CHAPTER 6

SPC BASED ON INTERVAL DOMAIN: MLE

6.1 Introduction.

Software reliability assessment is important to evaluate and predict the reliability and performance of software system, since it is the main attribute of software. To identify and eliminate human errors in software development process and also to improve software reliability, the Statistical Process Control concepts and methods are the best choice. SPC concepts and methods are used to monitor the performance of a software process over time in order to verify that the process remains in the state of statistical control. It helps in finding assignable causes, long term improvements in the software process. Software quality and reliability can be achieved by eliminating the causes or improving the software process or its operating procedures (kimura et al., 1995).

The most popular technique for maintaining process control is control charting. The control chart is one of the seven tools for quality control. Software process control is used to secure the quality of the final product which will conform to predefined standards. In any process, regardless of how carefully it is maintained, a certain amount of natural variability will always exist. A process is said to be statistically “in-control” when it operates with only chance causes of variation. On the other hand, when assignable causes are present, then we say that the process is statistically “out-of-control.”

Control charts can be classified into several categories, according to several distinct criteria. Depending on the number of quality characteristics under investigation, charts can be divided into univariate control charts or multivariate control charts. Furthermore, the quality characteristic of interest may be a continuous random variable or alternatively a discrete attribute. Control charts should be capable to create an alarm when a shift in the level of one or more parameters of the underlying distribution or a non-random behavior occurs. Normally, such a situation will be reflected in the control chart by points plotted outside the control limits or by the presence of specific patterns. The most common non-random patterns are cycles, trends, mixtures
and stratification (Koutras, 2007). For a process to be in control the control chart should not have any trend or nonrandom pattern.

SPC is a powerful tool to optimize the amount of information needed for use in making management decisions. Statistical techniques provide an understanding of the business baselines, insights for process improvements, communication of value and results of processes, and active and visible involvement. SPC provides real time analysis to establish controllable process baselines; learn, set, and dynamically improve process capabilities; and focus business areas needing improvement. The early detection of software failures will improve the software reliability. The selection of proper SPC charts is essential to effective statistical process control implementation and use. The SPC chart selection is based on data, situation and need (MacGregor and Kourt, 1995).

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 (Xie et al., 2002).

6.2 ML (Maximum Likelihood) Parameter Estimation.

Parameter estimation is of primary importance in software reliability prediction. Once the analytical solution for $m(t)$ is known for a given model, parameter estimation is achieved by applying a technique of Maximum Likelihood Estimate (MLE). 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 and yields estimators with good statistical properties. Assuming that the data are given for the cumulative number of
detected errors $y_i$ in a given time-interval $(0,t_i)$ where $i = 1, 2, \ldots, n$ and $0 < t_1 < t_2 < \ldots < t_n$ then the log likelihood function (LLF) takes on the following form.

Likelihood function by using $\lambda(t)$ is:

$$L = e^{-\lambda(t)} \prod_{i=1}^{n} \lambda(t_i)$$

The logarithmic likelihood function for interval domain data (Pham, 2006) is given by:

$$\log L = \sum_{i=1}^{n} (y_i - y_{i-1}) \log [m(t_i) - m(t_{i-1})] - m(t_i)$$

### 6.3 Illustrating the MLE Method.

**Parameter estimation**

To estimate ‘a’ and ‘b’, for a sample of $n$ units, first obtain the likelihood function:

The parameter ‘a’ is estimated by taking the partial derivative w.r.t ‘a’ and equating to ‘0’. (i.e $\frac{\partial \log L}{\partial a} = 0$)

$$a = \frac{n_k}{\sum_{i=1}^{k} \left[ \frac{1 - e^{-bt_i}}{1 + e^{-bt_i}} - \frac{1 - e^{-bt_{i-1}}}{1 + e^{-bt_{i-1}}} \right]}$$

The parameter ‘b’ is estimated by iterative Newton Raphson Method using

$$b_{n+1} = b_n - \frac{g(b_n)}{g'(b_n)}.$$ This is substituted in finding ‘a’. where $g(b)$ & $g'(b)$ are expressed as follows. $g(b) = \frac{\partial \log L}{\partial b} = 0$ ; $g'(b) = \frac{\partial^2 \log L}{\partial b^2} = 0$

$$\frac{\partial \log L}{\partial b} = \frac{n_k}{\log(1 + bt_k)} \sum_{i=1}^{k} \frac{(n_i - n_{i-1})}{\log(1 + bt_i) - \log(1 + bt_{i-1})} \left[ \frac{t_i}{1 + bt_i} - \frac{t_{i-1}}{1 + bt_{i-1}} \right]$$
\[
\frac{\partial^2 \log L}{\partial b^2} = \sum_{i=1}^{k} \left( \frac{n_i - n_{i-1}}{\log(1 + bt_i) - \log(1 + bt_{i-1})} \right)^2 \left[ \frac{t_i}{1 + bt_i} - \frac{t_{i-1}}{1 + bt_{i-1}} \right] - \left[ \frac{n_t t_k}{1 + bt_k} \right] \left[ \frac{1}{\log(1 + bt_k)} \right]^2
\]

/*N-R METHOD FOR M.L.Es OF ‘a’&‘b’of Failure Count DATA*/

#include<stdio.h>
#include<conio.h>
#include<math.h>

#define N 21

double g(double b, int s[],int n[]);

double gdash(double b, int s[], int n[]);

main()
{
int i,k;

int s[N]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21} ;
int n[N]={1,1,2,3,5,5,6,8,9,10,11,12,13,14,15,19,19,22,22,23,24,24,24,26} ;
double g1,g2,a,v1=0.0;

double b[N],x;
clrscr();

b[0]=0.002805;
i=-1;
do
{

i=i+1;

g1=g(b[i],s,n);

printf("g1 vale %f",g1);

g2=gdash(b[i],s,n);

printf("g2 value %f",g2);

b[i+1]=b[i]-(g1/g2);

printf("\nb[%d] = %f b[%d] = %f",i,b[i],i+1,b[i+1]);

printf("\n\t\t\t|b[%d] - b[%d]| = %f",i+1,i,fabs(b[i+1]-b[i]));

}while((fabs(b[i+1]-b[i])>=0.0001));

x=b[i+1];

for(i=1;i<N;i++)

v1=v1+(((1-exp(-x*s[i]))/(1+exp(-x*s[i]))) - 
            ((1-exp(-x*s[i-1]))/(1+exp(-x*s[i-1]))));

a=n[N]/v1;

printf("\n\n\nb[%d] = %f is the M.L.E of b = %f a=%f",i+1,x,x,a);

getch();

} /*Function for calclating g(b)*/}

double g(double b, int s[N],int n[N])
{

int i;

double t1=0.0,t2=0.0,g_val;

t1=n[N]/log(1+b*s[N]);
for(i=1;i<N;i++)
{
    t2=t2+(n[i]-n[i-1])/((log(1+b*s[i]))-(log(1+b*s[i-1])));
}
g_val=t1-t2;
return g_val;
}

/*function for calculating g'(b)*/

double gdash(double b,int s[N], int n[N])
{
    int i;
    double t1=0,t2=0,t21=0,t22=0;
    double gdash_val;
    t1=((n[N]*s[N])/(1+b*s[N]))/pow(log(1+b*s[N]),2);
    for(i=1;i<N;i++)
    {
        t21=(n[i]-n[i-1])/pow((log(1+b*s[i]))-(log(1+b*s[i-1])),2);
        t22=(s[i]/(1+b*s[i]))-(s[i-1]/(1+b*s[i-1]));
        t2=t21*t22;
    }
gdash_val=t2-t1;
return gdash_val;
}
6.4 Distribution of failures

Based on the failure data given in Table 1.9.3.1 and 1.9.3.2, we compute the software failures process through Mean Value Control chart. We used cumulative time failures data for software reliability monitoring through SPC using HSRGM.

The MLEs of the parameters for HLSRGGM based on the Test Data, which is of Interval domain data are as follows.

Table 6.4.1: Estimates of a and b for Interval domain

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Estimate of ‘a’</th>
<th>Estimate of ‘b’</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 2</td>
<td>124.619400</td>
<td>0.000280</td>
</tr>
<tr>
<td>Release 3</td>
<td>77.109358</td>
<td>0.000290</td>
</tr>
</tbody>
</table>

‘\( \hat{a} \)’ 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. Using ‘\( a \)’ and ‘\( b \)’ values we can compute \( m(t) \). Now the control limits are calculated by equating the cumulative distribution function to the standard values 0.00135, 0.99865, and 0.5.

These limits are converted to \( m(t_u), m(t_c) \) and \( m(t_l) \) form. They are used to find whether the software process is in control or not by placing the points in failure control chart. 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. The values of control limits are as follows.

Table 6.4.2: Control limits

<table>
<thead>
<tr>
<th>Data Set</th>
<th>( m(t_u) )</th>
<th>( m(t_c) )</th>
<th>( m(t_l) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 2</td>
<td>124.451200</td>
<td>62.309700</td>
<td>0.168236</td>
</tr>
<tr>
<td>Release 3</td>
<td>77.005260</td>
<td>38.554680</td>
<td>0.104098</td>
</tr>
</tbody>
</table>
### Table 6.4.3 Release 2 Successive differences of cumulative mean values

<table>
<thead>
<tr>
<th>TT (day)</th>
<th>CF</th>
<th>( m(t) )</th>
<th>Succ.diff</th>
<th>TT (day)</th>
<th>CF</th>
<th>( m(t) )</th>
<th>Succ.diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>0.22680706</td>
<td>0.087233</td>
<td>11</td>
<td>95</td>
<td>1.65734033</td>
<td>0.087217</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>0.31404023</td>
<td>0.139572</td>
<td>12</td>
<td>100</td>
<td>1.74455765</td>
<td>0.069773</td>
</tr>
<tr>
<td>3</td>
<td>26</td>
<td>0.45361262</td>
<td>0.139571</td>
<td>13</td>
<td>104</td>
<td>1.81433029</td>
<td>0.104657</td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>0.59318387</td>
<td>0.104677</td>
<td>14</td>
<td>110</td>
<td>1.91898709</td>
<td>0.034885</td>
</tr>
<tr>
<td>5</td>
<td>40</td>
<td>0.69786136</td>
<td>0.139568</td>
<td>15</td>
<td>112</td>
<td>1.9538721</td>
<td>0.034885</td>
</tr>
<tr>
<td>6</td>
<td>48</td>
<td>0.83742978</td>
<td>0.226794</td>
<td>16</td>
<td>114</td>
<td>1.9887568</td>
<td>0.052326</td>
</tr>
<tr>
<td>7</td>
<td>61</td>
<td>1.06422382</td>
<td>0.244232</td>
<td>17</td>
<td>117</td>
<td>2.04108326</td>
<td>0.017442</td>
</tr>
<tr>
<td>8</td>
<td>75</td>
<td>1.30845564</td>
<td>0.157001</td>
<td>18</td>
<td>118</td>
<td>2.05852526</td>
<td>0.034884</td>
</tr>
<tr>
<td>9</td>
<td>84</td>
<td>1.46545661</td>
<td>0.087221</td>
<td>19</td>
<td>120</td>
<td>2.09340901</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>89</td>
<td>1.5526774</td>
<td>0.104663</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Figure 6.4.1: Failure control chart of Release 2
Table 6.4.4 Release 3 Successive differences of cumulative mean values

<table>
<thead>
<tr>
<th>TT (day)</th>
<th>CF</th>
<th>m(t)</th>
<th>Succ.diff</th>
<th>TT (day)</th>
<th>CF</th>
<th>m(t)</th>
<th>Succ.diff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>0.06708512</td>
<td>0.033543</td>
<td>7</td>
<td>48</td>
<td>0.53667247</td>
<td>0.067081</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>0.10062766</td>
<td>0.044723</td>
<td>8</td>
<td>54</td>
<td>0.60375393</td>
<td>0.033540</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>0.14535097</td>
<td>0.078266</td>
<td>9</td>
<td>57</td>
<td>0.63729433</td>
<td>0.022360</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>0.22361651</td>
<td>0.089446</td>
<td>10</td>
<td>59</td>
<td>0.65965446</td>
<td>0.011180</td>
</tr>
<tr>
<td>5</td>
<td>28</td>
<td>0.31306227</td>
<td>0.134167</td>
<td>11</td>
<td>60</td>
<td>0.67083449</td>
<td>0.011180</td>
</tr>
<tr>
<td>6</td>
<td>40</td>
<td>0.44722926</td>
<td>0.089443</td>
<td>12</td>
<td>61</td>
<td>0.68201449</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.4.2: Failure control chart of Release 3

The software quality is determined by detecting failures at an early stage. Figure 6.4.1 and 6.4.2 are obtained by placing the successive differences of mean values on y axis, the time or day of the failures on x axis and the values of control limits are placed on failure control chart.
6.5 Conclusion.

The failure control chart shows that, in both the data sets many of the successive differences have gone out of lower control limit i.e \( m(t_L) \). This indicates the failure process is identified. It is significantly early detection of failures using failure control Chart. No failure data fell outside the upper control limit i.e \( m(t_U) \). It does not indicate any alarm signal.