International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 2, Issue 4, July – August 2013 ISSN 2278-6856

Two step approach for Software Process Control: HLSRGM

Dr. R. Satya Prasad[1], Shaheen[2] and G. Krishna Mohan3

[1]Associate Professor, Dept. of Computer Science & Engg., Acharya Nagrjuna University
Nagarjuna Nagar, Guntur.

[2]Asst. Professor, IPE, Osmaina university campus, Hyderabad.

[3]Reader, Dept. of Computer Science, P.B.Siddhartha college
Vijayawada.

Abstract: ***Software reliability process can be monitored efficiently by using Statistical Process Control (SPC). Control charts are widely used for process monitoring. It assists the software development team to identify failures and actions to be taken during software failure process and hence, assures better software reliability. SRGM is a mathematical model of how the software reliability improves as faults are detected and repaired. 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 Half Logistic Software Reliability Growth Model (HLSRGM), which is based on Non Homogenous Poisson Process (NHPP). The model parameters are estimated by a two step approach.***

Keywords: Statistical Process Control, Software reliability, HLSRGM, Mean Value function, two step approach, Probability limits, Control Charts.

1. Introduction

Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment (Lyu, 1996; Musa et al., 1987). Among all SRGMs developed so far a large family of stochastic reliability models based on a non-homogeneous Poisson Process known as NHPP reliability models, has been widely used. Some of them depict exponential growth while others show S-shaped growth depending on nature of growth phenomenon during testing. The success of mathematical modeling approach to reliability evaluation depends heavily upon quality of failure data collected.

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 (Kimura et al., 1995).

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. 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 (Koutras et al., 2007). For a process to be in control the control chart should not have any trend or nonrandom pattern.

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 (MacGregor and Kourti., 1995). 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. We followed a two step approach in estimating the parameters.

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 (Gokhale and Trivedi, 1998).

2. LITERATURE SURVEY

This section presents the theory that underlies NHPP based SRGMs, the HLSRGM and the parameter estimation methods.

2.1 NHPP SRGM

The Non-Homogenous Poisson Process (NHPP) based software reliability growth models (SRGMs) 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. Model parameters can be estimated by using two step approach (i.e one parameter is estimated through Maximum Likelihood Estimate (MLE) and another parameter is estimated through Least Square Estimation (LSE)). 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 (Ohba, 1984; Pham, 1993).

If 't' is a continuous random variable with probability density function: $f(t;\theta_1,\theta_2,\dots,\theta_k)$. Where $\theta_1,\theta_2,\dots,\theta_k$ are k unknown constant parameters which need to be estimated, and cumulative distribution function: $F(t)$. Let 'a' denote the expected number of faults that would be detected given infinite testing time in case of finite failure NHPP models and 'b' represents the fault detection rate. In software reliability, the initial number of faults and the fault detection rate are always unknown. The two step approach is used to evaluate the unknown parameters. Then, the mean value function of the finite failure NHPP models can be written as: $m(t) = aF(t)$, representing the expected number of software failures by time 't'. The failure intensity function $\lambda(t)$ in case of the finite failure NHPP models is given by: $\lambda(t) = aF'(t)$. which is proportional to the residual fault content (Pham, 2006).

Let $N(t)$ be the cumulative number of software failures by time 't'. A non-negative integer-valued stochastic process $N(t)$ is called a counting process, if $N(t)$ represents the total number of occurrences of an event in the time interval [0, t] and satisfies these two properties:

If $t_1 < t_2$, then $N(t_1) \le N(t_2)$

If $t_1 < t_2$, then $N(t_2) - N(t_1)$ is the number of occurrences of the event in the interval $[t_1, t_2]$.

One of the most important counting processes is the Poisson process. A counting process, $N(t)$, is said to be a Poisson process with intensity λ. if The initial condition is N(0) = 0

The failure process, N(t), has independent increments

The number of failures in any time interval of length s has a Poisson distribution with mean λ s, that is,

$$P\{N(t+s)-N(t)=n\} = \frac{e^{-\lambda s}(\lambda s)^n}{n!}$$

Describing uncertainty about an infinite collection of random variables one for each value of 't' is called a stochastic counting process denoted by $[N(t), t \ge 0]$.

The process $\{N(t), t \ge 0\}$ is assumed to follow a Poisson distribution with characteristic MVF (Mean Value Function) m(t). Different models can be obtained by using different non decreasing m(t).

A Poisson process model for describing about the number of software failures in a given time (0, t) is given by the probability equation.

$$P[N(t)=y] = \frac{e^{-m(t)}[m(t)]^y}{y!}, \quad y = 0,1,2,\dots$$

Where, $m(t)$ is a finite valued non negative and non decreasing function of $'t'$ called the mean value function. Such a probability model for $N(t)$ is said to be an NHPP model.

2.2 Model description: HLSRGM

One simple class of finite failure NHPP model is the HLSRGM, assuming that the failure intensity is proportional to the number of faults remaining in the software describing an exponential failure curve. It has two parameters. Where, 'a' is the expected total number of faults in the code and 'b' is the shape factor defined as, the rate at which the failure rate decreases. The cumulative distribution function of the model is: $F(t)=\frac{\left(1-e^{-bt}\right)}{\left(1+e^{-bt}\right)}$.

The expected number of faults at time't' is denoted by $m(t)=\frac{a\left(1-e^{-bt}\right)}{\left(1+e^{-bt}\right)}\ ,a>0,b>0,t\geq 0$.

The corresponding failure intensity function is given by $\lambda(t)=\frac{2abe^{-bt}}{\left(1+e^{-bt}\right)^{2}}$. Where 't' can be calendar time (Krishna Mohan et al., 2012).

2.3 Parameter estimation methods

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. Method of least squares (LSE) or maximum likelihood (MLE) has been suggested and widely used for estimation of parameters of mathematical models (Kapur et al., 2008). Non-linear regression is a method of finding a nonlinear model of the relationship between the dependent variable and a set of independent variables. Unlike traditional linear regression, which is restricted to estimating linear models, nonlinear regression can estimate models with arbitrary relationships between independent and dependent variables. The model proposed in this paper is a non-linear and it is difficult to find solution for nonlinear models using simple Least Square method. Therefore, the model has been transformed from non linear to linear.

The least squares method is widely used to estimate the numerical values of the parameters to fit a function to a set of data. We will use the method in the context of a Linear regression problem. It exists with several variations: it simpler version is called Ordinary Least Squares (OLS), a more sophisticated version is called Weighted Least Squares (WLS). A recent variations of least squares method are Alternating least squares (ALS) and Partial Least Squares (PLS) (Lewis-Beck, 2003).

The standard approach of using derivatives is not always possible when estimating the parameters of a non linear function with OLS. Therefore, iterative methods are often used. The best value of the estimate is found by searching in a stepwise fashion. They proceed by using at each step a linear approximation of the function and refine this approximation by successive corrections. The techniques involved are gradient descent and Gauss-Newton approximations. A neural network constitutes a popular recent application of these techniques.

3. TWO STEP APPROACH FOR PARAMETER ESTIMATION

MLE and LSE techniques are used to estimate the model parameters (Lyu, 1996; Musa et al., 1987). Sometimes, the likelihood equations are difficult to solve explicitly. In such cases, the parameters estimated with some numerical methods (Newton Raphson method). On the other hand, LSE, like MLE, applied for small sample sizes and may provide better estimates (Huang and Kuo, 2002).

3.1 Algorithm for the 2-step approach.

- Consider the Cumulative distribution function $F(t)$ and equate to p_i, i.e $F(t)=p_i$, where $p_i=\frac{i}{n+1}$
- Express the equated equation $F(t)=p_i$ as a linear form, $y=mx+b$.
- Find model parameters of mean value function $m(t)$. Where $m(t)=aF(t)$
  - The initial number of faults $\hat{a}$ is estimated through MLE method. Since, it forms a closed solution.
  - The remaining parameters are estimated through LSE regression approach.

3.2 ML (Maximum Likelihood) Parameter Estimation

The idea behind maximum likelihood parameter estimation is to determine the parameters that maximize the probability of the sample data. The method of maximum likelihood is considered to be more robust 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 MLE 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 (Pham, 2003) may be given by the following product:

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

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

Log Likelihood function for ungrouped data (Pham, 2006),

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

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 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’.

3.3 LS (Least Square) parameter estimation

LSE is a popular technique and widely used in many fields for function fit and parameter estimation (Liu, 2011). The least squares method finds values of the parameters such that the sum of the squares of the difference between the fitting function and the experimental data is minimized. Least squares linear regression is a method for predicting the value of a dependent variable Y, based on the value of an independent variable X.

o ***The Least Squares Regression Line***

Linear regression finds the straight line, called the least squares regression line that best represents observations in a bivariate data set. Given a random sample of observations, the population regression line is estimated by: $\hat{y} = bx + a$. Where, ‘a’ is a constant, ‘b’ is the regression coefficient and ‘x’ is the value of the independent variable, and ‘$\hat{y}$’ is the predicted value of the dependent variable. The least square method defines the estimate of these parameters as the values which minimize the sum of the squares between the measurements and the model. Which amounts to minimizing the expression: $E = \sum_i \left(Y_i - \hat{Y}_i\right)^2$ (Xie, 2001).

Taking the derivative of E with respect to ‘a’ and ‘b’ and setting them to zero gives the following set of equations (called the normal equations):

$$\frac{\partial E}{\partial a} = 2Na + 2b\sum X_i - 2\sum Y_i = 0$$, and

$$\frac{\partial E}{\partial b} = 2b\sum X_i^2 + 2a\sum X_i - 2\sum Y_i X_i = 0$$

The Least Square Estimates of ‘a’ and ‘b’ are obtained by solving the above equations. Where, $a = \bar{Y} - b\bar{X}$ and

$$b = \frac{\sum\left(Y_i - \bar{Y}\right)\left(X_i - \bar{X}\right)}{\sum\left(X_i - \bar{X}\right)^2}.$$

4. ILLUSTRATING THE PARAMETER ESTIMATION: HLSRGM

4.1 ML Estimation

Procedure to find parameter ‘a’ using MLE.

The likelihood function of HLSRGM is given as,

$$L = \prod_{i=1}^{N} \frac{2abe^{-bt_i}}{\left(1 + e^{-bt_i}\right)^2}$$

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

$$\log L = n\log 2 + n\log a + n\log b - b\sum_{i=1}^{n} t_i - 2\sum_{i=1}^{n}\log\left(1+e^{-bt_i}\right) - a\left(\frac{1-e^{-bt_n}}{1+e^{-bt_n}}\right)$$

Taking the Partial derivative with respect to ‘a’ and equating to ‘0’.

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

Taking the Partial derivative with respect to ‘b’ and equating to‘0’.

$$g(b) = \sum_{i=1}^{n} t_i - \frac{n}{b} - 2\sum_{i=1}^{n}\frac{t_i e^{-bt_i}}{\left(1+e^{-bt_i}\right)} - \frac{2t_n e^{-bt_n}}{\left(1+e^{-bt_n}\right)}\left[1 - \frac{n}{\left(1-e^{-bt_n}\right)}\right]$$

Taking the partial derivative again with respect to ‘b’ and equating to ‘0’.

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

4.2 LS Estimation

Procedure to find parameter ‘b’ using regression approach.

- The cumulative distribution function of HLSRGM is, $F(t) = \frac{\left(1-e^{-bt}\right)}{\left(1+e^{-bt}\right)}$. The c.d.f is equated to p_i.

Where, $p_i = \frac{i}{n+1}$.

- The equation $F(t) = p_i$ is expressed as a linear form,

$V_i = \sigma U_i$. Where, $V_i = X_i$; $U_i = -\log\left(\frac{1-p_i}{1+p_i}\right)$;

and $\sigma = \frac{\sum V_i U_i - n\bar{V}\bar{U}}{\sum U_i^2 - n\bar{U}^2}$

- Where, ‘$1/\sigma$’ is nothing but the parameter ‘b’ estimated through regression approach.

5. DISTRIBUTION OF TIME BETWEEN FAILURES

Based on the inter failure data given in Table 1 and 2, we compute the software failures process through failure Control chart. We used cumulative time between failures data for software reliability monitoring using HLSRGM. The use of cumulative quality is a different and new approach, which is of particular advantage in reliability. ‘$\hat{a}$’ and ‘$\hat{b}$’ are Estimates of parameters and the values can be computed using iterative method for the given time between failures data (pham, 2006) shown in table 1 and 2. Using ‘a’ and ‘b’ values we can compute $m(t)$.

Table 1. Time between failures of a software, IBM

No. of Error	Inter-failure time	No. of Error	Inter-failure time	No. of Error	Inter-failure time
1	10	6	12	11	19
2	9	7	18	12	30
3	13	8	15	13	32
4	11	9	22	14	25
5	15	10	25	15	40

International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 2, Issue 4, July – August 2013 ISSN 2278-6856

Table 2. Time between failures of a software, NTDS

F.NO	TBF	F.NO	TBF	F.NO	TBF
1	9	10	7	19	6
2	12	11	1	20	1
3	11	12	6	21	11
4	4	13	1	22	33
5	7	14	9	23	7
6	2	15	4	24	91
7	5	16	1	25	2
8	8	17	3	26	1
9	5	18	3		

Assuming an acceptable probability of false alarm of 0.27%, the control limits can be obtained as (Xie, 2002):

$$T_U = \left(\frac{1-e^{-bt}}{1+e^{-bt}}\right) = 0.99865$$

$$T_C = \left(\frac{1-e^{-bt}}{1+e^{-bt}}\right) = 0.5$$

$$T_L = \left(\frac{1-e^{-bt}}{1+e^{-bt}}\right) = 0.00135$$

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 Failure control chart shown in figure 1 and 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. The values of control limits are as follows.

Table 3: Parameter estimates and Control limits

Data Set	Control Limits		
	UCL	CL	LCL
DS1 IBM	18.585601	9.305362 5	0.025124 5
DS2 NTD	28.728418	14.38362 7	0.038836

Table 4. Successive differences of mean values: DS1

F.NO	CTBF	m(t)	SDs
1	10	8.994191	5.193557
2	19	14.187748	3.190792
3	32	17.378540	0.836914
4	43	18.215454	0.313307
5	58	18.528761	0.058803
6	70	18.587565	0.019688
7	88	18.607253	0.002758
8	103	18.610011	0.000644
9	125	18.610655	0.000065
10	150	18.610720	0.000004
11	169	18.610724	0.000001
12	199	18.610725	0.000000
13	231	18.610725	0.000000
14	256	18.610725	0.000000
15	296	18.610725	

Table 5. Successive differences of mean values: DS2

F.NO	CTBF	m(t)	SDs
1	9	9.364585	9.547409
2	21	18.911995	5.079918
3	32	23.991912	1.160868
4	36	25.152781	1.421052
5	43	26.573833	0.295708
6	45	26.869540	0.580299
7	50	27.449840	0.587264
8	58	28.037104	0.226504
9	63	28.263607	0.204790
10	70	28.468398	0.021510
11	71	28.489908	0.100268
12	77	28.590176	0.012770
13	78	28.602946	0.080585
14	87	28.683532	0.021694
15	91	28.705225	0.004482
16	92	28.709707	0.011595
17	95	28.721302	0.009260
18	98	28.730563	0.013301
19	104	28.743863	0.001691
20	105	28.745554	0.012196
21	116	28.757750	0.008706
22	149	28.766456	0.000326
23	156	28.766782	0.000471
24	247	28.767253	0.000000
25	249	28.767254	0.000000
26	250	28.767254	

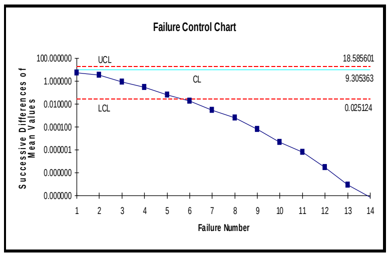

Figure: 1 Failure Control Chart, DS1

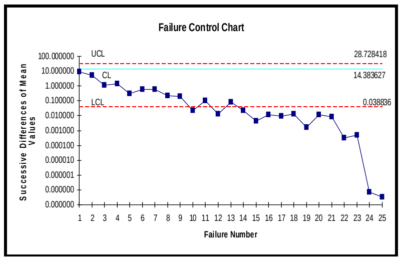

Figure: 2 Failure Control Chart, DS2

Figure 1 and 2 are obtained by placing the time between failures cumulative data shown in table 1 and 2 on y axis and failure number on x axis and the values of control limits are placed on Failure control chart. The Failure control charts show that from the 6th failure of DS1 and 10th failure of DS2, the data has fallen below $m(t_L)$ which indicates the failure process. It is significantly early detection of failures using Failure control chart. The software quality is determined by detecting failures at an early stage.

6. CONCLUSION

The given inter failure times are plotted through the estimated mean value function against the failure serial order. The parameter estimation is carried out by two step approach for the considered model. 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 Failure Control chart we identified that the failure situation is detected at 6th point of table-1 and 10th point of table-2 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 Ramchand(2011) control chart and then continued to fail. 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. From Figure 1 and 2 the process is stabilized by touching the X-axis. As SPC is to stabilize at some point of time the two-step approach is preferable.

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] Gokhale, S.S and Trivedi, K.S., 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., (2002). “Some effective control chart procedures for reliability monitoring” -Reliability engineering and System Safety 77 143 -150.

[11] Goel, A. L. and Okumoto, K., 1979, ‘Time-dependent error-detection rate model for software reliability and other performance measures’, IEEE Transactions on Reliability, vol. 28, pp. 206-211.

[12] Huang, C.Y and Kuo, S.Y., (2002). “Analysis of incorporating logistic testing effort function into software reliability modelling”, IEEE Transactions on Reliability, Vol.51, No. 3, pp. 261-270.

[13] Kapur, P.K., Gupta, D., Gupta, A. And Jha, P.C., (2008). “Effect of Introduction of Fault and Imperfect Debugging on Release Time”, Ratio Mathematica, 18, pp. 62-90.

[14] Krishna Mohan, G., Srinivasa Rao, B. and Satya Prasad, R. (2012). ”A Comparative study of Software Reliability models using SPC on ungrouped data”, International Journal of Advanced Research in Computer Science and Software Engineering. Volume 2, Issue 2, February.

[15] Lewis-Beck, M., Bryman, A. and Futing, T. (Eds) (2003). “Least Squares”, Encyclopedia of Social Sciences Research Methods. Thousand Oaks (CA):Sage.

[16] Liu, J., (2011). “Function based Nonlinear Least Squares and application to Jelinski-Moranda Software Reliability Model”, stat. ME, 25th August.

[17] Lyu, M.R., (1996). “Handbook of Software Reliability Engineering”, McGraw-Hill, New York..

[18] Ramchand. K. H., (2011) “ASSESSING SOFTWARE RELIABILITY USING SPC”, Acharya Nagarjuna University, Ph.D thesis.

[19] Xie, M., Yang, B. and Gaudoin, O. (2001). “Regression goodness-of-fit Test for Software Reliability Model Validation”, ISSRE and Chillarege Corp.

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 45 papers in National & International Journals.

Miss. Shaheen working as Assistant Professor, Institute of Public Enterprise (IPE), Osmania Univeristy Campus. She did her Master of Computer Science. She was associated with INFOSYS, Gacchibowli, Hyderabad and Genpact, Uppal as an Academic Consultant training fresh recruits on technical subjects. She is currently pursuing Ph. D. in Computer Science from Acharya Nagarjuna University.

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. He has 13 years of teaching experience. He has qualified APSLET in 2012. He published 15 papers in National & International journals.