International Journal of Advance Research in Science and Engineering
Vol. No.6, Issue No. 08, August 2017
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

ASSESSING BURR TYPE III SOFTWARE QUALITY USING SPC

Ch.Smitha Chowdary[1], Dr.R.Satya Prasad[2], Dr.R.Kiran Kumar[3]

[1]*Research Scholar, Dept. of Computer Science, Krishna University, Machilipatnam, Andhra Pradesh.*
[2]*Associate Professor, Dept. of C SE, Acharya Nagarjuna University, Guntur, Andhra Pradesh*
[3]*Assistant Professor, Dept. of Comp. Science, Krishna University, Machilipatnam, Andhra Pradesh*

ABSTRACT

Software reliability is assessed quantitatively by Software Reliability Growth Model (SRGM) for tracking and measuring the growth of reliability. Software Reliability is the probability of failure-free operation during precise period in precise environment. To improve reliability and quality of the selected process the execution of software process must be controlled and the accepted choice for monitoring software process is Statistical Process Control (SPC). This helps the professionals to identify anomalies while monitoring the process and take the necessary action. In this paper we proposed a control mechanism based on the cumulative observations of the time domain data using the mean value function of Burr type III distribution, which is based on Non-Homogenous Poisson Process (NHPP). To estimate the unknown parameters of the model, maximum likelihood estimation method is used. The failure data is analyzed with the proposed mechanism and the results are exhibited through control charts.

***Keywords*: Burr type III, Control Charts, NHPP, ML estimation, Software Reliability, SPC, Time domain data**

1. INTRODUCTION

Software Reliability is an important quality characteristic of a software which can evaluate and predict the operational quality of software system during its development. Software Reliability is the probability of failure free operation of software in a specified environment for a specified period of time [1,2]. Over few decades, statistical models of different types have been discussed for assessment of the software reliability. Software reliability is assessed quantitatively by using Software Reliability Growth Model (SRGM) for tracking and measuring the growth of reliability and is used to compute the reliability growth of products during software development phase. These models can be of two types i.e. static and dynamic. The static model uses software metrics in order to estimate the number of defects in the software and the dynamic model uses the past failure discovery rate to estimate the number of failures during software execution over time.

In software engineering one always wants to produce high quality software at low cost. As no one is perfect there is a possibility of errors in the software developed by humans. To improve software reliability these errors need to be identified while the software process is in development and a widely accepted choice for this is the Statistical Process Control.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

In this paper we probe applicability of SPC to Burr type III software reliability growth model to analyze the reliability of a software system using Time domain data. The layout of the paper is as follows: Section 2 describes the formulation and interpretation of the Burr type III model for the underlying NHPP. Section 3 discusses Maximum Likelihood (ML) estimation of Burr type III model based on time domain data. Section 4 describes SPC and its applicability. Section 5 result is presented by monitoring the system using the control charts. Section 6 presents the conclusion.

II. BACKGROUND AND FORMULATION OF NHPP

Here we present the theory that underlies NHPP models, the SRGMs under consideration and maximum likelihood estimation for ungrouped data. Let 't' be a continuous random variable in pdf: $f(t;\theta_1, \theta_2,\ldots. \theta_k)$ where, $\theta_1, \theta_2,\ldots. \theta_k$ are k unknown constant parameters that need to be estimated, and cdf: $F(t)$ where the mathematical relationship between the pdf and cdf is given by: $f(t) = F'(t)$. If 'a' denotes the expected number of faults that can be detected given infinite testing time then, the mean value function of the NHPP models can be written as: $m(t) = aF(t)$, where F(t) is a cumulative distribution function then the failure intensity function $\lambda(t)$ in case of NHPP models is given as: $\lambda(t) = aF'(t)$ [3].

2.1 NHPP for Model construction

The Non-Homogenous Poisson Process (NHPP) based software reliability growth models (SRGMs) are proved to be quite successful in practical software reliability engineering [2]. The main issue in the NHPP model is to determine an appropriate mean value function to order to denote the expected number of failures experienced up to a certain point in time. Model parameters can be estimated by using Maximum Likelihood Estimate (MLE). Many NHPP SRGMs have been built upon various assumptions. Some of the SRGMs assume that each time a failure occurs, the fault that caused it can be immediately removed and no new faults are introduced. This is usually called perfect debugging.

2.2 Burr Type III the considered model

Burr [4] introduced twelve different forms of cumulative distribution functions for modeling data. The probability density function of a three-parameter Burr type III distribution has the form: $f(t,b,c) = \frac{bct^{bc-1}}{[1+t^c]^{b+1}}$ where b,c are shape parameters. The corresponding cumulative distribution function is: $F(t) = [1+t^{-c}]^{-b}$. The mean value function $m(t) = a[1+t^{-c}]^{-b}$. The failure intensity function is given as: $\lambda(t) = \left[\frac{abc}{t^{c+1}(1+t^{-c})^{b+1}}\right]$

III. MAXIMUM LIKELIHOOD ESTIMATION FOR PARAMETER ESTIMATION

The preferred method for obtaining parameter estimates is to use the maximum likelihood equations. These equations are taken from the model equations and the assumptions which underlay them. The parameters are taken to be those values which maximize the likelihood functions. These values are identified by taking the partial derivate of the likelihood function with respect to the model parameters, the maximum likelihood equations, and setting them to zero. Later iterative routines are used to solve these equations.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

Log Likelihood function for ungrouped data [3] is given as,

$$: LLF = \sum_{i=1}^{n} \log[\lambda(t_i)] - m(t_n) \qquad (1)$$

The maximum likelihood estimators of θ_1, θ_2,….. θ_n obtained by maximizing L or $\wedge$, where $\wedge$ is in L . By maximizing $\wedge$, which is much easier to work with than L, the maximum likelihood estimators (MLE) of θ_1, θ_2,….. θ_n are the simultaneous solutions of n equations such

as: $\frac{\partial(\wedge)}{\partial\theta} = 0$, i=1,2,…,n.

3.1 Parameter Estimation using MLE

Cumulative time between failures data for software reliability monitoring is used. Using the estimators of 'a', 'b' and 'c' we computed m (*t*) [5].

The Log Likelihood function is given as:

$$: LogL = \sum_{i=1}^{n} \log\left[\frac{abc}{t_i^{c+1}(1+t_i^{-c})^{b+1}}\right] - \frac{a}{\left[1+t_n^{-c}\right]^b} \qquad (2)$$

Taking the Partial derivative with respect to 'a' and equating to '0'.

$$: a = n\left(1+t_n^{-c}\right)^b \qquad (3)$$

Taking the Partial derivative of log L with respect to 'b' and equating to'0'.

$$: b = \frac{n}{\sum_{i=1}^{n} \log\left(1+t_i^{-c}\right) - n\log\left(1+t_n^{-c}\right)} \qquad (4)$$

The parameter 'c' is estimated by iterative Newton-Raphson Method using $c_{i+1} = c_i - \frac{g(c_i)}{g'(c_i)}$ where g(c) and g'(c) is expressed as follows.

$$: g(c) = \frac{-n\log(t_n)}{1+t_n^c} + \frac{n}{c} + \sum_{i=1}^{n} \log t_i \left[-1 + \frac{2}{1+t_i^c}\right] = 0 \qquad (5)$$

$$: g'(c) = \frac{n(\log t_n)^2 t_n^c}{\left(t_n^c+1\right)^2} - \frac{n}{c} - \sum_{i=1}^{n} \frac{2t_i^c(\log t_i)^2}{\left(t_i^c+1\right)^2} \qquad (6)$$

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

IV. APPLICABILITY OF SPC TO SOFTWARE RELIABILITY

Software reliability growth models (SRGM's) are useful to assess the reliability for quality management and testing progress control of software development. To improve reliability and quality the execution of software process must be controlled and the choice for monitoring software process is Statistical Process Control.

The parameters estimated can be used to monitor the process through SPC concepts and methods over time, in order to verify that the process remains in the state of statistical control. SPC may help in finding assignable causes, long term improvements in the software process. Software quality and reliability can be achieved by eliminating the causes or improving the software process or its operating procedures [6].

4.1 Control Charts of SPC

The most popular technique of SPC for maintaining process control is control charting. The control chart is one of the seven tools for quality control. SPC is used to secure, that the quality of the final product will conform to predefined standards. In any process, regardless of how carefully it is maintained, a certain amount of natural variability will always exist. A process is said to be statistically "in-control" when it operates with only chance causes of variation. On the other hand, when assignable causes are present, then we say that the process is statistically "out-of-control". Control charts are capable to create an alarm when a shift in the level of one or more parameters of a distribution occurs. Normally, such a situation will be reflected in the control chart by points plotted outside the control limits or by the presence of specific patterns. The most common non-random patterns are cycles, trends, mixtures and stratification [7]. For a process to be in control the control chart should not have any trend or nonrandom pattern. The selection of proper SPC charts is essential to effective statistical process control implementation and use. The SPC chart selection is based on data, situation and need [8]. Chan et al.,[9] proposed a procedure based on the monitoring of cumulative quantity. This approach has been shown to have a number of advantages: it does not involve the choice of a sample size; it raises fewer false alarms; it can be used in any environment; and it can detect further process improvement. Xie et al.,[10] proposed t-chart for reliability monitoring where the control limits are defined in such a manner that the process is considered to be out of control when one failure is less than LCL or greater than UCL. The traditional false alarm probability is to set to be 0.27% although any other false alarm probability can be used. The actual acceptable false alarm probability should in fact depend on the actual product or process [11].

: $T_u = a(1+t^{-c})^{-b} = 0.99865$

: $T_c = a(1+t^{-c})^{-b} = 0.05$

: $T_l = a(1+t^{-c})^{-b} = 0.00135$

These limits when converted to $m(t_U)$, $m(t_C)$ and $m(t_L)$ form will be used to find whether the software process is in control or not by placing the points in Mean value chart. A point below the control limit $m(t_L)$ indicates an alarming signal. A point above the control limit $m(t_U)$ indicates better quality. If the points are falling within the control limits, it indicates the software process is in stable condition [12].

V. DATA ANALYSIS FOR NTDS DATA SET

In this section, we present the analysis of software failure data set, The set of software errors analyzed here is borrowed from software development project as published in Pham (2005) [3]. The data set consists of 26 failures in 250 days. During the production phase 26 software errors are found and during the test phase five

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

additional errors are found. During the user phase one error is observed and two more errors are noticed in a subsequent test phase indicating that a network of the module has taken place after the user error is found. In this paper, a numerical conversion of data (Failure Time (hours)*0.01) is done in order to facilitate the parameter estimation [13] [14] [15].The data named as NTDS data are summarized in the below table.

Solving equations in Section 3.1 by Newton-Raphson Method (N-R) method for the NTDS software failure data, the iterative solutions for MLEs of a, b and c are as below.

Table-1: NTDS Data Set

Failure Number n	Time between Failures S_k days	Cumulative Time $X_n = S_k$ days	Failure Time(hours)*0.01
Production (Checkout) Phase			
1	9	9	0.09
2	12	21	0.21
3	11	32	0.32
4	4	36	0.36
5	7	43	0.43
6	2	45	0.45
7	5	50	0.5
8	8	58	0.58
9	5	63	0.63
10	7	70	0.7
11	1	71	0.71
12	6	77	0.77
13	1	78	0.78
14	9	87	0.87
15	4	91	0.91
16	1	92	0.92
17	3	95	0.95
18	3	98	0.98
19	6	104	1.04
20	1	105	1.05
21	11	116	1.16
22	33	149	1.49
23	7	156	1.56
24	91	247	2.47
25	2	249	2.49
26	1	250	2.5
Test Phase			
27	87	337	3.37
28	47	384	3.84
29	12	396	3.96

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

30	9	405	4.05
31	135	540	5.4
User Phase			
32	258	798	7.98
Test Phase			
33	16	814	8.14
34	35	849	8.49

Solving equations in Section 3 by Newton-Raphson Method (N-R) method for the NTDS software failure data, the iterative solutions for MLEs of a, b and c are as below.

$\hat{a} = 34.465706$

$\hat{b} = 1.763647$

$\hat{c} = 1.810222$

Using 'a' and 'b' and 'c' values we can compute m(t). Now the control limits are calculated by the following equations taking the standard values 0.00135, 0.99865 and 0.5.

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

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

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

Table-2: NTDS Mean Successive Difference of BURR TYPE III

Error Number n	Cumulative Failures $x_n = \sum S_k$ days	Failure Time(hours)*0.01	m(t)	Sd
1	9	0.09	0.015452	0.198041
2	21	0.21	0.213493	0.520811
3	32	0.32	0.734304	0.286462
4	36	0.36	1.020766	0.626513
5	43	0.43	1.647279	0.206999
6	45	0.45	1.854278	0.56768
7	50	0.5	2.421958	1.039715
8	58	0.58	3.461673	0.717151
9	63	0.63	4.178824	1.068871
10	70	0.7	5.247695	0.157531
11	71	0.71	5.405226	0.963417
12	77	0.77	6.368643	0.162969

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

13	78	0.78	6.531613	1.482633
14	87	0.87	8.014246	0.661508
15	91	0.91	8.675753	0.165053
16	92	0.92	8.840807	0.493704
17	95	0.95	9.334511	0.490717
18	98	0.98	9.825228	0.968817
19	104	1.04	10.79404	0.159489
20	105	1.05	10.95353	1.707815
21	116	1.16	12.66135	4.481716
22	149	1.49	17.14306	0.817749
23	156	1.56	17.96081	7.227388
24	247	2.47	25.1882	0.105217
25	249	2.49	25.29342	0.051976
26	250	2.5	25.34539	-----

These limits are converted to the form m(tU), m(tC) and m(tL)

They are used to find whether the software process is in control or not by placing the points in Failure Control chart shown in figure 1.

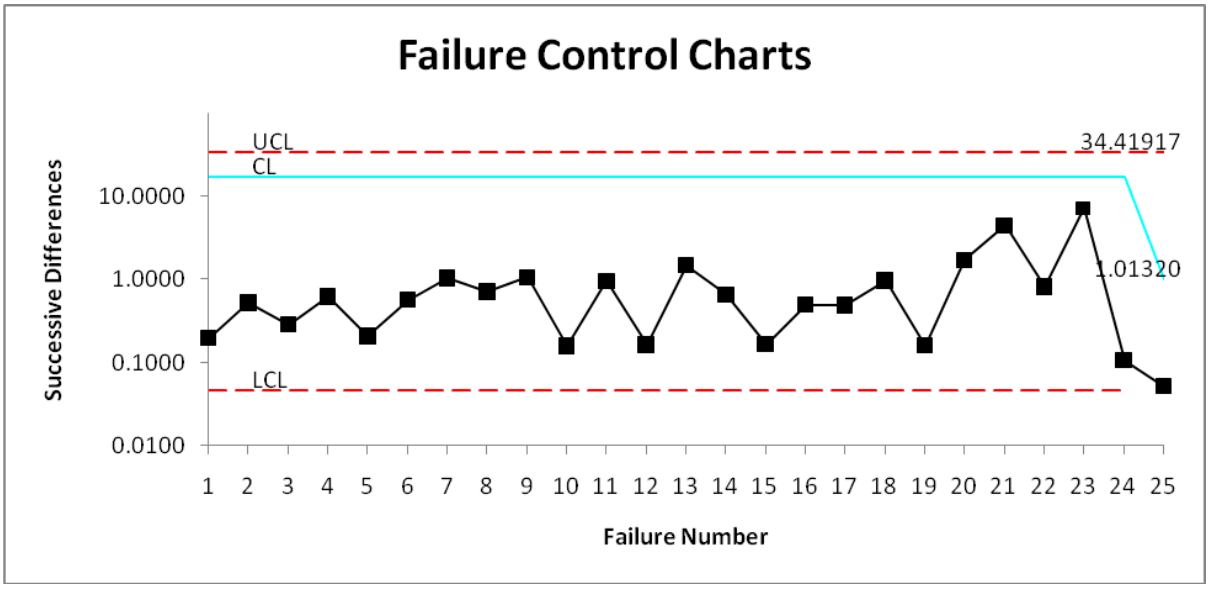

Fig. 1. Failure Control Charts

A point below the control limit m(t_L) indicates an alarming signal. A point above the control limit m(t_U) indicates better quality. If the points are within the control limits, it indicates the software process is in stable condition. The mean value control chart shows all the successive differences and their values are as follows

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

$$m(t_u) = 34.41917$$
$$m(t_l) = 0.46529$$
$$m(t_c) = 17.23285$$

Placing the time between failures cumulative data m(t) successive differences shown in Table 2 on y axis and failure number on x axis and the values of control limits on Failure Control chart we obtain Fig.1. The Failure control chart shows that the 25th failure data has fallen below $m(t_L)$ which indicates the failure process is identified. It is significantly early detection of failures using Failure Control Chart. The software quality is determined by detecting failures at an early stage.

VI. CONCLUSION

Software reliability is an important quality measure that quantifies the operational profile of computer systems. In this paper we proposed Burr type III software reliability growth model. This model is primarily useful in estimating and monitoring software reliability, viewed as a measure of software quality. Equations to obtain the maximum likelihood estimates of the parameters based on time domain data are developed. The analysis of NTDS data shows out of control signals i.e, below the LCL .We conclude that our method of estimation and control charts are giving +ve recommendations for their use in finding out preferable software. By observing the Failure control chart we have identified that the failure situation is detected at 25th point of Table-2. Hence our proposed Failure Chart detects out of control situation. This is a simple method for model validation and is very convenient for practitioners of software reliability. The early detection of software failure will improve the software reliability.

REFERENCES

[1] J. D. MUSA. Software Reliability Engineering.Wiley.1998.

[2] J. D. MUSA, A. IANNINO, AND K. OKUMOTO. Software Reliability Measurement Prediction Application. McGraw-Hill, 1987. ISBN 0-07-044093-X.

[3] Pham. H., 2006. “System software reliability”, Springer.

[4] Burr (1942), "Cumulative Frequency Functions", Annals of Mathematical Statistics, 13, pp. 215-232.

[5] Ch.Smitha Chowdary, Dr.R.Satya Prasad, K.Sobhana (2015),” Burr Type III Software Reliability Growth Model”, IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 1, Ver. I (Jan – Feb. 2015),PP 49-54.

[6] Kimura, M., Yamada, S., Osaki, S., (1995). “Statistical Software reliability prediction and itsapplicability based on mean time betweenfailures”. Mathematical and Computer Modelling Volume 22, Issues 10-12, Pages 149-155.

[7] Koutras, M.V., Bersimis, S., Maravelakis,P.E., 2007. “Statistical process control using shewart control charts with supplementary Runs rules” Springer Science + Business media 9:207-224.

[8] MacGregor, J.F., Kourti, T., 1995. “Statistical process control of multivariate processes”. Control Engineering Practice Volume 3, Issue 3, March 1995, Pages 403-414.

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

[9] Chan, L.Y, Xie, M., and Goh. T.N., (2000), "Cumulative quality control charts for monitoring production processes. Int J Prod Res; 38(2):397-408.

[10] Xie. M, T.N Goh and P.Ranjan. (2002). "Some effective control chart procedures for reliability monitoring", Reliability Engineering and System Safety. 77, 143-150.

[11] Swapna S. Gokhale and Kishore S.Trivedi, 1998. "Log-Logistic Software Reliability Growth Model". The 3rd IEEE International Symposium on High-Assurance Systems.

[12] MacGregor, J.F., Kourti, T., 1995. "Statistical process control of multivariate processes". Control Engineering Practice Volume 3, Issue 3, March 1995, Pages 403-414.

[13] N. R. Barraza., "Parameter Estimation for the Compound Poisson Software Reliability Model", International Journal of Software Engineering and Its Applications, http://www.sersc.org/journals/IJSEIA/vol7_no1_2013/11.pdf, vol. 7, no. 1, (2013) January, pp. 137-148.

[14] I. Inayat, M. Asim Noor and Z. Inayat, "Parameter Successful Product-based Agile Software Development without Onsite Customer: An Industrial Case Study", International Journal of Software Engineering and Its Applications,http://www.sersc.org/journals/IJSEIA/vol6_no2_2012/1.pdf, vol. 6, no. 2, (2012) April, pp. 1-14.

[15] Hassan Najadat and Izzat Alsmadi., "Enhance Rule Based Detection for Software Fault Prone Modules", International Journal of Software Engineering and Its Applications, Vol. 6, No.1, pp. 75-86, January (2012).