G.Krishna Mohan, Dr. Satya Prasad Ravi and Prof. R.R.L Kantam/ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 1, Issue 4, pp. 1486-1493

Software Reliability Using SPC and Weibull Order Statistics

G.Krishna Mohan*, Dr. Satya Prasad Ravi** and Prof. R.R.L Kantam**
*(Reader, Dept. of Computer Science, P.B.Siddhartha college, Vijayawada, Andhrapradesh, India)
*(Associate Professor, Dept. of Computer Science & Engg., Acharya Nagrjuna University, Nagarjuna Nagar, Guntur, Andhrapradesh, India)
*(Professor, Dept. of Statisitcs, Acharya Nagarjuna University, Nagarjuna Nagar, Guntur, Andhrapradesh, India)

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. In this paper we proposed a control mechanism based on order statistics of the cumulative quantity between observations of time domain failure data using mean value function of Weibull distribution, which is based on Non Homogenous Poisson Process (NHPP). The Maximum Likelihood Estimation (MLE) method is used to derive the point estimators of a two-parameter Weibull distribution.

Keywords-Statistical Process Control, Software reliability, Order statistics, Weibull Distribution, Mean Value function, Probability limits, Control Charts.

I. INTRODUCTION

Software Reliability is the application of statistical techniques to data collected during system development and operation to specify, predict, estimate, and assess the reliability of software-based systems. To identify and eliminate 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 [2].

The most popular technique for maintaining process control is control charting. 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 chance causes of variation. On the other hand, when assignable causes are present, 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 [3]. For a process to be in control the control chart should not have any trend or nonrandom pattern.

Statistical Process Control (SPC) is about using control charts to manage software development efforts, in order to effect software process improvement. The practitioner of SPC tracks the variability of the process to be controlled. 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 [4]. 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 item 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 causes may be removed. It can be noted here that 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 is. If the parameters are not known, they have to be estimated using a simple data by any admissible, efficient method of distribution. This is essential because the control limits depend on mean value function which intern depend on 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 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. BACK GROUND

This section presents the theory that underlies Weibull distribution 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: $F(t)$. Where, the mathematical relationship between the pdf and cdf is given by: $f(t)=\frac{d(F(t))}{dt}$. Let 'a' denote the expected number of faults that would be detected given infinite testing time in case of finite failure NHPP models. Then, the mean value function 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].

2.1 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 [5]. 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 [6,7].

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 $\lambda(t)=\frac{dm(t)}{dt}=b(a-m(t))$. 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 general NHPP SRGM $\lambda(t)$ can be expressed as $\lambda(t)=\frac{dm(t)}{dt}=b(t)\left[a(t)-m(t)\right]$. 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.

2.2 Order Statistics

Order statistics deals with properties and applications of ordered random variables and of functions of these variables. The use of order statistics is significant when failures are frequent or inter failure time is less. Let X denote a continuous random variable with probability density function (pdf) f(x) and cumulative distribution function (cdf) 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 [1].

2.3 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: $F(t) = 1 - e^{-(bt)^{\beta}}$. Mean Value Function of the Weibull distribution when $\beta = 2$ i.e. Rayleigh distribution. $m(t) = a\left[1 - e^{-(bt)^2}\right]$. For rth order statistics, the mean value function is expressed as $m(t)^r = \left(a\left[1 - e^{-(bt)^2}\right]\right)^r$. The failure intensity function is given as: $\lambda^r(t) = 2.a^r.b^2.r.t\left(1 - e^{-(bt)^2}\right)^{r-1}.e^{-(bt)^2}$.

2.4 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, $t_1, t_2, \ldots, t_N$. Then the likelihood function [8] may be given by the following product:

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

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

The logarithmic likelihood function is given by: $\Lambda = \ln L = \sum_{i=1}^{N} \ln f(t_i; \theta_1, \theta_2, \ldots, \theta_k)$

Log Likelihood function is: $(LogL = Log\left(\prod_{i=1}^{n} \lambda(t_i)\right))$

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

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:

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

The parameters 'a' and 'b' are estimated using iterative Newton Raphson Method, which is given as

$$x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}$$

III. ILLUSTRATING THE MLE METHOD

3.1 parameter estimation

To estimate 'a' and 'b' , for a sample of n units, first obtain the likelihood function: assuming $\beta = 2$.

The Likelihood function $L = \prod_{i=1}^{n} \lambda^r(t_i)$

Take the natural logarithm on both sides, The Log Likelihood function is given as: $LogL = \log[\prod_{i=1}^{n} \lambda(t_i)]$

$$\log L = \log\left[\prod_{i=1}^{n} 2.a^r.b^2.r.t\left(1-e^{-(bt)^2}\right)^{r-1}.e^{-(bt)^2}\right]$$

$$\log L = \sum_{i=1}^{n}\log\left(2.a^r.b^2.r.t\left(1-e^{-(bt)^2}\right)^{r-1}.e^{-(bt)^2}\right)-\left(a\left[1-e^{-(bt_n)^2}\right]\right)^r$$

Partially differentiating w.r.t 'a' and equating to 0.(i.e. $\frac{\partial \log L}{\partial a}=0$). $a^r = \frac{N}{\left[\left(1-e^{-(bt)^2}\right)\right]^r}$

Partially differentiating w.r.t 'b' and equating to 0.(i.e. $\frac{\partial \log L}{\partial b}=0$).

$$g(b)=\frac{2n}{b}-2b\sum_{i=1}^{n}t_i^2+2(r-1)\sum_{i=1}^{n}\frac{(t_i)^2 be^{-(bt_i)^2}}{\left(1-e^{-(bt_i)^2}\right)}-N=0.$$

Again Partially differentiating w.r.t 'b' and equating to 0.(i.e. $\frac{\partial^2 \log L}{\partial b^2}=0$).

$$g'(b)=-\frac{2n}{b^2}-2\sum_{i=1}^{n}t_i^2+$$
$$2(r-1)\sum_{i=1}^{n}\frac{(t_i)^2\left[e^{-(bt_i)^2}\left(1-e^{-(bt_i)^2}\right)-2b^2t_i^2e^{-(bt_i)^2}\right]}{\left(1-e^{-(bt_i)^2}\right)^2}=0$$

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'.

3.2 Distribution of Time between failures

We compute the software failures process through Mean Value Control chart based on the inter failure data given in Table 1,. We used cumulative time between failures data which is ordered through a transformation for software reliability monitoring using Weibull distribution. The transformation being applied is, the failure data is made into groups of 4, 5 and then cumulated. The inter failure time data represent the time laps 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 subgroups of size 4 or 5 and add the failures times with needs of groups. 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 totals in each subgroup would represent the time laps between every 5th failure. In the theory of statistics such a subtotal is defined as the 5th order statistics in a sample of size 5.

In general for inter failure data of size 'n' if 'r' is any natural number less than n and preferably a factor of 'n' we can conveniently divide the data into 'k' disjoint subgroups(k=n/r) and the cumulative total meets subgroup indicate the time between every rth failure. The probability distribution of such a time laps would be better in the rth order statistic in a subgroup of size 'r'. This would be equal to the rth power of the distribution function of the original variable.

The parameters of the mean value function with the revised distribution function would determine the control limits of a new control chart involving order statistics. Hence they need a separate study. In the present paper we have taken r = 4, 5 and the basic distribution as weibull. Choice of r beyond 5 may create an unduly long waiting time for the occurrence of every rth failure. '$\hat{a}$' 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 [9] shown in table 1. The data is documented in Lyu(1996). There are in total 136 faults reported and the time between failures in seconds. Using 'a' and 'b' values we can compute $m(t)$.

Table:1 Time between failures of a software

F.No	TBF(h)	F.No	TBF(h)	F.No	TBF(h)
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		

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 respectively. 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 & 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 estimated values of 'a' and 'b' and their control limits for both 4th-order and 5th-order statistics are as follows.

Table: 4 Parameter estimates and their control limits of 4 and 5 order

Order	a	b	$m(t_U)$	$m(t_C)$	$m(t_L)$
4	2.414736	0.000049	2.411476	1.207368	0.003260
5	1.933182	0.000058	1.930572	0.966591	0.002610

Table: 2 Successive differences of 4 order mean values

F. No	4-order Cumulatives	m(t)	SD
1	227	0.000299	0.000844
2	444	0.001143	0.002195
3	759	0.003338	0.003119
4	1056	0.006457	0.016303
5	1986	0.022760	0.018403
6	2676	0.041163	0.070175
7	4434	0.111338	0.034240
8	5089	0.145578	0.017062
9	5389	0.162639	0.062191
10	6380	0.224830	0.076215
11	7447	0.301045	0.036726
12	7922	0.337771	0.201342
13	10258	0.539113	0.086454
14	11175	0.625566	0.135687
15	12559	0.761253	0.093126
16	13486	0.854379	0.181537
17	15277	1.035915	0.108689
18	16358	1.144605	0.188294
19	18287	1.332898	0.207280
20	20567	1.540178	0.277691
21	24127	1.817869	0.251501
22	28460	2.069370	0.151410
23	32408	2.220781	0.113705
24	37654	2.334486	0.045404

25	42015	2.379889	0.001927
26	42296	2.381816	0.023993
27	48296	2.405810	0.005306
28	52042	2.411116	0.001081
29	53443	2.412197	0.001401
30	56485	2.413599	0.000943
31	62651	2.414541	0.000097
32	64893	2.414638	0.000096
33	76057	2.414734	0.000002
34	88682	2.414736	

Table:3 Successive differences of 5 order mean values

F.No	5-order Cumulatives	m(t)	SD
1	342	0.000760	0.0013587
2	571	0.002119	0.0039649
3	968	0.006084	0.0193965
4	1986	0.025481	0.0359380
5	3098	0.061419	0.0974547
6	5049	0.158873	0.0169449
7	5324	0.175818	0.0715680
8	6380	0.247386	0.0975878
9	7644	0.344974	0.2155448
10	10089	0.560519	0.0841946
11	10982	0.644713	0.1512690
12	12559	0.795982	0.2034536
13	14708	0.999436	0.1328680
14	16185	1.132304	0.1316687
15	17758	1.263973	0.2033211
16	20567	1.467294	0.2638261
17	25910	1.731120	0.0956919
18	29361	1.826812	0.0899189
19	37642	1.916731	0.0113546
20	42015	1.928085	0.0032165
21	45406	1.931302	0.0013569
22	49416	1.932659	0.0003875
23	53321	1.933046	0.0000935
24	56485	1.933140	0.0000386
25	62661	1.933178	0.0000035
26	74364	1.933182	0.0000000
27	84566	1.933182	

By placing the time between failures of cumulative data shown in tables 2 and 3 on y axis and failure number on x axis and the values of control limits are placed on Mean Value chart, figure 1 and 2 is obtained respectively. The Mean Value charts shows that the 1st,2nd,3rd,25th and 28th to 33rd failure data of 4th-order and 1st and 21st to 26th of 5th -order 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 differences of successive failure data shown in figures 1 and 2 are in stable. No failure data fall outside the $m(t_U)$. It does not indicate any alarming signal.

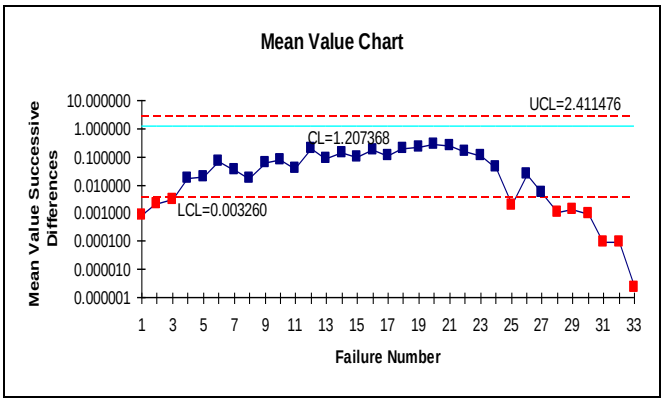

Figure: 1 Mean Value Chart

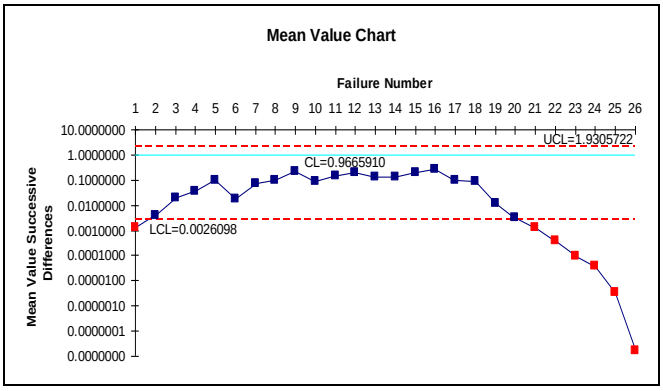

Figure: 2 Mean Value Chart

Conclusion

The 34 of 4th-order, 27 of 5th-order samples successive differences were plotted through the estimated mean value function against the failure number. 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 1st,2nd,3rd,25th and 28th to 33rd point of table-3 for the corresponding $m(t)$ in 4th-order statistics and at 1st and 21st to 26th point of table-4 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 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. This is an alternative method of the tr chart proposed by Xie et. al [11], who has just grouped the samples into each of size 'r'.

REFERENCES

[1] Balakrishnan. N., Clifford Cohen. A., "Order Statistics and Inference", Academic Press, INC. page 13. 1991.
[2] Kimura, M., Yamada, S., Osaki, S.,"Statistical Software reliability prediction and its applicability based on mean time between failures". Mathematical and Computer Modeling Volume 22, Issues 10-12, 1995. 149-155.
[3] Koutras, M.V., Bersimis, S., Maravelakis,P.E., "Statistical process control using shewart control charts with supplementary Runs rules" Springer Science + Business media 9: 2007. 207-224.
[4] MacGregor, J.F., Kourti, T., "Statistical process control of multivariate processes". Control Engineering Practice Volume 3, Issue 3, March 1995, 403-414.
[5] Musa, J.D., Iannino, A., Okumoto, k., "Software Reliability: Measurement Prediction Application". McGraw-Hill, New York. 1987.
[6] Ohba, M., "Software reliability analysis model". IBM J. Res. Develop. 28, 1984. 428-443.
[7] Pham. H., "Software reliability assessment: Imperfect debugging and multiple failure types in software development". EG&G-RAAM-10737; Idaho National Engineering Laboratory. 1993.
[8] Pham. H., "Handbook of Reliability Engineering", Springer. 2003.
[9] Pham. H., "System software reliability", Springer. 2006.
[10] Swapna S. Gokhale and Kishore S.Trivedi, "Log-Logistic Software Reliability Growth Model". The 3rd IEEE International Symposium on High-Assurance Systems Engineering. IEEE Computer Society. 1998.
[11] Xie, M., Goh. T.N., Ranjan.P., "Some effective control chart procedures for reliability monitoring" -Reliability engineering and System Safety 77¸ 2002. 143 -150.

AUTHOR'S 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.

Third Author

R.R.L. Kantam is a professor of statistics at Acharya Nagarjuna University, Guntur-India. He has 31 years of teaching experience in statistics at Under Graduate and Post Graduate programs. As researcher in Statistics, he has successfully guided 8 students for M.Phil in statistics and 5 students for Ph.D in statistics. He has authored 54 research publications appeared in various statistics journals published in India and other countries like US, UK, Germany, Pakistan, Srilanka and Bangladesh. He has been a referee for journal of Applied Statistics (U.K), METRON(Italy), Pakistan Journal of Statistics (Pakistan), IAPQR –Transactions (India), Assam Statistical Review(India) and Gujarat Statistical Review (India). He has been a special speaker in technical sessions of a number of seminars and conferences. His research interest are statistical Inference, Reliability studies, Quality Control Methods and Actuarial Statistics. As a teacher his present teaching areas are Probability theory, Reliability, and Actuarial Statistics. His earlier teaching topics include Statistical Inference, Mathematical Analysis, Operations Research, Econometrics, Statistical Quality Control, Measure theory.