Interval Domain based Software Process Control using Weibull Mean Value Function

G.Krishna Mohan
Dept. of Computer Science, P.B.Siddhartha college, Vijayawada, India.

Dr. R.Satya Prasad
Dept. of Computer Science & Engg., Acharya Nagrjuna University, Nagarjuna Nagar, India.

***Abstract*—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. In this paper we propose a control mechanism based on the cumulative observations of interval domain failure data using mean value function (average number of failures) of Weibull model, which is based on Non-Homogenous Poisson Process (NHPP). The maximum likelihood estimation approach is used to estimate the unknown parameters of the model.**

Keywords-Statistical Process Control, Software reliability, Weibull model, Interval domain data, Mean Value Function, Control Charts, NHPP.

I. INTRODUCTION.

Software reliability assessment is important to evaluate and predict the reliability and performance of software system, since it is the main attribute of 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 the quality of the final product which 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 characteristics under investigation, charts can be divided into 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 or a non-random behavior occurs. 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.

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 [3].

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 in control, 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 [10].

II. LITERATURE SURVEY.

This section presents the theory that underlies NHPP models, Weibull model and maximum likelihood estimation for Interval domain (i.e grouped) 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: $f(t) = \frac{d}{dt}F(t)$. 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: $\lambda(t) = aF'(t)$ [9].

A. Data Analysis.

There are two common types of failure data: time-domain and interval-domain. Some software reliability models can handle both types of data. The time domain approach involves recording the individual times at which failure occurred. The interval domain approach is characterized by counting the number of failures occurring during a fixed period (e.g., test session, hour, week, day). The collected data are a count of the number of failures in the interval.

B. 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 [4]. 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 [6].

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 the failure intensity function, which is proportional to the residual fault content. Thus $m(t) = a(1 - e^{-bt})$ and $\lambda(t) = b(a - m(t))$. where 'a' denotes the initial number of faults contained in a program and 'b' represents the 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. In NHPP SRGM $\lambda(t)$ can be expressed in a more general way as $\lambda(t) = b(t)[a(t) - m(t)]$. 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.

A constant $a(t)$ implies the perfect debugging assumption, i.e no new faults are introduced during the debugging process. A constant $b(t)$ implies the imperfect debugging assumption, i.e when the faults are removed, then there is a possibility to introduce new faults. The present paper deals with Weibull model applied on On-line Data Entry Software Package Test Data [5] which is of Interval domain data (i.e grouped).

C. Weibull distribution.

The probability density function of a two-parameter Weibull distribution has the form: $f(t) = b\beta(bt)^{\beta-1} e^{-(bt)^{\beta}}$, where b > 0 is a scale parameter and $\beta > 0$ is a shape parameter. The corresponding cumulative distribution function is: $F(t) = 1 - e^{-(bt)^{\beta}}$. The mean value function $m(t) = a\left[1 - e^{-(bt)^{\beta}}\right]$. The failure intensity function is given as: $\lambda(t) = \beta a b^{\beta} t^{\beta-1}.e^{-(bt)^{\beta}}$.

D. ML (Maximum Likelihood) Parameter Estimation.

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). Depending on the format in which test data are available, two different approaches are frequently used. A set of failure data is usually collected in one of two common ways, time domain data and interval 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 many 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. Assuming that the data are given for the cumulative number of detected errors yi in a given time-interval (0,t_i) where i = 1,2, …, n. and $0 < t_1 < t_2 < \ldots < t_n$ then the log likelihood function (LLF) takes on the following form. Likely hood function by using λ(t) is: $L = \prod_{i=1}^{n} \lambda(t_i)$

The logarithmic likelihood function for interval domain data [7, 8] is given by:

$$LogL = \sum_{i=1}^{n} (y_i - y_{i-1}).\log\left[m(t_i) - m(t_{i-1})\right] - m(t_n)$$

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 maximizingΛ, 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:

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

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, first obtain the likelihood function: assuming $\beta = 2$.

$$L = \prod_{i=1}^{N} 2ab^2 t_i e^{-(bt_i)^2}$$

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

$$LogL = \sum_{i=1}^{n} \left(y_i - y_{i-1}\right).\log\left[m(t_i) - m\left(t_{i-1}\right)\right] - m(t_n)$$

$$= \sum_{i=1}^{n} \left(y_i - y_{i-1}\right)\left[\log a + \log\left(e^{-(bt_{i-1})^2} - e^{-(bt_i)^2}\right)\right] - a\left(1 - e^{-(bt_n)^2}\right)$$

The parameter 'a' is estimated by taking the partial derivative w.r.t 'a' and equating to '0'. (i.e $\frac{\partial \log L}{\partial a} = 0$)

$$a = \frac{y_n - y_0}{\left[1 - e^{-(bt_n)^2}\right]}$$

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'. where $g(b) \& g'(b)$ are expressed as follows. $g(b) = \frac{\partial \log L}{\partial b} = 0$; $g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0$.

$$g(b) = \sum_{i=1}^{n} \left(y_i - y_{i-1}\right)\left[\frac{t_i^2 e^{-(bt_i)^2} - t_{i-1}^2 e^{-(bt_{i-1})^2}}{e^{-(bt_{i-1})^2} - e^{-(bt_i)^2}}\right] - \frac{\left(y_n - y_0\right) t_n^2 e^{-(bt_n)^2}}{\left(1 - e^{-(bt_n)^2}\right)}$$

$$g'(b) = \frac{\left(y_n - y_0\right)\left(2bt_n^4 e^{-(bt_n)^2}\right)}{\left(1 - e^{-(bt_n)^2}\right)^2} - \sum_{i=1}^{n}\left(y_i - y_{i-1}\right)\left[\frac{2be^{-(bt_{i-1})^2} e^{-(bt_i)^2}\left(t_i^2 - t_{i-1}^2\right)^2}{\left(e^{-(bt_{i-1})^2} - e^{-(bt_i)^2}\right)^2}\right]$$

Assuming $\beta = 2$, The MLEs of the parameters for weibull model based on the 21 failure data of On-line Data Entry Software Package Test Data, which is of Interval domain data are as follows.

$$a = 54.765902$$

$$b = 0.062527$$

B. Distribution of failures

Based on the failure data given in Table 1, we compute the software failures process through Mean Value Control chart. We used cumulative time failures data for software reliability monitoring through SPC using Weibull model.

TABLE I. Time between failures of a software

Testing time (day)	Failures	Testing time (day)	Failures
1	2	12	2
2	1	13	2
3	1	14	4
4	1	15	1
5	2	16	6
6	2	17	1
7	2	18	3
8	1	19	1
9	7	20	3
10	3	21	1
11	1		

' $\hat{a}$ ' and ' $\hat{b}$ ' 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. Using 'a' and 'b' 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.

$$T_U = 1 - e^{-(bt)^\beta} = 0.99865$$

$$T_C = 1 - e^{-(bt)^\beta} = 0.5$$

$$T_L = 1 - e^{-(bt)^\beta} = 0.00135$$

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. 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 [3]. The values of control limits are as follows.

$$m(t_U) = 54.69197$$

$$m(t_C) = 27.38295$$

$$m(t_L) = 0.073934$$

TABLE II. SUCCESSIVE DIFFERENCES OF CUMULATIVE MEAN VALUES

TT (day)	Cumulative Failure	M(t)	Succ.diff
1	2	0.84979464	1.043724338
2	3	1.89351898	1.427357992
3	4	3.32087697	1.77870039
4	5	5.09957736	4.448285044
5	7	9.54786241	5.317570126
6	9	14.8654325	5.776397407
7	11	20.6418299	2.934566781
8	12	23.5763967	17.83714878
9	19	41.4135455	5.097440335
10	22	46.5109858	1.331732272
11	23	47.8427181	2.166495577
12	25	50.0092137	1.589163022
13	27	51.5983767	1.888731288
14	31	53.487108	0.2791829
15	32	53.7662909	0.806102204
16	38	54.5723931	0.050303161
17	39	54.6226963	0.087824367
18	42	54.7105206	0.015658581
19	43	54.7261792	0.025736816
20	46	54.751916	0.004263309
21	47	54.7561793	

Figure 1 is obtained by placing the number of failures cumulative data shown in table 2 on y axis and the day of the failures on x axis and the values of control limits are placed on Mean Value chart. The Mean Value chart shows that the 16th, 18th, 19th and 20th day 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 are in stable condition. No failure data fall outside the $m(t_U)$. It does not indicate any alarm signal.

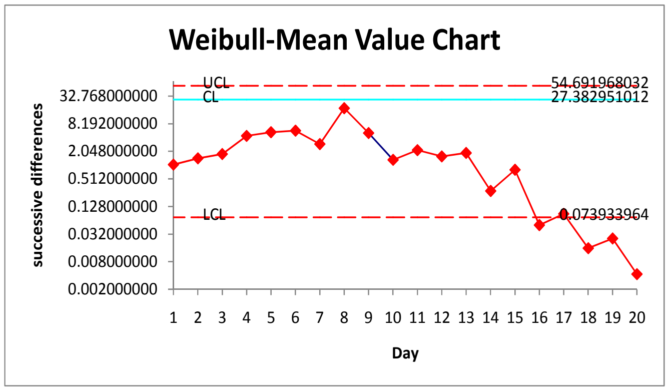

Figure 1. Mean Value Chart

IV. CONCLUSION.

The given 21 day failures are plotted through the estimated mean value function against the days of testing. The parameter estimation is carried out by Newton Raphson Iterative method. 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 +ve 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 16th, 18th, 19th and 20th point of Table-2 for the corresponding $m(t)$, which is below $m(t_L)$. Hence our proposed Mean Value Chart detects out of control situation. The early detection of software failure will improve the software Reliability.

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

[5] Ohba, M., 1984. "Software reliability analysis model". IBM J. Res. Develop. 28, 428-443.

[6] Pham. H., 1993. "Software reliability assessment: Imperfect debugging and multiple failure types in software development". EG&G-RAAM-10737; Idaho National Engineering Laboratory.

[7] Pham. H., 2003. "Handbook Of Reliability Engineering", Springer.

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

[9] 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.

[10] Xie. M., Goh. T.N., Ranjan. P., "Some effective control chart procedures for reliability monitoring", Reliability engineering and system safety. 77, 2002. 143-150.

AUTHORS PROFILE

First Author

Mr. G. Krishna Mohan is 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 in 2000, M.Tech from JNTU, Kakinada, M.Phil from Madurai Kamaraj University and pursuing Ph.D at Acharya Nagarjuna University. His research interests lies in Data Mining and Software Engineering.

Second Author

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 out standing performance in Masters Degree. He is currently working as Associate Professor and H.O.D, in the department of Computer Science & Engineering, Acharya Nagarjuna University. His current research is focused on Software Engineering. He has published several papers in National & International Journals.