Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 13 Issue 13 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Simulation of Reliability of Software Component

By Dr. P. K. Suri & Er. Karambir
University Institute of Eengg and Technology, India

Abstract- Component-Based Software Engineering (CBSE) is increasingly being accepted worldwide for software development, in most of the industries. Software reliability is defined as the probability that a software system operates with no failure within a specified time on specified operating conditions. Software component reliability and failure intensity are two important parameters that Estimates the reliability of system after integration of component. The estimation of reliability of software can save loss of time, life and cost. In this paper, software reliability has been estimated by analyzing the failure data. The Imperfect Software Reliability Growth Models (SRGMs) model have been used for simulating the software reliability by estimating the number of remaining faults and the model parameters of the fault content rate function. We aim for simulating software reliability by connecting the imperfect debugging and Goel-Okumoto model. The estimation of reliability gives the time of stopping the unending testing of that component or time of release of software component.

Keywords: component based software engineering (cbse), software reliability growth model (srgm), reliability, goel- okumoto model.

GJCST-C Classification : D.2.4

Strictly as per the compliance and regulations of:

Simulation of Reliability of Software Component

Dr. P. K. Suri [α] & Er. Karambir [σ]

Abstract- Component-Based Software Engineering (CBSE) is increasingly being accepted worldwide for software development, in most of the industries. Software reliability is defined as the probability that a software system operates with no failure within a specified time on specified operating conditions. Software component reliability and failure intensity are two important parameters that Estimates the reliability of system after integration of component. The estimation of reliability of software can save loss of time, life and cost. In this paper, software reliability has been estimated by analyzing the failure data. The Imperfect Software Reliability Growth Models (SRGMs) model have been used for simulating the software reliability by estimating the number of remaining faults and the model parameters of the fault content rate function. We aim for simulating software reliability by connecting the imperfect debugging and Goel-Okumoto model. The estimation of reliability gives the time of stopping the unending testing of that component or time of release of software component.

Keywords: component based software engineering (cbse), software reliability growth model (srgm), reliability, goel- okumoto model.

I. Introduction

With the popularity of the web and networked computers are finding their way into a wide and spread range of working environments. This new computing model have made a competition of early development, reliable and distributed software components that communicate with one another across the underlying networked and extendable infrastructure as per the requirement of different user. A distributed software component can be plugged into distributed applications and can be used for some specific purpose. The intention of most of the developers behind is reuse or slight modification of old or reliable component and this makes more reliable by using distributed software components to build new systems. Even though, it is also important for developer to know the functionality of distributed or compatible software component in any system. The design of component and requirement specification should clearly document the functional input, output with conditions and moreover it is the reliability percentage wise. Software reliability has been defined as the probability that a software system operates with no failure for a specified time on specified operating conditions. In other words, by estimating or predicting the reliability [1] of component, the quality of software product can be estimated. The satisfaction of customers is directly dependent on the quality of that software. The analysis report that is commonly used to describe software reliability has been derived from observed or failure intensity. Failure intensity is defined as the number of failures observed per unit time period. Failure intensity is a also good measure for reflecting the user perspective of software quality. As Computer applications are going more diverse and spreading through almost every area of everyday life then reliability factor becomes a very important characteristic of software or component systems. The reliable component is a base of system and part of system i.e. client, administrator and working environment. Since it is a matter of cost and performance to produce a system having documented and estimated reliability [2] of system. Therefore, it is necessary to measure its reliability before releasing any software. When reliability reaches at threshold level then the software component can be released for further use. To do this, a number of models [3] have been proposed and has been being developed. Software modeling is a statistical estimation [4] method applied to failure data collected or simulated the software component or system developed after integration of software component by different approach of joining in software engineering .This can be one after a component testing has been executed so that failure data are available. The implementation of newly developed and modified models tries to make system better and help in predicting the reliability in a accurate way. The most important parameter of any software product are level of quality, time of delivery, and final cost of the product. The time of delivery and cost should be quantitative and pre decided, whereas these attributes is difficult to define Quantitatively. Reliability is one, and probably the most Software reliability is related directly to operation and performance instead of designing of a component.

Therefore, software reliability is estimated by analyzing the observed failure data [5] of component and then applying Goel –Okumoto Model [6][7] , rather than the number of remaining faults in a component. So, estimation of reliability of system is more useful than finding the number of remaining fault. The uncertainty involved in the estimation for a specific interval expressed in terms of confidence interval and estimation of parameter used. This paper evaluates the estimates the reliability of component by using the Goel- Okumoto

Author α: Dean (R &D), Professor & Chairman (CSE/IT/MCA) HCTM Technical Campus, Kaithal (Haryana) India.
e-mail: pksurikuk@gmail.com
Author σ: Assistant Professor, Department of Computer Science and Engineering, University Institute of Engg and Technology, Kurukshetra Univeristy, Kurukshetra (Haryana) India.
e-mail: bidhankarambir@rediffmail.com

model on the set of failure data taken from simulating the real software applications. This should be done before any component release .The reusability of that component enhances the overall reliability of system and gives accurate estimation of value of reliability. The result shows that the proposed model has a technical point for improving software reliability and providing additional metrics for development project evaluation, management and time of delivery of new developed system.

II. Goel Okumoto Model : NHPP SRGM Exponential Model

Non Homogeneous Poisson Process (NHPP) based software reliability growth models are generally classified into two groups. The first group contains models, which use the machine execution time or calendar time [8] as a unit of fault detection/removal period. Such models are called continuous time models. The second group contains models, which use the number of test occasions/cases as a unit of fault detection period. Such models are called discrete time models [9], since the unit of software fault detection period is countable. A Goel Okumoto Model also known as exponential NHPP model is based on the following assumptions:(a) All fault in a component are independent from the failure detected. (b)The number of failures detected at any time is proportional to the current number of fault in a component. (c) The fault is removed immediately as soon as the failure happens, no new faults are introduced during the removal of fault.

The following differential Equation1 include the above assumptions. Where m(t) is expected number of component failures by time t, a is Total fault content rate function, i.e., the sum of expected number of initial software faults and introduced faults by time t and b is Failure detection rate per fault at time t.
important, aspect of software quality. S

$$\frac{\partial m(t)}{\partial t} = b[a - m(t)] \tag{1}$$

The mean value solution of above differential equation is given by Equation 2 where tn is time of nth failure occurrence

$$m(t) = a\left(1 - \mathrm{e}^{-\mathrm{bt_n}}\right) \tag{2}$$

Failure intensity function is given by Equation 3 as follows

$$\lambda(t) = abe^{-bt_n} \tag{3}$$

III. Estimation of Parameter

The different a and b parameter also reflect different assumptions of the software testing processes. In this section, we derive a new NHPP model for an interrelationship dependent function between a and b parameter by a common parameter from a generalized class of model. The most common method for the estimation of parameters is the Maximum Likelihood Estimation (MLE) method. MLE method of estimation of a broad collection of software reliability for grouped data is discussed in detail. To estimate a and b for a sample of n units, first obtain the likelihood function: take the natural logarithm on both sides. The equation for estimation of a and b is given in Equation 4 where

$$\mathrm{a} = \frac{\mathrm{y_n}}{(1-e^{-bt_n}} \tag{4}$$

Where yn is actual value of nth failure observed at time t. The parameter a can be estimated using MLE method based on the number o failures in a particular interval. Suppose that an observation interval {0, tk} is divided into set of sub intervals (0,ti],(t1,t2].....(tk-1,tk] , Equation 5 was used to determine the value of b .

$$\frac{\mathrm{y_n t_n e^{-bt_n}}}{1-\mathrm{e^{-bt_n}}} = \sum_{k=1}^{n}\left(\frac{(\mathrm{y_k}-y_{k-1})(\mathrm{t_k e^{-bt_k}} - \mathrm{t_{k-1}e^{-bt_{k-1}}})}{(\mathrm{e^{-bt_{k-1}}} - \mathrm{e^{-bt_k}})}\right) \tag{5}$$

The number of failures per subinterval [8] is recorded as ni(i=1,2,3..,k) with respect to the number of failures in(ti-1,ti].The parameters a and b are estimated using iterative Newton Raphson Method, which is given as in Equation 6 Equation 7 and Equation 8 .

$$b = \mathrm{b_0} - \frac{\mathrm{f(b_0)}}{\mathrm{f'(b_0)}} \tag{6}$$

$$f(b) = \frac{\mathrm{y_n t_n e^{-bt_n}}}{1-\mathrm{e^{-bt_n}}} - \sum_{k=1}^{n}\left(\frac{(\mathrm{y_k}-y_{k-1})(\mathrm{t_k e^{-bt_k}} - \mathrm{t_{k-1}e^{-bt_{k-1}}})}{(\mathrm{e^{-bt_{k-1}}} - \mathrm{e^{-bt_k}})}\right) = 0 \tag{7}$$

$$f'(b) = \sum_{k=1}^{n}\left(\frac{(\mathrm{y_k}-y_{k-1})(\mathrm{t_k}-\mathrm{t_{k-1}})^2\mathrm{e^{-b(t_k+t_{k-1})}}}{(\mathrm{e^{-bt_{k-1}}} - \mathrm{e^{-bt_k}})^2}\right) \tag{8}$$

IV. Model Analysis and Results

a) Data and Model Criteria

Once the analytical expression for the mean value function m(t) is derived, in this paper, the model parameters to be estimated in the mean value function can then be obtained with the help of a developed octave program based on the least squares estimate (LSE) method. Goel and Okumoto described failure detection as a non-homogeneous Poisson process with an exponentially decaying rate function .It is a simple non-homogeneous Poisson process model. The data of failure of 25 days have been observed here for estimating the reliability [10]. In table 1, the data of 25 days failure and cumulated failure have been shown here.

The two function of Reliability and Remaining fault function can be used to find the release of date or the additional testing time is required to reach ready state. After simulation the result of 25 days of testing were observed. Based on these data and using the MLE

method, the estimated values for the two parameter are given in the table. Each data set provides the cumulative number of faults by each week up to 25 weeks. The Fig. 1 represents the cumulative number of faults versus the cumulative system test hours at the end of each The Phase 2 data set is given in Table 2. We developed a Octave program to perform the analysis and all the calculations for LSE estimates. The parameter a is the number of initial faults in the software and the parameter b is related to the failure detection rate during testing process.

Table 1 : Number of Failure Observed

Days	Failure Observed	Cumulative Failure	Days	Failure Observed	CumulativeFailure
1	32	32	14	5	127
2	23	55	15	5	132
3	11	66	16	6	138
4	10	76	17	3	141
5	11	87	18	5	146
6	7	94	19	1	147
7	2	96	20	1	148
8	5	101	21	3	151
9	6	107	22	1	152
10	2	109	23	2	154
11	4	113	24	1	155
12	7	120	25	1	156
13	2	122			

The software reliability R(x/t) is defined as the probability of a failure free operations of a complete software for a specified time i.e. interval (t, t +x) in a specified environment in Equation 9. The interval methods of estimation are explained by applying the results to the software failure data .The set of software errors analyzed here is borrowed from a simulated data (an 1 days interval). where R(s|t) is reliability of component during (t, t+s) time.

$$R(xIt) = e^{-a(e^{-bt} - e^{-b(t+x)})} \quad (9)$$

Table 2 : Remaining Fault,Reliability, Failure Intensity

Day	A	B	Remaining Fault	Reliability	Failure Intensity
15	138.38	0.1333	16	80.5	2.1844
16	133.71	0.1432	12	84.66	1.6769
17	141.25	0.1274	14	83.48	1.8162
18	139.72	0.1304	12	85.91	1.5286
19	138.85	0.1322	10	87.86	1.3030
20	140.34	0.1290	9	88.71	1.2052
21	140.10	0.1295	8	90.09	1.0495
22	141.91	0.1255	8	90.60	0.9929
23	142.03	0.1252	7	91.62	0.8801
24	142.3154	0.1246	6	92.48	0.7869
25	141.13	0.1275	5	93.71	0.6538

b) Analysis

In fig 1 the cumulative failure observed have been shown as per number of days of testing. It is obviously seen that the number of fault observed is decreasing with days. The number of fault is initially is more but with time the number of fault is decreasing. The estimation of remaining fault decreases rapidly then it becomes straight and it is at low level in fig 2. The remaining fault is never zero as per fig 2. The reliability graph 3 shows that the growth is initially is fast but with time it is also decreasing. It reaches above of 90% after 21 days. The graph of comparision of reliability versus remaining fault describes that the as the reliability is not inversely proportional to the remaining fault as shown in fig 4. The failure intensity of component is slightly decreasing with the number of days of testing from fig 5. As per the solution of problem definition mentioned in 25 days of additional testing is required to get an benchmarks level of reliability and acceptable number of remaining faults so that the software can be released for final delivery.

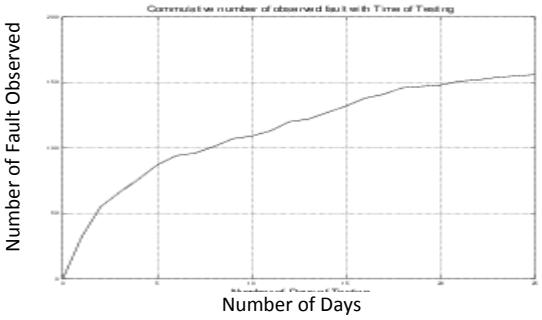

Figure 1 : Number of Fault Observed Wrt Number of Days of Testing

For example the component can be released the software if the expected reliability is greater than the threshold value i.e 90.09778 %and above 90%.

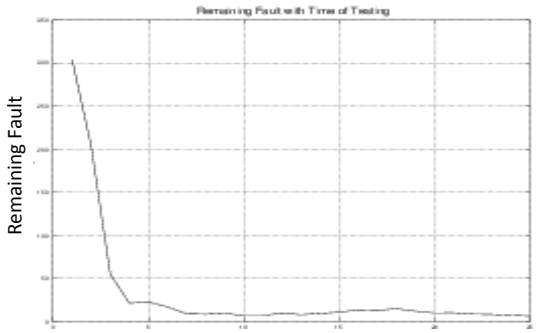

Figure 2 : Remaining Fault wrt number of Days of testing

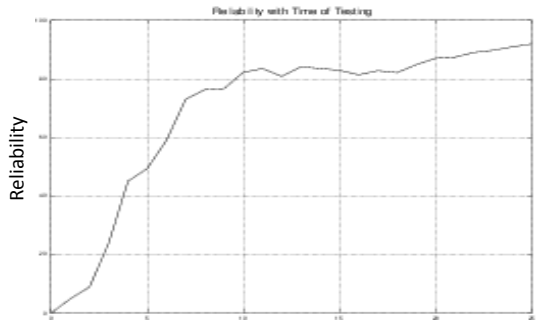

Figure 3 : Reliability wrt Number of Days of Testing

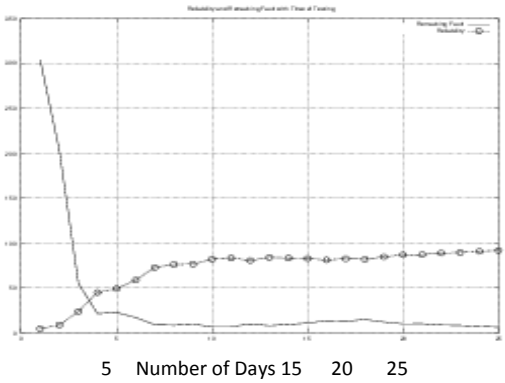

Figure 4 : Comparision of Remaining Fault and Reliability wrt Number of Days of Testing

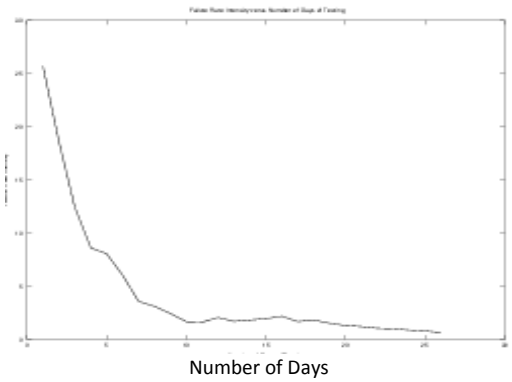

Figure 5 : Failure Intensity wrt Number of Days of Testing

V. Conclusion

This work has proposed a method of estimating the reliability of reusable architecture which can be used to build a software by using Goel-Okumoto Software Reliability Growth Model. The data available from an exponential distribution are grouped and the this model used to illustrate the parameter estimation problem. The measurement of reliability decides the quality and level of reliability decides the time of delivery of any software the reliability is increased with testing time but the reliability never becomes 100% even when the observed fault is close to zero. As per the other criteria in the above analysis, the best estimate for the remaining fault is less than 10 and then the component can be released. The integration of more reliable component can make the system more reliable. This solution will help the developer of third party component to predict the release the component with the specified marked reliability.

References Références Referencias

1. H. Pham. "System Software Reliability", Springer Series in Reliability Engineering, Springer, London, pp.149-149, 2006.
2. X. Teng, H. Pham. "A New Methodology for Predicting Soft-ware Reliability in the Random Field Environments". IEEE Transactions on Reliability, vol. 55, no. 3, pp. 458-468, 2006.
3. L. Pham, H. Pham. "Software Reliability Models with Time dependent Hazard Function Based on Bayesian Approach", IEEE Transactions on Systems, Man, and Cybernetics Part A, vol. 30, no. 1, pp. 25-35, 2000.
4. H. Pham. "Software Reliability Assessment: Imperfect Debugging and Multiple Failure Types in Software Development", EG and G-RAAM-10737, Idaho National EngineeringLaboratory, 1993.
5. H. Pham. "A Software Cost Model with Imperfect Debugging, Random Life Cycle and Penalty Cost", International Journal of Systems Science, vol. 27, no. 5, pp. 455-463, 1996.

6. A. L. Goel, K. Okumoto. "Time-dependent Fault-detection Rate Model for Software and Other Performance Measures", .IEEE Transactions on Reliability, vol. 28, pp. 206-211, 1979.
7. S. Yamada, S. Osaki., "Software Reliability Growth Modeling: Models and Applications", IEEE Transactions on Software Engineering, vol. 11, no. 12, pp. 1431-1437, 1985.
8. Dr. R.Satya Prasad, Bandla Srinivasa Rao, Dr. R.R. L. Kantham, "Assessing Software Reliability using Inter Failures Time Data", International Journal of Computer Applications Volume18-No-7, March 2011
9. X. Zhang, X. Teng, H. Pham. "Considering Fault Removal Efficiency in Software Reliability Assessment", IEEE Transactions on Systems, Man, and Cybernetics - Part A, vol.33, no. 1, pp. 114-120, 2003.
10. H. Pham, X. Zhang," An NHPP Software Reliability Models and its Comparison", International Journal of Reliability, Quality and Safety Engineering, vol. 4, no. 3, pp. 269-282, 1997.

This page is intentionally left blank