International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 6, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Control Chart Procedure for Software Reliability: Inflection S-Shaped Model

Dr. R. Satya Prasad, K. Prasada Rao, G.Krishna Mohan

***Abstract* — Over the last four decades, many software reliability growth models have been developed in tracking the growth of reliability as software is being developed. This paper presents the use of SPC in combination with one of those models. Software reliability process can be monitored efficiently using Statistical Process Control (SPC). It assists the software development team to identify and actions to be taken during software failure process and hence, assures better software reliability. The performance of an inflection S-shaped software reliability growth model based on the cumulative observations of failure data is presented using mean value function, which is based on Non-Homogenous Poisson Process (NHPP). The maximum likelihood approach is used to estimate the unknown parameters of the model. Using SPC, we can identify where the failure process has gone out of control.**

***Keywords* — Statistical Process Control, Software reliability, mean value function, Probability limits, Control Charts, Inflection S-shaped.**

I. INTRODUCTION

Reliability is one of the many important attributes of the software. It is the primary concern for both the developers and users. As there is a great demand in reliable software products, its assessment is important to evaluate and predict the reliability and performance of software system. SPC is known to be a powerful tool to improve processes to enhance quality and productivity [1],[7],[2]. 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 [3].

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 can be classified into several categories, according to several distinct criteria. Depending on the number of quality ch aracteristics under investigation, we may define univariate control charts or multivariate control charts. Furthermore, the quality characteristic of interest may be a continuous random variable or alternatively a discrete attribute. 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 [4]. For a process to be in control the control chart should not have any trend or nonrandom pattern.

SPC is a powerful tool to optimize the amount of information needed for use in making management decisions. Statistical techniques provide an understanding of the business baselines, insights for process improvements, communication of value and results of processes, and active and visible involvement. SPC provides real time analysis to establish controllable process baselines; learn, set, and dynamically improve process capabilities; and focus business areas needing improvement. The early detection of software failures will improve the software reliability. 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 [6].

II. BACKGROUND THEORY

This section presents the theory that underlies a distribution and maximum likelihood estimation for complete 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: $f(t)=\frac{d(F(t))}{dt}$. Let 'a' denote the expected number of faults that would be detected given infinite testing time in case of finite failure NHPP models. Then, the mean value function of the finite failure 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 the finite failure NHPP models is given by: $\lambda(t)=aF'(t)$ [11].

A. NHPP model

The Non-Homogenous Poisson Process (NHPP) based software reliability growth models (SRGMs) are proven to be quite successful in practical software reliability engineering [8]. 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).

Let $\{N(t),t\geq 0\}$ denote a counting process representing the cumulative number of faults detected by the time 't'. An SRGM based on an NHPP with the mean value

International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 6, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

function (MVF) $m(t)$ is the mean value function, representing the expected number of software failures by time 't' can be formulated as [5].

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

$\lambda(t)$ is the failure intensity function, which is proportional to the residual fault content. In a more general NHPP SRGM $\lambda(t)$ can be expressed as $\lambda(t)=\frac{dm(t)}{dt}=b(t)[a(t)-m(t)]$. where $a(t)$ is the time-dependent fault content function which includes the initial and introduced faults in the program and $b(t)$ is the time-dependent fault detection rate. In software reliability, the initial number of faults and the fault detection rate are always unknown. The maximum likelihood technique can be used to evaluate the unknown parameters.

B. Inflection S-shaped 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 inflection S-shaped model was proposed by Ohba in 1984. This model assumes that the fault detection rate increases throughout a test period. The model has a parameter, called the inflection rate, that indicates the ratio of detectable faults to the total number of faults in the target software. True, sustained exponential growth cannot exist in the real world. Eventually all exponential, amplifying processes will uncover underlying stabilizing processes that act as limits to growth. The shift from exponential to asymptotic growth is known as sigmoidal, or S-shaped, growth.

Ohba models the dependency of faults by postulating the following assumptions:

- Some of the faults are not detectable before some other faults are removed.
- The detection rate is proportional to the number of detectable faults in the program.
- Failure rate of each detectable fault is constant and identical.
- All faults can be removed.

Assuming [9]: $b(t)=\frac{b}{1+\beta e^{-bt}}$

This model is characterized by the following mean value function:

$$m(t)=\frac{a}{1+\beta e^{-bt}}\left(1-e^{-bt}\right)(2)$$

where 'b' is the failure detection rate, and 'β' is the inflection factor. The failure intensity function is given as:

$$\lambda(t)=\frac{abe^{-bt}(1+\beta)}{\left(1+\beta e^{-bt}\right)^2}.(3)$$

C. Maximum Likelihood Parameter Estimation

The idea behind maximum likelihood parameter estimation is to determine the parameters that maximize the probability (likelihood) of the sample data. The method of maximum likelihood is considered to be more robust (with some exceptions) and yields estimators with good statistical properties. In other words, MLE methods are versatile and apply to most models and to different types of data. Although the methodology for maximum likelihood estimation is simple, the implementation is mathematically intense. Using today's computer power, however, mathematical complexity is not a big obstacle. conduct an experiment and obtain N independent observations, $t_1,t_2,\ldots,t_N$. Then the likelihood function [10] is given by the following product: $L(t_1,t_2,\ldots,t_N\mid\theta_1,\theta_2,\ldots,\theta_k)=L=\prod_{i=1}^{N}f(t_i;\theta_1,\theta_2,\ldots,\theta_k)$

Likely hood function by using λ(t) is: $L=\prod_{i=1}^{n}\lambda(t_i)$

The logarithmic likelihood function is given by:

$$\Lambda=\ln L=\sum_{i=1}^{N}\ln f(t_i;\theta_1,\theta_2,\ldots,\theta_k)$$

Log Likelihood function is: $LogL=\log\left(\prod_{i=1}^{n}\lambda(t_i)\right)$

which can be written as $\sum_{i=1}^{n}\log[\lambda(t_i)]-m(t_n)$

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 that:

$$\frac{\partial(\Lambda)}{\partial\theta_j}=0,\ j=1,2,\ldots,k$$

The parameters 'a' and 'b' are estimated using iterative Newton Raphson Method, which is given as

$$x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$$

III. ILLUSTRATING THE MLE METHOD

A. Parameter estimation

To estimate 'a' and 'b' , for a sample of n units (all tested to failure), first obtain the likelihood function: assuming $\beta=0.05$.

$$L=\prod_{i=1}^{N}\frac{abe^{-bt}(1+\beta)}{\left(1+\beta e^{-bt}\right)^2}\quad(4)$$

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

$$\begin{aligned}LogL&=\log[\prod_{i=1}^{n}\lambda(t_i)]\\&=\log[\prod_{i=1}^{n}\frac{abe^{-bt}(1+\beta)}{\left(1+\beta e^{-bt}\right)^2}\\&=\sum_{i=1}^{n}\log(\frac{abe^{-bt}(1+\beta)}{\left(1+\beta e^{-bt}\right)^2})-\frac{a}{1+\beta e^{-bt}}\left(1-e^{-bt}\right)\end{aligned}\quad(5)$$

International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 6, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Taking the Partial derivative w.r.t 'a' equating to '0'. (i.e $\frac{\partial \log L}{\partial a} = 0$)

$$a = \frac{n\left(1+\beta e^{-bt_n}\right)}{\left(1-e^{-bt_n}\right)} \quad (6)$$

Taking the Partial derivative w.r.t 'b' and equating to '0'.(i.e $g(b) = \frac{\partial \log L}{\partial b} = 0$)

$$g(b) = \frac{n}{b} - \sum_{i=1}^{n} t_i + 2\sum_{i=1}^{n} \frac{\beta t_i e^{-bt_i}}{\left(1+\beta e^{-bt_i}\right)} - \frac{nt_n e^{-bt_n}\left(1+\beta\right)}{\left(1-e^{-bt_n}\right)\left(1+\beta e^{-bt_n}\right)} \quad (7)$$

Taking the partial derivative again w.r.t 'b' and equating to '0'. (i.e $g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0$)

$$g'(b) = \frac{nt_n^2 e^{-bt_n}\left(1+\beta\right)\left(1+\beta e^{-2bt_n}\right)}{\left[\left(1-e^{-bt_n}\right)\left(1+\beta e^{-bt_n}\right)\right]^2} - \frac{n}{b^2} - 2\sum_{i=1}^{n} \frac{\beta t_i^2 e^{-bt_i}}{\left(1+\beta e^{-bt_i}\right)^2} \quad (8)$$

The parameter 'b' is estimated by iterative Newton Raphson Method using $b_{n+1} = b_n - \frac{g(b_n)}{g'(b_n)}$, which is substituted in finding 'a'.

B. Distribution of Time between failures

Based on the time between failure and inter failure data given in Table I and II, we compute the software failures process through Mean Value Control chart. We used cumulative time between failures data for software reliability monitoring using inflection S-shaped distribution.

By taking the standard values 0.00135, 0.99865, and 0.5 and substituted in $m(t)$. The control limits are calculated by solving the following equations.

$$T_U = \frac{1}{1+\beta e^{-bt}}\left(1-e^{-bt}\right) = 0.99865$$

$$T_C = \frac{1}{1+\beta e^{-bt}}\left(1-e^{-bt}\right) = 0.5$$

$$T_L = \frac{1}{1+\beta e^{-bt}}\left(1-e^{-bt}\right) = 0.00135$$

Table:I Time between failures of a software:DS1

FNo	TBF(h)	FNo	TBF(h)
1	30.02	16	15.53
2	1.44	17	25.72
3	22.47	18	2.79
4	1.36	19	1.92
5	3.43	20	4.13
6	13.2	21	70.47
7	5.15	22	17.07
8	3.83	23	3.99
9	21	24	176.06
10	12.97	25	81.07
11	0.47	26	2.27
12	6.23	27	15.63
13	3.39	28	120.78
14	9.11	29	30.81
15	2.18	30	34.19

Table:II Inter failure Times of a software: DS2

FNo	IFT	FNo	IFT
1	10	9	22
2	9	10	25
3	13	11	19
4	11	12	30
5	15	13	32
6	12	14	25
7	18	15	40
8	15		

'' and '' are Maximum Likely hood Estimates (MLEs) of parameters and the values can be computed using iterative method for the given cumulative time between failures data shown in table I and II. Using 'a' and 'b' values we can compute $m(t)$.

These limits are converted to $m(t_U)$, $m(t_C)$ and $m(t_L)$ form. They are used to find weather the software process is in control or not by placing the points in Mean value chart shown in figure 1 and 2. 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. The values of control limits are as follows.

DATA SET	$m(t_U)$	$m(t_C)$	$m(t_L)$
DS1	33.19474	16.61981	0.044873
DS2	23.266601	11.649026	0.031452

Table: III Successive differences of cumulative mean values.

FNo	C_TBF	m(t)	SD
1	30.02	2.922726	0.133851
2	31.46	3.056577	2.016861
3	53.93	5.073438	0.117837
4	55.29	5.191275	0.295098
5	58.72	5.486374	1.108119
6	71.92	6.594493	0.420710
7	77.07	7.015203	0.308731
8	80.90	7.323934	1.631604
9	101.90	8.955539	0.957770
10	114.87	9.913309	0.034014
11	115.34	9.947322	0.446361
12	121.57	10.393683	0.239400
13	124.96	10.633083	0.631389
14	134.07	11.264472	0.148541
15	136.25	11.413014	1.030412
16	151.78	12.443426	1.603638
17	177.50	14.047064	0.166558

International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 6, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

18	180.29	14.213622	0.113804
19	182.21	14.327426	0.242558
20	186.34	14.569984	3.700637
21	256.81	18.270621	0.782407
22	273.88	19.053028	0.177028
23	277.87	19.230056	5.981508
24	453.93	25.211564	1.824108
25	535.00	27.035672	0.044672
26	537.27	27.080344	0.299052
27	552.90	27.379396	1.873595
28	673.68	29.252991	0.373686
29	704.49	29.626677	0.374013
30	738.68	30.000690	

Table: IV Successive differences of cumulative mean values.

FNo	C_TBF	m(t)	SD
1	10	0.784833	0.684779
2	19	1.469612	0.954157
3	32	2.423769	0.776117
4	43	3.199886	1.013935
5	58	4.213821	0.775635
6	70	4.989456	1.106895
7	88	6.096351	0.872897
8	103	6.969247	1.203308
9	125	8.172555	1.263270
10	150	9.435825	0.890981
11	169	10.326806	1.294501
12	199	11.621307	1.241843
13	231	12.863150	0.879410
14	256	13.742560	1.258149
15	296	15.000709	

By placing the time between failures cumulative data shown in table II on y axis and failure number on x axis and the values of control limits are placed on Mean Value chart, we obtained figure 1 and 2. The Mean Value chart shows that the 10th and 25th failure data has fallen below $m(t_L)$ which indicates the failure process is identified. It is significantly early detection of failures using Mean Value Chart. The software quality is determined by detecting failures at an early stage. The remaining failure data are shown in Figure 1 is in stable. No failure data fall outside the $m(t_U)$. It does not indicate any alarm signal.

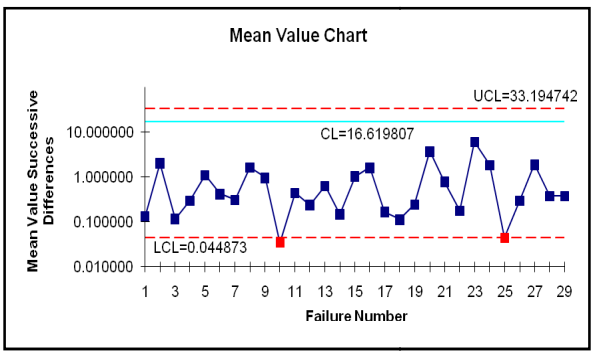

Fig.1. Mean Value Chart

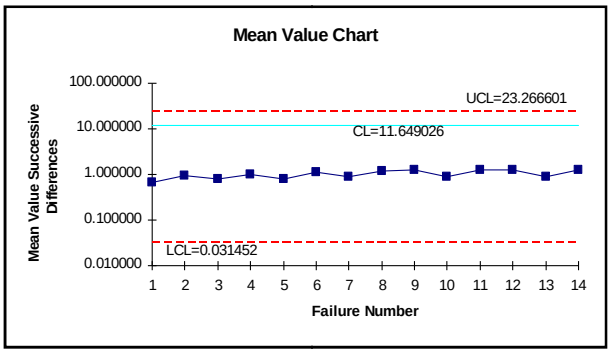

Fig.2. Mean Value Chart

IV. CONCLUSION

The given 30 and 15 inter failure times are plotted through the estimated mean value function against the failure serial order. The parameter estimation is carried out by Newton Raphson Iterative method for inflection s-shaped model. The graph in Fig:1 has shown out of control signals i.e below the LCL. Where as, the graph in Fig:2 has shown that the failure process is within the specified control limits. 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 Mean value Control chart we identified that the failure situation is detected at 10th and 25th point of Table-II for the corresponding $m(t)$, which is below $m(t_L)$. It indicates that the failure process is detected at an early stage compared with Xie et a1, (2002) control chart, which detects the failure at 23rd point for the inter failure data above the UCL. Hence, our proposed Mean Value Chart detects out of control situation at an earlier instant than the time control chart. The early detection of software failure will improve the software Reliability.

REFERENCES

[1] Florac, A.W and Carletan, A.D., "Using Statistical Process Control to Mewasure Software Process"; 9th International Conference on Software Quality, 1999, Software Engineering Institute.
[2] Humphrey, W.S. Managing the Software Process", Addison-Wesley Publishing Company, Inc., 1989, reading, Massachusetts.
[3] Kimura, M., Yamada, S., Osaki, S., "Statistical Software reliability prediction and its applicability based on mean time between failures". Mathematical and Computer Modelling Volume 22, Issues 10-12, 1995, Pages 149-155.
[4] 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.
[5] Lyu, M.R., 1996. "Handbook of Software Reliability Engineering", McGrawHill.
[6] MacGregor, J.F., Kourti, T., 1995. "Statistical process control of multivariate processes". Control Engineering Practice Volume 3, Issue 3, March 1995, Pages 403-414.
[7] Montgomery, D. C. and Woodall, H. L. (1997). "A Discussion on Statistically-Based Process Monitoring and Control", Journal of Quality Technology, 29(2), 121-162.
[8] Musa, J.D., Iannino, A., Okumoto, k., 1987. "Software Reliability: Measurement Prediction Application". McGraw-Hill, New York.
[9] Ohba, M., 1984. "Software reliability analysis model". IBM J. Res. Develop. 28, 428-443.
[10] Pham. H., 2003. "Handbook Of Reliability Engineering", Springer.

[11] Pham. H., 2006. “System software reliability”, Springer.
[12] Xie, M., Goh. T.N., Ranjan.P., “Some effective control chart procedures for reliability monitoring” -Reliability engineering and System Safety 77 143 -150¸ 2002.

AUTHOR’S PROFILE

Dr. R. Satya Prasad

Received Ph.D. degree in Computer Science in the faculty of Engineering in 2007 from Acharya Nagarjuna University, Andhra Pradesh. He received gold medal from Acharya Nagarjuna University for his outstanding performance in a first rank in Masters Degree. He is currently working as Associative Professor and H.O.D, in the Department of Computer Science & Engineering, Acharya Nagarjuna University. His current research is focused on Software Engineering. He published 50 research papers in National & International Journals.

Mr. K.Prasad Rao

Working as a Professor and Director, Dept. of M.C.A, CMR Institute of Technology. He is having 22 years of experience as a Head & Lecturer in Computer Science field. He Published papers in 3 National and 1 International journals. He is pursuing Ph.D. at Acharya Nagarjuna University. His research interests lies in Software Engineering.

Mr. G. Krishna Mohan

Working as a Reader in the Department of Computer Science, P.B.Siddhartha College, Vijayawada. He obtained his M.C.A degree from Acharya Nagarjuna University, M.Tech from JNTU, Kakinada, M.Phil from Madurai Kamaraj University and pursuing Ph.D from Acharya Nagarjuna University. He qualified AP State Level Eligibility Test. His research interests lies in Data Mining and Software Engineering. He published 13 research papers in various National and International journals.