Review on Software Reliability Growth Models and Software Release Planning

Rashmi Upadhyay
Scholar, Galgotias University
Greater Noida
India

Prashant Johri
Professor, Galgotias University
Greater Noida
India

ABSTRACT
In this paper, Software Reliability Engineering is a field that developed from ancestry in the reliability disciplines of structural, electrical, and hardware engineering. Reliability models are powerful tools of Software Reliability Engineering for estimating, predicting, devious, and assessing software reliability. On the basis of the review the cataloging of software reliability models has been presented as a major part. This categorization is based on the various dimensions of reliability models. Models under review reflect either infinite or finite number of failures. This paper discusses a two-dimensional software reliability growth modeling framework. We measured that an actual software reliability growth progression depends not only on testing time but also on testing effort and also enables us to portray software release planning problem in software reliability growth process. Thus, we can say that software project managers can demeanor more viable and accurate software reliability appraisal by using two-dimensional SRGM.

General Terms
SRGM(Software Reliability Growth Model), Software Release Planning.

Keywords
Software Reliability, SRGM, Two dimensional, Non-Homogeneous Poisson Process(NHPP), Release Time.

.

1. INTRODUCTION
Software reliability is essential because of our craving on computer software system in our everyday life and to the actuality that software system cannot be made error free. In the former two decades diverse methodologies and techniques have been developed and put in to observe in the hope of producing high quality, low cost software systems. Software is a resolute mechanism that comprised of computer programs, procedures, rules, data and related documentation. The enlarge in number of software failures inadequately affected the performance of transportation, telecommunication, military, industrial process, entertainment offices, aircrafts and business. Therefore software reliability has become more & more important. Reliability is the competence of software to sustain a determined level of performance within the time period.

Usually the software development progression is composed of four phases: requirement phase, design phase, coding and testing phase. The testing phase aims to perceive and get rid of the dormant software faults in order to ensure, as far as possible, error free process of software in a given time. In other words, the testing phase quantifies the quality of the software in requisites of its consistency. Therefore software reliability is reliant on the number of errors enduring in the software.

This paper presents a review on the software reliability Models. The study throws the light on various dimensions of reliability models. Section 2 have the literature review, Section 3 describes the difference between one dimension vs. Two Dimension, Section 4 Discussed Two dimensional modeling Framework. Section 5 describes Software release planning problem. At last paper concludes in section 6.

2. LITERATURE REVIEW

2.1 Software Reliability Engineering:

The genesis of Reliability Engineering can be found in the early 19th century industrial world. Musa (1999) has defined Software Reliability Engineering (SRE) as:

"SRE is a practice that helps one develop software that is more reliable, and helps one develop it faster and cheaper. It is a standard, proven, widespread best practice that is widely applicable to systems that include software. SRE is low in cost and its implementation has virtually no schedule impact".

SRE works by quantitatively characterizing and applying two things about the product: the expected relative use of its functions and its required major quality characteristics.

Software Reliability is an important to trait of software quality, together with functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation. Software Reliability is hard to achieve, because the complexity of software tends to be high. While any system with a high degree of complexity, including software, will be hard to reach an assured level of reliability, system developers lean to push complexity into the software layer, with the hasty growth of system size and ease of doing so by upgrading the software. Software reliability is often defined as ―the probability of failure-free operation of a computer program for a specified time in a specified

environment. Various approaches can be used to look up the reliability of software, nevertheless, it is hard to balance development time and budget with software reliability.

Software reliability engineering is also anxious with the characteristics of the software development process. In this regard, it deals with characteristics such as cost of development, duration of development, and risks in development of software. Consequently choice of the Software Development Life Cycle (SDLC) model adopted is grave to the accomplishment of any software development project [30, 31, and 32].

2.2. Reliability and the software lifecycle:

The methods and needs of software reliability appraisal and prediction vary by the phase of software development lifecycle [12,16].

• In the requirements and design phases, when no implementation is vacant, early prediction models can be used. Reliability must be analyzed based the architecture and stated requirements.

• In the implementation and testing phases, software reliability appraisal is pleasing to make the stopping decision pertaining to testing and debugging: when the mean time to failure is long adequate, the software can be released. Models most relevant here are *reliability growth models*.

• When the software is released, it is ordinary to suppose that all pragmatic faults have been debugged and corrected. As a result, after release, a *reliability model* is used to forecast the mean time to failure that can be predictable. The resulting reliability estimate may be used in system reliability assessment, as a source of maintenance recommendation, and auxiliary upgrading, on a basis of the recommendation to suspend the use of the software. Hence, when the software is in equipped use, the model to be used depends on upholding policies and incidence of failures.

•If no failures are detected in the software, or if the software is not maintained, a reliability model is most suitable. If failures are detected and the software is modernized, a reliability growth model is in order.

2.2.1. Software Reliability:

According to ANSI, Software Reliability is defined as: the probability of failure-free software operation for a specified period of time in a specified environment. Software reliability is one of the important parameters of software quality and its is defined as a probabilistic function, and comes with the concept of time. Software is an appliance for transforming a distinct lay down of input into an obligatory distinct set of outputs. It comprises a set of coded statements or directives whose functions may be to assess an appearance and accumulate the result in a provisional or eternal location, to decide which proclamation to carry out, or to execute input/output operations. Software is a logical method rather than a physical system element. Consequently, *software* has distinctiveness that is significantly different than those of hardware.

1. Software is developed or engineered; it is not manufactured in the conventional sense.

2. Software doesn't "wear out".

3. Even though the trade is moving toward component-based assembly, most software continues to be convention built.

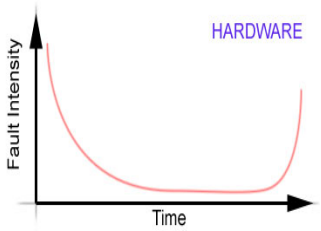

Figure 1.: Failure Curve for Hardware

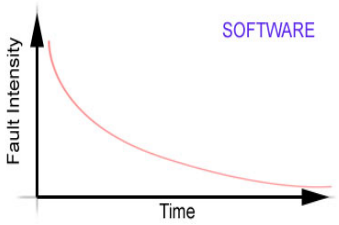

Figure 2: Failure Curve for Software

2.3. Software reliability growth models:

Software reliability models are used for the prediction and estimation of software reliability [9].This section reviews some existing software reliability growth models. These models describe how observation of failures and correcting the underlying faults. Such as occurs in software development when the software is being tested and debugged, influence the reliability of software. These models are appropriate also to assessing the reliability of software in equipped use, when the latest reliability approximation given by the model is used.

2.3.1 NHPP based Software Reliability Growth Models:

NHPP based SRGM are broadly classified into two categories first – *continuous time models*, which uses time (CPU time, calendar time or execution time) as a unit of fault detection period and second – *Discrete time models*, which assume the number of test occasions/cases as a constituent of fault detection period. Models can also be categorized as *concave* and *S-shaped* depending upon the shape of the failure curve described by them. Concave models describe an exponential failure curve while second category of models describes an S-shaped failure curve.

These models are generally used to forecast the release date of a software product. SRGMs base their predictions on data from the testing process and, thus, reflect the testing, the fault prologue and the fault pronouncement processes. Kapur et al [2, 3] proposed an SRGM with three types of fault. For each type, the FRR per remaining faults is assumed to be time independent. The first type is modeled by an Exponentional model of Goel and Okumoto [1]. The second type is modeled by Delayed S-shaped model of Yamada et al. [5]. The third type is modeled by three stage Erlang model proposed by Khoshogoftaar [4]. Later they extended their model to cater for more types of faults [3] Therefore, SRGMs are, in our

opinion, potentially appropriate for predicting fault inflow as well. Many architecture based software trustworthiness models has grown during the earlier period in a very imposing manner. Numerous software reliability models have been exposed since the early 1970s and lots of work has been done on the models that approximate reliability development during testing stage.

Various general model assumptions are as follows:

1. Software system is subject to failure during execution caused by faults remaining in the system.
2. Failure rate of the software is equally affected by faults remaining in the software.
3. The number of faults detected at any time instant is proportional to the remaining number of faults in the software.
4. On a failure, repair effort starts and fault causing the failure is removed with certainty.
5. All faults are mutually independent from failure detection point of view.
6. The proportionality of failure detection / fault isolation / fault removal is constant.
7. The fault detection / removal phenomenon is modeled by NHPP.

Notations:

$m(t)$: Expected number of faults identified in (0,t]

a:Representing initial fault content

b: Rate of fault removal per remaining faults for software.

p, q:Proportionality constants

The brief description of some models is given in subsequent section.

2.3.1.1. Goel-Okumoto Model (Goel and Okumoto 1979):

Goel and Okumoto [1] proposed an SRGM, which describes the fault detection rate, as a non homogeneous poisson process (NHPP) assuming the hazard rate is proportional to remaining fault number.

Following differential equation results from assumption-3

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

The above first order linear differential equation when solved with the initial condition $m(0) = 0$ gives the following mean value function

$$m(t) = a(1 - e^{-bt}) \quad (2)$$

The mean value function is exponential in nature and doesn't provide a good fit to the S-Shaped growth curves that generally occur in Software Reliability. But the model is popular due to its simplicity.

Now we briefly discuss below some S-Shaped SRGMs.

2.3.1.2. Delayed S-Shaped SRGM (Yamada, Ohba and Osaki 1983)

Yamada et al. [7] proposed a modified exponential SRGM assuming the software contains two types of faults. The model is based on the observation that in the early stages of the software phase, the testing team removes a large number of simple faults (faults that are easy to remove) while the hard faults are removed in the later stages of the testing phase. Accordingly, they assumed the fault removal process to be the superposition of two NHPP, the first NHPP models the removal of the simple fault while the second models the removal of the hard faults Failure rate and isolation rate per fault are assumed to be same and equal to b.

Thus

$$\frac{d}{dt}m_f(t) = b[a - m_f(t)] \quad (3)$$

$$\frac{d}{dt}m(t) = b[m_f(t) - m(t)] \quad (4)$$

$m_f(t)$ is the expected number of failures in $(0,t]$. Solving (3) and (4), we get the mean value function as

$$m(t) = a\{1 - (1 + bt)e^{-bt}\} \quad (5)$$

2.3.1.3. Inflection S-Shaped SRGM (Ohba 1984):

The model attributes S-Shapedness to the mutual dependency between software faults. Other than assumption-3 it is also implicit that the software contains two types of faults, namely reciprocally dependent and reciprocally independent. The mutually self-regulating faults are those to be found on different execution paths of the software, consequently they are similarly expected to be detected and removed. The mutually dependent faults are those faults sited on the same execution path. According to the order of the software execution, some faults in the execution path will not be impassive until their foregoing faults are removed.

Let r denote the ratio of independent faults to the total number of faults in the software. This ratio is called the inflection parameter $(0 < r \leq 1)$. If all faults in the software system are mutually independent ($r = 1$) then the faults are randomly removed and the growth curve is exponential. According to the assumptions of the model, the fault removal intensity per unit time can be written as

$$\frac{d}{dt}m(t) = b(t)[a - m(t)] \quad (6)$$

$b(t)$, the fault removal rate at time t is defined as

$$b(t) = b\phi(t) \quad (7)$$

where, $\varphi(t)$ the inflection function is defined as

$$\varphi(t) = r + (1-r)\frac{m(t)}{a}, \qquad \varphi(0) = 0 \text{ and } \varphi(\infty) = 1 \quad (8)$$

b is the fault removal rate in the steady state. Solving (8) under the initial condition $m(0)=0$ we get

$$m(t) = a\left[\frac{1-e^{-bt}}{1+\left(\frac{1-r}{r}\right)e^{-bt}}\right] \quad (9)$$

If $r=1$, the model reduces to the Goel-Okumoto model (1979). For different values of r different growth curves can be obtained and in that sense the model is flexible.

2.3.1.4 Flexible SRGM (Bittanti et al 1988) :

The model is based on the following differential equation:

$$\frac{d}{dt}m(t) = k(m)(a - m(t)) \quad (10)$$

Where

$$k(m) = k_i + (k_f - k_i)\frac{m(t)}{a} \quad (11)$$

Here k_i and k_f are initial and final values of Fault Exposure Coefficient. If ki = k_f , then it reduces to Exponential model. If $k_f >> k_i$; the failure growth curve takes S-shape. If k_f is very small as compared to k_i that it is almost equal to zero, the failure growth curve becomes flat at the end.

The solution of equation (10) with initial condition $m(t=0)=0$ is :

$$m(t) = a\left(\frac{1-e^{-k_f t}}{1+\left(\frac{k_f - k_i}{k_i}\right)e^{-k_f t}}\right) \quad (12)$$

For different values of k_f and k_i , it describes different growth curves.

2.3.1.5. SRGM for an Error Removal Phenomenon (Kapur and Garg 1992)

This model is based upon the following additional assumption: On a failure observation, the fault removal phenomenon also removes proportion of remaining faults, without their causing any failure.

Based on the assumption the fault removal intensity per unit time can be written as

$$\frac{d}{dt}m(t) = p[a - m(t)] + q\frac{m(t)}{a}[a - m(t)] \quad (13)$$

Solving equation (13) with the usual initial condition, the expected number of faults detected in $(0,t]$ is given as

$$m(t) = a\left[\frac{1-e^{-(p+q)t}}{1+\frac{q}{p}e^{-(p+q)t}}\right] \quad (14)$$

3. ONE DIMENSIONAL SRGM V/S TWO DIMENSIONAL SRGM

Traditionally, one-dimensional models have been proposed in the literature with respect to testing time or testing coverage, even though not much has been done to capture the combined effect of the testing coverage and the testing time. Ishii and Dohi[8] proposed a two dimensional software reliability growth model and their application .In this paper we discuss a two-dimensional model which shows the mutual effect of testing time and testing coverage to remove the faults lying dormant in the software. We imagine that the number of faults removed in the software by a fixed time is reliant on the total testing resources accessible to the testing team. This testing resource is a fusion of both testing time and testing coverage. We employ Cobb-Douglas production function [17] to exhibit the effect of both testing time and testing coverage in removing the faults in the software. Inoue proposed a two dimensional software reliability growth model with testing coverage using the Cobb Douglas production function. Further in this paper we discuss a release policy which gives us an optimal value of the testing time and testing coverage which minimizes the total testing cost subject to a pre requisite level of reliability.

During the last three decades, a huge number of SRGMs have been proposed in the literature [1, 6, 10, 11, 13, 18, and 19]. But, almost all of the SRGMs have been developed to model a single release software development process. Moreover, they are developed under the assumption that the software reliability growth process depends only on testing-time. An alternative approach based on the NHPP was proposed by Yamada *et al.* [14, 15], and Huang and Kuo [20].They developed some testing-effort dependent SRGMs. All such models can be termed as one-dimension one release software reliability growth models (1-D; 1-R SRGMs). However, the time and resource usage together govern the software reliability growth process. And 1-D; 1-R SRGMs do not incorporate these factors simultaneously for multi releases. Thus, to confine the mutual effect of testing time and resources, a two dimensional multi-release software reliability growth model (2-D; M-R SRGM) is needed.

In recent years, Ishii and Dohi [21] proposed a two dimensional software reliability growth model and their application. They investigate the dependence of test-execution time as a testing effort on the software reliability appraisal, and certify quantitatively the software reliability models with two-time scales. Inoue and Yamada [8,22] also proposed two dimensional software reliability growth models.

However their modeling framework was not a direct representative of using mean value functions to represent the fault removal process. They discuss software reliability estimation method by using two dimensional Weibull type SRGM. This study aims to compare the predictive capability of two popular software reliability growth models, say flexible logistic growth and exponentiated exponential growth. In this paper we discuss two dimensional SRGMs which enable us to expect more feasible software reliability assessment than the conventional software reliability measurement approach. To start with, we describe one-dimensional unified approach for describing failure-occurrence or fault-detection phenomenon before discussing our two dimensional software reliability growth modeling framework

4. TWO DIMENSIONAL MODELING

Lately, two dimensional software reliability models have been developed to assess the software quantitatively. The need for developing a two dimensional model is an ideal solution to the problem of software reliability at the hands of software engineers. In one dimensional analysis the object variable is dependent on one basic variable although the object takes on many different roles based upon its dependence on various other factors. Two dimensional models are used to capture the joint effect of testing time and testing coverage on the number of faults removed in the software. In economics, the Cobb-Douglas functional form of production functions is widely used to represent the relationship of an output to inputs. It was proposed by Knut Wicksell (1851 - 1926), and tested against statistical evidence by Charles Cobb and Paul Douglas in 1928. In 1928 Charles Cobb and Paul Douglas published a study in which they modeled the growth of the American economy during the period 1899 - 1922. They considered a simplified view of the economy in which production output is determined by the amount of labor involved and the amount of capital invested. While there are many other factors affecting economic performance, their model proved to be remarkably accurate.

The mathematical form of the production function is given as follows:

$Y = AL^{v}K^{1-v}$

Where:

Y = total production (the monetary value of all goods produced in a year), L = labor input; K = capital input, A = total factor productivity v is elasticity of labor. This value is constants and determined by available technology. Fig 1 shows graphically how the total production is influenced due to change in the proportion of labor and capital.

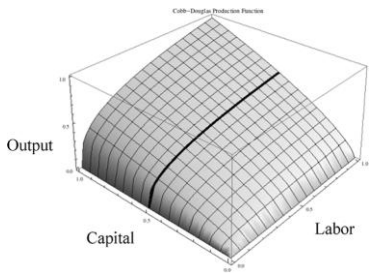

Figure 3. A two-input Cobb–Douglas production function [29].

The assumptions made by Cobb and Douglas can be stated as follows:

1. If either labor or capital vanishes, then so will production.

2. The marginal productivity of labor is proportional to the amount of production per unit of labor.

3. The marginal productivity of capital is proportional to the amount of production per unit of capital.

The Cobb- Douglas function based on the above assumptions is very appealing. The basic characteristic of this function is linearly homogeneous with constant return to scale i.e. a proportion increase in all inputs leads to same proportion increase in output the testing team has many resources of testing to make sure that software hence formed is of quality. These include software testing man hours, CPU time, testing effort testing coverage etc.

$\tau \cong s^{r}t^{1-r} \qquad 0 \le r \le 1 \qquad (15)$

Where

r : testing resources

s : testing time

u : testing coverage

τ : Effect of testing time

Let $\{N(s,u), s \ge 0, u \ge 0\}$ be a two-dimensional stochastic process representing the cumulative number of software failures by time s and testing coverage u. A two-dimensional NHPP with a mean value function m (s, u) is formulated as:-

$$\Pr(N(s,u) = n) = \frac{(m(s,u))^{n}}{n!}\exp((-m(s,u)),$$

n=0,1,2…

5. SOFTWARE RELEASE PLANNING PROBLEM:

Choosing and developing good software is a crucial step for organizations. A strong tendency is to follow the so-called iterative and incremental development [25]. This methodology is a recurring software development process in which the diverse components of any system are tearing into numerous parts which are developed at different stages (iterations) and then incorporated as their developments are accomplished. In each iteration, a preliminary version of the software must be released to the stakeholders. There are various optimistic characteristics that can be pointed out in this tactic, like early criticism from stakeholders, upgrading of subset of features in the system, better risks supervision and incremental tests execution. A key aspect to the success of a software project based on iterative and incremental life cycle is the planning of which necessities are going to be delivered in each release of the software. When starting a new project, stakeholder's requirements must be recognized. Next, the development team must decide which requirements have to be

implemented in each release. This requires a meticulous analysis concerning numerous aspects such as stakeholder's ease, costs, deadlines, available resources, and efforts needed, risks, requirements interdependencies, and so on. The superior and additional multifaceted the project is the more difficult and error vulnerable the release planning will be. This paper discusses this multifaceted task, named *Software Release Planning*.

The objective was to diminish the total software development cost focus to reliability less than a predefined reliability level, or maximize reliability subject to cost not exceeding a predefined budget. In 1991, Kapur and Garg [27] formulated release policies incorporating the effect of testing resource costs for an exponential SRGM under the other assumption that testing resource curves are described by moreover exponential, Rayleigh, or Weibull curves. Huang and Lyu [26] proposed an SRGM with a generalized testing effort function, and studied optimal release policies based on cost and reliability allowing for testing effort and efficiency. In 2007, Kapur *et al.* [11] proposed an SRGM with two types of imperfect debugging, and determined the optimal release time of the software.

Nowadays, in mainly organizations, software release planning is done by using *ad hoc* approaches, which are based merely on manager's proficiency, knowledge in previous experiences and insight. There are also methods for prioritizing software requirements [28], which are often costly and also based on manager's proficiency and perception to evaluate the requirements. Some harmful points can be shown, most of them due to the fact that the development gets reliant on folks instead of depending on process. To conquer this unwanted situation, we have to focus on software release planning.

5.1 Optimal release planning problem for software with multiple releases:

The release time problems discussed above were considered under the assumption that software comes in a single release. In forecast the release decisions for software that is to be brought into the market with new versions, the organization has to take into consideration two things:

(i) Testing data from the new code, and

(ii) Log reports of the previous release, i.e. bugs reported by the users in the equipped phase of the version that has been in the market.

Where in single release software systems only (i) prevails, and if (ii) is not taken into concern for the release planning of multi-release software systems, then the opinion of coming up with several versions gets lost. In the present problem, we consider minimizing the testing cost of the release that is under testing, with a constraint of removing a desired proportion of faults.

6. CONCLUSION:

Various software reliability models have been revealed since 1970s. Lots of work has been done on software reliability assessment. Some of major models that have appeared in the literature are discussed in this paper. Reliability models are based on the various dimensions. The major verdict of the study is that the models under review reflect either infinite or finite number of failures. All exponential distribution based models reflect finite failures and logarithmic distribution based model reflect infinite failures. The conventional one dimensional model has been dependent upon the testing time, testing effort or testing coverage. Conversely if the reliability of software is considered on the basis on the number of hours depleted on testing the software or the entitlement of software that has been enclosed then the results are not decisive. To accommodate the need of high accuracy software reliability we require a software reliability growth model which caters not only the testing time but also the testing effort. For this we discuss a two dimensional software reliability growth model incorporating the joint effect of testing time and testing effort on the number of faults removed in the software. And we also discuss the software release planning problem and optimal release planning for multi releases.

In Future, we will use a two dimensional approach to develop a flexible software reliability growth model using Cobb Douglas production function. This function is used to capture the combined effect of testing time and testing effort, and we will also use the Genetic Algorithm (GA) for solving optimal release planning problem as it is a complex, non linear optimization problem.

7. REFERENCES

[1] Goel AL, Okumoto K. Time dependent error detection rate model for software reliability and other performance measures. *IEEE Transactions on Reliability* 1979; R-28(3): 206-211.

[2] Kapur PK and R.B. Garg (1992), A software reliability growth model for an error removal phenomenon, Software Engineering Journal 7, 291-294.

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

[4] Khoshogoftaar TM and Woodcock TG (1991), "Software Reliability Model Selection: A case study", Proceedings of the international symposium on software reliability Engineering, pp.183-191.

[5] Yamada S, Ohba M, and Osaki S. (1984) S-shaped software reliability growth models and their applications, IEEE Transactions on Reliability, 1984; R-33: 169-175.

[6] Ohba, M. (1984), Software reliability analysis models, IBM Journal of research and Development 28, 428-443.

[7] Yashwant K. Malaiya, Michael Naixin, James M. Bieman and Rick Karcich; Software Reliability Growth with Testing Coverage, IEEE Transactions on Reliability,Vol. 51(4), pp.420-426, 2002.

[8] Inoue S, Yamada S "Testing-Coverage Dependent Software Reliability Growth Modeling" International Journal of Reliability, Quality and Safety Engineering, Vol. 11, No. 4, 303-312, 2004.

[9] Jintao zeng, Jinzhong Li, Xiaohui Zeng, Wenlang Luo "A Prototype System of Software Reliability Prediction and Estimation". IITSI 2010.

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

[11] P.K. Kapur, H. Pham, Anshu Gupta, P.C. Jha; "Software reliability Assessment with OR application", Springer London, 2011.

[12] Asad, C.A., Ullah, M.I. & Rehman, M.J.-U. An approach for software reliability model selection. *Proceedings of the 28th Annual International Computer Software and*

[13] *Applications Conference* (COMPSAC.04), Vol. 1, 534.539.

[14] J.D. Musa, D. Iannio and K. Okumoto; "Software Reliability Measurement", Prediction, Application, McGraw-Hill, New York, 1987.

[15] S. Yamada, H. Ohtera, and H. Narihisa; "Software reliability growth models with testingeffort", IEEE Trans. Reliab., Vol. R-35(1), pp. 19-23, 1986.

[16] S. Yamada, J. Hishitani and S. Osaki; "Software reliability growth with a Weibull test-effort", IEEE Trans. Reliab., Vol. 42(1), pp. 100-106, 1993.

[17] Hamlet, D. Are we testing for true reliability? *IEEE Software*, Vol. 9, No. 4 (July 1992),21.27.

[18] Shinji Inoue and Shigeru Yamada "Two-Dimensional Software Reliability Assessment with Testing-Coverage" 2008 Second International Conference on Secure System Integration and Reliability Improvement July 14-July 17.

[19] H. Pham, *System Software Reliability*. : Springer, 2006.

[20] M. R. Lyu, *Handbook of Software Reliability Engineering*. New York: McGraw-Hill, 1996.

[21] C. Y.Huang and S. Y. Kuo, "Analysis of incorporating logistic testingeffort function into software reliability modeling," *IEEE Trans. Reliability*, vol. 51, no. 3, pp. 261–270, 2002.

[22] T. Ishii and T. Dohi, "Two-dimensional software reliability models and their application," in *Proc. 12th Pacific Rim Intern. Symp. Dependa. Comput.*, 2006, pp. 3–10.

[23] S. Inoue and S. Yamada; "Two-dimensional software reliability measurement technologies",

[24] Proceedings of IEEE IEEEM, pp. 223-227, 2009.

[25] S. Inoue and S. Yamada; "Two-dimensional software reliability assessment with testing coverage", Second International Conference on Secure System Integration and Reliability Improvement, pp. 150-156, 14-17 July 2008.

[26] P.K. Kapur, R.B. Garg, G.A. Aggarwal and A. Tandon; "Two-dimensional flexible software reliability growth model and release policy", Proceedings of the Fourth National Conference on Computing for Nation Development-INDIA Com-2010, 25-26, New Delhi, pp. 431-438, February 2010.

[27] C. Larman and V. R. Basili, "Iterative and incremental development: A brief history," *Computer*, vol. 36, no. 6, pp. 47–56, 2003.

[28] C. Y. Huang andM. R.Lyu, "Optimal release time for software systems considering cost, testing effort, and testing efficiency," *IEEE Trans. Reliability*, vol. 54, no. 4, Dec. 2005.

[29] P. K. Kapurand and R. B. Garg, "Optimal release policies for software systems with testing effort," *International J. System Science*, vol. 22, no. 9, pp. 1563–1571, 1991.

[30] J. Karlsson, C. Wohlin, and B. Regnell, "An evaluation of methods for prioritizing software requirements," *Information and Software Technology*, vol. 39, no. 14-15, pp. 939–947, 1998.

[31] .P. K. Kapur, H. Pham, *Fellow, IEEE*, Anu G. Aggarwal, and Gurjeet Kaur, "Two Dimensional Multi-Release Software Reliability Modeling and Optimal Release Planning", IEEE TRANSACTIONS ON RELIABILITY, VOL. 61, NO. 3, SEPTEMBER 2012.

[32] Lyu, M.R. (ed.). *Handbook of Software Reliability Engineering*. IEEE Computer Society Press and McGraw-Hill, 1996.

[33] Boehm, B.W. (1981), 'Software Engineering Economics', Englewood Cliffs, N.J.: Prentice Hall.

[34] Pressman R.S.(2001), 'Software Engineering: A Practitioner's Approach', McGraw-Hill: Fifth Edition.