Software Process Control on Ungrouped Data: Log-Power Model

Satya Prasad Ravi *profrsp@gmail.com*
Associate Professor,
Dept. of Computer Science & Engg.,
Acharya Nagarjuna University
Andhrapradesh, India

B. Indira Reddy *Indira2259@yahoo.co.in*
Associate professor,
St. Pauls college of Management & IT
Hyderabad, India.

Krishna Mohan Gonuguntla *km_mm_2000@yahoo.com*
Reader, Dept. of Computer Science,
P.B.Siddhartha College
Andhrapradesh, India.

Abstract

Statistical Process Control (SPC) is the best choice to monitor software reliability process. It assists the software development team to identify and actions to be taken during software failure process and hence, assures better software reliability. In this paper we propose a control mechanism based on the cumulative observations of failures which is ungrouped data using an infinite failure mean value function of Log-Power model, which is Non-Homogenous Poisson Process (NHPP) based. The Maximum Likelihood Estimation (MLE) approach is used to estimate the unknown parameters of the model.

Keywords: MLE, SPC, Log-Power, Ungrouped Data.

1. INTRODUCTION

Many software reliability models have been proposed in last 40 years to compute the reliability growth of products during software development phase. These models can be of two types i.e. static and dynamic. A static model uses software metrics to estimate the number of defects in the software. A dynamic model uses the past failure discovery rate during software execution over time to estimate the number of failures. Various software reliability growth models (SRGMs) exist to estimate the expected number of total defects (or failures) or the expected number of remaining defects (or failures).

The goal of software engineering is to produce high quality software at low cost. As, human beings are involved in the development of software, there is a possibility of errors in the software. To identify and eliminate human errors in software development process and also to improve software reliability, the Statistical Process Control concepts and methods are the best choice. SPC concepts and methods are used to monitor the performance of a software process over time in order to verify that the process remains in the state of statistical control. It helps 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 [1].

The most popular technique for maintaining process control is control charting. The control chart is one of the seven tools for quality control. Software process control 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 should be capable to create an alarm when a shift in the level of one or more parameters of the underlying distribution occurs or a non-random behavior comes into. 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 [2]. 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 [3].

Chan et al.,[4] proposed a procedure based on the monitoring of cumulative quantity. This approach has 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.,[5] 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. Assuming an acceptable false alarm □=0.0027 the control limits were defined. In section 5 of present paper, a method is presented to estimate the parameters and defining the limits. The process control is decided by taking the successive differences of mean values.

2. BACKGROUND THEORY

This section presents the theory that underlies NHPP models, the SRGMs under consideration and maximum likelihood estimation for ungrouped data. If 't' is a continuous random variable with pdf: $f(t;\theta_1,\theta_2,\ldots,\theta_k)$. Where, $\theta_1,\theta_2,\ldots,\theta_k$ are k unknown constant parameters which need to be estimated, and cdf: $F(t)$. Where, The mathematical relationship between the pdf and cdf is given by: $f(t)=F'(t)$. Let 'a' denote the expected number of faults that would 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. The failure intensity function $\lambda(t)$ in case of NHPP models is given by: $\lambda(t)=aF'(t)$ [6].

2.1. NHPP model

The Non-Homogenous Poisson Process (NHPP) based software reliability growth models (SRGMs) are proved to be quite successful in practical software reliability engineering [7]. The main issue in the NHPP model is to determine an appropriate mean value function to denote the expected number of failures experienced up to a certain time point. Model parameters can be estimated by using Maximum Likelihood Estimate (MLE). Various NHPP SRGMs have been built upon various assumptions. Many 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. Which is usually called perfect debugging. Imperfect debugging models have proposed a relaxation of the above assumption [8][9].

2.2. Model under consideration: Log-Power model

Software reliability growth models (SRGM's) are useful to assess the reliability for quality management and testing-progress control of software development. They have been grouped into two classes of models concave and S-shaped. The most important thing about both models is that they have the same asymptotic behavior, i.e., the defect detection rate decreases as the number of defects detected (and repaired) increases, and the total number of defects detected asymptotically approaches a finite value. The Log Power NHPP model has several interesting properties, such as simple graphical interpretations and simple forms of the maximum likelihood estimates for the parameters. This model is characterized by the following mean value function:

$m(t) = a\log^b(1+t)$. Where, $a, b > 0, t \geq 0$. The failure intensity function of the model, which is defined as the derivative of the mean value function $m(t)$, is given by $\lambda(t) = \frac{ab\log^{b-1}(1+t)}{1+t}$.

3. MAXIMUM LIKELIHOOD ESTIMATION

In much of the literature the preferred method of obtaining parameter estimates is to use the maximum likelihood equations. Likelihood equations are derived from the model equations and the assumptions which underlie these equations. The parameters are then taken to be those values which maximize these likelihood functions. These values are found by taking the partial derivate of the likelihood function with respect to the model parameters, the maximum likelihood equations, and setting them to zero. Iterative routines are then used to solve these equations. Unfortunately, the SRGM literature is sadly lacking in advice on which iterative routines to use, and with what starting values. This is unfortunate because the accuracy of parameter estimates and thus the accuracy of the models themselves greatly depend on the ability of the iterative search methods used to overcome local minima and find good values for the parameters.

If we conduct an experiment and obtain N independent observations, $t_1, t_2, \ldots, t_N$. The likelihood function may be given by the following product:

$$L(t_1, t_2, \ldots, t_N \mid \theta_1, \theta_2, \ldots, \theta_k) = \prod_{i=1}^{N} f(t_i; \theta_1, \theta_2, \ldots, \theta_k)$$

Likelihood function by using □(t) is: $L = e^{-m(t)} \prod_{i=1}^{n} \lambda(t_i)$

Log Likelihood function for ungrouped data [10] is given as,

$$\log L = \sum_{i=1}^{n} \log\left[\lambda(t_i)\right] - m(t_n)$$

The maximum likelihood estimators (MLE) of $\theta_1, \theta_2, \ldots, \theta_k$ are obtained by maximizing L or Λ, where Λ is ln L. By maximizing Λ, which is much easier to work with than L, the maximum likelihood estimators (MLE) of $\theta_1, \theta_2, \ldots, \theta_k$ are the simultaneous solutions of k equations such as: $\frac{\partial(\Lambda)}{\partial \theta_j} = 0$, j=1,2,...,k.

3.1. Illustration: Parameter Estimation

We used cumulative time between failures data for software reliability monitoring. The use of cumulative quality is a different and new approach, which is of particular advantage in reliability. Using the estimators of 'a' and 'b' we can compute $m(t)$.

The likelihood function of Log-power model is given as,

$$L = e^{-a\log^b(1+t)} \prod_{i=1}^{N} \frac{a\,b\log^{b-1}(1+t_i)}{1+t_i} \qquad (3.1.1)$$

Taking the natural logarithm on both sides, The Log Likelihood function is given as:

$$\log L = \sum_{i=1}^{n} \log\left(\frac{ab\log^{b-1}(1+t_i)}{1+t_i}\right) - a\log^b(1+t_n). \qquad (3.1.2)$$

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

$$a = \frac{n}{\log^{b}(1+t_n)} \tag{3.1.3}$$

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

$$b = \frac{n}{n\log\left(\log\left(1+t_n\right)\right) - \sum_{i=1}^{n}\log\left[\log\left(1+t_i\right)\right]} \tag{3.1.4}$$

4. TIME DOMAIN FAILURE DATA SETS

The techniques examined here deal with data about the time at which failures occurred; or alternatively, data about the time between failure occurrences. These two forms can be considered equivalent. Although most software reliability growth models use data of this form, and such models have been in use for several decades, finding suitable data to verify models and improvement techniques is difficult. Early work generally focused on data based on calendar or wall clock time. Musa asserts that CPU execution time is a better measure than wall clock time, during which the actually time spent running a program can vary greatly based on CPU load, man hours, and other factors [11].

DS #1: On-Line Data Entry IBM Software Package

The data reported by Ohba [8] are recorded from testing an on-line data entry software package developed at IBM. The following table shows the number of errors and the inter failure time.

Failure Number	Time Between Failure(h)	Failure Number	Time Between Failure(h)	Failure Number	Time Between Failure(h)
1	10	**6**	12	**11**	19
2	9	**7**	18	**12**	30
3	13	**8**	15	**13**	32
4	11	**9**	22	**14**	25
5	15	**10**	25	**15**	40

TABLE 4.1: DS #1.

DS #2: AT&T System T Project

The AT&T's System T is a network-management system developed by AT&T that receives data from telemetry events, such as alarms, facility-performance information, and diagnostic messages, and forwards them to operators for further action. The system has been tested and failure data has been collected [12]. The following Table shows the failures and the inter-failure times (in CPU units).

Failure Number	Inter Failure Time	Failure Number	Inter Failure Time	Failure Number	Inter Failure Time
1	5.5	**9**	11.39	**17**	125.67
2	1.83	**10**	19.88	**18**	82.69
3	2.75	**11**	7.81	**19**	0.46
4	70.89	**12**	14.6	**20**	31.61
5	3.94	**13**	11.41	**21**	129.31
6	14.98	**14**	18.94	**22**	47.6
7	3.47	**15**	65.3		
8	9.96	**16**	0.04		

TABLE 4.2: DS #2.

5. RESULTS

The performance of the model under consideration is exemplified by applying on the data sets given in tables 4.1 and 4.2.

5.1 Calculation of Control Limits

The control limits for the chart are defined in such a manner that the process is considered to be out of control when the time to observe exactly one failure is less than LCL or greater than UCL. Our aim is to monitor the failure process and detect any change of the intensity parameter. When the process is normal, there is a chance for this to happen and it is commonly known as false alarm. 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 [13].

$$T_u = \log^b(1+t) = 0.99865$$

$$T_c = \log^b(1+t) = 0.5$$

$$T_l = \log^b(1+t) = 0.00135$$

Data set	a	b	$m(t_U)$	$m(t_C)$	$m(t_L)$
DS#1	0.022149	3.747340	0.022136	0.016392	0.001256
DS#2	0.073480	3.040257	0.073437	0.054379	0.004168

TABLE 5.1.1: Estimated Parameters and the Control Limits.

5.2 Distribution of Failures

The $m(t)$ values were calculated at each cumulative value of 't'. The successive differences of these values are calculated to plot as a failure control chart along with the calculated control limits which vary with the considered data. The following tables 5.2.1, 5.2.2 and graphs given in figures 5.2.1, 5.2.2 shows the performance of Log-Power model in software process control.

FN	m(t)	SD	FN	m(t)	SD	FN	m(t)	SD
1	0.587104	0.764908	**6**	5.069872	1.082693	**11**	10.191812	1.262075
2	1.352012	1.060919	**7**	6.152565	0.838939	**12**	11.453888	1.249374
3	2.412931	0.832128	**8**	6.991504	1.145406	**13**	12.703262	0.917759
4	3.245060	1.047639	**9**	8.136909	1.201109	**14**	13.621021	1.378979
5	4.292699	0.777173	**10**	9.338018	0.853794	**15**	15.000000	

TABLE 5.2.1: Successive Differences of Mean Values IBM.

FN	m(t)	SD	FN	m(t)	SD	FN	m(t)	SD
1	0.494209	0.227281	**9**	8.843671	0.842176	**17**	16.753902	1.699392
2	0.721490	0.337608	**10**	9.685847	0.312257	**18**	18.453294	0.008876
3	1.059098	5.614150	**11**	9.998104	0.559258	**19**	18.462171	0.596590
4	6.673248	0.218518	**12**	10.557362	0.417022	**20**	19.058761	2.206733
5	6.891766	0.784428	**13**	10.974384	0.658035	**21**	21.265494	0.734506
6	7.676194	0.172320	**14**	11.632419	2.008932	**22**	22.000000	
7	7.848515	0.477406	**15**	13.641351	0.001129			
8	8.325921	0.517750	**16**	13.642479	3.111423			

TABLE 5.2.2: Successive Differences of Mean Values ATT.

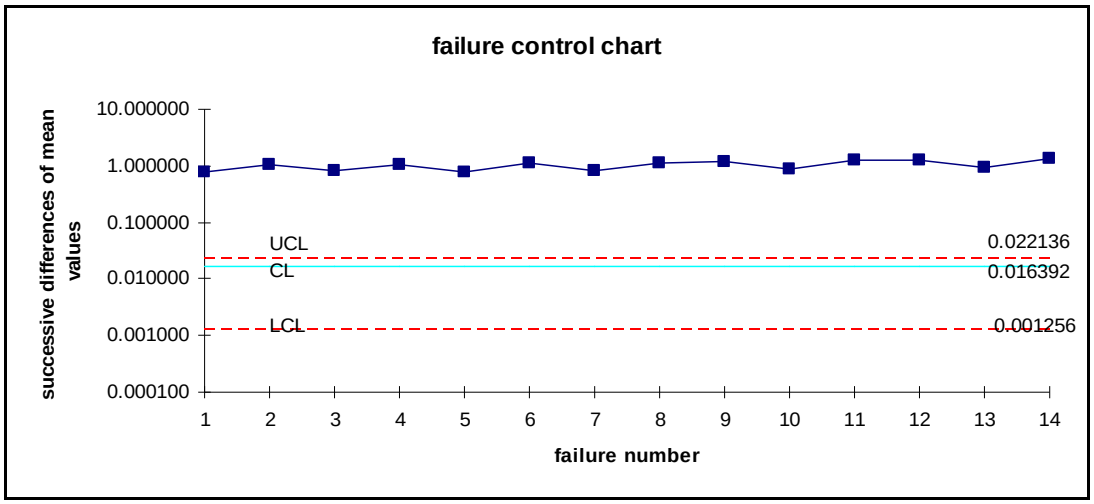

FIGURE 5.2.1: Failure Control Chart.

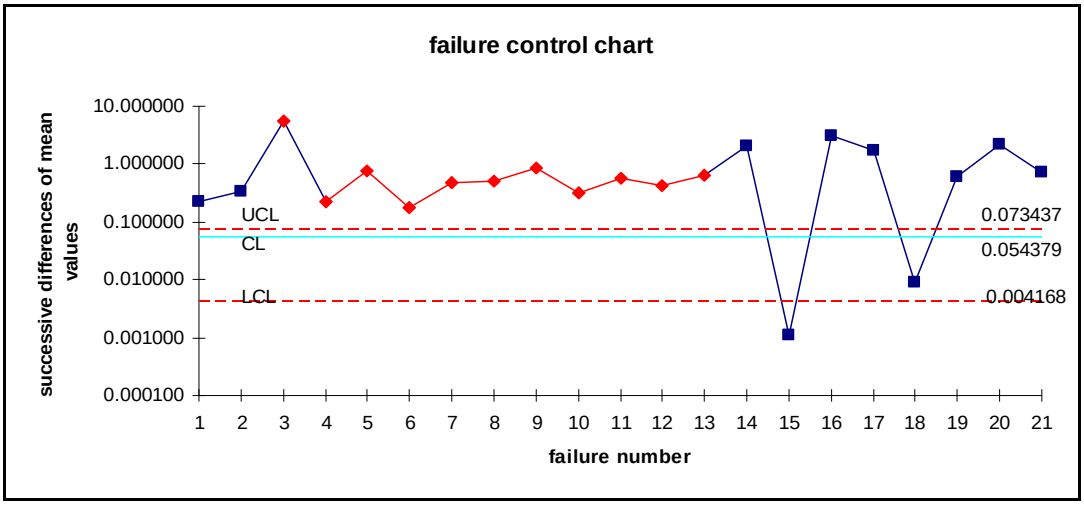

FIGURE 5.2.2: Failure Control Chart.

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. By placing the failure cumulative data shown in tables 5.2.1 and 5.2.2 on y axis and failure number on x axis and the values of control limits are placed on Control chart, we obtained Figures 5.2.1 and 5.2.2.The software quality is determined by detecting failures at an early stage.

6. CONCLUSION

The given Time between failures data are plotted through the estimated mean value function against the failure serial order. The graphs have shown out of control signals i.e below the LCL. Hence we conclude that our method of estimation and the control chart are giving a positive recommendation for their use in finding out preferable control process or desirable out of control signal. By observing the Control chart it is identified that, for DS#1 the failure process out of UCL. For DS#2 the failure situation is detected at 15th point below LCL. Hence our proposed Control Chart detects out of control situation. The performance of this model is compared with Rayleigh and exponential Rayleigh models [14][15]. Many of the successive differences have gone out of

upper control limits for the present model. But, in the case of exponential and Rayleigh model the successive differences have gone out of Lower control limits and within the upper and lower control limits.

7. REFERENCES

[1] Kimura, M., Yamada, S., Osaki, S., (1995). ”Statistical Software reliability prediction and its applicability based on mean time between failures”. Mathematical and Computer ModellingVolume 22, Issues 10-12, Pages 149-155.

[2] 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.

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

[4] 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.

[5] 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.

[6] Swapna S. Gokhale and Kishore S.Trivedi, (1998). “Log-Logistic Software Reliability Growth Model”. The 3rd IEEE International Symposium on High-Assurance Systems Engineering. IEEE Computer Society.

[7] Musa, J. D.; Iannino, A.; Okumoto, K. (1987). “Software Reliability - Measurement, Prediction, Application”, New York.

[8] Ohba, M., 1984. “Software reliability analysis model”. IBM J. Res. Develop. 28, 428-443.

[9] Pham. H., 1993. “Software reliability assessment: Imperfect debugging and multiple failure types

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

[11] Musa J.D. (1975). “A Theory of Software reliability and its applications” IEEE Trans. On Software Engineering ,vol SE-1(3)

[12] Ehrlich, W., Prasanna, B., Stampfel, J. and Wu, J. (1993). “Determining the cost of a stop testing decision”, IEEE Software: 33-42.

[13] Gokhale, S.S and Trivedi, K.S., 1998. “Log-Logistic Software Reliability Growth Model”. The 3rd IEEE International Symposium on High-Assurance Systems Engineering. IEEE Computer Society.

[14] R.Satya Prasad, G.Krishna Mohan and Prof. R.R.L. Kantham. “Time Domain based Software Process Control using Weibull Mean Value Function”, International Journal of Computer Applications (IJCA). 18(3):18-21, March 2011.

[15] G.Krishna Mohan, B.Srinivasa Rao and Dr. R.Satya Prasad. ”A Comparative study of Software Reliability models using SPC on ungrouped data”, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE). Volume 2, Issue 2, February 2012.