International Journal of Enterprise Computing and Business Systems
ISSN (ONLINE) : 2230-8849 Volume 2 Issue 1 January 2012

Analyzing the Impact of Software Reliability Growth Models on Object Oriented Systems during Testing

Sujata Khatri
Deen Dyal Upadhyaya College, Delhi, India

R.S.Chhillar
Department Of Computer Sc. And Applications, Rohtak, India

Arti Chhikara
Maharaja Agrasen College, Delhi, India.

ABSTRACT*:*Software applications are 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. Since development of large software products involves several activities which are need to be suitably coordinated to meet desired requirements. Hence in today's world the importance of developing quality software is no longer an advantage but a necessary factor. Software testing is one the most powerful methods to improve the software quality directly.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.Through this paper, Wehave proposed a model that will determine the proportion of different bug complexity. The paper also suggests the suitability of the proposed model for a particular data set. We haveapplied these models on two data sets based on object oriented methodology namely **SQL for Python** and **SQuirreLSQL Client**software developed under open source environment.

Keywords
-Bug Complexity, Objects Oriented approach,Software Reliability Growth Model, Testability.

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. Over the last two decades the traditional procedure-oriented methodology in software development has been supplanted by a more sophisticated, flexible and reusable object oriented approach. Object oriented programming systems are characterized by several traits, with information is localized around objects rather than functions and data as in procedure oriented approach. An object oriented approach treats the data as a critical element in program development and does not allow it to flow freely around the system. Meyer defines object-oriented design as "the constructionof software systems as structured collections of abstract data type implementations" [1]. The emphasis on object oriented language is on defining abstraction of a model conceptrelatedto an application domain [2].To understand Object Oriented Programming Systems thefollowing 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].
Software testing is one of the most powerful methods to improve software quality. 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].

Software testing is difficult and expensive and testing object oriented system is even more difficult due to its features like inheritance, dynamic binding and polymorphism. Testing of object oriented software has presented numerous challenges due to its features. A tester often needs to spend significant time in developing lengthy testing code to ensure that system under test is completely tested.

The object oriented approach has been widely used for the development of closed source software and open source software. In open source software developers are also the users meaning thereby the one 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].

Software reliability plays an important role in determining reliability growth of software during testing. The software reliability growth model is the tool which can be used to evaluate the software quantitatively, provide development test status, schedule status and monitor the changes in reliability performance. Most of these models are based either on failure count or time between failures. These models are used as estimators i.e. they are used during the software testing and are based on historical failure data.It was Kapur et al. [10] who firstlydeveloped reliability growth model for object oriented software system. The model assumes that the software system may fail due to three types of errors, namely erroneous communication between objects, erroneous execution of private(loca1) variable/data, or erroneous execution of public (global) variable/data (if they exist). The model further assumes that the time dependent behavior of the instruction execution follows either Exponential or Rayleigh curve, while the error removal phenomenon follows Non Homogeneous Poisson Process (NHPP). The model has been validated using a simulated error data. Recently, Singh et al. [19] have also developed a software reliability growth model for object oriented system which categorizesbugs into simple, hard and complex types.

In real practice, it is important to know that how many types of bugs exist in the software at any time, so that different testing strategy and testing effort can be applied to remove those bugs. Bugs may be simple, hard, complex, more complex or even more complex. 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. Recently, Singh et al. [18] havedeveloped a generalized software reliability growth model that determines proportion of bugs of different complexity from open source software.

To the best of our knowledge, no research paper has addressed the issue of complexity of bugs in object oriented software systems. In this paper,Wepropose a generalized model which determines the proportion of complexity of bugs lying dormant in object oriented software system.

In this paper, wehave used actual failure data of two softwares namely SQL for python and SQuirreL SQL Client software developed under open source environment. And the development of software follows object oriented approach.

2. MODEL DEVELOPMENT

In this section, we have formulated a generalized model that will determine the proportion of bug complexity. We also assume that bugs are occurring in software due to accession of private, protected and public variables.

2.1. Notations:

$m(t)$: mean value function of the expected number of detected/removed bugs the time interval [0,t].

i	: type of bug.
p_i	: proportion of type i bugs.
q_i	: proportion of type i accession
$a_i\left(=ap_i\right)$	: initial content of type i bugs.
b_i	: bug removal rate per bug for type i.
J	: number of stages for removing a bug in the software.

2.2. Assumptions of the Model

Following assumptions have been taken for measuring the bug complexity of object oriented software system.
1. A finite number of test cases are prepared to ensure that the software works according to the requirements andspecifications. 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 resulting in different types of bugs.
   3.1.1 Bugs due to private (local) variable/data.
   3.1.2 Bugs due to protected variable/data.
   3.1.3 Bugs due to public (global) Variable/data.
4.The Bugs existing in the software are of n types and each type of bug is modeled by a different growth curves as per their complexity.
5.We assume that first two type of bugs (namely simple and hard) are occurring due to accession of private and protected variables whereas other type of bugs name complex or more complex are occurring due to accession of public variables.
6.The failure observation/error removal phenomenon follows NHPP.
7. No new bug is introduced during removal process.
8.The bug removal intensity per execution is proportional to the remaining bugs in the software.
9. Debugging time lag will increase as complexity of bug increases.
10.The number of instructions executed per unit of time is proportional to the remaining number of instructions not executed.

2.3. Model Formulation

Let total number of instructions execution is W(t) at any give time t. These instructions cause an accession to private, protected and public variable [10 and 19].
Based on the assumptions 2 and 10, the number of instructions per unit of time can be written as

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

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

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

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.

$$W(t) = A\left(1-\exp\left(-Bt\right)\right)$$

If x(t)=Bt, then it follows Rayleigh curve means instructions are not uniformly executed with respect to time.

$$W(t) = A\left(1-\exp\left(-\frac{Bt^2}{2}\right)\right)$$

On each Instruction Execution (IE), the software may execute any of its variable types, namely, private, protected and public. The expected number of accessions to private, protected and public variables is respectively given as

$E_1(t) = q_1W(t)$, $E_2(t) = q_2W(t)$ and $E_3(t) = q_3W(t)$

The total number of accessions due toW(t) number of instruction execution is

$$E(t) = E_1(t) + E_2(t) + E_3(t) = (q_1+q_2+q_3)W(t)$$

Using assumptions (4-9), the bug removal intensity per accession to private, protected and public variableis given as:

Bug removal equation due to accession of private variable is

(Case 1: Simple bugs):

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

Hereb(t) is the bug detection/removal rate and this rate will be different for bugs of different complexity as follows in [20, 22 and 23].a is the initial bug content in software, $p_1$is proportion of simple bugs, $m_1(t)$ is the number of bugs due accession of private variable and $b(t)$ is the bug detection rate.

Firstly by taking $b(t)=b$ in case of simple bugs and solving above differential with initial condition m(0)=0, we get $m_1(t) = ap_1\left(1-\exp\left(-bE_1(t)\right)\right)$ (4)

Bug removal equation due to accession of protected variable is

(**Case 2:** Hard bugs):

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

Where $m_2(t)$ is the number of bugs due accession of protected variable and p_2 is proportion of hard bugs. Solving above differential with initial conditionm(0)=0, we get

$m_2(t) = ap_2\left(1-\left(1+bE_2(t)\right)\exp\left(-bE_2(t)\right)\right)$ (5)

Bug removal equation due to accession of public variable is

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

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

Where $m_3(t)$ is the number of bugs due accession of public variable and p_3 is proportion of complex bugs. Solving above differential with initial condition m(0)=0, we get

$m_3(t) = ap_3\left(1-\left(1+bE_2(t)+\frac{\left(bE_2(t)\right)^2}{2}\right)\exp\left(-bE_3(t)\right)\right)$ (6)

The total error removal is given as

$$m(t) = ap_1\left(1 - \exp\left(-bE_1(t)\right)\right) + ap_2\left(1 - \left(1 + bE_2(t)\right)\exp\left(-bE_2(t)\right)\right) + ap_3\begin{pmatrix} 1 - \left(1 + bE_3(t) + \dfrac{\left(bE_3(t)\right)^2}{2}\right) \\ \exp\left(-bE_3(t)\right)\end{pmatrix} \qquad (7)$$

Here $a = a(p_1 + p_2 + p_3)$ and $E_1 = q_1E, E_2 = q_2E, and\ E_3 = q_3E$

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

$$m(t) = \sum_{i=1}^{n} ap_i\left[1 - \exp\left(-bE_i(t)\right)\left[\sum_{j=0}^{i-1}\frac{\left(b_iE_i(t)\right)^j}{j!}\right]\right]$$

Further assuming that due to accession of public variables more complex bugs are generated in the software and the mean value function obtained in equation (7) can be generalized to n different types of bugs depending upon their complexity. We may write

Or

$$m(t) = \sum_{i=1}^{n} ap_i\left[1 - \exp\left(-bq_iE(t)\right)\left[\sum_{j=0}^{i-1}\frac{\left(bq_iE(t)\right)^j}{j!}\right]\right] \qquad (8)$$

Here, p_1 and p_2 are proportions of bugs due to accession of private and protected variable whereas $p_3, p_4 \ldots.. p_n$ are proportions of bugs due to accession of public variables as follows in assumptions 3 , 4 and 5. q_1, q_2 are proportions of accession due to private and protected variables whereas $q_3, q_4 .. q_n$ are proportions of accession due to public variables. We derive different combination of models namely model 1, model 2, model 3, model 4, model 5 and model 6 from above equation.

3. MODEL VALIDATION

To verify the proposed model that determines types of bugs present in the software due to accession of private, protected and public variable and proportion of accession of private, protected and public variable, we estimated the unknown parameters by using SPSS software tool.We have taken only six models as it increase the number of parameters which makes parameter estimation moredifficult.

3.1 Description of Data sets

Data set-1: MySQL for Python software has been developed under open source environment. 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 (www.sourceforge.net). We have considered only valid bugs which are fixed.

Data set-2:SQuirreL SQL Client is a graphical SQL client written in Java that allow to view the structure of a JDBC compliant database, browse the data in tables, issue SQL commands etc. This software has been developed under open source environment www.sourceforge.net). I collected failure data of SQuirreL SQL Client from 10/3/2001(first bug reported) to 4/26/2010, during this period 298 bugs were reported on bug tracking system.I have used these data sets earlier also, but here I used different models.

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 .03 ,.003 for exponential and rayleigh growth pattern respectively as follows in [10]

We have estimated the parameters of proposed model(equation 8) using SPSS tool for two data sets.

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

Parameter estimates are shown intable [1-4].

Sample of bug reported data of My SQL for Python

Sample of bug reported data of SQuirreL SQL Client Sample

ID	Summary	Status	Opened	Assignee	Submitter	Resolution	Priority
467386	Remember state of tree when refreshing	Closed	10/3/2001	gmackness	Colbell	Fixed	5
467979	Cannot type sql undr JDK1.4	Closed	10/4/2001	Colbell	Nobody	Fixed	5
472539	wrong path separator	Closed	10/18/2001	Colbell	Nobody	Fixed	5
474592	SQL Exception with Sybase ASE 12	Closed	10/24/2001	Colbell	Diegodalcero	Fixed	5
489985	Cannot 'Cancel' from a SQL query	Closed	12/6/2001	Colbell	Gmackness	Fixed	5
520500	Databased not displayed in object browsr	Closed	2/20/2002	Colbell	Nobody	Fixed	5
525621	SQL History	Closed	3/4/2002	Colbell	Dmishee	Fixed	5
526656	Incorrect linefeed replacement	Closed	3/6/2002	Colbell	Tlarsen	Fixed	5
526989	skinLF and kunstoff not available	Closed	3/7/2002	Colbell	Nobody	Fixed	5

3.3. Data Analysis:

Estimation results of different proposed model:

Table-1: Parameter Estimates of My-SQL Dataset (Rayleigh)

Models	A	b	p_1	p_2	p_3	p_4	p_5	p_6	q_1	q_2	q_3	q_4	q_5	q_6

Model 1	155	.006	-	-	-	-	-	-	-	-	-	-	-	-
Model 2	155	.109	.226	.774	-	-	-	-	.917	.083	-	-	-	-
Model 3	150	.160	.183	.709	.108	-	-	-	.754	.157	.089	-	-	-
Model 4	155	.333	.205	.750	.044	.001	-	-	.747	.026	.227	.000	-	-
Model 5	147	.256	.964	.025	.000	.007	.005		.052	.190	.467	.076	.215	
Model 6	174	.033	.542	.013	.371	.000	.001	.073	.537	.053	.390	.007	.013	.000

Table-2: Parameter Estimates of My-SQL Dataset (Exponential)

Models	A	B	p_1	p_2	p_3	p_4	p_5	p_6	q_1	q_2	q_3	q_4	q_5	q_6
Model 1**		-	-	-	-	-	-	-	-	-	-	-	-	-
Model 2	356	.007	.076	.924	-	-	-	-	.732	.268	-	-	-	-
Model 3	207	.011	.378	.000	.622	-	-	-	.425	.061	.000	-	-	-
Model 4	190	.734	.052	.896	.021	.032	-	-	.004	.017	.010	.968	-	-
Model 5	453	.006	.474	.526	.000	.000	.000	-	.105	.268	.000	.028	.599	
Model 6	251	.011	.650	.000	.000	.343	.007	.000	.178	.000	.011	.714	.095	.002

Table-3: Parameter Estimates of Squirrel Dataset (Rayleigh)

Models	A	B	p_1	p_2	p_3	p_4	p_5	p_6	q_1	q_2	q_3	q_4	q_5	q_6
Model 1	332	.004	-	-	-	-	-	-	-	-	-	-	-	-
Model 2	313	.077	.188	.812		-	-	-	.910	.090	-	-	-	-
Model 3	282	.804	.084	.882	.035	-	-	-	.943	.039	.019	-		-
Model 4	309	.176	.921	.030	.038	.011	-	-	.048	.756	.050	.146	-	-
Model 5	467	.032	.478	.000	.184	.338	.000		.274	.000	.316	.000	.410	
Model 6	399	.014	.369	.575	.038	.000	.004	.015	.719	.281	.000	.000	.000	.000

Table-4: Parameter Estimates SQuirrel Dataset (Exponential)

Models	A	B	p_1	p_2	p_3	p_4	p_5	p_6	q_1	q_2	q_3	q_4	q_5	q_6
Model 1**		-	-	-	-	-	-	-	-	-	-		-	-

Model 2	473	.036	.015	.985	-	-	-	-	.923	.077		-	-	-
Model 3	382	.026	.049	.481	.470	-	-	-	.185	.571	.244	-		-
Model 4	361	.078	.050	.589	.223	.138	-	-	.122	.392	.092	.393		
Model 5	442	.022	.041	.814	.000	.039	.107	-	.078	.322	.167	.003	.429	
Model 6	593	.009	.397	.000	.574	.018	.000	.010	.273	.189	.506	.000	.000	.032

Here, p_1 , p_2 , p_3 , p_4 p_5 , p_6 are the proportion of different complexity of bugs and $q_1, q_2, q_3, q_4, q_5, q_6$ are proportion of accession of different type of variables namely private, private and protected and public variables respectively.

** Parameters are not estimated

3.4. Description of Tables:

Table 1 shows proportion of different complexity of bugs. Here,Model 2 estimates the presence of two types of bugs mainly with proportion of 23% and 77%; Model 3 estimatesthe presence of three types of bugs with proportion of 18%, 71% and 11%; Model 4 estimates the presence of two types ofbugs mainly with proportionof 21% and 75%; Model 5 estimates maximumbugs of Type 1 with proportion of96%; and Model 6 estimatesfour types of bugs with different proportions. In this table q1...q6 shows proportion of accession of private, protected and public variable.

In Table 2, Model 2 estimates the presence of two types of bugs and mainly of Type 2;Model 3 estimates the presence of mainly two types of bugs with proportion of 38% and 62%;Model 4 estimates the presence of four types ofbugs with different proportion; Model 5 estimates maximum bugs of Type 1 and Type 4with proportion of47% and 53%; finally,Model 6 estimates maximum bugs of Type 1 and Type 4 with proportion of 65% and 34%.

In Table 3, Model 2 estimates the presence of two types of bugs with proportion of19% and 81%; Model 3 estimates the presence of three types of bugs where maximum bugs are of Type 2; Model 4 estimates maximum bugs of Type 1 with proportion of 92%;Model 5 estimates maximum bugs of Type 1 and Type 4 with proportion of 48% and 34%;and Model 6 estimates maximum bugs of Type 1 and Type 2 with proportionof 37% and 58%.

In Table 4, Model 2 estimates the presence of two types of bugs with proportion of 99% and 1%; Model 3 estimates the presence of two types of bugsmainly with proportion of48% and47%; Model 4 estimates the presence of four types ofbugs;Model 5 estimates maximum bugs of Type 2 with proportion of 81% and Model 6 estimatesmaximum bugsof Type 1 and Type 3.

4. CONCLUSION:

Object oriented approach has eased the software development process and in making a clear understanding of the requirements of the real world problems. In this paper, we proposed a generalized model that determines the proportion of bug complexity present in the software. The prior knowledge of distribution of bugs of different complexity will help project manager in allocation of testing efforts and tools. We have provided the numerical results of combinations of up to six types of models. The component model of these combinations represents different types of bugs which follows different typesof growth curves depending upon their complexity.The knowledge about failure distribution and their complexity can also improve the testability of software. Testing effort allocation can be made easy by knowing the failure distribution and complexity of faults, and this will ease the process of revealing faults from the software. As a result, the testability of the software will be improved.

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

6. 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*HawaiiInternational 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 Symposiumon theproduction 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,* BhartiyaVidyapith's Institute of Computer Applicationsand Management, New Delhi, pp.279-284, held during23rd-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, 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.

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

[21] 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.

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

[23] 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.

[24] 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