International Journal of Computer Applications (0975 – 8887)
Volume 18– No.7, March 2011

Assessing Software Reliability using Inter Failures Time Data

Dr. R Satya Prasad
Associate Prof,
Dept. of Computer Science & Eng.,
Acharya Nagarjuna University

Bandla Srinivasa Rao
Associate Prof,
Dept. of Computer Science
VRS & YRN College

Dr. R.R. L Kantham
Professor,
Dept. of Statics
Acharyan Nagarjuna University

ABSTRACT
For critical business applications continuous availability is a requirement. Software reliability is an important component of continuous application availability. A single software defect can cause a system failure. To avoid these failures, reliable software is required. Due to schedule pressure, resource limitations, and unrealistic requirements in software development process, developing reliable software is difficult. To monitor software process variations and to improve reliability, the statistical Process Control (SPC) can be applied to software development process. SPC is a methodology that aims to provide process control in statistical terms. Control charts are the most common tools for determining whether a software process is under statistically control or not. In this paper we proposed a control mechanism, based on time between failures observations using exponential distribution, which is based on Non Homogeneous Poisson Process (NHPP).

General Terms
Control Chart, SQC, Probability limits, pdf, cdf, Time between failures

Keywords
Software reliability, Statistical Process Control (SPC), NHPP, MLE, Probability limits, Exponential Distribution

1. INTRODUCTION
Software reliability is the probability of failure free operation of a software in a specified environment during specified duration [Musa, 1998]. Statistical Process Control (SPC) is known to be a powerful tool to improve process, to enhance quality and productivity [Florac, 1999]. One of the possible measures for software reliability is the use of mean time between failures (MTBF) data. As a preliminary study for applying SPC, we tried on time between failures data [Xie, 2002] to predict software reliability using some control chart mechanism [Florac, 1999].

2. BACK GROUND THEORY
This section presents the theory that underlines Goel-Okumoto NHPP exponential model, and Maximum Likelihood Estimation to time domain, 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:

$$f(t) = \frac{d(F(t))}{dt}.$$

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 can be written as:

$$m(t) = aF(t)$$

Where F(t) is a cumulative distribution function. The failure intensity function is given by (Swapna *et al.,1998)*:

$$\lambda(t) = aF'(t)$$

2.1. NHPP Model
The Non Homogeneous Poisson Process (NHPP) based software reliability growth models are proved to be quite successful in practical software reliability engineering (Musa *et al.*, 1987). 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. The parameters can be estimated by using Maximum Likelihood Estimate (MLE) based on various assumptions. Like 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. (Ohba, 1984, Pham, 1993). The present paper deals with Goel-Okumoto model applied on Inter failure times data (Xie *et al.*, 2002) which is of time domain.

Let $\{N(t), t \geq 0\}$ be the cumulative number of software failures by time 't'. m(t) is the mean value function, representing the expected number of software failures by time 't'. $\lambda(t)$ is failure intensity function, which is proportional to the residual fault content [Goel-Okumoto, 1979]. Thus

$$m(t) = a(1 - e^{-bt}) \quad a>0, b>0, t>=0 \qquad (2.1.1)$$

$$\lambda(t) = \frac{dm(t)}{dt} = b(a - m(t)) \qquad (2.1.2)$$

Here 'a' denotes the initial fault contained in a program and 'b' represents the fault detection rate. In software reliability, the initial number of faults and faults detection rate are always unknown.

3. PARAMETER ESTIMATION USING MAXIMUM LIKELIHOOD ESTIMATION
Parameter estimation is of primary importance in software reliability prediction. Once the analytical solution for $m(t)$ is known for a given model, the parameter in the solution needs to be determined. Parameter estimation is achieved by applying a technique of Maximum Likelihood Estimate (MLE) using Goel-Okumoto model. The MLE is consistent and asymptotically normally distributed as the sample size increases (Zhao, 1996). To estimate 'a' and 'b', for a sample of *n* units, first obtain the likelihood function (L):

$$L = e^{-m(s_n)} \prod_{k=1}^{n} m'(s_k) \qquad (3.1)$$

To solve equation (3.1), we take the logarithm of both sides

$$\log L = -m(s_n) + \sum_{k=1}^{n} \log m'(s_k) \qquad (3.2)$$

In order to estimate the parameters '*a*' and '*b*', we can take the derivative of the above equation (3.2) with respect to '*a*' and '*b*', and equating these derivatives to zero and solving the resulting equations for '*a*' and '*b*', we find the estimates as follows.

i.e. $\frac{\partial \log L}{\partial a} = 0$, $\frac{\partial \log L}{\partial b} = 0$

$$a = \frac{n}{(1-e^{-bs_n})} \qquad (3.3)$$

$$g(b) = \sum_{k=1}^{n} s_k - \frac{n}{b} + ns_n \frac{e^{-bs_n}}{(1-e^{-bt})} \qquad (3.4)$$

$$g'(b) = \frac{n}{b2} - ns_{n^2}\left[\frac{1}{(1-e^{-bs_n})} + \frac{e^{-bs_n}}{(1-e^{-bs_n})^2}\right] e^{-bs_n} \qquad (3.5)$$

The value of 'b' in the above equation can be obtained using Newton Raphson method

4. ESTIMATION BASED ON TIME BETWEEN FAILURES DATA

Based on the time between failures data give in Table-1, we compute the software failure process through mean value control chart. We use cumulative time between failures data for software reliability monitoring through SPC. The parameters obtained for Goel-Okumoto model applied on the given time domain data are as follows:

a = 31.698171, b = 0.003962

'$\hat{a}$' and '$\hat{b}$' are Maximum Likelihood Estimates (MLEs) of parameters and the values can be computed using numerical iterative method for the given time between failures data shown in Table 1. Using values of 'a' and 'b' we can compute $m(t)$. Now equate the *pdf* of m*(t) to* 0.00135, 0.99865, and 0.5 and the respective control limits are given by

$$T_U = (1 - e^{-bt}) = 0.99865$$
$$T_C = (1 - e^{-bt}) = 0.5$$
$$T_L = (1 - e^{-bt}) = 0.00135$$

These limits are convert at $m(t_u)$, $m(t_C)$ and $m(t_L)$ are given by

$m(t_u) = 31.6767602$, $m(t_C) = 21.1321140$,
$m(t_L) = 0.08546968$

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. 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 (MacGregor and Kourti). The values of control limits are as shown in Table-2.

Table-1: Time between failures data (Xie et al., 2002**)**

Failure No.	Time between failures	Failure No.	Time between failures	Failure No.	Time between failures	Failure No.	Time between failures	Failure No.	Time between failures
1	30.02	7	5.15	13	3.39	19	1.92	25	81.07
2	1.44	8	3.83	14	9.11	20	4.13	26	2.27
3	22.47	9	21	15	2.18	21	70.47	27	15.63
4	1.36	10	12.97	16	15.53	22	17.07	28	120.78
5	3.43	11	0.47	17	25.72	23	3.99	29	30.81
6	13.2	12	6.23	18	2.79	24	176.06	30	34.19

Table-2: Successive Difference of mean value function

Failure No	Cumulative failures	m(t)	m(t) Successive Difference	Failure No	Cumulative failures	m(t)	m(t) Successive Difference
1	30.02	3.554577564	0.160109911	16	151.78	14.32535611	1.683122175
2	31.46	3.714687474	2.383586679	17	177.5	16.00847828	0.172478506
3	53.93	6.098274153	0.137569469	18	180.29	13.18095679	0.117592238
4	55.29	6.235843622	0.34368381	19	182.21	16.298549.2	0.249934515
5	58.72	6.579527432	1.2799.4674	20	186.34	16.54848354	3.690660937
6	71.92	7.859432106	0.481483904	21	256.81	20.23914448	0.74936348
7	77.07	8.34091601	0.351758112	22	273.88	20.98850796	0.167971248
8	80.9	8.692674122	1.836637975	23	277.87	21.15647921	5.293999998
9	101.9	10.5293121	1.060330131	24	453.93	26.4504792	1.441652925
10	114.87	11.58964223	0.037410054	25	535	27.89213213	0.034077054
11	115.34	11.62705223	0.489356342	26	537.27	27.92620918	0.226497314
12	121.57	12.11640862	0.261247815	27	552.9	28.1527065	1.348363376
13	124.96	12.37765644	0.684916199	28	673.68	29.50106987	0.252475261
14	134.07	13.06257264	0.160265529	29	704.49	29.75354513	0.246358013
15	136.25	13.22283817	1.10251794	30	738.68	29.99990315	---------

5. CONTROL CHART

Control charts are sophisticated statistical data analysis tools, which include upper and lower limits to detect any outliers. They are frequently used in SPC analysis [Koutras *et.al*, 2007]. We used control chart mechanism to identify the process variation by placing the successive difference of cumulative mean values shown in table 2 on y axis and failure number on x axis and the values of control limits at mean value function are placed on Mean Value Chart, we obtained Figure 1. The Mean Value Chart shows that the successive differences of *m(t)* at 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 shown in Figure-1 is stable. No failure data fall outside $m(t_u)$. It does not indicate any alarm signal.

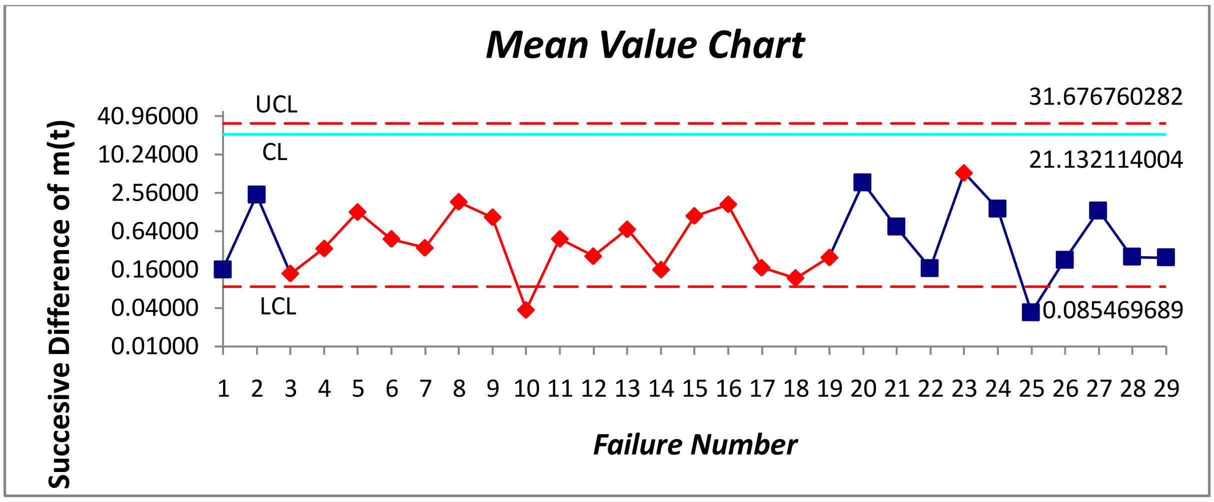

Fig-1: Mean Value Chart

6. CONCLUSION

This mean value chart exemplifies that, the first out – of – control situation is noticed at the 10th failure and the second at 25th failure with the corresponding successive difference of mean values falling below the LCL. 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* (2002) reveal that an out - of - control for the first time above the UCL occurred 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 Mean Value Chart proposed by us detects out - of - control in a positive way much earlier than the time-control chart. Therefore, earlier detections are possible in failures control chart

7. ACKNOWLEDGMENTS

We gratefully acknowledge the support of Department of Computer Science and Engineering, Acharya Nagarjuna University for providing necessary facilities to carry out the research work.

8. REFERENCES

[1] Florac, W.A., Carleton, A.D., "Measuring The Software Process:" Addison-wesley Professional, Jul 1999

[2] 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.

[3] 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.

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

[6] Musa, J.D, Software Reliability Engineering McGraw-Hill, 1998

[7] Musa, J.D., Iannino, A., Okumoto, k., 1987. "Software Reliability: Measurement Prediction Application". McGraw-Hill, New York.

[8] M.Xie, T.N. Goh, P.Ranjan, "Some effective control chart procedures for reliability monitoring" , Elsevier, Reliability engineering and System safety (2002).

[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] 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.