Volume 2, Issue 2, February 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Comparative Study of Software Reliability Models Using SPC on Ungrouped Data

G.Krishna Mohan
Reader,
Dept. of Computer Science
P.B.Siddhartha college
Vijayawada.
km_mm_2000@yahoo.com

B. Srinivasa Rao
Associate Professor
Dept. of Compute Science
VRS & YRN College
Chirala
sreenibandla@yahoo.com

Dr. R.Satya Prasad
Associate Professor
Dept. of CS&E.
Acharya Nagrjuna University
Nagarjuna Nagar.
profrsp@gmail.com

Abstract—**Control charts are widely used for process monitoring. Software reliability process can be monitored efficiently by using Statistical Process Control (SPC). It assists the software development team to identify failures and actions to be taken during software failure process and hence, assures better software reliability. If not many, few researchers proposed SPC based software reliability monitoring techniques to improve Software Reliability Process. In this paper we propose a control mechanism based on the cumulative quantity between observations of time domain failure data using mean value function of Weibull and Goel-Okumoto distribution, which are based on Non Homogenous Poisson Process (NHPP). The Maximum Likelihood Estimation (MLE) method is used to derive the point estimators of the distributions.**

Keywords— **Statistical Process Control, Software reliability, Weibull Distribution, Goel-Okumoto distribution, Mean Value function, Probability limits, Control Charts.**

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."

The control charts can be classified into several categories, as per several distinct criteria. Depending on the number of quality characteristics under investigation, charts can be divided into univariate control charts and 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 improves process capabilities; and focus business areas which need 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]. Many factors influence the process, resulting in variability. The causes of process variability can be broadly classified into two categories, viz., assignable causes and chance causes.

The control limits can then be utilized to monitor the failure times of components. After each failure, the time can be plotted on the chart. If the plotted point falls between the calculated control limits, it indicates that the process is in the state of statistical control and no action is warranted. If the point falls above the UCL, it indicates that the process average, or the failure occurrence rate, may have decreased which results in an increase in the time between failures. This is an important indication of possible process improvement. If this happens, the management should look for possible causes for this improvement and if the causes are discovered then action should be taken to maintain them. If the plotted point falls below the LCL, It indicates that the process average, or the failure occurrence rate, may have increased which results in a decrease in the failure time. This means that process may have deteriorated and thus actions should be taken to identify and the causes may be removed. It can be noted here that the parameter a, b should normally be estimated with the data from the failure process. Since a, b are the parameters in the proposed distributions, any traditional estimator can be used.

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 normal, 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 [9].

II. LITERATURE SURVEY

This section presents the theory that underlies the proposed distributions 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: $F\left(t\right)$. Where, The mathematical relationship between the pdf and cdf is given by: $f(t)=\dfrac{d\left(F\left(t\right)\right)}{dt}$. 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: $\lambda(t)=aF'(t)$ [8].

A. 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 [5,6].

Let $\left\{N\left(t\right),t\geq 0\right\}$ 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'. $\lambda\left(t\right)$ is the failure intensity function, which is proportional to the residual fault content. Thus $m\left(t\right)=a(1-e^{-bt})$ and $\lambda\left(t\right)=\dfrac{dm(t)}{dt}=b(a-m(t))$. 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 $\lambda\left(t\right)$ can be expressed in a more general way as $\lambda\left(t\right)=\dfrac{dm(t)}{dt}=b\left(t\right)\left[a\left(t\right)-m\left(t\right)\right]$. where $a\left(t\right)$ is the time-dependent fault content function which includes the initial and introduced faults in the program and $b\left(t\right)$ is the time-dependent fault detection rate. A constant $a\left(t\right)$ implies the perfect debugging assumption, i.e no new faults are introduced during the debugging process. A constant $b\left(t\right)$ implies the imperfect debugging assumption, i.e when the faults are removed, then there is a possibility to introduce new faults.

B. Goel-Okumoto distribution

The Goel-Okumoto model is a simple NonHomogenous Poisson Process (NHPP) model with the mean value function $m\left(t\right)=a\left(1-e^{-bt}\right)$ [12]. Where the parameter 'a' is the number of initial faults in the software and the parameter 'b' is the fault detection rate. The corresponding failure intensity function is given by $\lambda(t) = abe^{-bt}$. The probability density function of a Goel-Okumoto model has the form: $f(t) = be^{-bt}$. The corresponding cumulative distribution function is: $F(t) = 1 - e^{-bt}$.

C. Weibull distribution

The Weibull distribution is a generalization of exponential distribution, which is recovered for β = 1. Although the exponential distribution has been widely used for times-between-event, Weibull distribution is more suitable as it is more flexible and is able to deal with different types of aging phenomenon in reliability. Hence in reliability monitoring of equipment failures, the Weibull distribution is a good alternative. The probability density function of a two-parameter Weibull model 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 $m(t) = a\left[1 - e^{-(bt_n)^\beta}\right]$. The failure intensity function is given as: $\lambda(t) = \beta a b^\beta t^{\beta-1}.e^{-(bt)^\beta}$.

D. MLE (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 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. If we conduct an experiment and obtain N independent observations, $t_1, t_2, \ldots, t_N$. The likelihood function [7] may be given by the following product:

$$L(t_1, t_2, \ldots, t_N \mid \theta_1, \theta_2, \ldots, \theta_k) = \prod_{i=1}^{N} f(t_i; \theta_1, \theta_2, \ldots, \theta_k)$$

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

The logarithmic likelihood function is given by:

$$\log L = \log\left(\prod_{i=1}^{n} \lambda(t_i)\right)$$

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

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

The parameters 'a' and 'b' are estimated as follows. 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'.

III. ILLUSTRATING THE MLE METHOD

A. Goel-Okumoto parameter estimation

The likelihood function is given as, $L = \prod_{i=1}^{N} abe^{-(bt)}$

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

$$\log L = \sum_{i=1}^{n} \log(abe^{-(bt_i)}) - a[1 - e^{-(bt_n)}] \quad .$$

Taking the Partial derivative with respect to 'a' and equating to '0'. (i.e $\frac{\partial \log L}{\partial a} = 0$).

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

Taking the Partial derivative with respect to 'b' and equating to'0'.(i.e $g(b) = \frac{\partial \log L}{\partial b} = 0$).

$$g(b) = \sum_{i=1}^{n} t_i - \frac{n}{b} + nt_n \frac{e^{-(bt_n)}}{\left(1 - e^{-(bt_n)}\right)} = 0$$

Taking the partial derivative again with respect to 'b' and equating to '0'. (i.e $g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0$).

$$g'(b) = \frac{n}{b^2} - nt_n^2 \left\{ \frac{1}{\left(1 - e^{-(bt_n)}\right)} + \frac{e^{-(bt_n)}}{\left(1 - e^{-(bt_n)}\right)^2} \right\} e^{-bt_n}$$

B. Weibull parameter estimation

The likelihood function, assuming $\beta = 2$ (Rayleigh) is given as, $L = \prod_{i=1}^{N} 2ab^2 t.e^{-(bt)^2}$

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

$$\log L = \sum_{i=1}^{n} \log(2ab^2 t_i e^{-(bt)^2}) - a[1 - e^{-(bt_n)^2}] \quad .$$

Taking the Partial derivative with respect to 'a' and equating to '0'. (i.e $\frac{\partial \log L}{\partial a} = 0$).

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

Taking the Partial derivative with respect to 'b' and equating to'0'.(i.e $g(b) = \frac{\partial \log L}{\partial b} = 0$).

$$g(b) = \frac{2n}{b} - 2b\sum_{i=1}^{n} t_i^2 - \frac{2.n.b.t_n^2.e^{-(bt_n)^2}}{\left(1 - e^{-(bt_n)^2}\right)} = 0$$

Taking the partial derivative again with respect to 'b' and equating to '0'. (i.e $g'(b)=\frac{\partial^2 \log L}{\partial b^2}=0$).

$$g'(b)=2n\left(\frac{-1}{b^2}\right)-2\sum_{i=1}^{n}t_i^2-2nt_n^2\left\{\frac{e^{-(bt_n)^2}}{\left(1-e^{-(bt_n)^2}\right)}-\frac{2b^2t_n^2.e^{-(bt_n)^2}}{\left(1-e^{-(bt_n)^2}\right)^2}\right\}$$

C. Distribution of Time between failures

Based on the inter failure data given in Table 1, we compute the software failures process through Mean Value Control chart. We used cumulative time between failures data for software reliability monitoring using Goel-Okumoto and Weibull distributions. The use of cumulative quality is a different and new approach, which is of particular advantage in reliability.

'$\hat{a}$' and '$\hat{b}$' are Maximum Likely hood Estimates of parameters and the values can be computed using iterative method for the given cumulative time between failures data [10] shown in table 1. Using 'a' and 'b' values we can compute $m(t)$.

TABLE 1. TIME BETWEEN FAILURES OF A SOFTWARE

Failure Number	Time between failure(h)	Failure Number	Time between failure(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

Assuming an acceptable probability of false alarm of 0.27%, the control limits can be obtained as [10]:

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

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

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

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 and figure 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 condition [11]. The values of parameter estimates and the control limits are given in table 2 and 3 respectively.

TABLE 2. PARAMETER ESTIMATES

model	a	b
GO	31.698171	0.003962
Weibull	30.051592	0.003416

TABLE 3. CONTROL LIMITS.

model	$m(t_U)$	$m(t_C)$	$m(t_L)$
GO	31.676760	21.132114	0.085469
Weibull	30.011170	15.025870	0.040570

TABLE 4. MEAN SUCCESSIVE DIFFERENCES OF GO

FN	m(t)	SD
1	3.554578	0.160101
2	3.714687	2.383587
3	6.098274	0.137569
4	6.235844	0.343684
5	6.579527	1.279946
6	7.859432	0.481484
7	8.340916	0.351758
8	8.692674	1.836638
9	10.529312	1.060330
10	11.589642	0.037410
11	11.627052	0.489356
12	12.116408	0.261248
13	12.377656	0.684916
14	13.062573	0.160266
15	13.222838	1.102518
16	14.325356	1.683122
17	16.008478	0.172479
18	13.180956	0.117592
19	16.298549	0.249935
20	16.548483	3.690661
21	20.239144	0.749363
22	20.988508	0.167971
23	21.156479	5.293999
24	26.450479	1.441653
25	27.892132	0.034077
26	27.926209	0.226497
27	28.152706	1.348363
28	29.501069	0.252475
29	29.753545	0.246358
30	29.999903	

TABLE 5. MEAN SUCCESSIVE DIFFERENCES OF WEIBULL

FN	m(t)	SD
1	0.314371	0.030704
2	0.345076	0.657725
3	1.002801	0.050307
4	1.053108	0.132025
5	1.185134	0.575065
6	1.760199	0.252180
7	2.012380	0.197261
8	2.209641	1.219663

9	3.429305	0.859242
10	4.288547	0.032507
11	4.321054	0.439360
12	4.760415	0.245447
13	5.005863	0.680255
14	5.686118	0.166975
15	5.853094	1.230688
16	7.083782	2.161267
17	9.245050	0.240957
18	9.486008	0.166350
19	9.652358	0.359124
20	10.011482	6.120127
21	16.131610	1.396357
22	17.527968	0.317624
23	17.845592	9.491850
24	27.337443	1.649185
25	28.986628	0.029822
26	29.016451	0.186649
27	29.203101	0.697874
28	29.900976	0.058849
29	29.959825	0.040168
30	29.999994	

Figure 1 and 2 are obtained by placing the time between failures cumulative data shown in tables 3, 4 on y axis and failure number on x axis, and the values of control limits are placed on Mean Value chart. The Mean Value chart of Goel-Okumoto shows that the 10^{th} and 25^{th} failure data has fallen below $m(t_L)$. The Mean Value chart of weibull shows that the 1^{st}, 10^{th} and 25^{th} failure data has fallen below $m(t_L)$. The successive differences of mean values below $m(t_L)$ indicates the failure process. In the present scenario, It is significantly early detection of failure through weibull using Mean Value Chart. The software quality is determined by detecting failures at an early stage. The Remaining Failure data shown in figure 1 are in stable condition. No failure data fall outside the $m(t_U)$. It does not indicate any alarm signal.

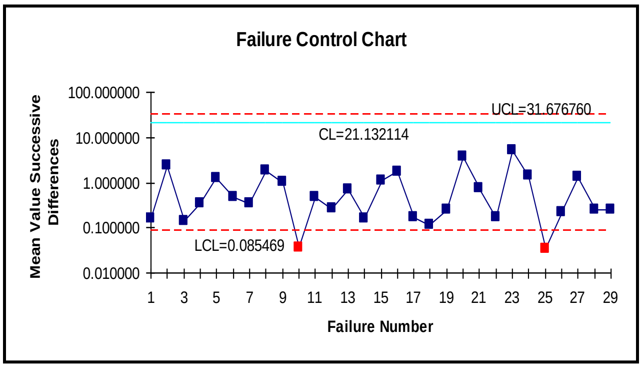

Figure: 1 GO Failure Control Chart

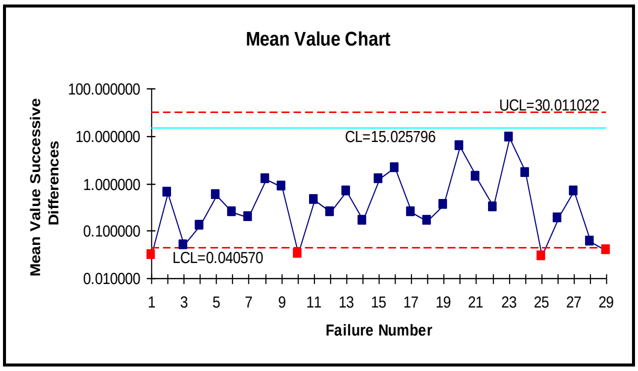

Figure: 2 Weibull Mean Value Chart

IV. CONCLUSION

The given 30 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 the models. 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 10th and 25th point of table-4, 1st ,10th, 25th and 29th point of table-5 i.e ailure data has fallen below $m(t_L)$. The successive difference of mean values below $m(t_L)$ indicates the failure process. In the present scenario, It is significantly early detection of failure through weibull using Mean Value Chart. The software quality is determined by detecting failures at an early stage 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 [10], 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 than the situation in the time control chart. The early detection of software failure will improve the software Reliability. When the time between failures is less than LCL, it is likely that there are assignable causes leading to significant process deterioration and it should be investigated. On the other hand, when the time between failures has exceeded the UCL, there are probably reasons that have lead to significant improvement.

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 Modeling 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 77 143 -150¸ 2002.
[11] Satya Prasad, R., "Half logistic distribution for software reliability growth model", Ph.D thesis, 2007.

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

AUTHOR PROFILE:

FIRST AUTHOR:

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. He has published several research papers in National and International Journals.

SECOND AUTHOR:

B. Srinivasa rao received the Master Degree in Computer Science and Engineering from Dr MGR Deemed University, Chennai, Tamil Nadu, India. He is Currently working as Associate Professor in PG Department of Computer Applications, VRS & YRN College, Chirala, Andhra Pradesh, India. His research interests include software reliability, Cryptography and Computer Networks. He has published several papers in National and International Journals.

THIRD 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 outstanding 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.