of the Institute of Statistical Mathematics, 2004, 56, 193–204. 7720-7725.

Authors' profiles

Md. Nasar received his BCA degree from T. M. Bhagalpur University, Bhagalpur in 2002, Master in Computer Science from G. B. Pant University of Agriculture & Technology, Pantnagar, India in 2006. He has also received Microsoft Certified Technology Specialist (MCTS). At present, he is pursuing Ph.D in Computer Science from Galgotias University, Gr. Noida, INDIA. He is having 8 years of experience in Teaching, and Software Development. His research interest includes Software Reliability and soft computing.

Dr. Prashant Johri working as a professor in school of computing science and Engineering, Galgotias University, Gr. Noida. .He received his Ph.D degree in Software Reliability from Jiwaji University Gawalior, India. He has more than 15 years of experience in teaching. He has published numerous papers in the area of software reliability in international journals and conference proceedings His area of research is software reliability, soft computing, parallel distribution and information security.

Dr. Udayan Chanda is currently working as Assistant Professor in Department of Management, Birla Institute of Technology & Science (BITS) Pilani. Earlier he was associated with Industrial Statistics Lab., Department of Information & Industrial Engineering Yonsei University as Post-Doctoral Fellow and Department of Operational Research, University of Delhi as Assistant Professor (Ad-hoc). He received his Ph.D. degree in Marketing Models and Optimization (Operational Research) from University of Delhi, Delhi. He has published numerous papers in the area of Marketing Models, Optimization, Software Reliability and Inventory Management in international journals and conference proceedings. His current research interests include Marketing Models, Inventory Modeling, Software Reliability Growth Modeling, and Dynamic Optimization Techniques.

Copyright © 2013 MECS