Designing Debugging Models for Object Oriented Systems

Sujata Khatri[1], R.S.Chhillar[2]

[1] D.D.U.College University , Delhi University
New Delhi, 110078

[2] D.C.S.A Maharishi Dayanand University
Rohtak , Haryana

Abstract

Bugs are inevitable in any software development life cycle. Most bugs are detected and removed in the testing phase. In software, we can classify bugs into two categories: (1) bugs of different severity (2) bugs of different complexity. Prior knowledge of bug distribution of different complexity in software can help project managers in allocating testing resources and tools. Various researchers have proposed models for determining the proportion of bugs present in software of different complexity but none of these models have been applied to object oriented software. Software reliability growth models have been used during later stages of testing to predict the number of latent bugs dormant in the software. Once a bug is found in the software, efforts have been made by the development team to debug it. It is found in practice that debugging may not be perfect and during removal of bugs, some new bugs may be generated and this phenomenon is called imperfect debugging. In this paper, we have developed a software reliability growth model for object oriented software system for perfect debugging in which new bugs are not generated during removal process and also by incorporating imperfect debugging, where new bugs are generated during removal process in a proportion removed bugs. Here, the proposed paper is used to assess the reliability growth of object oriented software developed under concurrent distributed development environment. We have collected bug reported data of MySQL for python. Numerical illustration has been also presented in the paper.

Keywords: *Open source software, software reliability growth model, and Object oriented approach.*

1. Introduction:

Software applications are the fastest growing trend in the virtual world and the possibilities regarding the features and functions provided by a specific application is generating tremendous interest amongst a vast number of people around the globe. As the interest grows, so does the demand for application.

Development of large software products involves several activities that need to be suitably coordinated to meet desired requirements. Meyer defines object-oriented design as "the construction of software systems as structured collections of abstract data type implementations" [1].The emphasis on object oriented language is on defining abstraction of a model concept related to an application domain [2].To understand Object Oriented Programming Systems the following high level concepts must be introduced: objects, classes, inheritance, polymorphism, and dynamic binding.

Software objects are conceptually similar to physical objects; they too consist of state and related behavior. An object stores its state in fields (variables) and exposes its behavior through methods (functions). Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication [3].

The IEEE defines testing as "the process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements or to identify differences between expected and actual results" [3]. Instead of bugs being in the software units, the complexity is now primarily in the way in which we connect the software.

The object oriented approach has been widely used for the development of closed source and open source software. In open source software, developers are also the users, meaning there by those who remove the bugs are also responsible for generating bugs [4]. Open source project has more advantage in terms of fewer bugs, better reliability, no vendor dependence, shorter development cycles, quick support and educational benefits. In the available literature, many papers address the issue of open source software [5, 6, 7, 8, and 9].

Over the last three decades various software reliability growth models have been developed but very few of them have been applied in object oriented software system. It was Kapur et al. [10] who firstly developed reliability growth model for object oriented software system. Recently, Singh et al. [19] have also developed a software reliability growth model for object oriented system which categorizes bugs into simple, hard and complex types.

During last three decades various software reliability growth models have been developed in the literature for closed source software in dealing with bug complexity. Kapur et al. [11] introduced a flexible model called the generalized Erlang SRGM by classifying the bugs in the

software system as simple, hard and complex. It is assumed that the time delay between the failure observation and its removal represent the complexity of bugs. Another model according to Kapur et al.[12], describes the implicit categorization of bugs based on the time of detection of fault. However, an SRGM should explicitly define the different types of bugs as it is expected that any type of fault can be detected at any point of testing time. Therefore, it is desired to study testing and debugging process of each type of bugs separately [13 and 14]. The mean value function of SRGM is described by the joint effect of the type of bugs present in the system. Such an approach can capture the variability in the reliability growth curve due to errors of different severity depending on the testing environment. Another model of Kapur et al. [16] describes the errors of different severity in software reliability growth model using different debugging time lag functions. Kapur et al.[17] also describe flexible software reliability growth model using a power function of testing time for defining errors of different severity. Singh et al. [18] have developed a generalized software reliability growth model which determines proportion of bugs of different complexity from open source software. Recently, Singh et al. [25] have developed a generalized software reliability growth model which determines the proportion of bugs of different complexity from open source software where software has been developed on object oriented methodology.

Due to the complexity of software systems and an incomplete understanding of software, the testing team may not be able to remove/correct the fault perfectly on observation/detection of a failure, and the original fault may remain resulting in a phenomenon known as imperfect debugging, or get replaced by another fault causing error generation. In the case of imperfect debugging, the fault content of the software remains the same; while in the case of error generation, the fault content increases as the testing progresses and removal/correction results in the introduction of new faults while removing/correcting old ones[25]. In this paper we have considered the second case where new errors are generated during removal process.

To the best of our knowledge, no research paper has addressed the issue of imperfect debugging in object oriented software systems. In this paper, we propose a Perfect and Imperfect debugging model for object oriented software system.

The rest of the paper is organized as follows: Section 2 provides model development, assumptions and formulation. Section 3 describes model validation and data analysis. And, finally, section 4 concludes the paper with a future research direction.

In this paper, we have taken actual failure data of software namely SQL for python developed under open source environment. And the development of software follows object oriented approach.

A. Assumptions of the Model

Following assumptions have been taken for developing software reliability growth model for software which has been developed under open source environment using object oriented approach.

1. A finite number of test cases are prepared to ensure that the software works according to the requirements and specifications. Each test case is designed to execute a finite number of instructions.
2. The time dependent behavior of the instruction execution is represented by Exponential or Rayleigh curve.
3. The software is prone to failure due to the following causes
   - 3.1 Erroneous execution of internal variable/data of the objects. In this case we have three types of errors:
     - 3.1.1 Error due to private (local) variable/data.
     - 3.1.2 Error due to public (global) Variable/data.
     - 3.1.3 Error due to protected variable/data.
4. The failure observation/error removal phenomenon follows NHHP.

5.1 No new error is introduced during removal process for perfect debugging.

5.2 New errors are generated for imperfect debugging.

6. The error removal intensity per execution is proportional to the remaining errors in the software.
7. The number of executions per unit of time is proportional to the remaining number of instructions not executed.
8. The software faults are classified in to three categories.
   1. simple faults, (easy to detect and remove)
   2. hard faults (difficult to detect and remove) and
   3. complex faults. (very difficult to detect and remove)
9. We assume that accession to private, protected and public variable resulted in simple, hard and complex faults.

2. MODEL FORMULATION (Perfect debugging)

The total number of instructions executed is E(t) at any given time t. These instructions cause an accession to private, protected and public variable [10 and 19].

The sum of errors (mean vale function) due accession of private, protected and public variable is a.

Based on the assumptions 2 and 7, the number of instructions per unit of time can be written as

$$\frac{dE(t)}{dt} = x(t)\left(A - E(t)\right) \qquad (1)$$

A is total number of instructions to be eventually executed and x(t) is the rate of instruction execution per instruction. Solving above equation we get:

$$E(t) = A\left(1 - \exp\left(-\int_0^t x(t)\right)dt\right) \quad (2)$$

Depending upon the value of x(t), different types of instruction execution functions can be formulated. If x(t)=B i.e. instructions execution rate is independent of time then it follows exponential curve (instructions are uniformly executed) i.e.

$$E(t) = A\left(1 - \exp(-Bt)\right) \quad (3)$$

If x(t)=Bt, then it follows Rayleigh curve means instructions are no uniformly executed with respect to time.

Using assumptions (4-9), we can write the following differential equation for error removal phenomenon in case of errors due to private, protected and public variable :

$$\frac{\frac{dm_1(t)}{dt}}{\frac{dE_1(t)}{dt}} = b\left(ap_1 - m_1(t)\right) \qquad (4)$$

Where $E_1(t)$ is the number of instructions causes an accession to private variable. Solving above differential with initial condition m(0)=0, we get

$$m_1(t) = ap_1\left(1 - \exp\left(-bE_1(t)\right)\right) \quad (5)$$

Error removal equation due to accession of protected variable is

$$\frac{\frac{dm_2(t)}{dt}}{\frac{dE_2(t)}{dt}} = \frac{b^2 t}{1+bt}\left(ap_2 - m_2(t)\right)$$

Where $E_2(t)$ is the number of instructions causes an accession to protected variable. Solving above differential with initial condition m(0)=0, we get

$$m_2(t) = ap_2\left(1 - \left(1 + bE_2(t)\right)\exp\left(-bE_2(t)\right)\right) \quad (6)$$

Error removal equation due to accession of public variable is

$$\frac{\frac{dm_3(t)}{dt}}{\frac{dE_3(t)}{dt}} = \frac{b^3 t^2}{2\left(1 + bt + \frac{b^2 t^2}{2}\right)}\left(ap_3 - m_3(t)\right)$$

Where $E_3(t)$ is the number of instructions causes an accession to public variable. Solving above differential with initial condition m(0)=0, we get

$$m_3(t) = ap_3\left(1 - \left(1 + bE_2(t) + \frac{\left(bE_2(t)\right)^2}{2}\right)\exp\left(-bE_3(t)\right)\right) \quad (7)$$

The total error removal is given as

$$m(t) = ap_1\left(1 - \exp\left(-bE_1(t)\right)\right) + ap_2\left(1 - \left(1 + bE_2(t)\right)\exp\left(-bE_2(t)\right)\right) + ap_3\left(1 - \left(1 + bE_2(t) + \frac{\left(bE_2(t)\right)^2}{2}\right)\exp\left(-bE_3(t)\right)\right) \quad (8)$$

Here $a = a\left(p_1 + p_2 + p_3\right)$ and $E_1 = pE, E_2 = qE, and\ E_3 = rE$

Here E is the total number of instructions executed due to accession of private, protected and public variables. p ,q, and r is the proportion of instructions causes an accession to private , protected and public variable. p_1, p_2 and p_3 are proportion of faults due to accession of private, protected and public variables.

3. MODEL FORMULATION(Imperfect debugging)

(Case 1: Simple bugs):

$$\frac{\frac{dm_1(t)}{dt}}{\frac{dE_1}{dt}} = b_1\left(ap_1 + \alpha_1 m_1(t) - m_1(t)\right)$$

Here, b_1 is the bug removal rate for simple bug. a is the initial bug content in software, p_1 is proportion of simple bugs, $m_1(t)$ is the number of bugs removed due to accession of private variable and α_1 is bug generation rate per remaining bug

.

Solving above differential with initial condition $m_1(0) = 0$ m(0)=0, we get

$$m_1(t) = \frac{ap_1}{(1-\alpha_1)}\left(1 - \exp\left(-(1-\alpha_1)b_1 E_1(t)\right)\right) \quad (4.1)$$

(**Case 2:** Hard bugs):
Bug removal equation due to accession of protected variable by considering bug generation during removal of bug is

$$\frac{\frac{dm_2(t)}{dt}}{\frac{dE_2}{dt}} = b_2\left(ap_2 + \alpha_2 m_2(t) - m_2(t)\right)$$

Here, b_2 is the bug removal rate for simple bug. a is the initial bug content in software, p_2 is proportion of simple bugs, $m_2(t)$ is the number of bugs removed due to accession of private variable and α_2 is bug generation rate per remaining bug.
Solving above differential with initial condition $m_2(0) = 0$, we get

$$m_2(t) = \frac{ap_2}{(1-\alpha_2)}\left(1 - \exp\left(-(1-\alpha_2)b_2 E_2(t)\right)\right)$$

(5.1)

Bug removal equation due to accession of public variable by considering bug generation during removal of bug is

(**Case 3:** Complex bugs):

$$\frac{\frac{dm_3(t)}{dt}}{\frac{dE_3}{dt}} = b_3\left(ap_3 + \alpha_3 m_3(t) - m_3(t)\right)$$

Here, b_3 is the bug removal rate for simple bug. a is the initial bug content in software, p_3 is proportion of simple bugs, $m_3(t)$ is the number of bugs removed due to accession of private variable and α_3 is bug generation rate per remaining bug.
Solving above differential with initial condition $m_3(0) = 0$, we get

$$m_3(t) = \frac{ap_3}{(1-\alpha_3)}\left(1 - \exp\left(-(1-\alpha_3)b_3 E_3(t)\right)\right)$$

(6.1)

The total bug removal is given as

$$m(t) = \frac{ap_1}{(1-\alpha_1)}\left(1 - \exp\left(-(1-\alpha_1)b_1 E_1(t)\right)\right) + \frac{ap_2}{(1-\alpha_2)}\left(1 - \exp\left(-(1-\alpha_2)b_2 E_2(t)\right)\right) + \frac{ap_3}{(1-\alpha_3)}\left(1 - \exp\left(-(1-\alpha_3)b_3 E_3(t)\right)\right)$$

(7.1)

Here $a = a(p_1 + p_2 + p_3)$ and $E_1(t) = q_1 E(t), E_2(t) = q_2 E(t), and\ E_3(t) = q_3 E(t)$ or

$$E(t) = E(t)(q_1 + q_2 + q_3)$$

Here, p_1 , p_2 and p_3 are proportions of bugs due to accession of private, protected and public variables.
q_1 , q_2 and q_3 are proportions of accession due to private, protected and public variables.

4. MODEL VALIDATION

To verify the proposed model that determines types of fault present in the software due to accession of private, protected and public variable and proportion of instructions execution causes to accession of private, protected and public variable , we estimated the unknown parameters by using SPSS software tool.

A. Description of Data set

Data set-: MySQL for Python software has been developed under open source environment www.sourceforge.net. We collected failure data of MySQL for Python from 4/25/2001 (first bug reported) to 11/23/2009, during this period 144 bugs were reported on bug tracking system.
We have considered only valid bugs which are fixed.

We have simulated the instruction executed data with assumption that expected total number of instructions executed is 2000K and the rate of instructions execution is .003 for Rayleigh growth pattern respectively as follows in [10].

Sample of bug reported data PF MySQL for Python Software

ID	Summary	Status	Opened	Assignee	Submitter	Resolution	Priority
418713	Python 1.5.2 adds an L	Closed	4/25/2001	nobody	nobody	Wont Fix	5
419004	_mysql_timestamp_converter	Closed	4/26/2001	adustman	nobody	Fixed	5
424878	Date_or_None	Closed	5/17/2001	adustman	nobody	Fixed	5
440332	Need to #ifdef around things	Closed	7/11/2001	adustman	ads	Fixed	5
440327	setup.py configuration for my platform	Closed	7/11/2001	adustman	gimbo	Wont Fix	5
442299	core-dump. Python2.1,config_pymalloc	Closed	7/18/2001	adustman	nobody	Fixed	5
445489	Execeptions don't follow DB-API v2.0	Closed	7/28/2001	adustman	nobody	Fixed	5
464875	Limit bug in ZMySQLDA	Closed	9/25/2001	adustman	nobody	Wont Fix	5
464873	Limit bug	Closed	9/25/2001	nobody	nobody	Wont Fix	5

A. Parameter Estimation and Comparison Results

The performance of an SRGM is judged by its ability to fit the past software bugs and to predict satisfactorily the future behavior of the software bug removal process. Therefore, we use various comparison criteria for goodness of fit as mentioned in figure[5-8] .We have estimated the parameters of proposed model (equation 8) using SPSS tool for this data set. Parameter estimates are also shown in table [1-4].

Parameter Estimates of My-SQL Dataset (Rayleigh) for perfect debugging.

Table 1

Parameter Estimates of proposed model (Equation-7)	Rayleigh
a	161.905
b	.090
p_1	.232
p_2	.275
p_3	.493
p	.713
q	.181
r	.106

Table 2

Datasets	Comparison Results(Rayleigh)				
	R^2	MSE	Bias	Variation	RMSPE
MySQL for Python	.995	10.05	-0.078	3.19	3.29

Parameter Estimates of My-SQL Dataset (Rayleigh) for Imperfect debugging

Table 3

Parameter estimates of the proposed model equation(7)	MySQL for Python
a	174
b1	.000
b2	.473
b3	.016
p1	.000
p2	.097
p3	.903
q1	.000
q2	.774
q3	.226
α	.041

Table 4

Datasets	Comparison Results				
	R^2	MSE	Bias	Variation	RMSPE
MySQL for Python	.998	4.69304	-0.074	2.25348	2.2547

3.5 Goodness of fit Curves

This section describes the goodness of fit curves of different models for given data sets.

For Perfect Debugging:

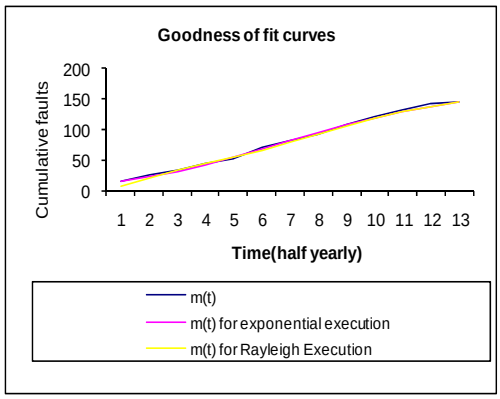

Figure 1

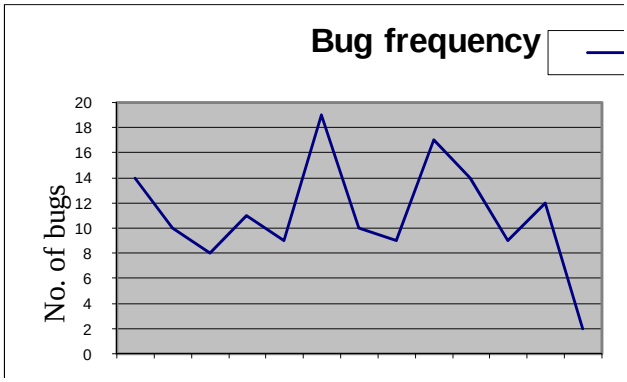

Figure 2

For Imperfect Debugging:

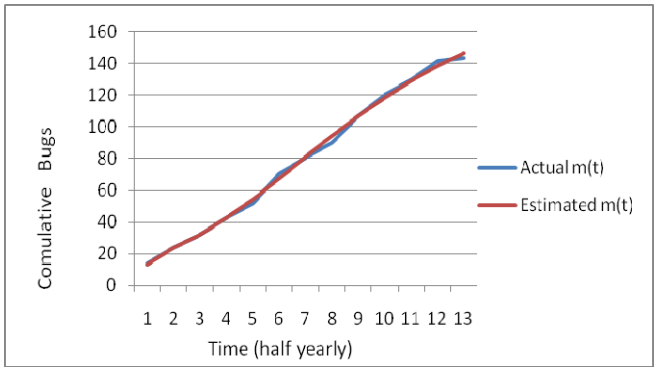

Figure 3

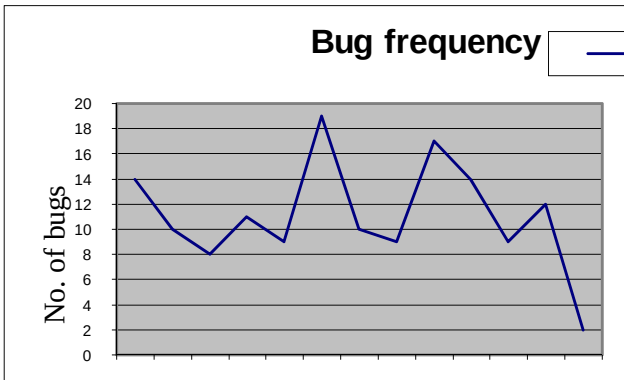

Figure 4

Figure 1, 3 gives mean value function of equation (7 and 7.1) vs time for MySql data set of Rayleigh type for Imperfect.

5. CONCLUSION:

Object oriented approach has become integral part of software development process. Traditional software engineering approach has been converting into object oriented software engineering. In this paper, we firstly discussed about object oriented approach. We also mentioned the main elements and advantages of using this approach. We see how accession to different variable namely private, protected and public causes an error to occur of different severity. In this paper, different modeling has been done for a failure resulting from accession to different types of variables. We have considered perfect as well as imperfect debugging occurs during bug removal. It also occurs in traditional and Object oriented development methodology. In this paper, we have proposed a software reliability growth model that determines the proportion of bug complexity in presence of perfect and imperfect debugging. The prior knowledge of distribution of bugs of different complexity will help project manager in allocation of testing efforts, tools and thus provide a better software production. We have provided the numerical results for the proposed model along with different types of growth curves depending upon their complexity.

This study can be further extended and applied on more data sets to increase confidence in the proposed model. In future, we will try to develop software reliability growth model for object oriented system by incorporating imperfect debugging and error generation by taking different models for different complexity of bugs.

REFERENCES:

[1] Meyer, Bertrand (1988): Object-oriented Software Construction. Prentice- Hall, *New York, NY, 1988, p. 59, 62.*

[2] Binder RV: Testing object oriented software: A survey**.** *Journal of software testing, Verification and Reliability 31996;6(3/4):125-252*

[3] IEEE 729-1983: Glossary of Software Engineering Terminology, *September 23, 1982.*

[4] Gacek Cristina and Arief Budi (2004):The Many meanings of Open Source, *IEEE Software, Vol. 21, issue 1, 2004, pp.34- 40.*

[5] Ruben van Wendel de Joode and Mark de Bruijne(2006): The organization of open source communities: Towards a Framework to Analyze the relationship between openness and reliability**,** Proceedings of 39th *Hawaii International Conference on System Sciences,* , 2006, pp.1-6.

[6] Mary Paul Li, Shaw, Herbsleb Jim , Bonnie Ray, Santhanam P., Empirical Evaluation of Defect Projection Models for Widely-deployed Production Software systems, in the proceedings of the *12th International Symposium on the production of Software Engineering (FSE-12),* PP.263-272.

[7] Tamura Y. and Yamada S., Optimization analysis for Reliability Assessment based on stochastic differential equation modeling for Open Source Software, *International Journal of Systems Science, Vol. 40,* No.4 , 2009, pp 429- 438.

[8] Zhou Ying and Davis Joseph (2005):Open Source Software Reliability Model: An empirical approach, *Proceedings of the 5th WOSSE*, 2005, pp 1-6.

[9] Singh V.B. and P.K Kapur.(2009): Measuring Reliability Growth of Open Source Software, Accepted for poster presentation in IBM-Indian Research Laboratory Collaborative Academia Research Exchange held during October 26, 2009 at IBM India Research Lab, ISID Campus, Institutional Area, Vasant Kunj , New Delhi, India.

[10] .Kapur P.K, Min Xie and Younes Said (1994): Reliability Growth Model for Object Oriented Software System, Software Testing, Reliability and Quality Assurance, , Dec.21-22 1994,. pp. 148 – 153

[11] Kapur P.K., Younes S. and Agarwala S. (1995) 'Generalized Erlang Software Reliability Growth Model with n types of bugs", *ASOR Bulletin*,14,5-11.

[12] Kapur P.K., Bardhan A.K., and Kumar S. (2000) :On Categorization of Errors in a Software**,** *Int. Journal of* Kapur *Management and System*, 16(1),37-38

[13] P.K., Bardhan A.K.; Shatnawi O.; (2002) Why Software Reliability Growth Modelling Should Define Errors of Different Severity**.** *Journal of the Indian Statistical Association*, Vol. 40, 2, 119-142.

[14] Kapur P.K., Younes S and Grover P.S.; (1995), Software Reliability Growth Model with Errors of Different Severity, *Computer Science and Informatics (India)* 25(3):51-65.

[15] Kapur P.K. Kumar Archana ,Yadav Kalpana and Khatri Sunil(2007) :Software Reliability Growth Modelling for Errors of Different Severity using Change Point, *International Journal of Quality ,Reliability and Safety engineering* Vol.14,No.4,pp. 311-326.

[16] P .K Kapur. Kumar Archana Singh V.B. and Nailana F.K.(2007):On Modeling Software Reliability Growth Phenomanon for Errors of Different Severity**,** *In the Proceedings of National Conference on Computing for Nation Development*, Bhartiya Vidyapith's Institute of Computer Applications and Management, New Delhi, pp.279-284, held during 23rd-24 th February.

[17] P.K., Kapur Kumar Archana , Mittal Rubina and Gupta Anu (2005):Flexible Software Reliability Growth Model Defining Errors of Different Severity, Reliability, Safety and Hazard, pp. 190-197 Narosa Publishing New Delhi.

[18] Singh V.B., Singh O. P., Kumar.Ravi,Kapur P.K.(2010) A Generalized Software Reliability Model for Open Source Software published in proceedings of 2nd International Conference on Reliability Safety and Hazard, organized by Bhabha Atomic Research Center, Mumbai held during December, 14-16, 2010, published by IEEE Explore, pp.479-484

[19] Singh V.B., Khatri Sujata and Kapur P.K.(2010): A Reliability Growth Model for Object Oriented Software Developed Under Concurrent Distributed Development Environment, published in proceedings of 2nd International Conference on Reliability Safety and Hazard, organized by Bhabha Atomic Research Center, Mumbai held during December, 14-16, 2010, Pp 479-484,Published by IEEE Explore. Kapur P.K., Garg R.B. and Kumar S. (1999) "*Contributions to Hardware and Software Reliability*", World Scientific, Singapore.

[20] K. Pillai and V.S.S. Nair, A Model for Software Development effort and Cost Estimation, *IEEE Transactions on Software Engineering; vol. 23(8)*, 1997, pp. 485-497.

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

[22] S. Yamada, M. Ohba and S. Osaki, S-shaped Software Reliability Growth Models and their Applications, *IEEE Transactions on Reliability R-33*, 1984, PP. 169-175.

[23] Singh V.B., Kapur P.K. and Abhishek Tandon "Measuring Reliability Growth of Software by Considering Fault Dependency, Debugging Time Lag Functions and Irregular Fluctuation" published in May issue Vol. 25, No. 3 ACM SIGSOFT Software Engineering note,2010.

[24] Kapur P.K., H. Pham, Anand Sameer, and Yadav Kalpana A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation, IEEE Transaction on Reliability, March 2011 Volume: 60 Issue: 1 On page(s): 331 - 340

[25] Khatri Sujata, Chhilar R.S. and Singh V.B. "A Generalized Software Reliability Growth Model for Object Oriented Software" ACM SIGSOFT, volume 36, Issue 6, 2011.