International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 12 (2015) pp. 32055-32070

http://www.ripublication.com

Monitoring Software Quality Using SPC – An Order Statistics Approach

K.Sobhana [(1)], Dr. R. Satya Prasad [(2)] and Ch.Smitha Chowdary [(3)]

(1) Research Scholar, Department of Computer Science, Krishna University, Machilipatnam, Andhra Pradesh (India)
E-mail: msobhana@yahoo.com
(2) Associate Professor, Dept. of Computer Science & Engg., Acharya Nagarjuna University, Guntur, Andhra Pradesh (India)
E-mail: prof_rsp@gmail.com
(3) Research Scholar, Department Of Computer Science, Krishna University, Machilipatnam, Andhra Pradesh (India)
E-mail: smitha_csc@yahoo.co.in

ABSTRACT

Software reliability assessment is an important aspect to be considered during the software development process. Software reliability is the probability that given software functions work without failure in a specific environment during a specified time. It can be monitored using Statistical Process Control(SPC). SPC is a method of quality control that uses statistical methods to control and monitor a software process and thereby contributes significantly to the improvement of software reliability. Control charts are widely used SPC tools to monitor software quality. The proposed model involves estimation of the parameters of the mean value function and hence these values are used to develop the control charts. The Maximum Likelihood Estimation (MLE) method is used to derive the estimators of the distribution. In this paper we propose a mechanism to monitor software quality based on order statistics of cumulative observations of time domain failure data using mean value function of Burr type III distribution based on Non-Homogeneous Poisson Process.

Keywords- Burr Type III Distribution,, Control Charts, Mean Value Function, Non-Homogeneous Poisson Process, Order Statistics, Probability Limits, Statistical Process Control.

1. INTRODUCTION

Reliability is a primary concern for software developers during software development process to achieve software quality. As the computer technology has developed rapidly, computers are used to control safety and critical systems. High quality software products are mainly needed in those application areas. To determine system quality, the software reliability must be evaluated carefully[1][2].Many Software reliability models have been developed to evaluate reliability using software failures based on assumptions.

The main goal of these models is to fit a theoretical distribution to time-between-failure data, to estimate the time-to-failure based on software test data, to estimate software system reliability and to design a stopping rule to determine the appropriate time to stop testing and to release the software into the market place[3].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 behavior [4].

The software reliability growth model is required to have a good performance in terms of goodness-of-fit, predictability, and so forth. In order to estimate as well as to predict the reliability of software systems, failure data need to be properly measured by various means during software development and operational phases. Numerous SRGMS have been developed during the last three decades and they can provide very useful information about how to improve reliability[2].SRGM can be classified based on the nature of the failure process as Times Between Failures Models, Failure Count Models, Fault Seeding Models, Input Domain Based Models [4][5].

We are more concerned about Time Between Failure models that deals with the random number of software failures in an inter failure time of a developed software. Software reliability growth model based on time domain data is proposed. An important class of SRGM that has been widely studied is Non-Homogenous Poisson Process (NHPP).NHPP models are used in describing failure processes, providing trends such as reliability growth and fault content [19].

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. Various NHPP SRGMs have been proposed 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. Our proposed model uses the assumption of perfect debugging.

The unknown parameters of the mean value function in the NHPP model can be estimated using Maximum Likelihood Estimation (MLE) technique. The

Maximum Likelihood Parameter Estimation is used to determine the parameters that maximize the probability (likelihood) of the sample data [15]. The mean value function we considered is the Cumulative distribution function of Burr Type III with order statistic approach.

To improve the quality of software a quality control method called Statistical Process Control can be used. This method has been widely used for manufacturing processes but recently it has been applied to software processes. There are a few pitfalls in its use of SPC for software[12].The application of SPC mainly involves understanding the process and the limits, eliminating assignable sources of variation, monitoring the process using control charts. The main advantage of using SPC over other quality control methods is it emphasizes on early detection and prevention of problems, rather than correction of problems after they take place. Control charts are the key tools that are used in SPC. The proper use of control charts brings stability and predictability to key processes.

SPC is used in our proposed model and the results exhibited that the failures are detected at early stages. The main objective is to show that the software reliability and thereby the quality of a software process can be improved by applying SPC technique in the software development process [7]. The objective of SPC is to establish and maintain statistical control over a random process. To achieve this objective, it is necessary to detect assignable causes of variation that contaminate the random process. The SPC had proven useful for detecting assignable causes [17].

PROPOSED WORK

A. BURR Type III NHPP Model

NHPP software reliability growth models have been proposed to assess the reliability of software [13].In these models, the number of software failures display the behavior of non-homogenous Poisson Process [3][20].These models consider the debugging process as a counting process characterized by its mean value function. Software reliability can be estimated once the mean value function is determined. Model parameters are usually estimated using Maximum Likelihood method.

The various notations used in NHPP model are:
{N(t), t>0} represents the cumulative number of failures by time ‘t’.
m(t) denotes the expected number of software failures by time ‘t’.
‘a’ represents the expected number of software failures eventually detected.
‘b’ denotes the failure detection rate.
λ (t) corresponds to intensity function of software failures [3][10][14][25].

The Assumptions of NHPP Model are:

1. A Software system is subject to failures during execution caused by faults remaining in the system.
2. All faults are mutually independent from a failure detection point of view.
3. Failure rate of the software depends on the faults remaining in the system.

4. The number of faults detected at any time is proportional to the remaining number of faults in the software.

Since the expected number of errors remaining in the system at any time is finite, m(t) is bounded, non-decreasing function of 't' with the boundary conditions

$$m(t) = \begin{cases} 0, t = 0 \\ a, t \to \infty \end{cases}$$

For t≥0 N(t) is known to have a Poisson Probability mass function with parameters m(t) i.e.,

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

The behavior of software failure phenomena can be illustrated through N(t) process. Several time domain models exist in the literature which specify that the mean value function m(t) will be varied for each NHPP process.

In this paper we consider the mean value function of Burr Type III software reliability growth model as

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

where t ∈ (0, ∞), *c* and b are shape parameters.

Here, we consider the performance given by the Burr Type III software reliability growth model based on order statistics and whose mean value function is given by

$$m(t) = \left(a\left(1 + (t_i)^{-c}\right)^{-b}\right)^r \qquad (2)$$

Where [m(t)/a] is the cumulative distribution function of Ordered Burr distribution model

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

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

This is considered as Poisson model with mean 'a'.

B. Parameter Estimation Based on Inter Failure Times

The mean value function of Order Burr Type III is given by

$$m(t) = \left(a\left(1 + (t_i)^{-c}\right)^{-b}\right)^{r} \tag{3}$$

The constants a, b and c in the mean value function are called parameters of the proposed model. To assess the software reliability, it is necessary to compute the expressions for finding the values of a, b and c. For doing this, Maximum Likelihood estimation is used whose Log Likelihood function is given by

$$\text{LLF} = \sum_{i=1}^{n} Log[\lambda(t_i)^{r} - m(t_n)^{r} \tag{4}$$

Differentiating m(t) with respect to 't' we get λ (t)

$$\lambda(\text{t}) = \frac{rabc}{(t_i)^{(c+1)} * [1 + (ti)^{-c}]^{(br+1)}} \tag{5}$$

The log likelihood equation to estimate the unknown parameters a, b, c after substituting (3) in (4) is given by

$$\text{LogL} = -[a[1+(t_n)^{-c}]^{-b}]^{r} + \sum_{i=1}^{n}[\log r + \log a + \log b + \log c] +$$

$$\sum_{i-1}^{n}[-(br+1)\log(1 + (t_i)^{-c}) - (c+1)\log(t_i)] \tag{6}$$

Differentiating LogL with respect to 'a' and equating to 0 (i.e., $\frac{\partial \log L}{\partial a} = 0$) we get

$$a^{r} = \frac{n(1+(t_n)^{-c})^{br}}{r} \tag{7}$$

Differentiating LogL with respect to 'b' and equating to 0 (i.e., $\frac{\partial \log L}{\partial b} = 0$) we get

$$\text{g(b)} = \frac{n}{b} + \sum_{i=1}^{n} r\log(1+(t_i)^{-1}) + \frac{n^2(1+(t_n)^{-1})^{br}}{r}\log(1+(t_n)^{-1}) \tag{8}$$

Again Differentiating g(b) with respect to 'b' and equating to 0 (i.e.,

$$\frac{\partial^2 \log L}{\partial b^2} = 0)$$

$$g'(b) = \frac{-n}{b^2} + n^2 (1 + (t_n)^{-1})^{br} . \log^2 (1 + (t_n)^{-1}) \quad (9)$$

Differentiating LogL with respect to 'c' and equating to 0 (i.e., $\frac{\partial \log L}{\partial c} = 0)$ we get

$$g(c) = \frac{n}{c} + \sum_{i=1}^{n} (\frac{(r+1)(t_i) - c}{1 + (t_i) - c} - 1) \log t_i - \frac{n(t_n) - c \log t_n}{(1 + (t_n)^{-c})} \quad (10)$$

Again Differentiating g(c) with respect to 'c' and equating to 0 (i.e.,

$$\frac{\partial^2 \log L}{\partial c^2} = 0)$$

we get

$$g'(c) = \frac{-n}{c^2} + \sum_{i=1}^{n} \frac{(r+1)(\log t_i)^2 (t_i)^{-c}}{(1 + (t_i)^{-c})^2} + \frac{n \log (t_n)^2 (t_n)^{-c}}{(1 + (t_n)^{-c})^2} \quad (11)$$

The parameters 'b' and 'c' are estimated by iterative Newton-Raphson Method using

$$b_{n+1} = b_n - \frac{g(b_n)}{g'(b_n)} \quad (12)$$

$$c_{n+1} = c_n - \frac{g(c_n)}{g'(c_n)} \quad (13)$$

Order Statistics

Order Statistics can be used in several applications like data compression, survival analysis, Study of Reliability and many others [12]. Let X denote a continuous random variable with probability density function f(x) and cumulative distribution function F(x), and let (X1, X2, …, Xn) denote a random sample of size n drawn on X. The original sample observations may be unordered with respect to magnitude. A transformation is required to produce a corresponding ordered sample. Let (X(1), X(2), …, X(n)) denote the ordered random sample such that X(1) < X(2) < … < X(n);

then (X(1), X(2), …, X(n)) are collectively known as the order statistics derived from the parent X. The various distributional characteristics can be known from Balakrishnan and Cohen [12].

The inter-failure time data represent the time lapse between every two consecutive failures. On the other hand if a reasonable waiting time for failures is not a serious problem, we can group the inter-failure time data into non overlapping successive sub groups of size 4 or 5 and add the failure times with in each sub group.

For instance if a data of 100 inter-failure times are available we can group them into 20 disjoint subgroups of size 5. The sum total in each subgroup would denote the time lapse between every 5^{th} order statistic in a sample of size 5. In general for inter-failure data of size 'n', if r (any natural number) less than 'n' and preferably a factor n, we can conveniently divide the data into 'k' disjoint subgroups (k=n/r) and the cumulative total in each subgroup indicate the time between every r^{th} failure. The probability distribution of such a time lapse would be that of the ordered statistic in a subgroup of size r, which would be equal to power of the distribution function of the original variable (m(t)).

The whole process involves the mathematical model of the mean value function and knowledge about its parameters. If the parameters are known they can be taken as they are for the further analysis, if the parameters are not known they have to be estimated using a sample data by any admissible, efficient method of estimation. This is essential because the control limits depend on mean value function, which in turn depends on the parameters. If software failures are quite frequent, keeping track of inter-failure is tedious. If failures are more frequent order statistics are preferable [12].

C. Monitoring the time between failures using control chart

Software process monitoring is an essential activity that has to be performed during software process improvement. Monitoring involves measuring a quantifiable characteristic of software process over time and detecting out anomalies. A process must be characterized before it is monitored, for example by using upper and lower threshold values for process performance limits. When the observed performance falls outside these limits one can understand that there is something wrong in the process. Statistical process control is an time series analysis technique that has been effective in manufacturing and recently used in software contexts. It uses control charts as a tool to establish operational limits for acceptable process variation [13].

Control charts are an essential tool used for continuous quality control. Control charts monitor processes to show how the process is performing and how the process and capabilities are affected by changes to the process. This information is then used to make quality improvements. Control charts are also used to determine the capability of the process. These charts have data points that are either averages of subgroup measurements or individual measurements plotted on the x/y axis and joined by a line. Time is always on the x-axis. These charts have an indicator of the process performances as average line, Upper Control Limit (UCL), Lower Control Limit (LCL).

Control charts are mainly classified as attribute charts and variable charts. Attribute Control Charts are used to monitor an organization‘s progress at removing defects that are inherently present in a process. Attribute charts are based on data that can be grouped and counted as present or not. Attribute charts are also called count charts and attribute data is also known as discrete data. Examples of attribute charts are p-charts, np-chart, c-chart, u chart.

Variable charts are based on variable data that can be measured on a continuous scale.Variables Control Charts monitor process parameters or product features. A variable‘s measurement can indicate a significant change in process performance without producing a non-conformance. Variables Control Charts are more sensitive to change and are more efficient than Attribute Control Charts. Two primary statistics are measured and plotted on a Variables Control Chart: central tendency and process dispersion. Examples of variable charts are X-bar, R charts and multivariate charts. We have named the control chart as ***Failures Control Chart*** in this paper. The said control chart helps to assess the software failure phenomena on the basis of the given inter-failure time data [16].

D. Distribution of Time Between Failures

For a software system during normal operation, failures are random events caused by, for example, problem in design or analysis and in some cases insufficient testing of software. In this paper we applied Burr Type III to time between failures data. This distribution uses cumulative time between failure data for reliability monitoring.

The equation for mean value function of Burr Type III from equation [1] is

$$m(t) = a[1 + t^{-c}]^{-b}$$

Equate the pdf of above m(t) to 0.99865, 0.00135, 0.5 and the respective control limits are given by.

$$T_u = [1 + t^{-c}]^{-b} = 0.99865$$

$$T_c = [1 + t^{-c}]^{-b} = 0.5$$

$$T_l = [1 + t^{-c}]^{-b} = 0.00135$$

These limits are converted to $m(t_u)$, $m(t_c)$and $m(t_l)$ form and are used to find whether the software process is in control or not by placing the points in control charts.

2. DATA ANALYSIS AND RESULTS

The procedure of a failures control chart for failure software process will be illustrated with an example here.

Table 1 shows the time between failures of a software product.

Table:1 Software failure data documented in Lyu(1996)

Failure number	Time Between Failures(hrs)	Failure number	Time Between Failures(hrs)	Failure number	Time Between Failures(hrs)
1	3	47	6	93	2930
2	30	48	79	94	1461
3	113	49	816	95	843
4	81	50	1351	96	12
5	115	51	148	97	261
6	9	52	21	98	1800
7	2	53	233	99	865
8	91	54	134	100	1435
9	112	55	357	101	30
10	15	56	193	102	143
11	138	57	236	103	108
12	50	58	31	104	0
13	77	59	369	105	3110
14	24	60	748	106	1247
15	108	61	0	107	943
16	88	62	232	108	700
17	670	63	330	109	875
18	120	64	365	110	245
19	26	65	1222	111	729
20	114	66	543	112	1897
21	325	67	10	113	447
22	55	68	16	114	386
23	242	69	529	115	446
24	68	70	379	116	122
25	422	71	44	117	990
26	180	72	129	118	948
27	10	73	810	119	1082
28	1146	74	290	120	22
29	600	75	300	121	75
30	15	76	529	122	482
31	36	77	281	123	5509
32	4	78	160	124	100
33	0	79	828	125	10
34	8	80	1011	126	1071
35	227	81	445	127	371
36	65	82	296	128	790
37	176	83	1755	129	6150
38	58	84	1064	130	3321
39	457	85	1783	131	1045
40	300	86	860	132	648

41	97	87	983	133	5485
42	263	88	707	134	1160
43	452	89	33	135	1864
44	255	90	868	136	4116
45	197	91	724		
46	193	92	2323		

Table: 2 Successive Differences of 4th order mean value function (m(t))

Failure Number	4-Order Cumulative	m(t)	Successive Difference of m(t)
1	227	9.063012	0.022075
2	444	9.085086	0.016993
3	759	9.10208	0.010183
4	1056	9.112262	0.01888
5	1986	9.131143	0.008645
6	2676	9.139788	0.014252
7	4434	9.15404	0.003805
8	5089	9.157845	0.001571
9	5389	9.159416	0.004595
10	6380	9.164012	0.004163
11	7447	9.168175	0.001652
12	7922	9.169827	0.006829
13	10258	9.176656	0.002236
314	11175	9.178891	0.003027
15	12559	9.181918	0.001834
16	13486	9.183752	0.003189
17	15277	9.186942	0.001737
18	16358	9.188679	0.002814
19	18287	9.191492	0.002942
20	20567	9.194434	0.003958
21	24127	9.198392	0.004047
22	28460	9.202439	0.00315
23	32408	9.205589	0.003601
24	37654	9.20919	0.002605
25	42015	9.211795	0.000158
26	42296	9.211953	0.003125
27	48296	9.215078	0.001746
28	52042	9.216824	0.000619
29	53443	9.217442	0.001285
30	56485	9.218728	0.002392
31	62651	9.22112	0.000807
32	64893	9.221927	0.00362
33	76057	9.225547	0.003461
34	88682	9.229008	

Table: 3 Successive Differences of 5th order mean value function(m(t))

Failure Number	5-Order Cumulative	m(t)	Successive Difference of m(t)
1	342	5.8967019	0.011044
2	571	5.9077458	0.010984
3	968	5.9187302	0.014332
4	1986	5.9330617	0.008513
5	3098	5.9415749	0.009046
6	5049	5.9506206	0.000963
7	5324	5.9515837	0.003258
8	6380	5.9548418	0.003212
9	7644	5.9580537	0.004849
10	10089	5.9629027	0.001462
11	10982	5.9643647	0.002294
12	12559	5.966659	0.002671
13	14708	5.9693304	0.001603
14	16185	5.9709335	0.001543
15	17758	5.9724762	0.002421
16	20567	5.9748968	0.003752
17	25910	5.9786491	0.002004
18	29361	5.9806533	0.003925
19	37642	5.9845787	0.001713
20	42015	5.9862912	0.001201
21	45406	5.9874919	0.001301
22	49416	5.9887927	0.001162
23	53321	5.9899545	0.000876
24	56485	5.9908304	0.001567
25	62661	5.9923972	0.002558
26	74364	5.994955	0.001898
27	84566	5.9968527	

Table: 4 4Th and 5th order Parameter Estimates and control limits

Order	a	b	c	$m(t_u)$	$m(t_c)$	$m(t_L)$
4	9.485651	0.099997	0.101228	9.472845	4.742826	0.012806
5	6.1561	0.099996	0.106223	6.147789	3.07805	0.008311

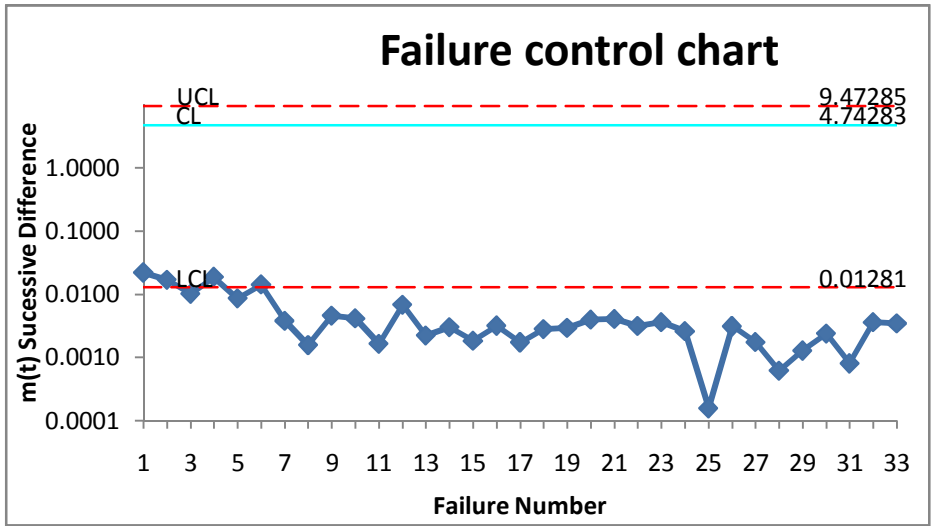

Figure 1: Failure Control Chart of Table 2

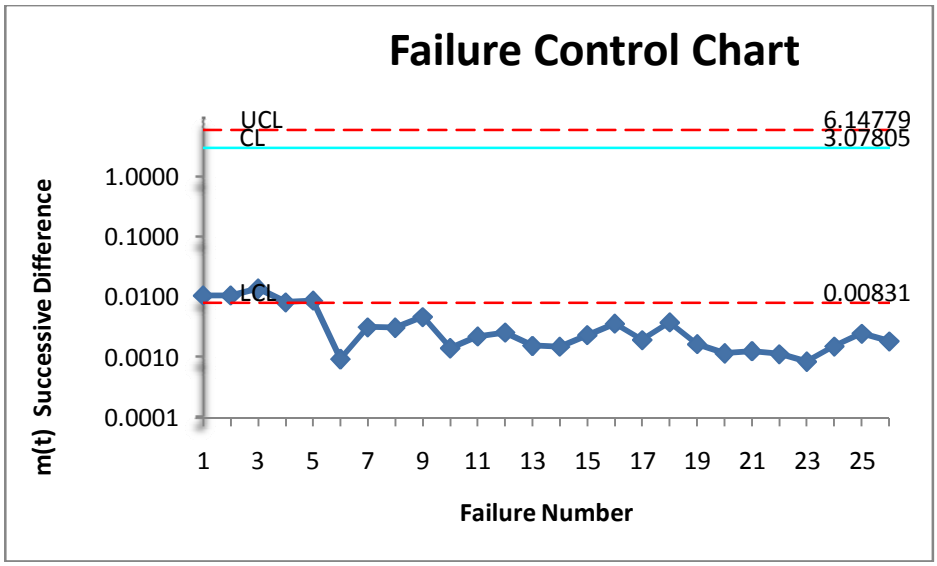

Figure 2: Failure Control Chart of Table 3

3. CONCLUSION

In this paper an SPC technique is applied to sample data with Order Statistic approach. In order to apply this method the parameters are estimated using MLE of Burr Type III ordered function. These parameters are used to calculate the limits and control charts are used on these limits. In the control charts 34 of 4^{th}-order, 27 of 5^{th}-order samples successive differences were plotted through the estimated mean value function against the failure number. 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 3rd point of table-2 for the corresponding m(t) in 4th-order statistics and at 6th point of table-3 for the corresponding m(t) in 5th-order statistics, which is below $m(t_L)$. It indicates that the failure process is detected at an early stage. The early detection of software failure will improve the software quality.

4. REFERENCES

[1] Hong-Wei Liu, Xiao-Zong Yang, Feng Qu, and Yan-Jun Shu, "A General NHPP Software Reliability Growth Model with Fault Removal Efficiency", Iranian Journal of Electrical and Computer Engineering, Vol. 4, No. 2 Summer-Fall 2005.

[2] M. R. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill and IEEE Computer Society, pp. 27-164, New York, 1996.

[3] Richard Lai, Mohit Garg, "A Detailed Study of NHPP Software ReliabilityModels" Journal of Software, Vol. 7, No. 6, June 2012.

[4] Jahir Pasha, S.Ranjitha, Dr. H. N. Suresh, " Certain Reliability Growth Models for Debugging in Software Systems, International Journal of Engineering and Technical Research (IJETR) Volume-2, Issue-4, April 2014

[5] Mohd Razeef and MohsinNazir (2012), ―Software Reliability Growth Models: Overview and Applications, Journal of Emerging Trends in Computing and Information Sciences, VOL. 3, NO. 9, SEP 2012

[6] Maria Teresa Baldassarre, Nicola Boffoli and Danilo Caivano, "Statistical Process Control for Software: Fill the Gap", www.intechopen.com, 2010

[7] Sargut.K.U and Demirors.O "Utilization of Statistical Process Control (SPC) in emergent Software Organization: Pitfalls and Suggestions", Springer, Software Quality Journal, 2014

[8] Mutsumi Komuro; Experiences of Applying SPC Techniques to software development processes; 2006 ACM 1-59593-085-x/06/0005.

[9] Dr.R.Satya Prasad, NGeetha Rani, Prof R.R.L.Kantham, Pareto Type II Based Software Reliability Growth Model, International Journal of Software Engineering, Vol (2), 2011.

[10] Goel. A.L and Okumoto. K., (1979). "A Time-dependent error-detection rate model for software and other performance measures", IEEE Trans. Reliability, vol R-28, Aug, pp 206 - 211.

[11] W. Burr, "Cumulative frequency functions, " *Annals of Mathematical Statistics*, vol. 13, pp. 215–232, 1942.

[12] Balakrishnan.N, Clifford Cohen; Order Statistics and Inference; Academic Press Inc; 1991.

[13] N.Boffoli, G.Bruno, D.Caivano, G.Mastelloni; "Statistical Process Control for Software: a Systematic Approach";2008 ACM 978-1-595933-971-5/08/10.

[14] Pham. H., "*Handbook of Reliability Engineering*", Springer. 2003.

[15] Hoang Pham, "System Software Reliability", Springer, 2006.

[16] M.Xie, T.N. Goh, P. Rajan; Some effective control chart procedures for reliability monitoring; Elsevier science Ltd, Reliability Engineering and system safety 77(2002) 143- 150

[17] K.Ramchand H Rao, R.Satya Prasad, R.R.L.Kantham; Assessing Software Reliability Using SPC – An Order Statistics Approach; IJCSEA Vol.1, No.4, August 2011.

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

[19] Omar Shatnawi, "Discrete Time NHPP Models for Software Reliability Growth Phenomenon", The International Arab Journal of Information Technology, Vol.6, No. 2 April 2009.

[20] Michael R.Lyu 1996a, Handbook of Software Reliability Engineering.

[21] K.Sita Kumari, R.Satya Prasad;Pareto Type II Software Reliability Growth Model – An Order Statistics Approach; IJCST Vol.2, Issue 4, Jul-Aug 2014.

[22] V.K.Gupta, Gaurav Aggarwal; Software Reliability Growth Model; IJARCSSE Vol.4, Issue 1, January 2014.

[23] Dr R.Satya Prasad, N.Geetha Rani, Prof R.R.L Kantham;Pareto Type II Based Software Reliability Growth Model; IJSE Vol.2, Issue 4, 2011.

[24] Hee-cheul Kim., "Assessing Software Reliability based on NHPP using SPC", International Journal of Software Engineering and its Applications, vol.7, No.6 (2013), pp.61-70.

[25] R.Satya Prasad, K.V Murali Mohan, G.Sridevi;Burr Type XII Software Reliability Growth Model;IJCA Volume 108 No-16 December 2014.

[26] Ch.Smitha Chowdary, Dr R.Satya Prasad, K.Sobhana;Burr Type III Software Reliability Growth Model;IOSR-JCE Volume17, Issue 1, Jan-Feb 2015.

[27] Dr R.Satya Prasad, K.Ramchand H Rao, Dr R.R.L.Kantham ; "Software Reliability with SPC" ;International Journal of Computer Science & Emerging Technologies, Volume 2, Issue 2, April 2011.

5. AUTHOR PROFILE

Mrs. K. Sobhana received MCA from Acharya Nagarjuna University in 2005 and M. Tech., (Computer Science & Engineering) from Acharya Nagarjuna University in 2010. Now she is pursuing Ph.D., in Computer Science & Engineering from Krishna University as Part-Time Research Scholar under the guidance of Dr. R.Satya Prasad. Currently, she is working as a Lecturer at Post Graduate Centre of P.B. Siddhartha College of Arts & Science, Vijayawada, AP, India. Her research interest lies in Software Reliability Engineering and Data Mining Artificial Intelligence. She published two research papers in various international journals.

Dr.R Satya Prasad received Ph.D.degree in Computer Science in the Faculty of Engineering in 2007 from Acharya Nagarjuna University, Andhra Pradesh, India. 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 & Engineering, Acharya Nagarjuna University. He performed various academic roles like practical examiner, project adjudicator, external member of board of examiners for various universities and colleges in and around Andhra Pradesh. He received Dr.Abdul Kalam Life Time Achievement Award for his remarkable achievements in the field of Teaching, Research and Publications. His current research is focused on Software engineering, Image processing & Database Management system. He has published several papers in National & International Journals.

Mrs. Ch. Smitha Chowdary received MCA from Kakatiya University in 2003 and M.Tech., (Computer Science & Engineering) from Acharya Nagarjuna University in 2010. Now she is pursuing Ph.D., in Computer Science & Engineering from Krishna University as Part-Time Research Scholar under the guidance of Dr. R.Satya Prasad. Currently, she is working as a Lecturer at Post Graduate Centre of P.B.Siddhartha College of Arts & Science, Vijayawada, AP, India. Her research interest lies in Software Reliability Engineering, Data Warehousing and Data Mining.