International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 2, Issue 3, March 2015, PP 21-24
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
www.arcjournals.org

Burr Type III Software Reliability with SPC-An Order Statistics Approach

K. Sobhana

Department of Computer Science
Krishna University
Machilipatnam, Andhra Pradesh,India
msobhana@yahoo.com

Dr. R. Satya Prasad

Dept. of Computer Science & Engg
Acharya Nagarjuna University
Guntur, Andhra Pradesh (India)
prof_rsp@gmail.com

Abstract: *In the past few decades research on software reliability has been conducted and several software reliability growth models have been developed for estimating software reliability. Order Statistics is an approach for estimating software reliability for time domain data based on NHPP with a distribution model. This paper presents the Burr Type III model as a software reliability growth model and derives the expressions for an efficient reliability function using order statistics. Statistical Process Control (SPC) can be used to monitor the software reliability process and thereby improve the software quality. Control Charts are one of the powerful SPC tools to analyze the failure frequency. It is proposed that the SPC can be applied to monitor the software failure process of Burr Type III based NHPP.*

Keywords: *Software reliability; Burr type III; Order Statistics; Statistical Process Control;NHPP*

1. INTRODUCTION

Software reliability is one of the most important characteristics of software quality. Reliable software systems can be produced and maintained by employing quality measurement and management technologies during the software life cycle. Software Reliability is the probability of failure free operation of software in a specified environment during specified time [1].

The monitoring of Software reliability process is a far from simple activity. In recent years, several authors have recommended the use of SPC for software process monitoring. A few others have highlighted the potential pitfalls in its use[2].

The main thrust of the paper is to formalize and present an array of guidelines in a disciplined process with a view to helping the practitioner in putting SPC to correct use during software process monitoring.

Over the years, SPC has come to be widely used among others, in manufacturing industries for the purpose of controlling and improving processes. Our effort is to apply SPC techniques in the software development process so as to improve software reliability and quality [3]. It is reported that SPC can be successfully applied to several processes for software development, including software reliability process. SPC is traditionally so well adopted in manufacturing industry. In general software development activities are more process centric than product centric which makes it difficult to apply SPC in a straight forward manner.

The utilization of SPC for software reliability has been the subject of study of several researchers. A few of these studies are based on reliability process improvement models. They turn the search light on SPC as a means of accomplishing high process maturities. Some of the studies furnish guidelines in the use of SPC by modifying general SPC principles to suit the special requirements of software development [3] (Burr and Owen[4]; Flora and Carleton[5]). It is especially noteworthy that Burr and Owen provide seminal guidelines by delineating the techniques currently in vogue for managing and controlling the reliability of software. Significantly, in doing so, their focus is on control charts as efficient and appropriate SPC tools.

It is accepted on all hands that Statistical process control acts as a powerful tool for bringing about improvement of quality as well as productivity of any manufacturing procedure and is particularly relevant to software development also. Viewed in this light, SPC is a method of process management through application of statistical analysis, which involves and includes the defining, measuring, controlling, and improving of the processes[6].

2.1. Model Development

A. NHPP Model

Software reliability probabilistic models can be classified as Markovian models and fault counting models. In Markovian model a Markov process represents the failure process. In fault counting model the failure process is described by stochastic process like Homogeneous Poisson Process (HPP), Non Homogeneous Poisson Process (NHPP) and

Compound Poisson Process etc. A majority of failure counting models are based upon NHPP described in the following lines[7].

A software system is subject to failures at random times due to the errors present in the system. Let {N(t), t>0} be a counting process representing the cumulative number of failures by time t. Since there are no failures at t=0 we have N(0) = 0.

It is assumed that the number of software failures during non-overlapping time intervals do not affect each other. It can be mentioned that for finite times t1<t2<t3<….< tn , the n random variables N(t1), {N(t2)-N(t1)}, ….. {N(tn)-N(tn-1)} are independent. It implies that the counting process {N(t), t>0} has independent increments [8].

Let m(t) denote the expected number of software failures by time 't'. Since the expected number of errors remaining in the system at any time is finite, m(t) is bounded, non-decreasing function of 't' with the boundary conditions

$$m(t) = \begin{matrix} 0, t = 0 \\ a, t \to \infty \end{matrix}$$

where 'a' is the expected number of software errors that need to be detected.

For t>=0 N(t) is known to have a Poisson Probability mass function with parameters m(t) i.e.,

$$P\{N(t) = n\} = \frac{[m(t)]^n e^{-m(t)}}{n!} \quad n = 0,1,2 \ldots\ldots \infty$$

The behaviour of software failure phenomena can be illustrated through N(t) process. Several time domain models exist in the literature [12] which specify that the mean value function m(t) will be varied for each NHPP process.

The mean value function of Burr Type III software reliability growth is given by

$$m(t) = a[1 + t^{-c}]^{-b} \tag{1}$$

Here, we consider the performance given by the Burr Type III software reliability growth model based on order statistics and whose mean value function is given by

$$m(t) = \left(a\left(1 + (t_i)^{-c}\right)^{-b} \right)^r \tag{2}$$

Where [m(t)/a] is the cumulative distribution function of Ordered Burr distribution model $P\{N(t) = n\} = \frac{m(t)^n e^{-m(t)}}{n!}$

$$\lim_{n \to \infty} P\{N(t) = n\} = \frac{a^n e^{-a}}{n!}$$

This is considered as Poisson model with mean a.

Let S_k be the time between $(k-1)^{th}$ and k^{th} failure of the software product. It is assumed that X_k be the time up to the k^{th} failure. We need to find out the probability of the time between $(k-1)^{th}$ and k^{th} failures. The Software Reliability function is given by

$$R = \frac{S_k}{X_{(k-1)}}(s / x) = e^{-[m(x+s) - m(s)]} \tag{3}$$

B. Estimation Based on Inter Failure Times

In this section, the expressions are generated for estimating the parameters of the Ordered Burr Type III model based on the time between the failures. The expressions for a, b, and c have to be derived.

Let S1, S2, ….be a sequence of times between successive software failures associated with an NHPP N(t). Let X_k be equal to

$$\sum_{i=1}^{k} S_i, k = 1,2,3 \ldots\ldots$$

This represents the time at which failure k occurs. Suppose we are given with 'n' software failure times say $x_1, x_2, \ldots, x_n$, there are 'n' time instants at which the first, second, third …. n^{th} failure of software is observed.

The mean value function of Order Burr Type III is given by

$$m(t) = \left(a\left(1 + (t_i)^{-c}\right)^{-b}\right)^r \tag{4}$$

The constants a, b and c in the mean value function are called parameters of the proposed model. To assess the software reliability, it is necessary to compute the expressions for finding the values of a, b and c. For doing this, Maximum Likelihood estimation is used whose Log Likelihood function is given by

$$LLF = \sum_{i=1}^{n} Log\ [\lambda(t_i)^r - m(t_n)^r \tag{5}$$

Differenciating m(t) with respect to 't' we get λ (t)

$$\lambda(t) = \frac{rabc}{(t_i)^{(c+1)} * [1 + (ti)^{-c}]^{(br+1)}} \tag{6}$$

The log likelihood equation to estimate the unknown parameters a, b, c after substituting (5) in (4) is given by

$$LogL = -[a[1+(t_n)^{-c}]^b]^r + \sum_{i=1}^{n} [\log r + \log a + \log b + \log c] + \sum_{i-1}^{n} [-(br+1)\log(1 + (t_i)^{-c}) - (c+1)\log(t_i)] \tag{7}$$

Differentiating LogL with respect to 'a' and equating to 0 (i.e $\frac{\partial \log L}{\partial a} = 0$) we get

$$a^r = \frac{n(1 + (t_n)^{-c})^{br}}{r} \quad (8)$$

Differentiating LogL with respect to 'b' and equating to 0 (i.e $\frac{\partial \log L}{\partial b} = 0$) we get

g(b) =

$$\frac{n}{b} + \sum_{i=1}^{n} r \log(1 + (t_i)^{-1}) + \frac{n^2 (1 + (t_n)^{-1})^{br}}{r} \log(1 + (t_n)^{-1})$$

We can group the inter-failure time data into non overlapping successive sub groups of size 4 or 5 and add the failure times within each sub group.

For instance if a data of 100 inter-failure times are available we can group them into 20 disjoint subgroups of size 5. The sum total in each subgroup would denote the time lapse between every 5th order statistic in a sample of size 5. In general for inter-failure data of size 'n', if r (any natural (9)

Again Differentiating g(b) with respect to 'b' and equating to 0 (i.e $\frac{\partial^2 \log L}{\partial b^2} = 0$)

g'(b) =

$$\frac{-n}{b^2} + n^2 (1 + (t_n)^{-1})^{br} . \log^2 (1 + (t_n)^{-1}) \quad (10)$$

Differentiating LogL with respect to 'c' and equating to 0 (i.e $\frac{\partial \log L}{\partial c} = 0$) we get

g(c) =

$$\frac{n}{c} + \sum_{i=1}^{n} \left(\frac{(r+1)(t_i) - c}{1 + (t_i) - c} - 1\right) \log t_i - \frac{n(t_n) - c \log t_n}{(1 + (t_n)^{-c})} \quad (11)$$

Again Differentiating g(c) with respect to 'c' and equating to 0

(i.e $\frac{\partial^2 \log L}{\partial c^2} = 0$) we get

g'(c) =

$$\frac{-n}{c^2} + \sum_{i=1}^{n} \frac{(r+1)(\log t_i)^2 (t_i)^{-c}}{(1 + (t_i)^{-c})^2} + \frac{n \log(t_n)^2 (t_n)^{-c}}{(1 + (t_n)^{-c})^2} \quad (12)$$

The parameters 'b' and 'c' are estimated by iterative Newton-Raphson Method using

$$b_{n+1} = b_n - \frac{g(b_n)}{g'(b_n)} \quad (13)$$

$$c_{n+1} = c_n - \frac{g(c_n)}{g'(c_n)} \quad (14)$$

which are substituted in (7) to determine 'a'.

C. Order Statistics

Order Statistics can be used in several applications like data compression, survival analysis, Study of Reliability and many others [9]. Let X denote a continuous random variable with probability density function f(x) and cumulative distribution function F(x), and let (X1 , X2 , …, Xn) denote a random sample of size n drawn on X. The original sample observations may be unordered with respect to magnitude. A transformation is required to produce a corresponding ordered sample. Let (X(1) , X(2) , …, X(n)) denote the ordered random sample such that X(1) < X(2) < … < X(n); then (X(1), X(2), …, X(n)) are collectively known as the order statistics derived from the parent X. The various distributional characteristics can be known from Balakrishnan and Cohen [9].

The inter-failure time data represent the time lapse between every two consecutive failures. On the other hand if a reasonable waiting time for failures is not a serious problem, we can group the inter-failure time data into non overlapping successive sub groups of size 4 or 5 and add the failure times within each sub group.

Number less than 'n' and preferably a factor n, we can conveniently divide the data into 'k' disjoint subgroups (k=n/r) and the cumulative total in each subgroup indicate the time between every r^{th} failure. The probability distribution of such a time lapse would be that of the ordered statistic in a subgroup of size r, which would be equal to power of the distribution function of the original variable (m(t)).

The whole process involves the mathematical model of the mean value function and knowledge about its parameters. If the parameters are known they can be taken as they are for the further analysis, if the parameters are not known they have to be estimated using a sample data by any admissible, efficient method of estimation. This is essential because the control limits depend on mean value function, which in turn depends on the parameters. If software failures are quite frequent, keeping track of inter-failure is tedious. If failures are more frequent order statistics are preferable [9].

D. Monitoring the time between failures using control chart

The selection of proper SPC charts is essential to effective statistical process control implementation and use. There are many charts which use statistical techniques. It is important to use the best chart for the given data, situation and need[10].

There are advances charts that provide more effective statistical analysis. The basic types of advanced charts, depending on the type of data are the variable and attribute charts. Variable control chats are

designed to control product or process parameters which are measured on a continuous measurement scale. X-bar, R charts are variable control charts.

Attributes are characteristics of a process which are stated in terms of good are bad, accept or reject, etc. Attribute charts are not sensitive to variation in the process variables charts. However, when dealing with attributes and used properly, especially by incorporating a real time pareto analysis, they can be effective improvement tools. For attribute data there are : p-charts, c-charts, np-charts, and u-charts. We have named the control chart as ***Failures Control Chart*** in this paper. The said control chart helps to assess the software failure phenomena on the basis of the given inter-failure time data[11].

E. Distribution of Time Between Failures

For a software system during normal operation, failures are random events caused by, for example, problem in design or analysis and in some cases insufficient testing of software. In this paper we applied Burr Type III to time between failures data. This distribution uses cumulative time between failure data for reliability monitoring.

The equation for mean value function of Burr Type III from equation [1] is

$$m(t) = a[1 + t^{-c}]^{-b}$$

Equate the pdf of above m(t) to 0.99865, 0.00135, 0.5 and the respective control limits are given by.

$$T_u = [1 + t^{-c}]^{-b} = 0.99865$$

$$T_c = [1 + t^{-c}]^{-b} = 0.5$$

$$T_l = [1 + t^{-c}]^{-b} = 0.00135$$

These limits are converted to $m(t_u)$,$m(t_c)$and $m(t_l)$ form. They are used to find whether the software process is in control or not by placing the points in control charts.

2. CONCLUSION

Software reliability is an important measure of quality that determines the failure free operation of a computer system. In this paper we proposed Burr type III software reliability model using order statistics for estimating and monitoring reliability.

Equations to estimate the parameters based on time domain data are obtained using Maximum likelihood. We conclude that our method of estimation can be used in applying the control charts an SPC tool for early detection of software failure and thereby improve software quality.

REFERENCES

[1] *Musa J.D, Software Reliability Engineering MCGraw-Hill, 1998.*

[2] *N. Boffoli, G. Bruno, D. Cavivano, G. Mastelloni; Statistical process control for Software: a systematic approach; 2008 ACM 978-1-595933-971-5/08/10.*

[3] *K. U. Sargut, O. Demirors; Utilization of statistical process control (SPC) in emergent software organizations: Pitfallsand suggestions; Springer Science + Business media Inc. 2006.*

[4] *Burr,A. and Owen ,M.1996. Statistical Methods for Software quality . Thomson publishing Company. ISBN 1-85032-171-X.*

[5] *Carleton, A.D. and Florac, A.W. 1999. Statistically controlling the Software process. The 99 SEI Software Engineering Symposimn, Software Engineering Institute, Carnegie Mellon University.*

[6] *Mutsumi Komuro; Experiences of Applying SPC Techniques to software development processes; 2006 ACM 1-59593-085-x/06/0005.*

[7] *Goel. A.L and Okumoto. K., (1979). "A Time-dependent error-detection rate model for software and other performance measures", IEEE Trans. Reliability, vol R-28, Aug, pp 206 - 211.*

[8] *R.R.L.Kantam and R.Subbarao, 2009. "Pareto Distribution: A Software Reliability Growth Model". International Journal of Performability Engineering, Volume 5, Number 3, April 2009, Paper 9, PP: 275- 281.*

[9] *Balakrishnan.N, Clifford Cohen; Order Statistics and Inference; Academic Press Inc; 1991.*

[10] *R.satyaprasad, Half Logistic Software Reliability Growth Model,Ph.D. Thesis,2007*

[11] *M.Xie, T.N. Goh, P. Rajan; Some effective control chart procedures for reliability monitoring; Elsevier science Ltd, Reliability Engineering and system safety 77(2002) 143-150*

[12] *J.D.Musa and K.Okumoto,"A Logorithmic Poisson Execution time modelfor software reliability measure-ment", proceeding seventh international conference on software engineering, Orlando, pp.230-238, 1984.*