Assessing Pareto Software Reliability Using SPC

Satya Prasad R[1], Sita Kumari K[2] and Sridevi G[3]

**[1]Department of CSE, Acharya Nagarjuna University,
Guntur, Andhra Pradesh, India**

**[2]Department of IT, V.R.Siddhartha Engineering College,
Kanuru, Vijayawada, Andhra Pradesh, India.**

**[3]Department of CSE, Nimra Women's College of Engineering,
Vijayawada, Andhra Pradesh, India.**

Abstract

Software reliability is one of the most important characteristics of software quality and is very much essential for producing reliable software systems. The reliability of software can be monitored efficiently using Statistical Process Control (SPC). It helps to identify when the failure takes place during the software development process. In this paper we proposed a control mechanism based on the cumulative observations of the time domain data using the mean value function of Pareto type II 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: *Control Charts, Mean Value Function, NHPP, Pareto type II distribution, Statistical Process Control, 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 [5], [6]. Software Process Control (SPC) concepts and methods are used for improving the software reliability by identifying and eliminating the human errors in the software development process. SPC is an important tool for monitoring and controlling manufacturing processes. SPC can be 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].

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 [2]. An advantage of SPC over other methods of quality control, such as "inspection", is that it emphasizes early detection and prevention of problems, rather than the correction of problems after they have occurred.

Control charts are the key tools which are used in SPC to monitor the quality. Basically there are two types of Control charts that can be used depending on the characteristics to be monitored. There are two main categories of Control Charts, those that display *attribute data*, and those that display *variables data*.

Attribute Data -- This category of Control Chart displays data that result from counting the number of occurrences or items in a single category of similar items or occurrences. These "count" data may be expressed as pass/fail, yes/no, or presence/absence of a defect.

Variables Data -- This category of Control Chart displays values resulting from the measurement of a continuous variable. Examples of variables data are elapsed time, temperature. The univariate control chart is a graphical display of one quality characteristic and the multivariate control chart is a graphical display of statistics that represents more than one quality characteristic. The control

chart is one of the seven tools for quality control. 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 [2].

The Non-Homogeneous Poisson Process (NHPP) based models are the most important models because of their simplicity, convenience and compatibility. The NHPP based software reliability growth models are proved quite successful in practical software reliability engineering [5]. The NHPP model represents the number of failures experienced up to certain time. 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 [3]. The Maximum likelihood estimation (MLE) is the most useful technique for deriving the point estimators. Parameter estimation is of primary importance in software reliability prediction. Once the analytical solution for m (t) is known for a given model, parameter estimation is achieved by applying a technique of Maximum Likelihood Estimate (MLE). The failure data is collected in time domain data. 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. In our proposed model the parameters are estimated using MLE. The Newton Raphson method is used for obtaining the parameter values.

This paper presents Pareto type II model for analyzing the reliability of a software system using time domain data. The layout of the paper is as follows: Section II gives the interpretation of the model for the underlying NHPP, Section III describes the proposed Pareto type II software reliability growth model, Section IV discusses parameter estimation of Pareto type II model based on time domain data. Section V describes the control charts that are used for analysing the live data for software failures and finally Section VI gives the Conclusion.

2. Model Formulation

Software reliability growth models can be used as an indication of the number of failures that may be encountered after the software has shipped and thus as an indication of whether the software is ready to ship. These models use system test data to predict the number of defects remaining in the software. There are essentially two types of software reliability models - those that attempt to predict software reliability from design parameters and those that attempt to predict software reliability from test data. The first type of models are usually called "defect density" models and use code characteristics such as lines of code, nesting of loops, external references, input/outputs, and so forth to estimate the number of defects in the software. The second type of models is usually called "software reliability growth" models. These models attempt to statistically correlate defect detection data with known functions such as an exponential function. If the correlation is good, the known function can be used to predict future behaviour. Software reliability growth models are the focus of this report. Most software reliability growth models have a parameter that relates to the total number of defects contained in a set of code. If we know this parameter and the current number of defects discovered, we know how many defects remain in the code (see Figure 1).Knowing the number of residual defects, it can be decided whether or not the code is ready to ship and how much more testing is required if we decide the code is not ready to ship. It gives us an estimate of the number of failures that our customers will encounter when operating the software. This estimate helps us to plan the appropriate levels of support that will be required for defect correction after the software has shipped and determine the cost of supporting the software.

Software reliability growth models are a statistical interpolation of defect detection data by mathematical functions. The functions are used to predict future failure rates or the number of residual defects in the code. There are different ways to represent defect detection data as discussed in Section 2.1. There are many types of software reliability growth models as described in Section 2.2, and there are different ways to statistically correlate the data to the models as discussed in Section 2.3. Current software reliability literature is inconclusive as to which data representation, software reliability growth model, and statistical correlation technique works best. The advice in the literature seems to be to try a number of the different techniques and see which works best in your environment. In Section 3, we describe the application of the techniques in the Tandem environment.

There are numerous software reliability growth models available for use according to probabilistic assumptions. The Non Homogenous Poisson Process (NHPP) based software reliability growth models are proved quite successful in practical software reliability engineering. NHPP model formulation is described in the following lines.

A software system is subject to failures at random times caused by errors present in the system. Let {N(t), t >0 } be the cumulative number of software failures by time ‘t’, where t is the failure intensity function, which is proportional to the residual fault content.

Let m (t) represents the expected number of software failures by time't'. The mean value function m (t) is finite valued, non-decreasing, non-negative and bounded with the boundary conditions.

$$m(t) = 0, t = 0$$
$$= a,\ t \to \infty$$

Where a is the expected number of software errors to be eventually detected.

Suppose N (t) is known to have a Poisson probability mass function with parameters m (t) i.e.,

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

Then N (t) is called an NHPP. Thus the stochastic behaviour of software failure phenomena can be described through the N (t) process. Various time domain models have appeared in the literature that describes the stochastic failure process by an NHPP which differ in the mean value functions m (t).

3. The Proposed Pareto Type II SRGM

In this paper we consider m (t) as given by

$$m(t) = a\left[1 - \frac{c^b}{(t+c)^b}\right] \qquad (3.1)$$

Where [m (t)/a] is the cumulative distribution function of Pareto type II distribution (Johnson et al, 2004) for the resent choice.

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

This is also a Poisson model with mean 'a'.

Let N (t) be the number of errors remaining in the system at time't'

$$N(t) = N(\infty) - N(t)$$
$$E[N(t)] = E[N(\infty)] - E[N(t)]$$
$$= a - m(t)$$
$$= a - a\left[1 - \frac{c^b}{(t+c)^b}\right]$$
$$= \frac{ac^b}{(t+c)^b}$$

4. Parameter Estimation Based On Time Domain Data

In this section we develop expressions to estimate the parameters of the Pareto type II model based on time domain data. Parameter estimation is of primary importance in software reliability prediction.

A set of failure data is usually collected in one of two common ways, time domain data and time domain data. In this paper parameters are estimated from the time domain data.

The mean value function of Pareto type II model is given by

$$m(t) = a\left[1 - \frac{c^b}{(t+c)^b}\right], \qquad t \geq 0 \qquad (4.1)$$

In order to have an assessment of the software reliability, a, b and c are to be known or they are to be estimated from software failure data. Expressions are now delivered for estimating 'a', 'b' and 'c' for the Pareto type II model [7].

We conduct an experiment and obtain N independent observations, t_1, t_2…, tn. The likelihood function for time domain data [8] is given by

$$Log\ l = -a\left[1 - \left(\frac{c}{t+c}\right)^b\right] +$$
$$\sum_{i=1}^{n}[\log a + \log b + b\ logc - (b+1)\log(t_i + c)] \qquad (4.2)$$

Accordingly parameters 'a','b' and 'c' would be solutions of the equations.

$$\frac{\partial \log L}{\partial a} = 0$$

$$a = \frac{n(t+c)^b}{(t+c)^b - c^b} \qquad (4.3)$$

The parameter 'b' is estimated by iterative Newton Raphson Method using

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

Where $g(b) and\ g'(b)$ are expressed as follows.

$$g(b) = \frac{\partial LogL}{\partial b} = 0$$

$$\frac{\partial Log\ L}{\partial b} = \frac{nlog\left(\frac{1}{t+1}\right)}{(t+1)^b - 1} + \frac{n}{b} - \sum_{i=1}^{n} log(t_i + 1) \qquad (4.4)$$

$$g'(b) = \frac{\partial^2 Log\ L}{\partial b^2} = 0$$

$$\frac{\partial^2 Log\ L}{\partial b^2} = -nlog\left(\frac{1}{t+1}\right)\left[\frac{(t+1)^b \log(t+1)}{\left[(t+1)^b - 1\right]^2}\right] - \frac{n}{b^2} \quad (4.5)$$

The parameter 'c' is estimated by iterative Newton Raphson Method using

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

Where $g(c)$ and $g'(c)$ are expressed as follows.

$$g(c) = \frac{\partial LogL}{\partial c} = 0$$

$$\frac{\partial Log\ L}{\partial c} = \frac{n}{(t+c)} + \frac{n}{c} - \sum_{i=1}^{n} \frac{2}{t_i + c} \quad (4.6)$$

$$g'(c) = \frac{\partial^2 LogL}{\partial c^2} = 0$$

$$\frac{\partial^2 logL}{\partial c^2} = \frac{-n}{(t+c)^2} - \frac{n}{c^2} + \sum_{i=1}^{n} \frac{2}{(t_i+c)^2} \quad (4.7)$$

The values of 'b' and 'c' in the above equations can be obtained using Newton Raphson Method. Solving the above equations simultaneously yields the point estimates of the parameters b and c. These equations are to be solved iteratively and their solutions in turn when substituted in equation (4.3) gives value of 'a'.

5. Data Analysis

In this section, we present the analysis of software failure data set. The set of software errors analysed here is borrowed from software development project as published in Pham (2005) [3].

The data named as NTDS data are summarized in the below table.

Table 1. NTDS Data

Failure Number n	Time Between Failure (x_k) days	Cumulative Time
1	9	9
2	12	21
3	11	32
4	4	36
5	7	43
6	2	45
7	5	50
8	8	58
9	5	63
10	7	70
11	1	71
12	6	77
13	1	78
14	9	87
15	4	91
16	1	92
17	3	95
18	3	98
19	6	104
20	1	105
21	11	116
22	33	149
23	7	156
24	91	247
25	2	249
26	1	250
Test Phase		
27	87	337
28	47	384
29	12	396
30	9	405
31	135	540
User Phase		
32	258	798
Test Phase		
33	16	814
34	35	849

Solving equations by Newton Raphson Method for the NTDS test data, the iterative solutions for MLEs of a, b and c are

$$a = 55.01871$$

$$b = 0.998899$$

$$c = 278.6101$$

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.

Table 2. Successive differences of Cumulative mean values

Failure number	Cumulative failures	Mean values	Successive differences
1	9	1.7198	2.132428
2	21	3.852228	1.810059
3	32	5.662288	0.626838
4	36	6.289126	1.059468
5	43	7.348594	0.294291
6	45	7.642885	0.720064
7	50	8.362949	1.107632
8	58	9.470581	0.66594
9	63	10.13652	0.90024
10	70	11.03676	0.125665
11	71	11.16243	0.739154
12	77	11.90158	0.120775
13	78	12.02235	1.057264
14	87	13.07962	0.453377
15	91	13.533	0.111816
16	92	13.64481	0.331858
17	95	13.97667	0.326574
18	98	14.30324	0.637793
19	104	14.94104	0.10436
20	105	15.0454	1.113071
21	116	16.15847	2.995794
22	149	19.15426	0.577016
23	156	19.73128	6.103281
24	247	25.83456	0.110506
25	249	25.94507	0.05494
26	250	26.00001	------

$$T_u = [1 - \frac{c^b}{(t+c)^b}] = 0.99865$$

$$T_c = [1 - \frac{c^b}{(t+c)^b}] = 0.5$$

$$T_l = [1 - \frac{c^b}{(t+c)^b}] = 0.00135$$

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 Mean value chart shown in figure 1.

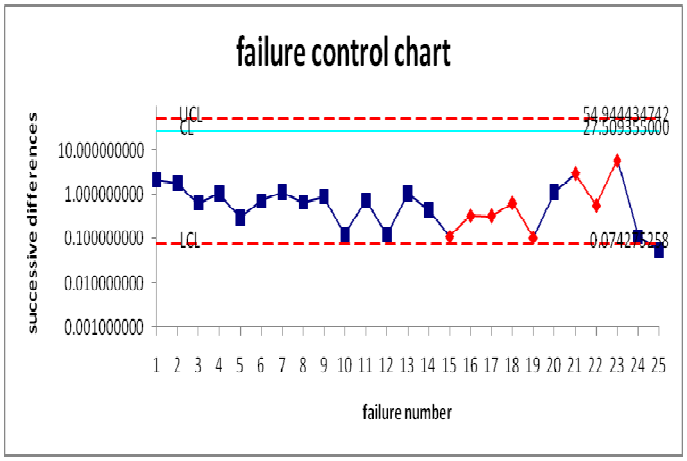

Fig 1. Mean Value Chart

A point falling below the control limit $m(t_l)$ indicates an alarming signal. A point above the control limit $m(t_u)$ indicates the better quality. If the points are falling within the control limits it indicates that the software process is in stable. The mean value chart shows all the successive differences. No failure data fall outside $m(t_u)$. It does not indicate any alarm signal. The values of control limits are as follows.

$$m(t_U) = 54.944434742$$

$$m(t_C) = 27.509355000$$

$$m(t_L) = 0.074275258$$

By placing the failure cumulative data shown in table 2 on y axis and failure week on x axis and the values of control limits are placed on Mean Value chart, we obtained Figure2. The Mean Value 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 Mean Value chart.

6. Conclusion

Software reliability is an important quality measure that quantifies the operational profile of computer systems. In this paper we proposed Pareto type II 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.

This analysis of NTDS data shows out of control signals i.e., below the LCL. We conclude that our method of estimation and the control chart are giving a +ve recommendation for their use in finding out preferable

By observing the Mean Value control chart we have identified that the failure situation is detected at 25th point of Table-2. Hence our proposed Mean Value 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. The methodology adopted in this paper is better than the methodology adopted by Xie et al, [2002] .Therefore; we may conclude that this model is the best choice for an early detection of software failures.

7. Acknowledgements

Our thanks to Department of Computer Science and Engineering, Acharya Nagarjuna University for providing necessary details to carry out the research work.

8. 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 Modelling Volume 22, Issues 10-12, Pages 149-155.

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

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

[4] Goel, A.L., Okumoto, K., 1979. Time-dependent error detection rate model for software reliability and other performance measures. IEEE Trans. Reliab. R-28, 206-211.

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

[6] Wood, A(1996), "Predicting software Reliability", IEEE Computer,2253-2264.

[7] Satya Prasad R and Geetha Rani N (2011), "Pareto type II Software Reliability Growth Model". International Journal of Software Engineering, Volume 2, Issue(4) 81-86.

[8] Musa J.D., Iannino, A.,Okumoto, K.,1987. Software Reliability: Measurement Prediction application. MC Graw Hill, NewYork.

9. Authors 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 master's degree. He is currently working as Associate Professor in the department of Computer Science &Engg., Acharya Nagarjuna University. His current research is focused on Software engineering. He has published several papers in National & International Journals.

Mrs. K.Sita kumari received M.Sc degree from Acharya Nagarjuna University, Guntur and M.Tech degree from Dr.MGR university, Chennai. She is currently pursuing her Ph.D from Department of Computer Science and Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh , India. Presently she is working as an Associate Professor in the department of Information Technology , V.R.Siddhartha Engineering College, Kanuru, Vijayawada.

Mrs. G. Sridevi received M.Sc. and M.Tech degree from Acharya Nagarjuna University. She is currently pursing Ph.D at Department of Computer Science and Engineering ,Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. She is currently working as a Vice-Principal and Associate professor in the Department of Computer Science, Nimra Women's College of Engineering, Jupudi, Ibrahimpatnam, Vijayawada, Andhra Pradesh. Her research interests lies in Data Mining and Software Engineering.