Software Release Time Determination: A Combination of Three Attributes

Dr. Subhadra Rajpoot

Abstract - The quality of the software system usually depends on how much time testing takes and what testing methodologies are used. More we spend on testing; more errors can be removed, which leads to more reliable software. However, the testing cost of the software will also increase during this process. On the other hand, if testing time is too short, the cost of the software could be reduced, but the customers may take higher risk of buying unreliable software. Therefore, it is important to determine when to stop testing, and release the software. In this paper, we propose a new method to estimate the optimal software release time by using Multi Attribute Utility Theory. More precisely, three significant attributes are used to determine the optimal release time. We apply a non-homogeneous Poisson process model to the formulation of software reliability, cost and behaviour of detection rate. It can be concluded throughout numerical examples that the existing optimal software release policy underestimates and overestimates the real optimal software release time.

Index Terms— Multi Attribute Utility function, Detection rate, Software Reliability Growth Model.

INTRODUCTION

With growing penetration of information technology, computer software is playing an important role in our lives. Software reliability becomes a problem that can't be overlooked. Many models (generally called software reliability growth models, SRGM) have been proposed to describe the software testing processes. A software reliability growth model (SRGM) can be considered to be a mathematical expression which fits the experimental data. G-O, Yamada, K-G models are most well-known models among these models [1, 8, 12, 17, 18, 19, 23]. GO model assumes that the number of defects detected up to time t follows a non-homogenous Poisson process (NHPP) with constant detection rate, Yamada and K-G assume that the total number of defects at the start of testing is a known constant, detection rate is function of time and the failure rate at any time is proportional to the number of defects remaining at that moment [2,3].

Software release is one of the most prominent issues involved in software development to decide upon the most appropriate software release plans. The problem of determining when we should stop testing emerges. If we stop testing too early, there may be too many defects in the software, which will result in too many failures during operation, and lead to significant losses due to the failure penalty or user dissatisfaction. On the other side, spending too much time in testing may result in a high testing cost and delay the introduction of the product into the market place. Therefore, there is a trade-off between software testing and releasing.

In late years, due to the significance of software application, professional testing of software becomes an increasingly important task. Once all detected faults are removed, project managers can begin to determine when to stop testing. Software reliability has important relations with many aspects of software, including the structure, the operational environment, and the amount of testing. Actually, software reliability analysis is a key factor of software quality and can be used for planning and controlling the testing resources during development [2,3]. Debian: in recent years, the project has faced increasingly delayed and unpredictable releases. Most notably, the release process of Debian 3.1 was characterized by major delays. Initially announced for December 1, 2003, the software was finally released in June 2005 – a delay of one and a half years. By the time the new version was released, the previous stable release was largely considered out of date and did not run on modern hardware [].

GNU tools: despite their popularity and importance, development has been slow in recent years and there is a long interval between releases. Version1.13 of tar came out in August 1999, followed by version 1.14 at the end of 2004. The compression utility gzip saw a new version in December 2006, more than a decade after the last stable release in 1993. As a consequence of these long delays between stable releases, several vendors started shipping pre-releases. Thus there is a trade-off, and the issue is to find an optimal point at which costs justify the stop decision [].

Yang, et. al worked on the approach taken is to minimize the expected total cost (ETC) of the software project, or further consider the software reliability requirement[24]. Yun et al. [25] study the optimal software release problem based on software reliability growth models with random life–cycle length. Yamada et al. [19,20] present an optimal software release problem based on a dual constraint of minimizing a total average software cost and satisfying a reliability requirement. Brown [2] describes a cost model to determine the optimal number of test cases.

This stream of research is closely related to the broader software reliability literature, a good summary of which is provided by Pham [6,11] and Kapur [3,4,5,7,31,32]. In all these studies, the optimal release time is determined from the cost and reliability viewpoint. Testing should continue until the gain from the improved reliability cannot justify the cost of continued testing. An implicit assumption made in these studies is that software testing stops completely after release. Recently many researchers have used Multi attribute utility theory to find the optimal release time of new version of software by combining different attributes like cost, reliability, and failure intensity. Several researchers have done work on multi attribute utility theory to determine the release time of software by combining two attributes like cost and reliability, cost and failure intensity etc. X. Li, et alproposed open source software release based on release indicator and reliability using MAUT [21]. Lately Kapur et al. solved a release problem using cost and failure intensity as attribute []. Recently Kapur et al. defined a scenario of release time using different structure of multi attribute utility function combining just two attribute reliability and cost [5]. These studies by different researcher are bound for two attribute only.

In this paper, we address the problem faced by most software managers, namely, time to stop testing or time of releasing software using three different attribute. This is a problem of decision-making under uncertainly and involves a trade-off among reliability; cost and detection rate indicator (rate of change of detection rate).

We will further investigate the modelling of fault removal process using logistic distribution functions. Specifically, the multi-attribute utility theory (MAUT) is adopted for determining the optimal time for release, where three important strategies are considered simultaneously: reliability of software, rate of change of detection rate and the acceptable level cost. The paper is outlined as follows: In Sec 2, we have discussed basic modeling of SRGM. In Section 3 we have described the theme of our discussion i.e. problem of releasing software in the market and later we have used MAUT as an evaluation approach to formulate the problem followed by numerical illustration in Sec 4. Conclusion and Acknowledgement are given in section 5 and 6 respectively.

SOFTWARE RELIABILITY MODELLING

SRGM is mathematical model. It shows how software reliability improves as faults are detected and repaired. SRGM can be used to predict when a particular level of reliability is likely to be attained. Software reliability models which assume that software failures display the behaviour of a non-homogeneous Poisson process (NHPP).The parameter of the stochastic process, which denotes the failure intensity of the software at time , is time dependent. Let denote the cumulative number of faults detected by time , and denote its expectation.

Then, and the failure intensity is related as follows:

$$m(t) = \int_0^t \lambda(s)ds$$

and, $$\frac{dm(t)}{dt} = \lambda(t)$$

$N(t)$ is known to have a Poisson probability mass function with parameter $m(t)$, that is:

$$Pr\left\{N(t) = n\right\} = \frac{m(t)^n \cdot \exp\left(-m(t)\right)}{n!}, \quad n = 0,1,2,\ldots$$

Various time dependent models have appeared in the literature which describes the stochastic failure process by an NHPP. These models differ in their failure intensity function and hence . Let ‘ ’ denote the expected number of faults that would be detected given infinite testing time in case of finite failure NHPP models. Then, the mean value function of the finite failure NHPP models can also be written as [1,6,8,9,10,11,12].

(1)

where is a distribution function.

APPLIED MODEL AND MODEL ASSESSMENT

In our model, NHPP is used to describe the time-dependent nature of the cumulative number of faults detected up to a specific testing time.
The differential equation for describing the removal phenomenon can be given by:

$$\frac{dm(t)}{dt} = b(t)\left[a - m(t) \right] = \frac{b}{(1+\beta e^{-bt})} \cdot \left[a - m(t) \right]$$

Solving the above differential equation (1), under initial condition, we get mean value function as[6, 7]:

$$m(t) = a.F(t) = a.\left(\frac{1-e^{-bt}}{1+\beta e^{-bt}} \right)$$

Logistic Distribution model proposed by Kapur and Garg is S-shape in nature. We use S-shape curve software reliability model to capture all possible behavior of data set. The analysis can be similarly carried out using any other model. Model assessment evaluates how well a data-set conforms to a chosen model. An important objective of model-based software reliability analysis is to guide decisions by providing accurate future predictions. Thus, a model that fits the observed data well, but makes poor predictions, raises serious doubts regarding its practical utility. The R2 is a complimentary measure of a model's statistical adequacy [13,14,15,16,17,18]. The value of the estimated parameters of all the models and value of R^2 are given in Table-1.

Parameter Estimation

SRGMs	Parameters			
K-G(Logistic)	a	b	β	R^2
	113	0.12	3.22	0.989

DETERMINATION OF RELEASE TIME

Decision-making is a process of choosing among alternative courses of action in order to attain goals and objectives. Release time of software, for example, involves deciding when should be done? Where? Other managerial functions, such as organizing, implementing, and controlling rely heavily on decision making.
Multi-Attribute Utility Theory (MAUT) is a structured methodology designed to handle the tradeoffs among multiple objectives. One of the first applications of MAUT involved a study of alternative locations for a new airport in Mexico City in the early 1970s. The factors that were considered included cost, capacity, access time to the airport, safety, social disruption and noise pollution. Utility theory is a systematic approach for quantifying an individual's preferences. It is used to rescale a numerical value on some measure of interest onto a 0-1 scale with 0 representing the worst preference and 1 the best [24,26,27,28,29,30].
To tackle these two conflicting factors simultaneously, multi attribute utility theory (MAUT) is adopted in our decision model. The application of MAUT can obtain a one-dimensional multi-attribute utility function, which is the measure of the attractiveness of the conjoint outcome of attributes given a specified alternative.

Let $x_1, x_2, x_3 .... x_n, n \geq 2$, be a set of attributes associated with the consequences of a decision problem. The utility of a consequence $(x_1, x_2, x_3 .... x_n)$ can be determined from
Decomposed assessment: estimate n conditional utilities $U_i(x_i)$ for the given values of the n attributes; and compute $U(x_1, x_2, x_3 .... x_n)$ by combining the $U_i(x_i)$ of all attributes:

$$U(x_1, x_2, x_3 .... x_n) = f(U_1(x_1), U_2(x_2), U_3(x_3) .... U_n(x_n)), i = 1....n$$

Identifying the Relevant Attributes
The attributes we are seeking should be the most important ones deemed relevant to the final decision. They should preferably be mutually exclusive: the attributes should be viewed as independent entities among which appropriate trade-offs may later be made. Most importantly, the chosen attributes should be measurable in a meaningful and practical way, for each of the proposed alternatives.

When considering the attributes for optimal release of any software the main objective of software industry is to prepare software which is much reliable and satisfy the customer needs. Software reliability represents a customer oriented view of software quality. Therefore maximizing software reliability is also 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. Hence, the attribute reliability can be represented by $\frac{m(t)}{a}$ and it should be maximized.

$$Maximize\,R = \frac{m(t)}{a} \qquad (5)$$

Where the approximated reliability R is attribute in MAUT. Since the reliability is an increasing function of time, it reaches its maximum when time goes to infinity.

Many software reliability models (SRMs) have been proposed to explain the software fault detection process and to quantify the different metrics that are related to system reliability [13,14,15,16,17,18]. These SRMs follow a wide variety of fault detection rates. A majority of these SRMs first choose a well-known mathematical distribution to characterize the software fault detection rate, and then interpret the parameters of the chosen distribution in the context of the testing process. For example, an early SRM developed by Goel and Okumoto assumed a constant fault detection rate. Yamada used time dependent detection rate. Common sense, however, suggests that one must first identify the characteristics or behaviour of the detection rate that drives the fault detection process followed by cumulative number of faults i.e. how detection rate is changing as testing process goes on. To measure the rate of change of detection rate we differentiate the detection rate $b(t)$ used in equation (2).

$$b^{'}(t) = \frac{b^2 \beta e^{-bt}}{(1+\beta e^{-bt})^2} \qquad (6)$$

We use $b^{'}(t)$ as an attribute to measure the behaviour of detection rate. It is reasonable to assume that the $b^{'}(t)$ follows a hump-shaped curve. It is worth noting here that $b^{'}(t)$ reaches its maximum value $b^{'}(t)_{\max} = \frac{b^2}{4}$ at $t = \frac{1}{b}\log\beta$. This new attribute is called Detection Rate Indicator (DRI) and defined as which is to maximize.

$$MaximizeD = \frac{b^{'}(t)}{b^{'}(t)_{\max}} \qquad (7)$$

On the other hand the software performance during the field is dependent on the reliability level achieved during testing. In general, it is observed the longer the testing phase, the better the performance. Better system performance also ensures less number of faults required to be fixed during operational phase. On the other hand prolonged software testing unduly delays the software release. Here we wish to determine the optimal testing time of software so that the total expected costs of the software can be minimized. For this purpose we construct a cost model for the software by assuming that there are three type of cost

$$C(T) = C_1 m(T) + C_2\left[a - m(T)\right] + C_3 T$$

Where,
C_1 be the cost of fixing a fault during testing phase.
C_2 be the cost of fixing a fault during operational phase.
C_3 is the testing cost per unit testing time.
$m(T)$ is the expected number of faults removed during testing phase.
$C(T)$ is the total cost.

A firm never wants to spend more than its capacity, therefore the next attribute that we consider is:

$$minC = \frac{C(T)}{C_{\mathbf{B}}}$$

Where, C_B is the total budget allocated to the firm.

Elicitation of single utility function for each attribute

The single utility function for each attribute represents management's satisfaction level towards the performance of each attribute. It is usually assessed by a few particular points on the utility curve [24,26,27]. More specifically, suppose that the single utility function for cost is to be determined, the worst and best values of cost are selected first as C^+ and C^-. These values are of great importance because C^+ and C^- represent its lowest cost and its highest cost expectation respectively. At these boundary points, we have $u(C^+) = 1$ and $u(C^-) = 0$. Finally, to determine functional form of utility functions either an additive or exponential form needs to examine through interviews, surveys or lottery. It may be noted that we use lottery when there is a preference or indifference between two lotteries. If they are equal to each other, management is risk neutral and the linear form should be used. Otherwise, if management is not risk neutral then the exponential form will be selected.

$$u(C) = l + m.C \; or \; u(C) = l + m\exp(k.C)$$

Where, l, m, k are constants. The single utility functions $u(R)$ and $u(D)$ for the reliability can be obtained as well [26,28].

Estimation of weight parameters

In this section we have discussed about estimation of weight parameters w_c, w_D and w_R. There are two common methods to assess the scaling constants: certainty scaling and probabilistic scaling [28,29,30]. Given that the number of attributes considered in our problem is only three and this is a small number, the probabilistic scaling technique is recommended for use.

In probabilistic scaling, management is asked to compare the choices. Let (R^+, D^+, C^+) and (R^-, D^-, C^-) denote the best and worst possible consequence respectively. There is a certain joint outcome (R^+, D^+, C^-) comprised three attribute R, D and C at the best and worst level with probability p, q and $(1-p-q)$, respectively. In these situations, the weight for attribute R equals p, where p is the indifference probability among them, [29,30]. At indifference, $q\, and\, (1-p-q)$ is equal to the weight parameters for the detection rate indicator and cost. Since the sum of weight parameters must be equal to one, the other weight parameter w_D w_R can be obtained with ease.

Structure of Multi Attribute Utility Function (MAUF)

Based on the previously estimated single utility functions and scaling constants, choosing the structure of the multi-attribute utility function is important.
The additive linear form of the MAUF is given as [26,27,29,30]:

$$U(R,D,C) = w_R \times u(R) + w_D \times u(D) + w_C \times u(C)$$

$$w_R + w_D + w_C = 1$$

Where w_R, w_D and w_c are the weight parameters for attribute R, D and C respectively. $u(R)$, $u(D)$ and $u(C)$ are the single utility function for each attribute i.e. for reliability, detection rate indicator and cost respectively. From the manager's point of view R and D are to be maximized while cost attribute C is to be minimized. To synchronize the two utility together, we convert minimization of cost utility by multiplying "–'sign before cost utility. By maximizing multi-attribute utility functions, the optimal time to release, T^* will be obtained.

NUMERICAL EXAMPLE

Data [22] comprises of four successive releases. The proposed decision model has been validated for its first release. We will find the release time of software for using first release data. The first version of software is released after 20 weeks.
For management, it is of utmost importance to predict the optimal release time. The Multi Attribute Utility Theory approach is used to determine the release time. Multi-Attribute Utility Theory (MAUT) is a label for a family of methods. These methods are a means to analyze situations and create an evaluation process. The objective of MAUT is to attain a conjoint measure of the attractiveness (utility) of each outcome of a set of alternatives.

Quantification of Attributes

As discussed, reliability R D and cost C are three important factors for management to determine the optimal release. Based on the failure data shown in Table 1, the model parameters can be estimated as shown in Table 2. Then, these three attributes are quantitatively measured by (5) (7) and (8).
The reliability attribute defined in equation (6) is the ratio of number of faults removed up to time t to the total number of faults in the software. The number of faults removed up to time t, $m(t)$ reaches its maximum value at $t_{\max} = \infty$.
Similarly for the second attribute given by equation (7) is the ratio of change in detection rate to maximum change in detection rate.
For other attribute i.e. cost we use the cost model as discussed earlier in Section (3).We set parameters $C_1 = 15, C_2 = 18$, $C_3 = 5$ and $C_B = 20000$ as parameters of cost function. The cost function is then calculated using the value of estimated parameters as given in the Table.2.

Elicitation of single utility function for each attribute:

The single utility function for each attribute is elicited based on the management's own scenarios. Since these management scenarios are subjective assessments from management, they may not be precise. Suppose that management scenarios in our application example are as follows:

- For Reliability, management has verified that at least 70% of software faults should be detected; its highest expected value is 100%.
- For Detection Rate Indicator, management has verified that 70% of change in detection rate show minimum, while its highest expected value is 100%.
- Under minimization cost strategy, management indicates that at least 60% of budget must be consumed.

According to the above management decision, some important points on the utility curve are obtained. In particular, for third release the lowest reliability requirement is $R^0 = 0.7$ and the maximum reliability expectation is $R^1 = 1$. The lowest change in detection rate is $D^0 = 0.7$ and the highest change is $D^1 = 1$. The lowest cost requirement is $C^0 = 0.6$ and the highest cost expectation $C^1 = 1$. Additionally, based on management's risk neutral attitude towards these attributes, three form of the single utility function should be used. Specifically, we have $u(C) = -\frac{5}{4} + \frac{5}{2}C$ $u(D) = -\frac{1}{4} + \frac{5}{2}D$ and $u(R) = -\frac{7}{3} + \frac{10}{3}R$ .It is worth noting here that although the linear form is simple for evaluating the utility for the attributes [21,26,27].

Estimation of weight parameters:

The weight parameter w_C is estimated by comparing the two choices by lottery approach [29,30]. Management has claimed that it is indifferent among these choices when *p* is equal to 0.3; hence $w_C = 0.3$ i.e. the weight related to cost parameter is 0.3.

It is easy to calculate w_R *and* w_D based on the sum of weight parameters is equal to one, therefore w_R, w_D are respectively .4 and .3.

Maximization of multi-attribute utility function

Finally, based on the estimated single utility functions and the weight parameter, the multi-attribute utility function is evaluated using three attributes.

The Multi Attribute Utility Function is maximized by using of Maple package Software and the optimal time to release the first version of software. The optimal Release time is given in table-4.

Table-4		
Model	Optimal Release time T^*	Utility Value
	20.07	.750

According to Tandem data failure, real time to release thefirstversion of software is 20weeks. Based on optimal result, we can say that software in first release must release after this time.
Figure-1 shows the multi attribute utility function for first release of software. Figure-2 represents the behavior of the cost function for first release.

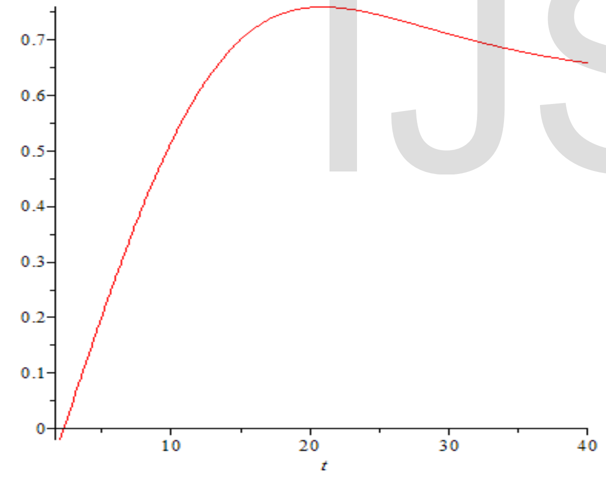

Figure-1: Utility function graph

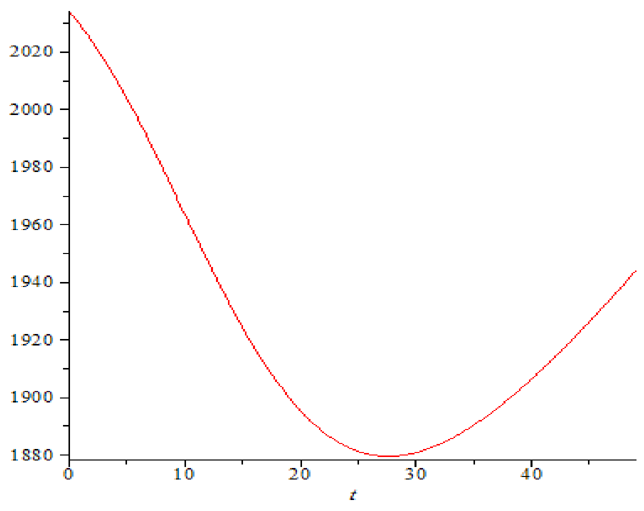

Figure-2: Cost function graph

SENSITIVITY ON WEIGHT OF ATTRIBUTES

Optimal release time can be determined by maximizing the multi-attribute function. However, since most parameters in the MAUT are obtained based on the subjective assessments from management, the optimal introduction time received may not be precise. The weight attached to each attribute is purely management decision. Different combination of attributes can be used by attaching different weights to them. In numerical example we have find the release time with fixed weight .3,.4,.3, for cost, reliability and detection rate indicator respectively. Accordingly for choice of different weight for attributes, sensitivity analysis is needed. Sensitivity analysis is generally done by changing one parameter and setting the other parameters at their fixed values. We choose different combination of weight for three attributes and find optimal release time as we have done in our numerical example section. Sensitivity for weight parameter and respective release time are summarized in following table.

Combination No.	Sensitivity on weight							
	Weights			Optimal Values		Single Attributes Utility		
	w_c	w_R	w_D	T^*	Multi Utility	$C = \frac{C(}{Bu}$	$R = \frac{m(}{}$	$D = \frac{b'(}{b}$
1	.30	.40	.30	20	.75	.094	.720	.650
2	.30	.30	.40	15	.95	.096	.730	.890
3	.29	.29	.42	14.97	.98	.091	.540	.910
4	.20	.40	.40	17	.81	.095	.613	.810

5	.40	.40	.20	31.8	.76	.094	.914	.264
6	.40	.30	.30	17.63	.87	.095	.633	.805

Optimal release times for different choice of weight for three attributes are given in above table. The final utility values along with single utility value for three attributes are given in last four columns in above table. Different combination of weight gives different release time which shows its sensitivity. Management has several choices for release time according to their preference in attributes. One thing is very clear from the table that changing weight for cost parameter does not affect its own utility value. While weight parameter is very sensitive for rest two attributes. If we increase or decrease the weight of reliability its own utility follows same. Similarly for detection rate indicator is happening (Combination No 5, 1, 2, 3). This shows the importance of reliability and detection rate indicator as an attribute. Less weight to detection rate indicator gives bad release time (Combination No 5).

CONCLUSION

The strategy behind the developing and releasing software is not an easy thing. Most of organizations are struggling to find the exact release time of software. From a management point of view it is important to understand the balance between reliability of software and cost. A vital decision problem that the software developer encounters is to determine when to stop testing and release the software system to the user. If the release of the software is unduly delayed, the manufacturer (software developer) may suffer in terms of penalties and revenue loss, while a premature release may cost heavily in terms of fixes (removals of faults) to be done after release, which consequently might harm the manufacturer's reputation. On one hand, when there is limited cost budget for testing, the software is expected to be tested in such a manner that it costs reasonable.In order to make a judicious decision on the optimal time of release of software, a decision model based on MAUTis proposed. We maximize the Multi attribute utility function using cost, reliability and detection rate indicator and obtained optimal release time.We conduct sensitivity analysis for weight parameter. We find different optimal release time for different combination of weight for attributes.

REFERENCE

1. K. Okumoto and A. L. Goel, "Optimal release time for computer software," *IEEE Transactions on Software Engineering*, vol. 9, pp. 323-327, 1983.
2. D. B. Brown, S. Maghsoodloo, and W. H. Deason "A cost model for determining the optimal number of test cases" *IEEE Trans. on Software Engineering*, 15(2):218–221, February 1989.
3. P. K. Kapur, H. Pham, Anu G. Aggarwal, Gurjeet Kaur, "Two-dimensional multi-release software reliability modelling and optimal release planning" *IEEE Trans on Reliability*, Vol. 61 (3), pp. 758-768, 2012
4. P. K. Kapur, A. tondon, G. Kaur, "Multi up- gradation software reliability model," 2nd international conference on reliability, safety & hazard (ICRESH-2010), pp-468-474, Mumbai, 2010.
5. P K Kapur, V B Singh, Ompal Singh, Jyotish N P Singh, software release time based on differentmulti-attribute utility functions" International Journal of Reliability, Quality and Safety Engineering, Vol. 20, No. 4 (2013) 1350012 (15 pages).
6. H. Pham and X. Zhang, "Software release policies with gain in reliability justifying costs," *Annals of Software Engineering*, vol. 8, 1999.
7. P. K. Kapur and R. B. Garg, "Cost reliability optimum release policies for a software system with testing effort," *Operations Research*, vol. 27, pp. 109-116, 1990.
8. H. Pham, *Software Reliability*. Singapore: Springer, 2000.
9. Kapur, P. K., Pham, H., Gupta, A. and Jha, P.C. (2011), "Software Reliability Assessment with OR Applications", Springer –UK.
10. H. Ohtera and S. Yamada., "Optimum software-release time considering an error detection phenomenon during operation" *IEEE Trans. on Reliability*, 39(5):596–599, December 1990.
11. H. Pham and X. Zhang., "A software cost model with warranty and risk costs". *IEEE Trans. on Computers*, 48(1):71–75, January 1999.
12. R. S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw Hill, 1997.
13. S. M. Ross. "Software reliability: The stopping rule problem". *IEEE Trans. on Software Engineering*, SE-11(12):1472–1476, December 1985.
14. Musa JD, Iannino A, Okumoto K. "Software reliability: Measurement, Prediction, Applications" 1987; New York: Mc Graw Hill.
15. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Trans. on Reliab.*, vol. 28, pp. 206–211, 1979.
16. M. Obha, "Software reliability analysis models," *IBM Journal of Research and Development*, vol. 28, pp. 428–443, 1984.

17. M. Xie, *Software Reliability Modeling*World Scientific Publishing, 1991.

18. H. Pham, *System Software Reliability*. U. K.: Springer-Verlag, 2006.

19. Yamada, S., Ohba, M. and Osaki, S. (1984), "S-shaped software reliability growth models and their applications", IEEE Trans. on Reliability, Vol. 33 No. 4, pp. 289–292.

20. Yamada, S., Narihisa, H., Osaki, S., (1984), "Optimum release policies for a software system with a cheduled software delivery time", International Journal of Systems Science, Vol. 15 No. 8, pp. 905–914.

21. Li, X., Li, Y.F., Xie, Min. and Ng, S.H. (2011), "Reliability analysis and optimal version-updating for open source software", Information and Software Technology, Vol. 53, pp. 929–936.

22. Wood,, "Predicting software reliability," *IEEE Computer,* vol. 9, pp. 69-77, 1996.

23. Kapur PK, Garg RB, Kumar S. "Contributions to hardware and software reliability" 1999; Singapore: World Scientific Publishing Co. Ltd.

24. M. C. K. Yang and A. Chao. "Reliability-estimation and stopping-rules for software testing based on repeated appearance of bugs". *IEEE Trans. on Reliability*, 44(2):315–321, June 1995.

25. W. Y. Yun and D. S. Bai., "Optimum software release policy with random life cycle" *IEEE Trans. on Reliability*, 39(2):167–170, June 1990.

26. Ferreira, R.J.P.,Almeida, A.T., Cavalcante, C.A.V., "A multi-criteria decision modelto determine inspection intervals of condition monitoring based on delay timeanalysis," Reliability Engineering and System Safety 94 (2009) 905–912.

27. Fishburn, C P, Utility Theory for Decision Making, Wiley, New York, 1970.

28. Keeney, R.L. and Raiffa, H. (1976), "Decisions with Multiple Objectives: Preferences and Value Tradeoffs, Wiley, New York.

29. Web reference, http://wiki.ece.cmu.edu/ddl/index.php/Multiattribute_utility_theory.

30. Winterfeldt, D. and Edwards, W. (1986), "Decision Analysis and Behavioral Research", Cambridge University Press, Cambridge, UK

31. Kapur PK, Agarwala S, Garg RB. "Bicriterion release policy for exponential software reliability growth model" Recherche Operationnelle – Operations Research 1994; 28: 165-180.

32. Kapur PK, Garg RB. "Optimal release policies for software systems with testing effort" Int. Journal System Science, 1990; 22(9), 1563-1571