Monitoring Software Failure Process using Half Logistic Distribution

R. Satya Prasad
PhD, Associate Professor
Dept. of CS & Engg.
Acharya Nagarjuna University

K. Sowmya
Assistant Professor
Dept. of CSE
Dhanekula Institute of Engineering & Technology

R. Mahesh
Dept. of CSE
Dhanekula Institute of Engineering & Technology

ABSTRACT
In this paper, Software reliability is the anticipation of operations which are free of error in the software in a stated environment during the detailed time duration. Statistical Process Control can survey the gauging of software failure and thereby devote significantly to the enhancement of software reliability. Such an assessment assists the software development team to pinpoint and diagnose their actions during software failure process and hence, assure superior software reliability. A control mechanism planted on the cumulative observations of interval domain failure data using mean value function of the Half Logistic Distribution (HLD) based on Non Homogeneous Poisson Process (NHPP) is proposed. The maximum likelihood estimation approach is used to estimate the unknown parameters of the model. A new mechanism is coded to analyze the observations instead of using regular control charts.

General Terms
Decision Rule, Software testing, Software failure data, Quality Software.

Keywords
Statistical Process Control (SPC); Software reliability; Probability limits; HLD; Maximum Likelihood Estimation; Failure count data

1. INTRODUCTION
Software reliability assessment is vital to appraise and envision the reliability and pursuance of software system. The gauging of Software reliability process is like biting of more than to be chewed. Over the years, several intellectuals have endorsed the benefits of SPC for software process monitoring. A few others have focal pointed the potential entanglements in its use[1]. Few cogitations equip the guidelines in the practice of SPC by modifying general SPC principles to costume the special requirements of software development [2] [3][4]. It is especially obvious that Burr and Owen supplied crucial ground rules by portraying the techniques currently in prevalent for governing the reliability of software SPC concepts and methods, which are further recycled to access the performance of a software process over time in order to authenticate that the process remains in the stand of statistical control. It guides in finding debatable causes, remote future upgrades in the software process. To pin point and wipe out human errors in software development process and also to enhance software reliability, the Statistical Process Control (SPC) concepts and methods are the best choice.

Software process control is employed to guarantee the quality of the eventual product which will adhere to preordained norms. A proceeding is treated as statistically "in-control" when it behaves with few and far between elements of deviation and when assignable variations are prevalent, then it can be claimed the process is statistically "out-of-control". The widely used convention for studying the control state of process is control charting. SPC provides problem solving analysis to inaugurate tractable process baselines; understand, specific and vigorously enhance process capabilities; and target business areas needing calibration. The initial inference of software missteps will mitigate the software reliability. The appropriation of suitable SPC charts is required for impressive statistical process control implementation and practice. The SPC chart choice is depended on data, circumstances and necessity [5].A from scratch alternative approach is devised, framed and successfully enforced to supplant the practice of classical control charts without comprising the standards of classical control chart.

The prime thrust of the manuscript is to designate and crop up a list of instructions in a disciplined manner with a prospect to guide the practitioner with correct usage of SPC during software process monitoring and propose an alternate control chart mechanism.

2. MODEL SELECTION
A reliability growth model is vital to figure out the prevailing reliability level, the time and resources needed to attain the intended reliability level. In the course of this aspect, reliability assessment is depended upon the study of failure data. The number of failures alighted can be signified as stochastic counting process featured by its mean value function. Aforementioned process can be expressed by a Poisson model. Software reliability can be predicted once with the persisting failure count, and then a value function is measured using this. The afresh charting model is coded and has been used in the software engineering so as to enhance the quality of software products.

A thrust employing SPC in gauging software reliability is conducted by Stieber (1997). The Sequential Probability Ratio Test [6] considers that software failure adapt a Homogenous Poisson Process (HPP). But, more frequently, the cases encountered in existence are non-homogenous. Counting on this, the failure process is audited both on Time domain data and Interval domain data.

There are two main sectors of software reliability models: the deterministic and the probabilistic. The degree of act of the deterministic type is secured by inspecting the program texture and does not get involved with any random event. Two outstanding eminent metric models are: McCabe's Cyclomatic complexity metric (McCabe, 1976) and Halstead's software metric (Halstead, 1977). The probabilistic model serves the failure occurrences and the faults elimination as probabilistic events. The probabilistic software reliability models can be categorized into different groups [7] such as,

Error seeding, Curve fitting, Failure rate, Reliability growth, Markov structure, Time-series and NHPP. Counting processes in reliability engineering are extensively contrived to portray the possibility of events in time, e.g., failures, number of perfect repairs, etc. The straightforward counting process is a Poisson process. Poisson-type models consider that the count of the failures obtained with in distinct time intervals is separate.

(1) Homogeneous Poisson Process (HPP): with the same rate of failure.
(2) Non-Homogeneous Poisson process (NHPP): with a varying rate of failure

NHPP type of software reliability models and methods for predicting software reliability are under limelight at present.

2.1 Interval Domain /Failure count models

In Failure count models, the variable in focal point is the failure count noticed in a specified time interval and this is designed with regard to a Poisson process. As faults are removed from the system it is expected that the observed number of failures per unit time will diminish. Here, the time metric can be calendar time or CPU time. The time intervals are fixed and recorded failure count in each interval is treated as a random variable. The pioneering models of this category are due to Goel and Okumoto (1979), Yamada et al., (1983) and Musa and Okumoto (1984). The key assumptions are given as follows [7]

Testing intervals are independent of each other. Testing with in intervals is reasonably homogeneous. Number of faults detected during non-overlapping intervals is independent of each other.

Due to the data essentials for both time domain and interval domain models, considerable testing need to be performed to estimate the parameters. There are several functions of fundamental importance in modern reliability engineering [8]. The first and foremost fundamental function of importance is the density function. For a continuous variable, the density function is denoted by f(t), gives the relative frequency with which the t-values occur. Characteristic of these density functions is the fact that $\int_D f(t)dt = 1$ for the continuous case. Here, 'D' denotes the domain of definition or interval of integration. All other functions considered depend on the density function and its characteristics.

The second important function from the estimation and interpretation standpoint is the cumulative density function and is denoted by F (t). Which is given as F (t) $= \int_{t_0}^{t} f(t)dt$. Where, $'t_0$ $'$is the lower limit of domain 'D'.

2.2.Half Logistic Distribution

The estimation of parameters for the Half Logistic Software Reliability Growth Model (SRGM)[14] introduced can be adapted for software failure count data in the pattern of interval domain failure data of a software product. Occasionally data would be at hand in the form of number of failures encountered in a span of time. The maximization of such inference would give us better and apt results.[7] Maximum Likelihood estimates of the parameters of an SRGM are totally different and needs a separate treatment.

$$\frac{\partial \log L}{\partial a} = 0 \,, \frac{\partial \log L}{\partial b} = 0 \; \frac{\partial^2 \log L}{\partial b^2} = 0$$

The equation for mean value function m(t) and intensity function λ(t) of HLD[9] is given by

$$m(t) = \frac{a(1-e^{-bt})}{(1+e^{-bt})},\ a>0,\ b>0, t\geq 0 \qquad (1)$$

$$\lambda(t) = \frac{2ab\,e^{-bt}}{(1+e^{-bt})^2},\ a>0,\ b>0, t\geq 0 \qquad (2)$$

In equations (1) and (2) 'a' indicates number of errors and 'b' indicates error rate.

2.3.MLE (Maximum Likelihood) Parameter Estimation

Assessment of parameters is very influential in predicting the software reliability. Upon concluding the analytical solution for m(t) for the specific model, the Maximum Likelihood Estimate (MLE) technique is enforced for attaining the parameter estimation. The crucial intention of Maximum Likelihood parameter Estimation is to resolve the parameters that magnify the probability of the fragment data. The MLE is deliberated as vigorous, robustious and mathematically fierce. They yield estimators with good statistical factors. In the outline analysis, MLE methods are resilient, versatile and can be employed to distinct models and data categories. Accomplishing to present day's computer

capability, the mathematical intensity is not a considerable hurdle.

The constants 'a', 'b' surfacing in the mean value function also appear in NHPP, through the intensity function to materialize error detection rate and in various other expressions are treated as parameters of the model. To assess the software reliability, the unknown parameters 'a' and' b' are to be treasured and they are to be predicted using the failure data of the software fragment data.

For a detail, let 'n' be the time instances where the first, second, third..., kth faults in the software are encountered. It can be consolidated as, if T_k is the total time to the kth failure, tk is an observation of random variable Tk and 'n' such similar failures are successively recorded. The combined probability of such failure time grasps t1, t2,…..tn is given by the Likelihood function as

$$L = e^{-m(t_n)} . \prod_{K=1}^{n} m'(t_k) \qquad (3)$$

The logarithmic application on the equation (3) would result a log likelihood function and is given in equation (4).

$$LogL = \sum_{i=1}^{k} \left[(n_i - n_{i-1}) . \log\left(m(t_i) - m(t_{i-1}) \right) \right] - m(t_k) \qquad (4)$$

The Maximum Likelihood Estimators (MLEs) is featured to maximize L and estimate the values of 'a' and 'b'. The process to maximize is by applying partial derivation with respective to the unknown variables and equate to zero to obtain a close form for the required variable. If the closed form is not destined, then the variable can be estimated using Newton Raphson Method. Subsequently 'a' and 'b' would be solutions of the equations.

Implanting the equations for m(t), λ(t) given by (1) and (2) in equation (4) and executing the aforementioned process and with the aid of few combined simplifications, and obtain a closure form for variable 'a' in terms of 'b'.

$$a = (n_k - n_0)\left(\frac{1+e^{-bt_k}}{1-e^{-bk}}\right) \tag{5}$$

$$g(b)= (n_k - n_0)\sum_{i=1}^{k} \frac{\left[\left(\frac{2.t_i e^{-bt_i}}{\left(1+e^{-bt_i}\right)^2}\right)-\left(\frac{2.t_i e^{-bt_{i-1}}}{\left(1+e^{-bt_{i-1}}\right)^2}\right)\right]}{\left[\left(\frac{1-e^{-bt_i}}{1+e^{-bt_i}}\right)-\left(\frac{1-e^{-bt_{i-1}}}{1+e^{-bt_{i-1}}}\right)\right]} - \frac{2.(n_k-n_0).t_k.e^{-bt_k}}{\left(1-e^{-bt_k}\right)\left(1+e^{-bt_k}\right)} \tag{6}$$

$$g'(b) =(n_k - n_0)\sum_{i=1}^{k} \left\{\left[\frac{\left(\frac{2.t_i^2.e^{-bt_i}\left(1-e^{-bt_i}\right)}{\left(1+e^{-bt_i}\right)^2}\right)-\left(\frac{2.t_{i-1}^2.e^{-bt_{i-1}}\left(1-e^{-bt_{i-1}}\right)}{(1+e^{-bt_{i-1}})^2}\right)\left(\frac{1-e^{-bt_i}}{1+e^{-bt_i}}-\frac{1-e^{-bt_{i-1}}}{1+e^{-bt_{i-1}}}\right)-\left(\frac{2.t_i.e^{-bt_i}}{(1+e^{-bt_i})^2}-\frac{1-e^{-bt_{i-1}}}{1+e^{-bt_{i-1}}}\right)\left(\frac{2.t_i.e^{-bt_i}}{(1+e^{-bt_i})^2}-\frac{2.t_i.e^{-bt_i}}{(1+e^{-bt_i})^2}\right)}{\left(\frac{1-e^{-bt_i}}{1+e^{-bt_i}}-\frac{1-e^{-bt_{i-1}}}{1+e^{-bt_{i-1}}}\right)^2}\right]\right\}$$

$$+ \frac{2.\left(n_k-n_0\right).t_k^2.e^{-bt_k}.(1+e^{-bt_k})}{(1-e^{-bt_k})^2(1+e^{-bt_k})^2} \tag{7}$$

Newton-Raphson method is utilized for attaining ‘b’ value and further it can be substituted in equation (5) to get value of ‘a’

3. CONTROL LIMITS

The control limits need to be stated in a matter of course that the process is scrutinized to be dissipated off balance immediately upon the time of marking exactly one failure is less than LCL or greater than UCL. The main aspiration here is to invigilate the failure process and unmask any deviation in the intensity parameter. Despite the process is proper governance, there might occur an offhand situation sometimes, and it is treated as a false alarm. The causes of the false alarm are diverse and not considered here. The continuous deviations help to determine the accessibilities of the software.

Accustomed to the data readings and the fragment capacity and using equations (5), (6), (7), the parameters number of errors and error rate are enumerated by working with the prominent NR method. A program coded in C#.Net [15] is used for this purpose. The equation for mean value function of HLD is given by (1) is used for obtaining control limits as follows [29].

$$m(t) =\frac{a\left(1-e^{-bt}\right)}{\left(1+e^{-bt}\right)}$$

Delete the term ‘a’ from the mean value function. Equating the remaining function successively to 0.99865, 0.00135, 0.5 and solve ‘t’, for HLD, in order to get the usual 3 sigma corresponding control limits, central line.

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

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

$$\Rightarrow e^{-0.95bt} = 0.0512825$$

$$\Rightarrow bt = \log(0.000675456)$$

$$\Rightarrow t = {}^{1}/_{b}\,(7.300122639) = t_U \tag{8}$$

It gives

$$t = 1/b\,(0.002700002) = t_L \tag{9}$$

$$t = 1/b\,(1.098612289) = t_C \tag{10}$$

The afore mentioned control limits are further used to compute specific limits UCL, LCL and CL by substituting the tU, tC and tL in m(t) and which results m(tU), m(tC) and m(tL) respectively. These mean time specifics are distinguished as UCL, CL and LCL consequently. The observations which are noticed above UCL and LCL are alarm signals. A point below the m(tL) that is LCL is an inkling of more desirable quality of software. The observations inward the control limits express durable process

4. DESIGNED CONTROL CHARTS

Walter A. Shewhart devised control charts during his contributions to Bell Labs in the 1920s. These control charts are also acknowledged as Shewhart charts (Nelson, 1984) or process behavior charts. A control chart is a precise category of run charts that grant extensive adjustments to be distinct from the regular usage of the process. They thus clearly distinct the required tailored special variation from the common. These are indeed graphical tools and thus extend the users to clearly picture the nature and quantity of deviation pertaining in a system. A control chart comprises few notable features like points representing the measurements of a quality characteristic, a CL is drawn at the value of the mean of the statistic and UCL, LCL that indicate the threshold at which the process output is considered statistically 'unlikely'. These features of control charts are incubated and a new control charts mechanism is code using C#.Net

```
private void calculations()
    {
      gettestdata();
      int i = 0;
      double g1, g2, a;
      List<double> b = new List<double>();
      b.Add(Convert.ToDouble(txtbxSeed_Val.Text));
      i = -1;
      do
      {
        i = i + 1;
        g1 = g(b[i]);
```

```
            g2 = gdash(b[i]);
            b.Add(b[i] - (g1 / g2));
        } while (Math.Abs(b[i + 1] - b[i]) >= 0.00001);
        txtbxFinal_b_val.Text = b[i + 1].ToString();
        a = (n[k-1] - n[0]) *
          (
            ((double)1 + Math.Exp(-b[i + 1] * t[k - 1]))
            /
            (1 - Math.Exp(-b[i + 1] * t[k - 1]))
          );
        txtbx_a_Value.Text = a.ToString();


    }
    private double g(double b)
    {
        int i;
        double sum = (double)0.0, cons = (double)0.0, g_val;
        double p, q, r, s;

        cons = (-2 * (n[k - 1] - n[0]) * t[k - 1] * Math.Exp(-b * t[k - 1]))
            /
            (1 - Math.Exp(-2 * b * t[k - 1]));
        for (i = 1; i < k; i++)
        {

            p = ((2 * t[i] * Math.Exp(-b * t[i])) / (sqr(1 + Math.Exp(-b * t[i]))));
            q = ((2 * t[i - 1] * Math.Exp(-b * t[i - 1])) / (sqr(1 + Math.Exp(-b * t[i - 1]))));
            r = (1 - Math.Exp(-b * t[i])) / (1 + Math.Exp(-b * t[i]));
            s = (1 - Math.Exp(-b * t[i - 1])) / (1 + Math.Exp(-b * t[i - 1]));
            sum = sum + ((p - q) / (r - s));
        }
        g_val = (n[k - 1] - n[0]) * sum + cons;
        return g_val;
    }
    double gdash(double b)
    {
        int i;
        double gdash_val, sum = 0, cons = 0;
        double p, q, r, s, pdash, qdash, rdash, sdash;
        cons = (2 * (n[k - 1] - n[0]) * sqr(t[k - 1]) * Math.Exp(-b * t[k - 1]) * (1 + Math.Exp(-2 * b * t[k - 1])))
            /
            (sqr(1 - Math.Exp(-b * t[k - 1])) * sqr(1 + Math.Exp(-b * t[k - 1])));
        for (i = 1; i < k; i++)
        {
            p = ((2 * t[i] * Math.Exp(-b * t[i])) / (sqr(1 + Math.Exp(-b * t[i]))));
            q = ((2 * t[i - 1] * Math.Exp(-b * t[i - 1])) / (sqr(1 + Math.Exp(-b * t[i - 1]))));
            r = (1 - Math.Exp(-b * t[i])) / (1 + Math.Exp(-b * t[i]));
            s = (1 - Math.Exp(-b * t[i - 1])) / (1 + Math.Exp(-b * t[i - 1]));
            rdash = p;
            sdash = q;
            pdash = ((2 * sqr(t[i]) * Math.Exp(-b * t[i])) * (-1 * (1 + Math.Exp(-b * t[i])) + 2 * Math.Exp(-b * t[i]))) / (sqr((1 + Math.Exp(-b * t[i]))) * (1 + Math.Exp(-b * t[i])));
            qdash = ((2 * sqr(t[i - 1]) * Math.Exp(-b * t[i - 1])) * (-1 * (1 + Math.Exp(-b * t[i - 1])) + 2 * Math.Exp(-b * t[i - 1]))) / (sqr((1 + Math.Exp(-b * t[i - 1]))) * (1 + Math.Exp(-b * t[i - 1])));
            sum += (((pdash - qdash) * (r - s)) - ((rdash - sdash) * (p - q))) / sqr((r - s));
        }
        gdash_val = (n[k - 1] - n[0]) * sum + cons;
        return gdash_val;
    }

    double sqr(double x)
    {
        return (x * x);
    }
    private void btnGenChart_Click(object sender, EventArgs e)
    {
  chart1.Series[0].Points.Clear();
  chart1.Series[1].Points.Clear();
  chart1.Series[2].Points.Clear();
  chart1.Series[3].Points.Clear();
        chart1.Visible = true;
  chart1.ChartAreas[0].AxisX.Minimum = 0;
  chart1.ChartAreas[0].AxisY.Minimum = 0.001;
  chart1.ChartAreas["ChartArea1"].AxisX.MajorGrid.Enabled = false;
```

```
chart1.ChartAreas["ChartArea1"].AxisY.MajorGrid.Enabled = false;
stringStrQuery;
stringconnetionString = null;
connetionString = HalfLogisticDistribution.Properties.Settings.Default.HalfLDis_DBConnectionString;
SqlConnection con = new SqlConnection(connetionString);
SqlCommandcmd = new SqlCommand();

doublem_t_u, m_t_l, m_t_c;

        List<double>succ_m_t = new List<double>();
        List<double>cumm_fd = new List<double>();
try     {
StrQuery = "select m_t_u,m_t_l,m_t_c from TestTableNames where TestDataName='" + cmbBxTestTableNames.SelectedItem.ToString().TrimEnd() + "'";
cmd.CommandText = StrQuery;
cmd.Connection = con;
con.Open();
SqlDataReaderdr;
dr = cmd.ExecuteReader();
dr.Read();
m_t_u = Convert.ToDouble(dr["m_t_u"]);
m_t_l = Convert.ToDouble(dr["m_t_l"]);
m_t_c = Convert.ToDouble(dr["m_t_c"]);
dr.Close();
StrQuery = "select cumm_fd,s_m_t from " + cmbBxTestTableNames.SelectedItem.ToString().TrimEnd();
cmd.CommandText = StrQuery;
dr = cmd.ExecuteReader();
while (dr.Read())
        {
cumm_fd.Add(Convert.ToDouble(dr[0]));
if(!DBNull.Value.Equals(dr[1]))
succ_m_t.Add(Convert.ToDouble(dr[1]));
        }
dr.Close();
chart1.Series["m(tl)"].Points.AddXY(0, m_t_l);
chart1.Series["m(tc)"].Points.AddXY(0, m_t_c);
chart1.Series["m(tu)"].Points.AddXY(0, m_t_u);
for (inti = 0; i< cumm_fd.Count-1;i++ )
        {
chart1.Series["m(tl)"].Points.AddXY(cumm_fd[i], m_t_l);
chart1.Series["m(tc)"].Points.AddXY(cumm_fd[i], m_t_c);
chart1.Series["m(tu)"].Points.AddXY(cumm_fd[i], m_t_u);
chart1.Series["Successive Diff of m(t)"].Points.AddXY(cumm_fd[i], succ_m_t[i]);
        }
    }
catch (Exception ex)    {
MessageBox.Show(ex.Message.ToString() + "Can not complete the operation. Try again!! ");
    }
finally     {
con.Close();
    }
  }
```

5. RESULTS

The procedure of monitoring software process with failure domain data through failure control chart will be illustrated with examples. With the stated process the appropriate values for parameters ‘a’ and ‘b’ for an apt seed value are obtained. Using these values, m(t)values and thus their successive differences can also be obtained. Here two real time datasets (Table 1 & Table 2) [10] which contain failure data of two distinct projects are considered and demonstrated. Table 1& Table 2 shows the time between failures of a software product [10]. Table 3 contains the estimated parameter values for both the datasets (1 &2) and their respective control limits.

6. CONCLUSION

A control mechanism is proposed for the cumulative observations of interval domain failure data using mean value function of the HLD based on Non Homogeneous Poisson Process. The (MLE) Maximum Likelihood Estimation approach is used to estimate the unknown parameters of the model. A new mechanism is coded to analyze the observations instead of using regular control charts.

Table1: Successive differences of mean values of Dataset 1

TT(day)	CF	m(t)	Successive differences	TT(day)	CF	m(t)	Successive differences
1	2	3.32911444	3.24619174	12	25	33.4736671	0.850365341
2	3	6.63027763	3.16599679	13	27	35.30622	1.185042
3	4	9.87647	3.063118	14	31	36.981926	0.21476762
4	5	13.0424662	5.74283743	15	32	38.5077	0.846567
5	7	16.1055851	5.14301157	16	38	39.8915977	0.08814403
6	9	19.046257	4.487447	17	39	41.1424	0.205616966
7	11	21.8484211	1.99478745	18	42	42.2693253	0.0526794866
8	12	24.4997063	9.808073	19	43	43.28174	0.122744448
9	19	26.9914322	2.44186473	20	46	44.1889458	0.0314152
10	22	29.3184566	0.6443972	21	47	45	
11	23	31.47888	1.08155239				

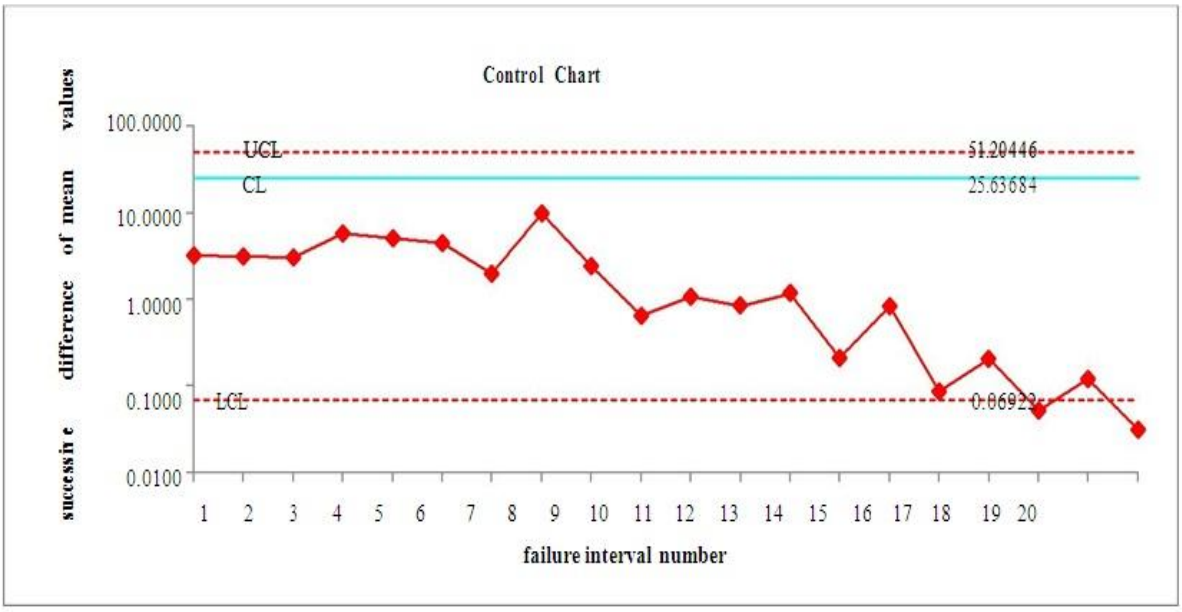

Figure 1: Mean Value Chart for Dataset 1

In figure 1, the control chart, at point 18 of x-axis, highlights an alarm and consequently at 20 and 22 it further indicates the failure of the process. The durability of the process is at stake.

Table 2: Successive differences of mean values of Dataset 2

TT(day)	CF	m(t)	Successive differences	TT(day)	CF	m(t)	Successive differences
1	1	2.73077273	2.681832	8	13	24.27901	1.33174026
2	2	5.412605	2.58739734	9	15	25.61075	1.61538172

3	3	8.000002	4.743524	10	19	27.2261314	0.6404373
4	5	12.7435265	5.71788836	11	22	27.86657	0.146042377
5	8	18.4614143	1.50995469	12	23	28.0126114	0.121249989
6	9	19.9713688	2.47105312	13	24	28.1338615	0.183970675
7	11	22.4424229	1.83658755	14	26	28.3178329	

In figure 2, the control chart indicates the reliability of the project as the mean values are within the limits.

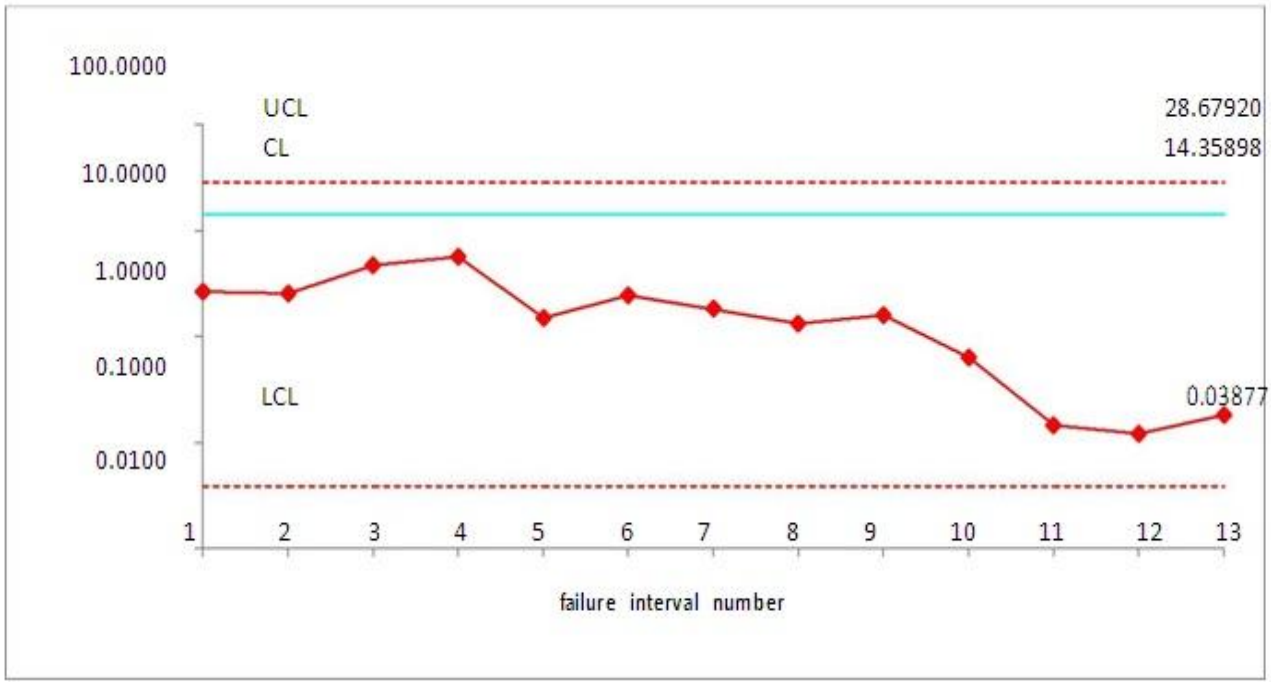

Figure 2: Mean Value Chart for Data Set 2

Table 3: Parameter estimates and Control limits of Interval domain data

DataSet	No. of Samples	Estimated Parameters		Control Limits		
		a	b	UCL	CL	LCL
1	**21**	51.27368	0.130039	51.204464	25.63684	0.069219
2	**14**	28.717968	0.190755	28.6792	14.358984	0.038769

7. REFERENCES

[1] N. Boffoli, G. Bruno, D. Cavivano, G. Mastelloni; Statistical process control for Software: a systematic approach; 2008 ACM 978-1-595933-971-5/08/10.

[2] K. U. Sargut, O. Demirors; Utilization of statistical process control (SPC) in emergent software organizations: Pitfallsand suggestions; Springer Science + Business media Inc. 2006.

[3] Burr,A. and Owen ,M.1996. Statistical Methods for Software quality .Thomson publishing Company.ISBN 1-85032-171-X.

[4] Carleton, A.D. and Florac, A.W. 1999. Statistically controlling the Software process. The 99 SEI Software Engineering Symposimn, Software Engineering Institute, Carnegie Mellon University.

[5] MacGregor, J.F., Kourti, T., 1995. "Statistical process control of multivariate processes". Control Engineering Practice Volume 3, Issue 3, March 1995, Pages 403-414 .

[6] Wald, A.(1947). "Sequential Analysis".Wiley, New York.

[7] Pham. H., 2006."System software reliability", Springer.

[8] E. E. Lewis, 1996 "Introduction to Reliability Engineering" John Wiley & Sons.

[9] Pham.H., 2003. "Handbook Of Reliability Engineering", Springer.

[10] Xie, M., Goh. T.N., Ranjan.P., "Some effective control chart procedures for reliability monitoring" -Reliability engineering and System Safety 77 143 -150, 2002.

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

[12] Ronald P.Anjard;SPC CHART selection process;Pergaman 0026-27(1995)00119-0Elsevier science ltd.

[13] Dr. R Satya Prasad ,K Ramchand H Rao and Dr. R.R. L Kantham (2011)," Software Reliability Measuring using Modified Maximum Likelihood Estimation and SPC" IJCA Journal, Number 7 – Article1

[14] R.Satya Prasad, Half Logistic Software Reliability Growth Model,Ph.D. Thesis,2007

[15] 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.

[16] 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.

[17] 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.

[18] Musa, J.D., Iannino, A., Okumoto, k., 1987. "Software Reliability: Measurement Prediction Application". McGraw-Hill, New York.

[19] Ohba, M., 1984. "Software reliability analysis model". IBM J. Res. Develop. 28, 428-443.

[20] Pham.H., 1993. "Software reliability assessment: Imperfect debugging and multiple failure types in software development". EG&G-RAAM-10737; Idaho National Engineering Laboratory.

[21] Huang, C.Y. and Kuo, S.Y., (2003). "A unified scheme of some Nonhomogenous Poisson process models for software reliability estimation", IEEE Transactions on Software Engineering, 29 (3): 261-269.

[22] ANSI/IEEE, (1991). "Standard Glossary of Software Engineering Terminology", STD-729-1991

[23] Ashoka. M., (2010). "Sonata software limited Data Set", Bangalore.

[24] Baldassarre, M.T., Boffoli, N., Caivano, D. and Visaggio, G., (2004). "Managing software process improvement (SPI) through Statistical Process Control (SPC)".In Proc.Of PROFES (kansai city Japan, 5-8. LNCS Springer, pp. 30-46.

[25] Caivano, D. (2005). "Continuous Software Process Improvement through Statistical Process Control", Proceedings of the European conference of Software Maintanance and Reengineering-CSMR 05, IEEE Computer Society.

[26] https://msdn.microsoft.com/

[27] Card, D., (1994). "Statistical Process Control for Software", IEEE Software, pp. 95-97.

[28] Chang, Y.P. (2001). "Estimation of Parameters for Non-homogeneous Poisson Process: Software Reliability with Change-point Model", Communications in Statistics: Simulation and Computation, 30(3):625–635.

[29] "Reliability Engineering Handbook" By DodsoN/NolaN, CRC Press, 1999.

8. AUTHOR PROFILE

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 135 papers in National & International Journals. So far 20 Ph.D"s awarded under his guidance.

K.Sowmya working as Assistant professor, Department of Computer Science & Engineering, Dhanekula Institute of Engineering & Technology, Vijayawada, Andhra Pradesh. Her research interests include Software Engineering and Cloud Computing.

R.Mahesh pursuing B.Tech in Dhanekula College of Engineering and Technology. Affiliated to JNTUK, Kakinada. His research interest in Data Mining, Software Engineering, CloudComputing.