Incorporating Features Enhancement Archetype in Software Reliability Growth Modeling and Optimal Release Time Determination

Adarsh Anand
Department of Operational Research, University of Delhi, Delhi 110007, India

Deepika
Department of Operational Research, University of Delhi, Delhi 110007, India

Ompal Singh
Department of Operational Research, University of Delhi, Delhi 110007, India

ABSTRACT

Requirement Analysis is one of the important phases of any Software Development Life Cycle (SDLC). The competitive market scenario has made clients dynamic and informative and so in between demand of changing or addition of new functionalities is a very common scenario that firms faces now-a-days. But this up-gradation of the system often brings new complexities that might alter/increase the total fault count in the software system. The add-ons or the interfacing of the modules with other applications is cumbersome task and adds to the load of the testing team. Taking this feature intensification affect (after a certain time point) into account, in this paper, an optimal release time of the software has been identified. The proposal has been validated on real life software failure data set and results show the impact of feature enhancement.

Keywords

Software Reliability Growth Models (SRGM), Features Enhancement, Release policy.

1. INTRODUCTION

In this competitive environment the role of software is expanding very rapidly; software products get integrated and more organization uses networks for critical business operations. For large scale or international software companies, successful development of a software system depends on its software components. For example, NASA is increasingly dependent upon systems in which software is a major component, and many of these systems are critical to the success of NASA's mission. These systems must execute successfully for a specified time under specified conditions, that is, they must be reliable. The capability to provide accurate measurement of the reliability of the software in these systems before NASA accepts them is an essential part of ensuring that NASA software will meet its mission [28]. Actually, compared with hardware, software is more difficult to measure the reliability that can be achieved. Reliability depends on how the software is used so it cannot be specified absolutely [4]. The software development and testing teams play a key role in the reliability, quality and the satisfaction of the customer for the defect free operation. Software testing is a major part of quality control during the development of software that requires rigorous testing. Agile software testing practice plays a key role to ensure the delivery of the bug free software by considering that the testing is not a separate phase, but an integral part of software development, along with coding. Once software is up for release in the market it has to be free from defects. Assessment of software reliability is an important task to predict and evaluate the reliability of the software system [13]. Many SRGMs have been developed and have been used for determining software reliability during testing; they belong to two categories such as exponential and S-shaped. Goel and Okumoto proposed Non homogenous Poisson Process (NHPP) based SRGM which is purely exponential and under the assumption that the faults are uniformly distributed where each fault has an equal chance of detection. The two stage problem or S-shaped model was introduced by Yamada et al [30].Yamada et al. described the fault removal in two stage process: failure observation and corresponding removal of cause for failure. Other S-shaped model have been discussed by Ohba [21] Bittani et al. [1] Kapur et al. [9,10 and 12], Kumar et al. [18]. In many software development project it is observed that relation between the mean value numbers of error removed and the testing time is S-shaped.

Software is developed using the tools and techniques of Software reliability engineering (SRE) that enables the developers to deliver enough reliability to the software avoiding both excessive costs and development time. Software development involves a set of ordered task; each task can be called as a generic process and the process of software development is known as Software development life cycle (SDLC). The IEEE computer dictionary has defined SDLC as the period of time in which the software is conceived, developed and used [13]. The software life cycle process model describes life of the software from the conceptualization stage to the final implementation and maintenance stage. Many life cycle process models have been described in the software engineering literature. The generic process framework applicable to the vast majority of software includes the following stages: Analysis and specification, Software development, Verification and validation, maintenance and evolution. Each framework activity is populated by a set of software engineering actions such as software project tracking and control, risk management, quality assurance and measurement, technical reviews, reusability measurement, etc. Following the generic framework activities every software development and engineering organization describes a unique set of activities. Requirement analysis forms the foundation stage for building successful software. It concludes with a feasibility study of user requirements, cost benefit estimation, and documentation of collected information and feasibility report. A well developed specification can reduce the occurrence of faults in the software and minimizes the cost [13]. The development cycle is an important component of any Software Development Life Cycle (SDLC). Software is released in the market at the end of testing phase of SDLC. With larger development and testing efforts, better quality of the software can be ensured.

Within this shell of software engineering, lies an important decision making for the firms about when to stop testing and release the software in the market. This class of problem has been termed as software release time problem (SRTP) in software reliability literature and many researchers have come up with their mathematical propositions to analyze the duration of testing phase of software[13, 23]. There are many attributes and important factors that a firm has to think of before making the final release decision. As, shipping the software too early might result in pendency of faults in the software and on the other hand, if testing proceeds a very long time, the surety of reliable product increases but the cost of testing, contract penalty and loss of market initiative may constitute and even larger portion of the cost of late delivery. Hence, both, economic factors and technical factors have to be taken care of while deciding the optimal release time of the software.

Many authors have previously discussed the importance of immediate launch of the product in the market [28]. Smith and Reinertsen [27] observed that an incremental approach to product innovation is important to reduce the release time of the product. Chang [33] studied the sequential software release policy based on a state space model. During last decade, numerous number of optimization models have also been developed in the field of software reliability. Although many cost models have been developed in past starting from the model developed by Goel and Okumoto [13] to Singh at al [13] etc. Goel and Okumoto [24] introduced a simplest optimal release time model in two ways. Firstly, they discussed an unconstrained cost objective. Secondly, they considered the unconstrained reliability objective. Later other researchers have addressed this approach with different scenarios. Yamada and Osaki [31] developed a release time problem with cost minimization objective under constraint based on exponential, modified exponential and S-shaped SRGMs. Kimura et al. [17] proposed a software release time problem in which they have considered the present value and warranty period during which the developer has to pay the cost for fixing any faults detected. Huang [3] proposed an optimal release policy based on cost and reliability incorporating testing effort and efficiency. Kapur and Garg [16] proposed a release time problem based on penalty cost using the concept of releasing of the software at appropriate delivery time. Further Kapur and Garg [7] developed a release policy for maximizing the expected gain function subject to achieving a given level of failure intensity considering modified exponential and S-shaped test effort based SRGMs. Yun and Bai [34] proposed that software release time problems should assume software life cycle to be random as several factors such as availability of alternative competitive product in the market, a better announcement by the developer himself etc. plays a key role for determining the length of software life cycle. Further, Kapur et al. [5] developed a multi objective optimization problem for determination of release time considering two simultaneous objective functions as reliability maximization and cost minimization, the assigned weights to the two objective functions according to their relative importance. Many studies have been proposed and much work is under progress to determine the software reliability quantitatively [13, 23].

Later on some researchers have also worked on software random life cycle, release time problem based on penalty cost, warranty and risk cost, bi-criterion release time problem, release policy based on before and after change point, multi release software release time problem etc. [6, 8, 15, 22, 23]. Singh et al. [25] proposed a software release time problem for multi versions of software incorporating the concept of stochastic differential equation. Further Singh et al. [26] introduced a bi-criteria released problem for multi version of software using genetic algorithm. Kapur et al. [11] developed scheduled delivery time problem for multi up-gradation of software considering the cost of debugging in warranty period. Further Singh et al. [25] developed an optimal release time problem for uncertainty based successive software releases.

Software products which are introduced in the market can be one-off type (i.e no enhancement in the software) or can be product which is periodically upgraded with new features. To optimize the goals, companies try to reduce the risk by staggering the new ideas to a sequence of product- features introduced over a period of time. Many authors have proposed software reliability growth models to measure the failure rate of software. All the models are based on the assumption that when the software is first manufactured, the initial number of faults is high but then it decreases as the fault components are identified and removed. The software then enters the useful life phase when more faults are removed. Thus, the traditional software reliability growth model failed to capture the error growth due to the software enhancement in the testing phase. In the useful life phase as the software firm introduces new add-ons or features on the basis of the user needs, software will experience a drastic increase in failure rate each time an upgrade is made.

Due to the features enhancement the complexities of software is likely to be increased as the functionality of software is enhanced. Even fixing bugs may induce more software failure by fetching other defects into software. In this paper, a software reliability growth model has been proposed under the assumption that software will experience a drastic increase in the number of failures when an upgrade is made at a certain time by the software firm. The objective is to optimize the release time of the software incorporating features enhancement during testing phase and operational phase. Numerical solution has shown that the optimal release policy with features improvement reduces the relevant cost by a significant amount. Carrying these assumptions, further, the rest article is designed as follow: section 2 consists of notations that have been used in mathematical modeling. Section 3 presents a brief overview of Kapur and Garg model. The proposed modeling framework is presented in Section 4 which provides the concept of fault removal process. Further, Section 5 shows data analysis & model validation for real software failure data set and the mathematical expression for cost minimization is supplemented in Section 6. Numerical illustration of the proposed mathematical cost and conclusion are given in Section 7 and 8 respectively. Lastly, acknowledgment and the list of references have been provided.

2. NOTATION

a	Expected number of faults in the software
b_1	Detection rate before the time ‘ τ ’
b_2	Detection rate after the time ‘ τ ’
$m(t)$	Expected number of faults removed by time ‘ t ’
τ	Time at which new add-ons are added to software
α	Constant rate of fault addition due to new features add-ons in the software

β_1	Learning parameter before the time ‘ τ ’
β_2	Learning parameter after the time ‘ τ ’
T^*	Optimal time to stop testing or release time
c_1	Cost of removing one fault before time
c_2	Cost of removing one fault after time
c_3	Cost per unit testing time

3. KAPUR AND GARG MODEL [8]: A REVIEW

The model is based on the assumption that the debugging team can also remove some additional errors while removing errors without causing any failure.

Using the logistic rate function, the K-G model can be described by following mathematical structure:

$$\frac{dm(t)}{dt} = \frac{b}{1+\beta e^{-bt}}(a - m(t)) \tag{1}$$

where $m(t)$ is the cumulative number of faults removed in the software by time ‘ t ’, a total number of faults present in the software, b is the constant fault detection rate and β is the learning parameter.

4. MODEL DEVELOPMENT

The formulation of the proposed model is based on the following assumptions.

1. Failure/fault removal phenomenon is modeled by NHPP.
2. Software is subject to failures during execution caused by faults remaining in the software.
3. Failure rate is equally affected by all the faults remaining in the software.
4. Fault detection/removal rate change at any time moment.
5. Up-gradation is done continuously after a fix time τ .

Here, a software reliability growth model incorporating the effect of adding new features has been proposed in the software system. Before adding any additional feature in the system (i.e. before the time τ) the initial fault content in the software system remain constant and are detected and eliminated during the testing phase, and the number of faults remaining in the software system gradually decrease as the testing phase goes on. On the other hand adding new features like preface of new format or change in the testing team or change in expenditure pattern in the software introduces more faults in the system. Enhancement in the functionality of the software increases the fault count which requires rigours testing of the software. It is considered that firm has started incorporating additional features after a certain time ‘ τ ’and each additional feature increases the fault content at the rate ‘ α ’.The model for different situation can be given by the following differential equations:

$$\frac{dm(t)}{dt} = \begin{cases} \dfrac{b_1}{1+\beta_1 e^{-b_1 t}}(a - m(t)) & 0 \le t < \tau \\ \dfrac{b_2}{1+\beta_2 e^{-b_2 t}}(a(1+\alpha t) - m(t)) & t > \tau \end{cases} \tag{2}$$

where α is the rate of fault addition due to adding new features in the software; which lies between 0 to 1. As a part of the assumption, α increases with time because from management perspective; it is always required that new add-ons are added to the software version as soon as possible and as frequently as possible; which can actively allow us to have competitive advantage over often rivals.

under the initial condition

$$t = 0, \quad m(t) = 0$$
$$t = \tau, \quad m(t) = m(\tau)$$

On solving the equation (2), the mean value function is obtained as follows:

$$m(t) = \begin{cases} a\left(\dfrac{1-e^{-b_1 t}}{1+\beta_1 e^{-b_1 t}}\right) & t \le \tau \\ \dfrac{a}{1+\beta_2 e^{-b_2 t}}\left[\begin{array}{l} (1-\dfrac{\alpha}{b_2})(1-e^{-b_2(t-\tau)}) + \\ \left(\dfrac{1+\beta_2 e^{-b_2\tau}}{1+\beta_1 e^{-b_1\tau}}\right)(e^{-b_2(t-\tau)} - \\ e^{-b_1\tau - b_2(t-\tau)}) + \alpha(t - \tau e^{-b_2(t-\tau)}) \end{array}\right] & t > \tau \end{cases} \tag{3}$$

Hence the mean value function of the proposed framework is expressed by (3) and from this equation it can be observed that the parameter α (i.e. rate of fault addition due to additional features added) plays a significant role during estimation of the expected number of faults removal.

5. DATA ANALYSIS AND MODEL VALIDATION

The proposed model has been validated on the fault removal data set of P1 system from Yang [31]. The fault removal time of system P1was about 86 months and the cumulative number of faults removed was 4312. The time point after which new functionalities are added is taken as 32th month i.e. $(\tau = 32)$

The proposed model has been compared with K-G model on the basis of different comparison criteria. Five comparison criterion such as MSE, Bias, R^2 , Variation and RMSPE been considered to compare the models. The parameter values of the developed model have been calculated using SPSS tool based on non-linear least square method are given in Table 1. Table 2 summarizes the value of all the comparison criteria for each of the models for fault removal dataset. The predicted and estimated values seem to be closely rated claiming a fine fitness curve as shown in Fig. 1. Fig. 2 represents the pictorially comparison between proposed model and K-G model.

Table1. Model Parameter Estimation Results

Parameter	K-G Model	Proposed Model
a	4394	4400
b_1	0.102	0.026
b_2	-	0.087
β_1	339.62	16.433
β_2	-	153.29
α	-	0.001

Table 2. Model Comparison Results

Comparison Criteria	K-G Model	Proposed Model
MSE	7881.42	7784.87
Bias	-17.87	-4.76
Variation	87.46	88.61
RMSPE	89.27	88.74
R^2	0.997	0.997

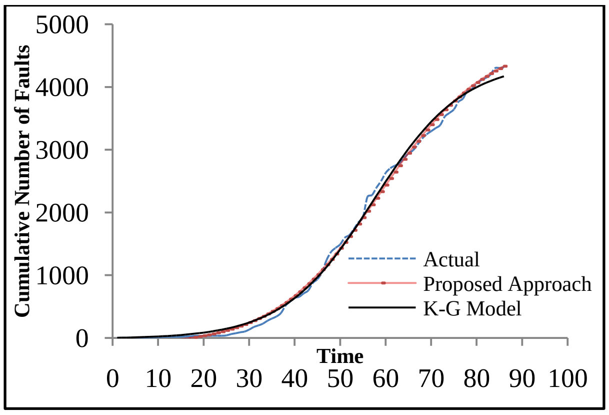

Figure 1: Goodness of Fit Curve

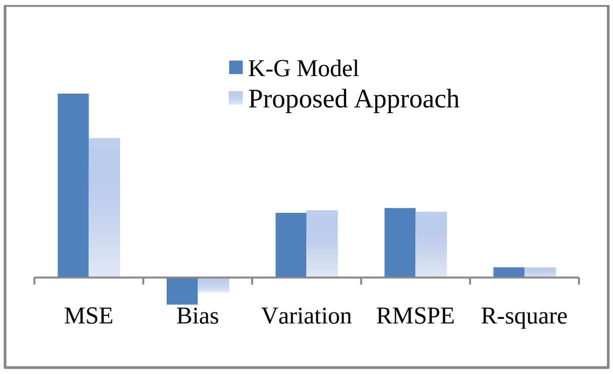

Figure 2: Comparison criteria for Proposed Approach and K-G Model

6. OPTIMAL SCHEDULING

As discussed above; the optimal software release time is a key factor for the firm. Through the use of SRGMs optimal release time of the software can be formalized and solved numerically. Here in this section the optimal software release time policy under the assumption that the software product is enhanced with new features has been discussed. Often, the introduction of new features or techniques, change in the testing team or change in the consumption pattern of testing resources may alter the fault removal rate. In such case, the optimal stopping time for testing with a desired level of reliability needs to be determined. To formulate the release time problem and minimize the cost incurred during testing and debugging under the assumption that the software product is enhanced with new features. The expected software cost function for the testing and operational phase can be written as:

$$C(T) = C_1 m(\tau) + C_2 (m(t) - m(\tau)) + C_3 \left(\frac{t}{1-\alpha} \right) \qquad (4)$$

Here $m(t)$ is the mean value function for the fault removal process. And the total cost ($C(T)$) comprises of following costs: First component in (4) describes the cost of removing one fault before time τ . Second component indicates the cost of removing faults after time τ . Third component presents the cost of testing per unit time. Here $t/(1-\alpha)$ is a time point where the time span is significantly large so more and more number of faults can be removed and the software can be considered perfectly debugged.

In addition, an optimal release time problem based on the methodology has been discussed to minimize the total expected cost of testing subjected to a reliability level to be achieved till the time of release of the software. The optimal release time problem can be formulated as follow:

$$\left.\begin{array}{ll} Min & \\ \multicolumn{2}{l}{C(T) = C_1 m(\tau) + C_2 (m(t) - m(\tau)) + C_3 \left(\dfrac{t}{1-\alpha} \right)} \\ Subject\ to & C(T) \le C_B \\ & R(t) \gtrsim R_0 \\ \text{where} & \mathrm{R(t)} \cong \dfrac{m(t)}{a} \\ & T \ge 0 \end{array}\right\} \qquad (5)$$

where C_B is the available budget.

During the testing process, maximizing software reliability is a major concern of management. A simple index to measure the reliability is the ratio of the number of cumulative detected faults at time ‘ t ’ to the mean value of initial faults in the software [19]. Hence, the reliability for release can be represented by ‘ $m(t)/a$ ’ and reliability is an increasing function of time, it reaches its maximum when time goes to infinity i.e. ‘ $m(t)/a$ ’ is the measurement of reliability.

The above formulated equation (5) is the set of optimization problem which can help the firm in determining the optimal release time of the software under the effect of features intensification.

7. NUMERICAL ILLUSTRATION

For the applicability of the above formulated methodology of determining the optimal release time and the determination of associated cost of testing. As stated earlier the data set from the system P1 reported by Yang [32] has been considered. Using this data set the parameters of proposed fault removal phenomena given in equation (4) are estimated (using SPSS) to be $a = 4400$, $b_1 = .026$, $b_2 = .087$, $\beta_1 = 16.43$, $\beta_2 = 153.29$. Assumed cost parameters are $C_1 = \$5$, $C_2 = \$9$, $C_3 = \$7$ and time after which new features are added i.e. $\tau = 32$. Further, assumed total budget $C_B = \$60,000$ and the reliability requirement by the release time is 0.77. The release time problem based on following data can be analyzed using an optimization solver (LINGO). On solving the equation (5), using the optimization tool, optimal value of the release time is $T^* = 69.59$ and expected cost is $C(T) = 29759.93$. Taking features intensification affect (after a certain point), the policy suggests that the software should be tested for a period of about 69 months. Numerical solution has shown that the optimal release policy with features improvement reduces the relevant cost by a significant amount.

8. CONCLUSION

Software market is tremendously globalized and rapidly expanding. Competition among firms require new features like preface of new format, change in the testing team or change in expenditure pattern to be inculcated in the software. Enhancement in the functionality of the software increases the fault count which requires rigours testing of the software. The time spend by the software in the testing phase will cost a firm in many ways. Early release will result into goodwill loss whereas delay will lead the firm under more competitive scenario and will also impact its market share. Thus timely release and determination of related testing period is very essential. In this situation, the optimal release time for testing with a required level of reliability needs to be determined. The objective of this paper was to propose the software reliability growth model that associates the effect of intensification of features in the software system during testing and debugging. Numerical solution has shown that the optimal release policy with features improvement reduces the relevant cost by a significant amount. Five different comparison criteria are considered for the analysis and to compare the models. The proposal has been validated on real life software failure data set and results have shown the impact of feature enhancement.

9. ACKNOWLEDGEMENT

The research work presented in this paper is supported by grants to the first and third author from University of Delhi, R&D Grant No. RC/2015/9677, Delhi, India. The authors sincerely thank the reviewers for providing valuable suggestions that resulted in clear presentation of the proposal.

10. REFERENCES

[1] Bittani, S., Bolzen, P., Pedrotti, E. and Scattolini. R. 1988. “A flexible modeling approach for software reliability growth” Software Reliabilty Modelling and Identification (Ed.) G. Goos and J. Harmanis, Springer Verlag, Berlin, pp. 101-104.

[2] Goel, A.L. and Okumoto, K. 1979. “Time dependent error detection rate model for software reliability and other performance measures”, IEEE Transactions on Reliability, 28(3).

[3] Huang, C.Y. 2004. “Performance analysis of software reliability growth models with testing effort and change-point”, The Journal of Systems and Software 76, pp. 181-194.

[4] Huang, C.Y. 2005. “Cost reliability optimal release policy for software reliability models incorporating improvement in testing efficiency”, The Journal of System Software, 77, pp. 139-155.

[5] Kapur, P.K., Agarwal, S. and Garg, R.B. 1994. “Bi-criterion Release Policy for Exponential Software Reliability Growth Models”, Recherche Operationanelle / Operational Research, 28, pp.165-180.

[6] Kapur, P.K., Garg, R.B. and Bhalla, V.K. 1993. “Release Policy with Random Software Life Cycle and Penalty Cost”, Microelectronics Reliability, 33(1), pp. 7-12.

[7] Kapur, P.K. and Garg R.B. 1991. “Optimal software release policies for software System with testing effort”, International Journal System Science, 22(9), pp. 1563-1571.

[8] Kapur, P.K. and Garg, R.B. 1992. “A Software Reliability Growth Model for an Error Removal Phenomenon”, Software Reliability Journal, pp. 291-294.

[9] Kapur, P.K., Garg, R.B. and Kumar, S. 1999. “Contribution to hardware and software reliability” Singapore World Scientific Publishing Co. Ltd.

[10] Kapur, P.K., Goswami, D. N. and Gupta, A. 2004. “ A Software reliability Growth model for distributed development Environment with learning Function and Errors of Different Severity” Published in the proceeding Mathematical Modeling Application Issue and Analysis. BITS Pilani, 8-9th October, pp. 99-110.

[11] Kapur, P.K., Pham, H. Singh, J. N. P. and Sachdeva, N. 2014. “When to Stop Testing Multi up-gradations of Software based on Cost Criteria”, International Journal of Systems Science: Operations & Logistics, Volume 1, Issue 2, pp. 84-93.

[12] Kapur, P.K., Jha, P.C. and Gupta, A. 2004. “Optimal Release Policy of a Software Fuzzy Enviroment” Published in the Proceeding Mathematical Modeling , Application Issue and Analysis, held BITS Pilani, 8-9th Oct pp. 125-134.

[13] Kapur, P.K., Pham, H., Gupta, A. and Jha, P.C. 2011. “Software reliability assessment with OR application”, Springer, Berlin.

[14] Kapur, P.K., Sachdeva, N. and Singh, J. N. P. 2014. “Optimal Cost-A Criterion to Release Multiple versions of software”, International Journal of System Assurance Engineering and Management, Volume 5, Issue 2, pp. 174-180.

[15] Kapur, P.K. and Garg, R. B. 1990. “Optimal Software Release policies for Software reliability Growth models under Imperfect Debugging”, Recherche Operationanelle /Operational Research, 24, pp. 295-305.

[16] Kapur, P.K. and Garg, R.B. 1989. “Cost reliability optimum release policies for software system under penalty cost”, International Journal System Science, 20, pp. 2547-2562.

[17] Kimura, M., Toyota, T. and Yamada, S. 1999. “Economic analysis of software release problems with warranty cost and reliability requirement”, Reliability Engineering and system safety, 66, pp. 49-55.

[18] Kumar, D., Kapur, P.K., Sehgal, V.K. and Jha, P.C. 2007 “ On the development of Software Reliability Growth Models with two types of Imperfect debugging” Communications in Dependability and Quality Management An International Journal, Volume 10, Number 3, pp. 105-122, Ohba described the software failure occurrence phenomenon with mutual dependency of faults.

[19] Lin, C.T. and Huang C.Y. 2008. “Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models”, Journal of Systems and Software 81, pp. 1025-1038.

[20] McDaid, K. and Wilson, S. P. 2001. “Deciding how long to test software”, The Statistician, 50(2), pp. 117-134.

[21] Ohba, M. 1984. “Software reliability analysis models” IBM Journal of Research and development, 28, pp. 428-443.

[22] Pham, H. 1996. “A Software Cost Model with Imperfect Debugging, Random Life Cycle and Penalty Cost, International Journal System Science, 27, pp. 455-463.

[23] Pham, H. 1999. “Software reliability”, Springer, Singapore.

[24] Pham, H. and Zhang, X. 1999. “A software cost model with warranty and risk costs”, IEEE Trans Comp 48(1), pp. 71-75.

[25] Singh, O., Aggrawal, D. and Kapur, P. K. 2012 “Reliability Analysis and Optimal Release time for a Software using Multi Attribute Utility Theory”, Communications in Dependability and Quality Management-An International Journal, Serbia, Vol. 5, No. 1, pp. 50-64.

[26] Singh, O., Kapur, P. K. and Anand, A. 2012. “A Multi Attribute Approach for Release time and Reliability Trend Analysis of a Software”, International Journal of System Assurance and Engineering Management (IJSAEM), Vol. 3, Issue 3, pp. 246-254.

[27] Smith, Preston G., and Donald G. Reinertsen. 1998. “Developing products in half the time: new rules, new tools”, New York, NY: Van Nostrand Reinhold.

[28] Stalk, G. 1998. “Time- the next source of competitive advantage”, Harvard Business Rev 66, pp. 41-51.

[29] Wallace, D. and Coleman, C. 2001. “Application and improvement of software reliability models”. Technical Report, Software Assurance Technology Centre, NASA Goddard Space Flight Centre (GSFC).

[30] Yamada, S. Obha, M. and Osaki, S. 1983. “S-shaped software reliability growth modeling for software error detection” IEEE Trans. On Reliability, R-32(5), pp. 475-484.

[31] Yamada, S. and Osaki, S. 1987. “Optimal software release policies with simultaneous cost and reliability requirements” European Journal of Operational Research, 31, pp. 46-51.

[32] Yang, K.Z. 1996. “An infinite server queueing model for software readiness assessment and related performance measures”, Ph.D. Dissertation, Department of Computer Engineering, Syracuse University, Syracuse, NY.

[33] Yen- Chang, C. 2004. “A sequential software release policy”, Ann Inst Statist Math 56 (1), pp. 193-204.

[34] Yun, W.Y. and Bai, D.S. 1990. “Optimum Software Release Policy with Random Life Cycle”, IEEE Transactions on Reliability, 39(2), pp. 338-353.