Software Reliability with SPC

Dr. R Satya Prasad[1], K Ramchand H Rao[2], Dr. R.R. L Kantham[3]

[1]Associate Prof, Dept. of Computer Science & Eng, Acharya Nagarjuna University, Guntur, INDIA, profrsp@gmail.com
[2] Dept. of Computer Science A.S.N. Degree College, Tenali, INDIA, ramkolasani@gmail.com
[3] Prof., Dept. of Statistics, Acharya Nagarjuna University, Guntur, INDIA, kantam_rrl@rediffmail.com

Abstract: *Software reliability is the probability of failure-free operation of software in a specified environment during specified time duration. Statistical Process Control can monitor the forecasting of software failure and thereby contribute significantly to the improvement of software reliability. Control charts are widely used for software process control in the software industry. Relatively little research work is, however, available on their use to monitor failure process of software. It is well known that Control charts can be used to analyze both small and large failure frequency. Some control charts can be used for monitoring the number of failures per fixed interval. However they are not effective especially when the failure frequency becomes small. To meet this desideratum, the control scheme adopted for our study is based on the cumulative data between observations of failure. It is proposed that the said control scheme can be easily and fruitfully applied to monitor the software failure process for Half Logistic Distribution based NHPP.*

Keywords: *Statistical Process Control (SPC), Software reliability, Control Charts, Probability limits, Half Logistic Distribution*

1. Introduction

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[1].

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 [2]. 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 [2] (Burr and Owen[3]; Flora and Carleton[4]). 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[5].

2. Model Formulation.

Let $\left[\mathrm{N}(\mathrm{t}), \mathrm{t} \geq 0\right], \mathrm{m}(\mathrm{t}), \lambda(t)$ be the counting process, mean value function and intensity function of a software failure phenomenon.

The mean value function m(t) is finite valued, non decreasing, non negative and bounded with the boundary conditions

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

Here 'a' represents the expected number of software failures eventually detected. If $\lambda(t)$ is the corresponding intensity function. $\lambda(t)$ is a decreasing function of $m(t)$ as a result of repair action following early failures. A relation between $m(t)$ and $\lambda(t)$ is given by

$$\lambda(t) = \frac{b}{2a}\left[a^2 - m^2(t)\right]$$

where 'b' is a positive constant, serving the purpose of constant of proportional fall in $\lambda(t)$. This relation indicates a decreasing trend for $\lambda(t)$ with increase in $m(t)$

From the fact that $\lambda(t)$ is the derivative of $m(t)$ we get the following differential equation

$$\frac{dm(t)}{dt} = \frac{b}{2a}\left[a^2 - m^2(t)\right]$$

whose solution is

$$m(t) = \frac{a\left(1-e^{-bt}\right)}{\left(1+e^{-bt}\right)} \qquad (2.1)$$

An NHPP with its mean value function given in equation (2.1). Its intensity function is

$$\lambda(t) = \frac{2abe^{-bt}}{\left(1+e^{-bt}\right)^2} \qquad (2.2)$$

3. Estimation Based on Inter Failure Times

The mean value function and intensity function of Half Logistic Model [6] are given by

$$m(t) = \frac{a\left(1-e^{-bt}\right)}{\left(1+e^{-bt}\right)} \ , a>0, b>0, t\geq 0 \qquad (3.1)$$

$$\lambda(t) = \frac{2abe^{-bt}}{\left(1+e^{-bt}\right)^2} \qquad (3.2)$$

The constants 'a', 'b' which appear in the mean value function and hence in NHPP, in intensity function (error detection rate) and various other expressions are called parameters of the model. In order to have an assessment of the software reliability 'a',' b' are to be known or they are to be estimated from a software failure data.

Suppose we have 'n' time instants at which the first, second, third..., n[th] failures of a software are experienced. In other words if S_k is the total time to the k[th] failure, s_k is an observation of random variable S_k and 'n' such failures are successively recorded. The joint probability of such failure time realizations $s_1, s_2, s_3, \ldots . s_n$ is

$$L = e^{-m(s_n).} \prod_{k=1}^{n} \lambda(s_k) \qquad (3.3)$$

The function given in equation (3.3) is called the likelihood function of the given failure data. Values of 'a', ' b' that would maximize L are called maximum likelihood estimators (MLEs) and the method is called maximum likelihood (ML) method of estimation. Accordingly 'a', 'b' would be solutions of the equations

$$\frac{\partial \log L}{\partial a} = 0 \ , \frac{\partial \log L}{\partial b} = 0 \ , \ \frac{\partial^2 \log L}{\partial b^2} = 0$$

Substituting the expressions for m(t), λ(t) given by equations (3.1) and (3.2) in equation (3.3), taking logarithms, differentiating with respect to 'a', 'b' and equating to zero, after some joint simplification we get

$$a = n\left[\frac{\left(1+e^{-bs_n}\right)}{\left(1-e^{-bs_n}\right)}\right] \qquad (3.4)$$

$$g(b) = \sum_{k=1}^{n} s_k - \frac{n}{b} - 2\sum_{k=1}^{n-1} \frac{s_k . e^{-bs_k}}{\left(1+e^{-bs_n}\right)} - \frac{2s_n . e^{-bs_k}}{\left(1+e^{-bs_n}\right)}\left[1 - \frac{n}{1-e^{-bs_n}}\right] = 0 \qquad (3.5)$$

$$g^{|}(b) = \frac{n}{b^2} + 2\sum_{K=1}^{n-1} \frac{s_k^2 e^{-bs_k}}{(1+e^{-bs_k})^2} \qquad (3.6)$$

The value of 'b' can be obtained using Newton-Raphson method which when substituted in equation (3.4) gives value of 'a'.

4. 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[7].

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 as

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[8].

4.1 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 *Half Logistic Distribution[6]* to time between failures data. This distribution uses cumulative time between failure data for reliability monitoring.

The equation for mean value function of Half Logistic Distribution from equation 2.1

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

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

$$m(t) = a\left[\frac{1-e^{-bt}}{1+e^{-bt}}\right] = 0.99865$$

It gives

$$t = \frac{7.300122639}{b} = t_U \qquad (4.1)$$

Similarly

$$t = \frac{0.002700002}{b} = t_L \qquad (4.2)$$

$$t = \frac{1.098612289}{b} = t_C \qquad (4.3)$$

The control limits are such that the point above the $m(t_U)$ (4.1)(UCL) is an alarm signal. A point below the $m(t_L)$(4.2) (LCL) is an indication of better quality of software. A point within the control limits indicates stable process.

4.2 **Example**

The procedure of a failures control chart for failure software process will be illustrated with an example here. Table 1 shows the time between failures of a software product [8].

Table -1: Time between failures of a component[8]

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

Table 2 shows the time between failures (cumulative) in hours, corresponding m(t) and successive difference between m(t)'s.

Table 2- Successive difference of mean value function (m(t))

Failure	***Time between Failure (hrs)***	***m(t)***	***Successive Difference of***	***Failure***	***Time between Failure (hrs)***	***m(t)***	***Successive Difference of***

number	***(cumulative)***		***m(t)***	***number***	***(cumulative)***		***m(t)***
1	30.02	2.364302301	0.112954122	15	136.25	10.35263373	1.078933884
2	31.46	2.477256423	1.75247064	16	151.78	11.43156761	1.720058168
3	53.93	4.229727062	0.105331424	17	177.5	13.15162578	0.181300217
4	55.29	4.335058486	0.265209098	18	180.29	13.332926	0.12414598
5	58.72	4.600267585	1.014117756	19	182.21	13.45707198	0.265317161
6	71.92	5.614385341	0.392552448	20	186.34	13.72238914	4.145675764
7	77.07	6.006937788	0.290696243	21	256.81	17.8680649	0.892060255
8	80.9	6.297634032	1.572927375	22	273.88	18.76012516	0.202130338
9	101.9	7.870561407	0.951568845	23	277.87	18.9622555	6.671109936
10	114.87	8.822130252	0.034170022	24	453.93	25.63336543	1.838854653
11	115.34	8.856300274	0.450773778	25	535	27.47222009	0.04287584
12	121.57	9.307074052	0.244275895	26	537.27	27.51509593	0.283919499
13	124.97	9.551349947	0.647546483	27	552.9	27.79901542	1.634249439
14	134.07	10.19889643	0.1537373	28	673.68	29.43326486	0.290356701
15	136.25	10.35263373	1.078933884	29	704.49	29.72362156	0.276439943

The values of 'a' and 'b' are computed by using the well know iterative Newton-Rapson method. These values are used to compute, T_u, T_L, T_c i.e. UCL, LCL, CL

The values of a and b are 31.524466 and 0.005006 and

$m(T_U)/UCL = 31.48190797$

$m(T_L)/LCL = 0.042558035$

$m(T_C)/CL = 15.762233$

The values of m(t) at T_c, T_u, T_L and at the given 30 inter-failure times are calculated. Then the m(t)'s are taken, which leads to 29 values. The graph with the said inter-failure times 1 to 30 on X-axis, the 29 values of m(t)'s on Y-axis, and the 3 control lines parallel to X-axis at $m(T_L)$, $m(T_U)$, $m(T_C)$ respectively constitutes failures control chart to assess the software failure phenomena on the basis of the given inter-failures time data.

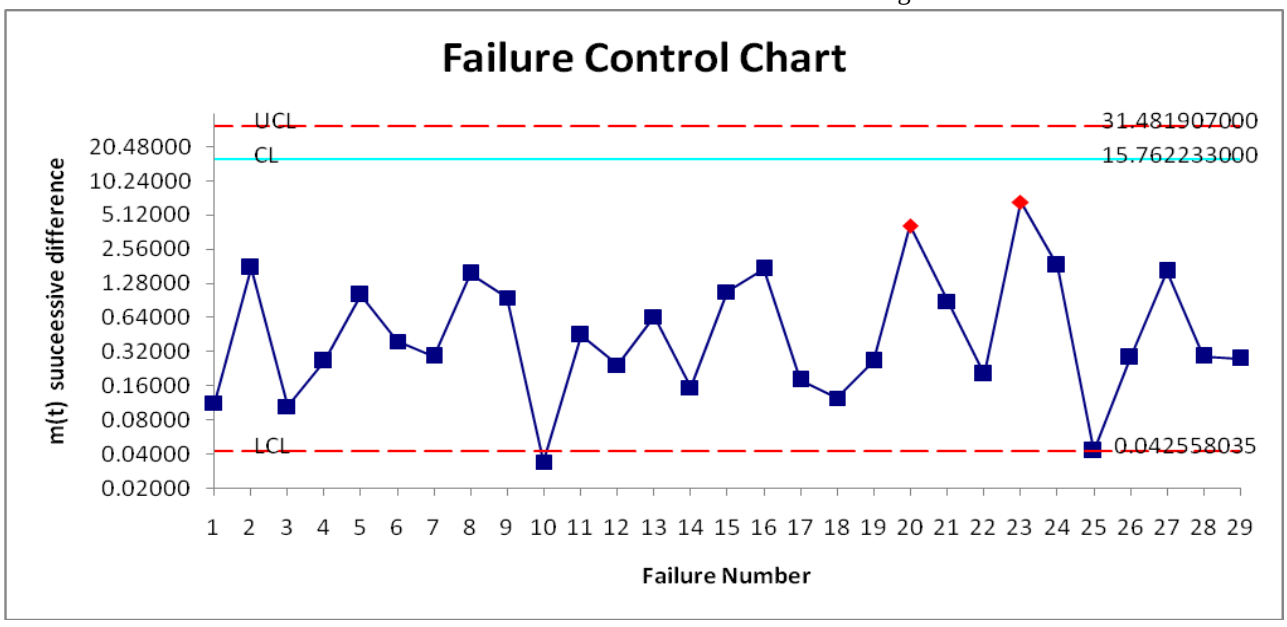

Figure 1: Failures Control Chart

5. Conclusion

This failures control chart (Figure 1) exemplifies that, the first out – of – control situation is noticed at the 10th failure with the corresponding successive difference of m(t) falling below the LCL. It results in an earlier and hence preferable out - of - control for the product. The assignable cause for this is to be investigated and promoted. In comparison, the time control chart for the same data given in Xie et a1 [8] reveals an out - of - control for the first time above the UCL at 23rd failure. Since the data of the time-control chart are inter-failure times, a point above UCL for time-control chart is also a preferable criterion for the product. The time control chart

gives the first out - of - control signal in a positive way, but at the 23rd failure. Hence it is claimed that the proposed failures control chart detects out - of - control in a positive way much earlier than the time-control chart. Therefore, earlier detections are possible in failures control chart

References

[1] 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.

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

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

[4] 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.

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

[6]Ronald P.Anjard;SPC CHART selection process;Pergaman 0026-27(1995)00119-0Elsevier science ltd.

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

[8]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

Author Biographies

Dr. R. Satya Prasad received Ph.D. degree in Computer Science in the faculty of Engineering in 2007 from Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. He have a satisfactory consistent academic track of record and received gold medal from Acharya Nagarjuna University for his out standing performance in a first rank in Masters Degree. He is currently working as Associate Professor and Head of the Department, in the Department of Computer Science & Engineering, Acharya Nagarjuna University. He has occupied various academic responsibilities like practical examiner, project adjudicator, external member of board of examiners for various Universities and Colleges in and around in Andhra Pradesh. His current research is focused on Software Engineering, Image Processing & Database Management System. He has published several papers in National & International Journals.

K Ramchand H Rao received Master's degree in Technology with Computer Science from Dr. M.G.R University, Chennai, Tamilnadu, India, . He is currently working as Associate Professor and Head of the Department, in the Department of Computer Science, A.S.N. Degree College, Tenali, which is affiliated to Acharya Nagarjuna University. He has 18 years experience and 2 years of Industry experience at Morgan Stanly, USA as Software Analyst. He is currently pursuing Ph.D., at Department of Computer Science and Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. His research include on Software Engineering.