Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Engineering

Elixir Comp. Engg. 79 (2015) 30335-30340

ISSN: 2229-712X

Assessing Burr Type XII software reliability for interval domain data using SPC

R.Satya Prasad[1], B.Rama Devi[2] and G.Sridevi[3]

[1]Department of CSE, Acharya Nagarjuna University, Guntur, India.

[2]Department of CSE, Acharya Nagarjuna University, Guntur, India.

[3]Department of CSE, Nimra Women's College of Engineering, Vijayawada, India.

ARTICLE INFO

Article history:

Received: 1 December 2014;

Received in revised form:

20 January 2015;

Accepted: 7 February 2015;

Keywords

SPC, ML Estimation,

Burr Type XII Model,

Interval Domain Data.

ABSTRACT

Statistical Process Control (SPC) is the best choice to monitor software reliability process. It assists the software development team to identify and actions to be taken during software failure process and hence, assures better software reliability. In this paper we propose a control mechanism based on the cumulative observations of Interval domain data using the mean value function of Burr type XII model, which is Non-Homogenous Poisson Process (NHPP) based. The Maximum Likelihood Estimation (MLE) approach is used to estimate the unknown parameters of the model.

Introduction

Many software reliability models have been proposed in last 40 years to compute the reliability growth of products during software development phase. These models can be of two types i.e. static and dynamic. A static model uses software metrics to estimate the number of defects in the software. A dynamic model uses the past failure discovery rate during software execution over time to estimate the number of failures. Various software reliability growth models (SRGMs) exist to estimate the expected number of total defects (or failures) or the expected number of remaining defects (or failures).

The goal of software engineering is to produce high quality software at low cost. As, human beings are involved in the development of software, there is a possibility of errors in the 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, that the quality of the final product will conform to predefined standards. In any process, regardless of how carefully it is maintained, a certain amount of natural variability will always exist. A process is said to be statistically "in-control" when it operates with only chance causes of variation. On the other hand, when assignable causes are present, then we say that the process is statistically "out-of-control". Control charts should be capable to create an alarm when a shift in the level of one or more parameters of the underlying distribution occurs or a non-random behavior comes into. 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 non random pattern. 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].

Chan et al.,[4] proposed a procedure based on the monitoring of cumulative quantity. This approach has shown to have a number of advantages: it does not involve the choice of a sample size; it raises fewer false alarms; it can be used in any environment; and it can detect further process improvement. Xie et al.,[5] proposed t-chart for reliability monitoring where the control limits are defined in such a manner that the process is considered to be out of control when one failure is less than LCL or greater than UCL. Assuming an acceptable false alarm =0.0027 the control limits were defined. In section 5 of present paper, a method is presented to estimate the parameters and defining the limits. The process control is decided by taking the successive differences of mean values.

Background Theory

This section presents the theory that underlies NHPP models, the SRGMs under consideration and maximum likelihood estimation for complete data. If 't' is a continuous random variable with pdf: $f(t;\theta_1,\theta_2,...,\theta_k)$. Where $\theta_1,\theta_2,...,\theta_k$ 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

Tele:

E-mail addresses: sridevi_gutta@yahoo.com

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 distributive function. The failure intensity function $\lambda(t)$ in case of the finite failure NHPP models is given by: $\lambda(t)=aF'(t)$ [6].

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 [7]. 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 [8][9].

A software system is subjected to failures at random times caused by errors present in the system. Let $\{N(t), t \geq 0\}$ be a counting process representing the cumulative number of failures by time 't', where t is the failure intensity function, which is proportional to the residual fault content. Let $m(t)$ represent the expected number of software failures by time's'. The mean value function $m(t)$ is finite valued, non-decreasing, non-negative and bounded with the boundary conditions.

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

Where 'a' is the expected number of software errors to be eventually detected.

Suppose $N(t)$ is known to have a Poisson probability mass function with parameters $m(t)$ i.e.,

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

Then $N(t)$ is called an NHPP. Thus the stochastic behaviour of software failure phenomena can be described through the $N(t)$ process. Various time domain models have appeared in the literature that describes the stochastic failure process by an NHPP which differ in the mean value function $m(t)$.

Model under consideration: Burr Type XII model

In this paper, we propose to monitor software quality using SPC based on Burr Type XII distribution model. The Burr distribution has a flexible shape and controllable scale and location which makes it appealing to fit to data. It is frequently used to model insurance claim sizes [16]. The mean value function and intensity function of Burr Type XII NHPP model are as follows.

The Cumulative distributive function (CDF) is given by

$$m(t)=\int_0^1\lambda(t)\,dt=a\left[1-\left(1+t^c\right)^{-b}\right]$$
$$=a\,F(t)$$

The Probability Density Function (PDF) of Burr XII distribution are given, respectively by

$$\lambda(t)=a\left(cbt^{c-1}\Big/\left(1+t^c\right)^{b+1}\right)=a\,f(t)$$

Where t>0, a>0, b>0 and c>0 denote the expected number of faults that would be detached given infinite testing time in case of finite failure NHPP models.

Maximum Likelihood Estimation (MLE)

In this section we develop expressions to estimate the parameters of the Burr type XII model based on interval domain data. Parameter estimation is of primary importance in software reliability prediction.

A set of failure data is usually collected in one of two common ways, time domain data and interval domain data. In this paper parameters are estimated from the interval domain data.

The mean value function of Burr type XII model is given by

$$m(t)=a\left[1-\left(1+t^c\right)^{-b}\right] \tag{3.1}$$

In order to have an assessment of the software reliability, a, b and c are to be known or they are to be estimated from software failure data. Expressions are now delivered for estimating 'a', 'b' and 'c' for the Burr type XII model.

Assuming the given data are given for the cumulative number of detected errors n_i in a given time interval $(0, t_i)$ where i=1,2, ….. n and $0 < t_1 < t_2 < \ldots t_n$, then the logarithmic likelihood function (LLF) for interval domain data [10] is given by

$$LogL=\sum_{i=1}^{k}(n_i-n_{i-1})\log\left[m(t_i)-m(t_{i-1})\right]-m(t_k) \tag{3.2}$$

$$LogL=\sum_{i=1}^{k}(n_i-n_{i-1})\log\left\{a\left[1-\left(1+t_i^c\right)^{-b}\right]-a\left[1-\left(1+t_{i-1}^c\right)^{-b}\right]\right\}-a\left[1-\left(1+t_k^c\right)^{-b}\right]$$

$$LogL=\sum_{i=1}^{k}(n_i-n_{i-1})\left\{Loga+\log\left[\left(1+t_{i-1}^c\right)^{-b}-(1+t_i^c)^{-b}\right]\right\}-a+a(1+t_k^c)^{-b} \tag{3.3}$$

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

(i.e., $\frac{\partial Log\,L}{\partial a}=0$)

$$\therefore a=\sum_{i=1}^{k}(n_i-n_{i-1})\frac{(1+t_k^c)^b}{(1+t_k^c)^b-1} \tag{3.4}$$

The parameter 'b' is estimated by iterative Newton Raphson Method using

$b_{n+1}=b_n-\frac{g(b)}{g'(b)}$, Where $g(b)\ and\ \ g'(b)$ are expressed as follows.

$$g(b)=\frac{\partial LogL}{\partial b}=0$$

$$\frac{\partial Log\,L}{\partial b}=g(b)=\sum_{i=1}^{k}(\mathrm{n}_i-n_{i-1})\left\{\left[-\log\left(t_{i-1}+1\right)-\log(t_i+1)+\frac{(t_i+1)^b\log(t_i+1)-(t_{i-1}+1)^b\log(t_{i-1}+1)}{(t_i+1)^b-(t_{i-1}+1)^b}\right]\right.$$
$$\left.+\left[\frac{1}{(t_k+1)^b-1}Log\left(\frac{1}{1+t_k}\right)\right]\right\} \tag{3.5}$$

Again partial differentiating with respect to 'b' and equate to 0 , we get

$$g'(b)=\frac{\partial^2 LogL}{\partial b^2}=0$$

$$\frac{\partial^2 LogL}{\partial b^2}=g'(b)=\sum_{i=1}^{k}(n_i-n_{i-1})\left[\frac{2(t_{i-1}+1)^b(t_i+1)^b\log(t_i+1)\log\left(\frac{t_{i-1}+1}{t_i+1}\right)}{\left[(t_i+1)^b-(t_{i-1}+1)^b\right]^2}\right] \tag{3.6}$$
$$+\sum_{i=1}^{k}(n_i-n_{i-1})\log(1+t_k)\frac{(t_k+1)^b\log(t_k+1)}{\left[(t_k+1)^b-1\right]^2}$$

The parameter 'c' is estimated by iterative Newton Raphson Method using

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

Where $g(c)\ and\ g'(c)$ are expressed as follows.

$$g(c)=\frac{\partial LogL}{\partial c}=0$$

$$\frac{\partial LogL}{\partial c}=g(c)=\sum_{i=1}^{k}(n_i-n_{i-1})\left[-\log t_{i-1}\frac{t_{i-1}^c}{(1+t_{i-1}^c)}-\log t_i\frac{t_i^c}{(1+t_i^c)}+\frac{t_i^c\log t_i-t_{i-1}^c\log t_{i-1}}{(t_i^c-t_{i-1}^c)}\right]-\sum_{i=1}^{k}(\mathrm{n}_i-n_{i-1})\frac{\log t_k}{(1+t_k^c)} \tag{3.7}$$

$$g'(c)=\frac{\partial^2 LogL}{\partial c^2}=0$$

$$\frac{\partial^2 Log\,L}{\partial c^2}=g'(c)=\sum_{i=1}^{k}(n_i-n_{i-1})\left[\begin{array}{l}\left(Log\left(\frac{t_{i-1}}{t_i}\right)\frac{t_i^c.t_{i-1}^c}{(t_i^c-t_{i-1}^c)^2}\left\{\log t_i-\log t_{i-1}\right\}\right)\\ -(\log t_{i-1})^2.\frac{t_{i-1}^c}{(1+t_{i-1}^c)^2}-(\log t_i)^2\frac{t_i^c}{(1+t_{i-1}^c)^2}\end{array}\right]+\sum_{i=1}^{k}(n_i-n_{i-1})(\log t_k)^2.\frac{t_k^c}{(1+t_k^c)^2} \tag{3.8}$$

Interval Domain Datasets

DS #1: Telecommunication System Data

The dataset was reported by Zhang *et al.* (2002) based on system test data for a telecommunication system [17] are shown in Table 4.1.

Table 4.1. DS #1

Week Index	Fault	Week Index	Fault
1	1	8	2
2	1	9	2
3	1	10	4
4	2	11	3
5	3	12	1
6	1	13	1
7	2	14	2

DS #2: Failure Data from Misra (1983)

A set of failure data taken from Misra (1983), given in Table 4.2 consists of the observation time (week) and the number of failures detected per week are errors: major and minor[18].

Table 4.2. DS #2

Week	Minor Errors	Week	Minor Errors	Week	Minor Errors
1	9	13	5	25	2
2	4	14	3	26	3
3	7	15	3	27	6
4	6	16	3	28	3
5	5	17	4	29	1
6	3	18	10	30	1
7	2	19	3	31	4
8	5	20	1	32	3
9	4	21	2	33	2
10	2	22	4	34	11
11	4	23	5	35	9
12	7	24	2		

Results

The performance of the model under consideration is exemplified by applying on the datasets given in tables 4.1 and 4.2.

Calculation of Control Limits

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 [13].

$$T_U = \left[1-\left(1+t^c\right)^{-b}\right] = 0.99865$$

$$T_C = \left[1-\left(1+t^c\right)^{-b}\right] = 0.5$$

$$T_L = \left[1-\left(1+t^c\right)^{-b}\right] = 0.00135$$

The estimated parameters and the control limits are shown in Tables 4.3 and table 4.4.

Table 4.3. Estimated Parameters for the datasets

Dataset	a	b	c
DS #1	26.623982	0.973637	1.066462
DS #2	142.175009	0.985185	1.079388

Table 4.4. Estimated Control Limits

Dataset	$m(t_U)$	$m(t_C)$	$m(t_L)$
DS #1	26.588039	13.311991	0.035942
DS #2	141.983072	71.087504	0.191936

Distribution of Failures

The $m(t)$ values are calculated at each cumulative value of 't'. The successive differences of these values are calculated to plot as a failure control chart along with the calculated control limits which vary with the considered data. The following tables 5.2.1, 5.2.2 and graphs given in figures 5.2.1, 5.2.2 shows the performance of the Burr type XII model in software process control.

Table 5.2.1. Successive Differences of Mean Values DS #1

FN	$m(t)$	SD	FN	$m(t)$	SD	FN	$m(t)$	SD
1	13.06649898	4.693181825	**6**	24.13700545	0.434032852	**11**	25.58644296	0.045203949
2	17.75968081	2.322150723	**7**	24.5710383	0.306969711	**12**	25.63164691	0.041497386
3	20.08183153	2.280123102	**8**	24.87800802	0.228336084	**13**	25.67314429	0.073550535
4	22.36195463	1.483230742	**9**	25.1063441	0.316567045	**14**	25.74669483	
5	23.84518537	0.29182008	**10**	25.42291114	0.163531812			

Table 5.2.2. Successive Differences of Mean Values DS #2

FN	$m(t)$	SD	FN	$m(t)$	SD	FN	$m(t)$	SD
1	129.588386	3.83236519	13	140.458879	0.08193940	25	141.173419	0.02937191
2	133.420751	3.09485021	14	140.540818	0.07467703	26	141.202790	0.05399569
3	136.515601	1.33800818	15	140.615496	0.06833068	27	141.256786	0.02487059
4	137.853609	0.71949381	16	140.683826	0.08254680	28	141.281657	0.00800213
5	138.573103	0.32965740	17	140.766373	0.17210562	29	141.289659	0.00786424
6	138.902761	0.18901777	18	140.938479	0.04392962	30	141.297523	0.03014629
7	139.091778	0.39112151	19	140.982408	0.01397754	31	141.327669	0.02132215
8	139.482900	0.24962793	20	140.996386	0.02702252	32	141.348992	0.01364538
9	139.732528	0.10866962	21	141.023408	0.05059012	33	141.362637	0.06785800
10	139.841197	0.19140361	22	141.073998	0.05747388	34	141.430495	0.04783265
11	140.032601	0.27048270	23	141.131472	0.02139033	35	141.478328	
12	140.303084	0.15579537	24	141.152863	0.02055585			

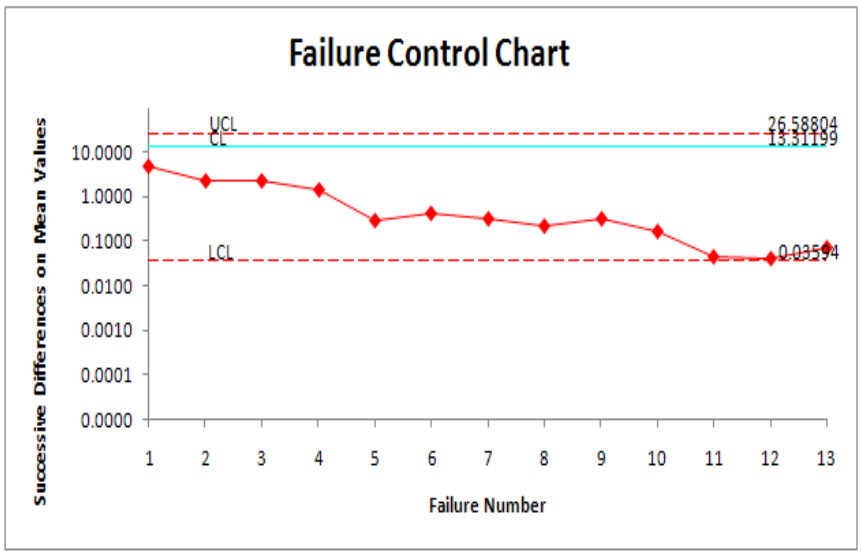

Figure 5.2.1. Failure Control Chart

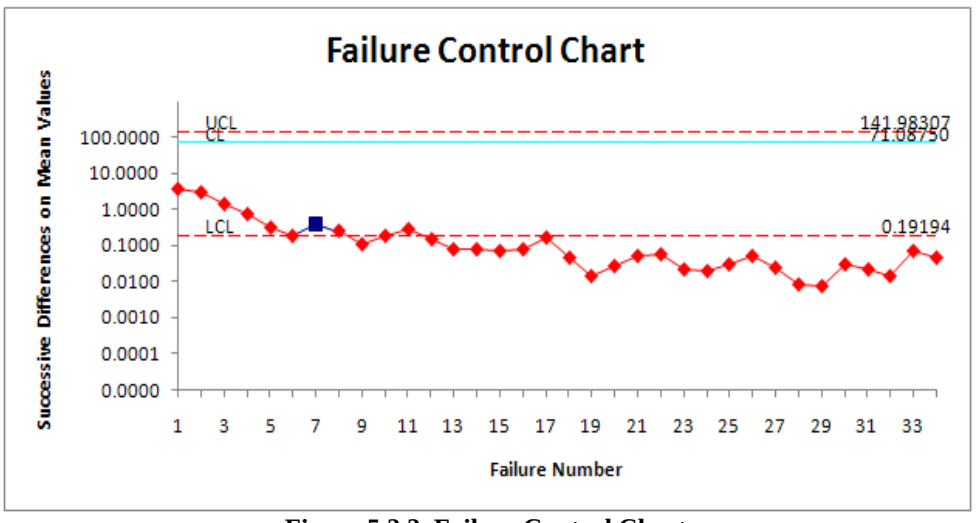

Figure 5.2.2. Failure Control Chart

A point below the control limit $m(t_L)$ indicates an alarm 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. By placing the failure cumulative data shown in tables 5.2.1 and 5.2.2 on y axis and failure number on x axis and the values of the control limits are placed on Control chart, we obtained figures 5.2.1 and 5.2.2. The software quality is determined by detecting failures at an early stage.

Conclusion

The given Interval domain failures data are plotted through the estimated mean value function against the failure serial order. 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 positive recommendation for their use in finding out preferable control processor desirable out of control signal. By observing the control chart it is identified that, for DS #1the failure process out of UCL. For DS #2 the failure situation is detected at 6^{th} point below LCL. Hence our proposed control chart detects out of control situation.

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 ModellingVolume 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 rule s” 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] Chan, L.Y, Xie, M., and Goh. T.N., (2000), “Cumulative quality control charts for monitoring production processes. Int J Prod Res; 38(2):397-408
[5] Xie. M, T.N Goh and P.Ranjan. (2002). “Some effective control chart procedures for reliability monitoring”, Reliability Engineering and System Safety. 77, 143-150.
[6] 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.
[7] Musa, J. D.; Iannino, A.; Okumoto, K. (1987). “Software Reliability - Measurement, Prediction, Application”, New York.
[8] Ohba, M., 1984. “Software reliability analysis model”. IBM J. Res. Develop. 28, 428-443.
[9] Pham. H., 1993. “Software reliability assessment: Imperfect debugging and multiple failure types
[10] Pham. H., 2006. “System software reliability”,Springer.
[11] Musa J.D. (1975). “A Theory of Software reliability and its applications” IEEE Trans. On Software Engineering ,vol SE-1(3)
[12] Ehrlich, W., Prasanna, B., Stampfel, J. and Wu, J. (1993). “Determining the cost of a stop testing decision”, IEEE Software: 33-42.
[13] Gokhale, S.S and Trivedi, K.S., 1998. “Log-Logistic Software Reliability Growth Model”. The 3rd IEEE International Symposium on High-ssurance Systems Engineering. IEEE Computer Society.
[14] R.Satya Prasad, G.Krishna Mohan and Prof. R.R.L. Kantham. “Time Domain based Software Process Control using Weibull Mean Value Function”, International Journal of Computer Applications (IJCA). 18(3):18-21, March 2011.
[15] G.Krishna Mohan, B.Srinivasa Rao and Dr. R.Satya Prasad. ”A Comparative study of Software Reliability models using SPC on ungrouped data”, International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE). Volume 2, Issue 2, February 2012.
[16] 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.
[17] Zhang, X. Jeske, D.R. and Pham, H. (2002). “Calibrating software reliability models when the test environment does not match the user environment”, “Applied stochastic models in Business and Industry, Vol. 18:87-99.
[18] Misra, P. (1983). "Software Reliability Analysis," IBM Systems Journal, vol. 22, pp. 262–270.

Authors Profile

Dr. R. Satya Prasad, received Ph.D.degree in computer science in the faculty of Engineering in 2007 from Acharya Nagarjuna University, Guntur, Andhra Pradesh. He have a satisfactory consistent academic track of record and 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 &Engg., Acharya Nagarjuna University. He has occupied various academic responsibilities like practical examiner, project adjudicator, external member of board of Examiners for various Universities and colleges in and around in Andhra Pradesh. His current research is focused on Software engineering. He has published several papers in National & International Journals.

Mrs. B.Ramadevi, received M.Sc. degree from Acharya Nagarjuna University and M.Tech from Vinayaka Missions University. She is currently pursing Ph.D at Department
of Computer Science and Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. She is currently working as Asst. professor in the Department of Computer Science, Acharya Nagarjuna University, Andhra Pradesh. Her research interests lies in Software Engineering.

Mrs. G. Sridevi, received M.Sc. and M.Tech degree from Acharya Nagarjuna University. She is currently pursing Ph.D at Department of Computer Science and Engineering ,Acharya Nagarjuna University, Guntur, Andhra Pradesh, India. She is currently working as a Vice-Principal and Associate professor in the Department of Computer Science, Nimra Women’s College of Engineering, Jupudi, Ibrahimpatnam, Vijayawada, Andhra Pradesh. Her research interests lies in Data Mining and Software Engineering.