ISSN: 0975-766X
CODEN: IJPTFI
Available Online through
Research Article
www.ijptonline.com

SOFTWARE RELIABILITY MONITORING USING GOMPERTZ DISTRIBUTION

V. Vallinayagam*[1], S. Parthasarathy[2], P. Venkatesan[3]
[1]St. Josephs Engineering College, Chennai.
[2]Department of Mathematics, SRM University Ramapuram, Chennai.
[3]Department of Statistics, Chennai NIRT, Chennai.
Email: vngam05@yahoo.com

Received on 29-04-2016 *Accepted on 29-05-2016*

Abstract:

Software reliability process can be monitor efficiently using statistical process control. It helps the software development team to rectify and necessary action to be taken during software failure process and hence assures better software reliability. In this paper we propose a control mechanism based on cumulative observations using mean value function of Gompertz model. The maximum likelihood estimation approach is used to estimate the unknown parameter of the model.

Keywords: Statistical process control, Software reliability, Gompertz distribution, Maximum Likelihood Estimator.

I Introduction

Software reliability is a key part in software quality. The nature and complexity of software have changed lot in the past few decades. In the recent years, it is very necessary to produce good quality of software with high precession of reliability. In the olden days software errors and bugs were fixed at a later stage in the software development. Today to produce high quality reliable software is a big challenge because nowadays most standard components and better process are introduced every day in this software engineering field. If not considered carefully, software reliability can be the reliability bottleneck of the whole system. Ensuring software reliability is no easy task. As hard as the problem is, promising progresses are still being made toward more reliable software.

Software Reliability measurement is a set of mathematical procedures that can be used to estimate and predict the reliability behavior of software during its development and operation. The primary goal of software reliability is to answer the following question, that is given a system, what is the probability that it will fail in a given time interval, or, what is the expected duration between successive failures? Also it is an important factor affecting system reliability see e.g.,

Musa [3,5], Lyu [6] and Pham H [7, 8, 9]. It differs from hardware reliability in that it reflects the design perfection, rather than manufacturing perfection. The high complexity of software is the major contributing factor of Software Reliability problems. Software Reliability is not a function of time - although researchers have come up with models relating the two. In order to estimate software reliability data we have to use some probability models. Various NHPP (Non-homogeneous Poisson Process) software reliability models are available to estimate the software reliability. Therefore, in this paper, we have taken Gompert distribution and its properties to estimate.

II Literature Survey

This section presents the theory that underlines the proposed distribution and maximum likelihood estimation for complete data. If 't' is 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 cdf and pdf 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 Non-homogeneous poisson process models. Then the mean value function of the finite failure Non-homogeneous poisson process models can be written as $m(t) = aF'(t)$. where, $F(t)$ is a cumulative distribution function $\lambda(t)$ in case of finite failure Non-homogeneous poisson process models is given by $\lambda(t) = aF'(t)$ (see [7])

II.(A). Non-Homogeneous Poisson Process Model

The Non-Homogeneous Poisson process (NHPP) based software reliability growth models are proved to be quite successful in practical software reliability engineering. 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 MLE. Various NHPP SRGMs (Software Reliability Model) 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.

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 $\lambda(t) = b(a - m(t))$. where a denotes the initial number of faults

contained in a program and ′b′ represents the fault detection rate. Inn 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 NHPP SRGM $\lambda(t)$ can be expressed in more general way as $\lambda(t) = b(t)[a(t) - m(t)]$. where $a(t)$ is the time dependent fault content function which includes the initial and introduced the 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 perfect debugging assumption, when the faults are removed, then there is a possibility to introduce new faults.

II.B. Gompertz Distribution

The pdf of a Gompertz distribution has the form,

$f(t, \beta, b) = b\beta e^{bt} e^{\beta} e^{-\beta e^{bt}}$ $t > 0, b > 0, \beta > 0$ where b is the scale parameter and β is the shape parameter. The corresponding cumulative distribution is $F(t) = 1 - e^{-\beta(e^{bt}-1)}$.

If we assume $\beta = 1$, the cumulative function for the distribution becomes, $F(t) = 1 - e^{-(e^{bt}-1)}$

$$= 1 - e^{(1-e^{bt})}$$

$$F'(t) = e^{(1-e^{bt})} e^{bt}.b$$

The Survival function for this distribution is given by $S(t) = 1 - F(t) = 1 - [1 - e^{-\beta(e^{bt}-1)} = e^{-\beta(e^{bt}-1)}$ and Hazard rate function is

$$h(t) = \frac{f(t)}{S(t)} = \frac{b\beta e^{bt} e^{-\beta(e^{bt}-1)}}{e^{-\beta(e^{bt}-1)}}$$

$$= b\beta e^{bt}$$

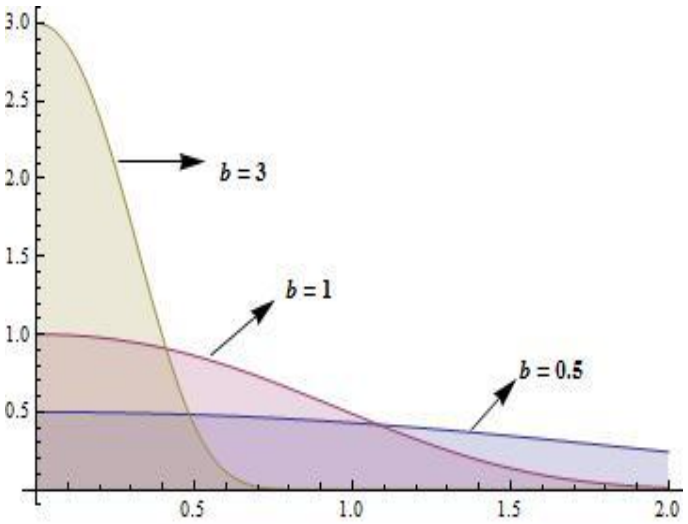

Fig 2.1 PDF of Gompertz with various values of b and $\beta = 1$

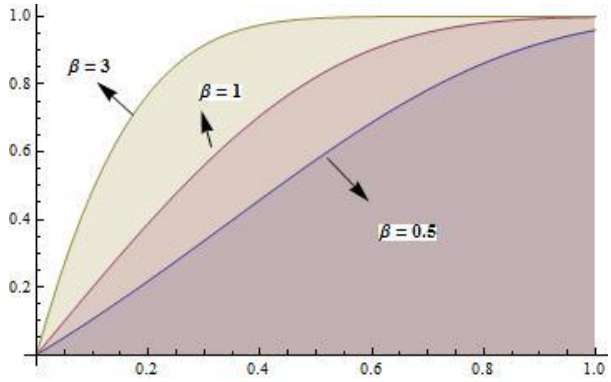

Fig 2.2 CDF of Gompertz with various values of β and b = 2

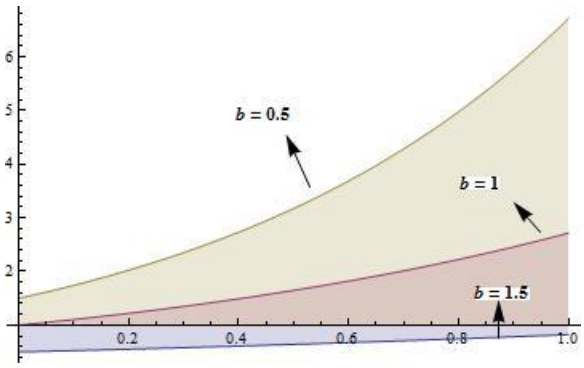

Fig 2.3 Hazard Function of Gompertz with various values of b and β = 1

II. C. Some Statistical Properties

(A)Quantile (x_q) $0 < q < 1$

$$q = p[x \le x_q]$$

$$= F(x_q)$$

$$= 1 - e^{-\beta\left(e^{bx_q}-1\right)}$$

$$\Rightarrow e^{-\beta\left(e^{bx_q}-1\right)} = 1 - q$$

Taking log on both sides,

$$-\beta\left(e^{bx_q} - 1\right) = \text{Log}(1 - q)$$

$$\Rightarrow \boldsymbol{e^{bx_q} - 1 = -\frac{1}{\beta} * Log(1 - q)}$$

$$\boldsymbol{e^{bx_q} = 1 - \frac{1}{\beta} * Log(1 - q)}$$

Again taking log on both sides,

$$bx_q = Log\left[1 - \frac{1}{\beta} * Log(1-q)\right]$$

$$x_q = \frac{1}{b} Log\left[1 - \frac{1}{\beta} * Log(1-q)\right]$$

(B) Median

Take $q = 1/2$

Median = $\left(\frac{1}{b}\right) Log\ [1 - \frac{1}{\beta} Log(1 - 1/2)]\Big)$

(C) Mode

To get mode, we first have to differentiate its pdf with respect to t

$$f'(t) = \beta b^2 e^{bt} e^{-\beta(e^{bt}-1)}[1 - \beta e^{bt}]$$

To find mode, $f'(t) = 0$. Since $f(t) > 0$,Then the mode is solution of the equation $f'(t) = 0$.

$$i.e, \beta b^2 e^{bt} e^{-\beta(e^{bt}-1)}\left[1 - \beta e^{bt}\right] = 0$$

D r^{th} moment

$$\mu^{(r)} = \int_0^\infty t^r f(t) dt$$

$$= \int_0^\infty t^r b\beta e^{bt} e^{-\beta(e^{bt}-1)} dt$$

$$= b\beta \int_0^\infty t^r e^{bt} e^{-\beta(e^{bt}-1)} dt = b\beta e^{\beta} \int_0^\infty t^r e^{bt} \sum_{n=0}^{\infty} \frac{\left(-\beta e^{bt}\right)^n}{n!}$$

$$= b\beta e^{\beta} \sum_{n=0}^{\infty} \int_0^\infty t^r e^{bt} (-\beta)^n e^{nbt}/n!$$

$$= b\beta e^{\beta} \sum_{n=0}^{\infty} \frac{(-\beta)^n}{n!} \int_0^\infty t^r e^{(n+1)bt} dt.$$

Put $t = -u/b(n+1)$

$$= b\beta e^{\beta} \sum_{n=0}^{\infty} \frac{(-\beta)^n}{n!} \int_0^\infty \left[-\frac{1}{b(n+1)}\right] (-1)^r u^r e^{-u}/b^r (n+1)^r du$$

$$= \boldsymbol{b\beta e^{\beta}} \sum_{\boldsymbol{n=0}}^{\infty} ((\boldsymbol{-1})^{\boldsymbol{n+r+1}} \boldsymbol{\beta^{n}})/\boldsymbol{n!}\, \boldsymbol{b^{r+1}}(\boldsymbol{n+1})^{\boldsymbol{r+1}} \int_{\boldsymbol{0}}^{\infty} \boldsymbol{u^{r}} \boldsymbol{e^{-u}} \boldsymbol{du}.$$

$$\mu^{(r)} = b\beta e^{\beta} \sum_{n=0}^{\infty} ((-1)^{n+r+1}\, \beta^{n})/(n!\, b^{r+1}(n+r)^{n+1}) * \Gamma_{(r+1)}$$

(E) Maximum Likelihood Estimator Assume $t_1, t_2, \ldots t_n$ be a random sample of size n. The likelihood function is given by

$$l = \prod_{i=1}^{n} f(t_i) = \prod_{i=1}^{n} \beta b e^{bt} e^{-\beta(e^{bt}-1)}$$

$$= (\beta b)^{n} e^{\sum_{i=1}^{n} bt_i} e^{\sum_{i=1}^{n} -\beta(e^{bt_i}-1)}$$

$$= (\beta b)^{n} e^{b\sum_{i=1}^{n} t_i} e^{\sum_{i=1}^{n} -\beta(e^{bt_i}-1)}$$

The likelihood function $L = \log(l)$is given by

$$L = nlog\beta b + b\sum_{i=1}^{n} t_i - \beta \sum_{i=1}^{n} (e^{bt_i} - 1)$$

The first derivative of Log of the likelihood L with respect to $b$and βas follows.

$$\frac{\partial L}{\partial b} = \left(\frac{n}{b}\right) + \sum_{i=1}^{n} t_i - \beta \sum_{i=1}^{n} t_i\, e^{bt_i}$$

$$\frac{\partial L}{\partial \beta} = \left(\frac{n}{\beta}\right) - \sum_{i=1}^{n} (e^{bt_i} - 1)$$

The likelihood equation are $\frac{\partial L}{\partial b} = 0$ and $\frac{\partial L}{\partial \beta} = 0$.

$$\beta = \left(\frac{n}{\sum_{i=1}^{n}(e^{bt_i} - 1)}\right)$$

Now $\left(\frac{n}{b}\right) + \sum_{i=1}^{n} t_i - \left(n\sum_{i=1}^{n} e^{bt_i} * \frac{t_i}{\sum_{i=1}^{n}(e^{bt_i}-1)}\right) = 0$

The non-linear equation does not have an analytical solution in b.

Also we can study these statistical properties for generalized Gompertz distribution. The pdf for Generalized Gompertz distribution is given by

$$f(x) = \lambda b\beta e^{bt}\ e^{-\beta(1-e^{bt})}[1 - e^{-\beta(1-e^{bt})}]^{\lambda-1}$$

Where $\lambda, \beta > 0$ and $b \geq 0, t \geq 0$.

Note that Generalized Exponential distribution with two parameters can be derived by setting b tends to 0 . Gompertz distribution with two parameters can be derived from setting $\lambda = 1$ Exponential distribution with one parameter can be derived by setting b tends to 0 and putting $\lambda = 1$.

III. Illustration Through Maximum Likelihood Estimator In Interval Domain (Complete Data)

Parameter estimation is very important in the field of software reliability engineering. Once the mean value function $m(t)$ is known for any model, then the parameter estimation can be done by using Maximum likelihood estimator. The parameter estimation can be achieved by two different ways. One method is we can compute through standard method using some mathematical software like MATLAB-R2013a, Mathematica and SAS procedures and the other one is using Newton Rapshon method.

Mean value function for Gompertz is given by $m(t) = aF(t) = a[1 - e^{(1-e^{bt})}]$ and Intensity function is given by $\lambda(t) = aF'(t) = abe^{bt}e^{(1-e^{bt})}$.

Assume that $0 < t_1 < t_2 \ldots < t_n$ then the log likelihood function takes on the following form. The likelihood function by using intensity function is given by see [ ] for MLE for interval domain.

$$L = \sum_{i=1}^{n} \lambda(t_i)$$

$$= \prod_{i=1}^{n} abe^{bt_i}e^{(1-e^{bt_i})}$$

Taking log on both sides

$$Log\ L = Log[\prod_{i=1}^{n} abe^{bt_i}e^{(1-e^{bt_i})}]$$

$$= \sum_{i=1}^{n} Log[abe^{bt_i}e^{(1-e^{bt_i})}] - a[1 - e^{(1-e^{bt_i})}]$$

$$= nLog(2ab) + \sum_{i=1}^{n} t_i + \sum_{i=1}^{n} 1 - e^{bt_i} - a\left[1 - e^{(1-bt_n)}\right]$$

Taking partial derivatives with respect to $'a'$

$$\frac{\delta LogL}{\delta a} = \left(\frac{n}{a}\right) - \left[1 - e^{(1-e^{bt_n})}\right]$$

Equate $\frac{\delta LogL}{\delta a} = 0$

$$\left(\frac{n}{a}\right) - \left[1 - e^{(1-e^{bt_n})}\right] = 0$$

$$\Rightarrow a = \left(\frac{n}{\left[1-e^{(1-e^{bt_n})}\right]}\right)$$

Taking partial derivatives with respect to $'\beta'$

$$a = \left(\frac{n}{\left[1 - e^{\beta(1-e^{bt_n})}\right]}\right) \; where \; \beta > 0$$

Taking partial derivatives with respect to b

$$\frac{\delta LogL}{\delta b} = \left(\frac{n}{b}\right) + \sum_{i=1}^{n} t_i - \sum_{i=1}^{n} t_i \, e^{bt_i} - \left(\frac{nt_n e^{bt_n} e^{(1-e^{bt_n})}}{1 - e^{(1-e^{bt_n})}}\right)$$

Take $\frac{\delta LogL}{\delta b} = 0$,

$$g(b) = \left(\frac{n}{b}\right) + \sum_{i=1}^{n} t_i - \sum_{i=1}^{n} t_i \, e^{bt_i} - \left(\frac{nt_n e^{bt_n} e^{(1-e^{bt_n})}}{1 - e^{(1-e^{bt_n})}}\right) = 0$$

Now the parameter $'b'$ is estimated by Newton-Raphson Method

i.e, $b_{n+1} = b_n - \frac{g(b_n)}{g'(b)}$.

In general for any β

$$g(b) = \left(\frac{n}{b}\right) + \sum_{i=1}^{n} t_i - \beta \sum_{i=1}^{n} t_i \, e^{bt_i} - \left(\frac{\beta n t_n e^{bt_n} e^{\beta(1-e^{bt_n})}}{1 - e^{\beta(1-e^{bt_n})}}\right) = 0$$

Where $\beta > 0$.

Based on the following failure data (Table 3.1), we have computed the software failure process using Mean value chart suggested by Xie et al [10]. We have used cumulative time between failures for software reliability monitoring using Gompertz distribution. The parameters a and b are computed using Maximum Likelihood estimator for complete data in an interval domain using Newton-Raphson iterative method. Using a and b values we can compute $m(t)$.

Failure Number	Time Between Failure	Failure Number	Time Between Failure
1	30.02	16	15.53
2	1.44	17	25.72

3	22.47	18	2.79
4	1.36	19	1.92
5	3.43	20	4.13
6	13.2	21	70.47
7	5.15	22	17.07
8	3.83	23	3.99
9	21	24	176.06
10	12.97	25	81.07
11	0.47	26	2.27
12	6.23	27	15.63
13	3.39	28	120.78
14	9.11	29	30.81
15	2.18	30	34.19

Table3.1Time Between Failures of a Software

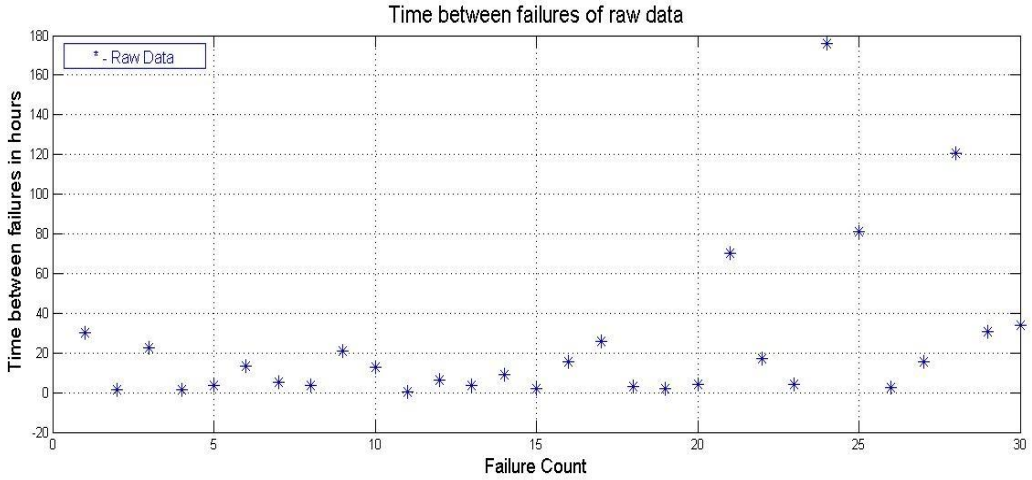

Fig.3.1Time Between Failures of Raw Data

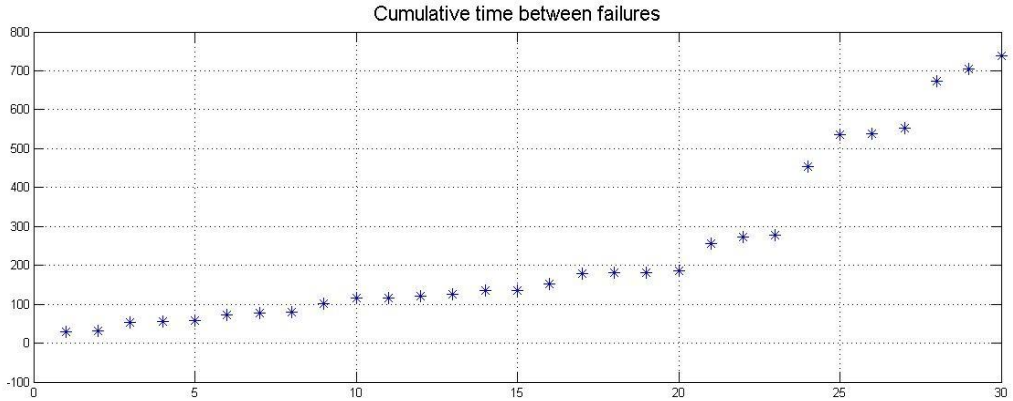

Fig.3.2Cumulative Time Between Failures

III. A. Monitoring Gompertz Distribution Time Between Failures

Model	a	b
Gompertz	30.0207	0.004

Table 3.2 Parameter Value

$$T_U = 1 - e^{(1-e^{bt})} = 0.99865$$

$$T_C = 1 - e^{(1-e^{bt})} = 0.5$$

$$T_L = 1 - e^{(1-e^{bt})} = 0.00135$$

More general,

$$T_U = 1 - e^{\beta(1-e^{bt})} = 1 - \frac{\alpha}{2}$$

$$T_C = 1 - e^{\beta(1-e^{bt})} = \frac{1}{2}$$

$T_L = 1 - e^{\beta(1-e^{bt})} = \frac{\alpha}{2}$, where α is the traditional false alarm probability is set to be 0.27%.

These limits are converted to $m(T_U),\ m(T_C),\ m(T_L)$ form. These limits are used to find whether the software process is control or not. A point below the control limit $m(T_L)$ is indicates an alarming signal. A point above the control limit $m(T_U)$ indicated 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 follows

Model	$\boldsymbol{m(T_U)}$	$\boldsymbol{m(T_C)}$	$\boldsymbol{m(T_L)}$
Gompertz	29.9802	15.0104	0.0405

Table 3.3 Control Limits

Failure Count	$\boldsymbol{m(t)}$	**Successive difference**
1	10.2033	0.048
2	10.2513	0.7696
3	11.0209	0.0478
4	11.0687	0.1213
5	11.19	0.4751
6	11.6651	0.1889

7	11.854	0.1418
8	11.9957	0.7962
9	12.792	0.5072
10	**13.2992**	**0.0186**
11	13.3177	0.2478
12	13.5656	0.1359
13	13.7015	0.3688
14	14.0702	0.089
15	14.1592	0.642
16	14.8013	1.0913
17	15.8925	0.1202
18	16.0128	0.0829
19	16.0957	0.1789
20	16.2745	3.1189
21	19.3935	0.7596
22	20.1531	0.1768
23	20.3299	6.6905
24	27.0204	1.7716
25	**28.792**	**0.0357**
26	28.8277	0.2263
27	29.054	0.8554
28	29.9094	0.0583
29	**29.9677**	**0.0323**
30	30	

Table 3.4 Successive Differences of Cumulative Mean Values

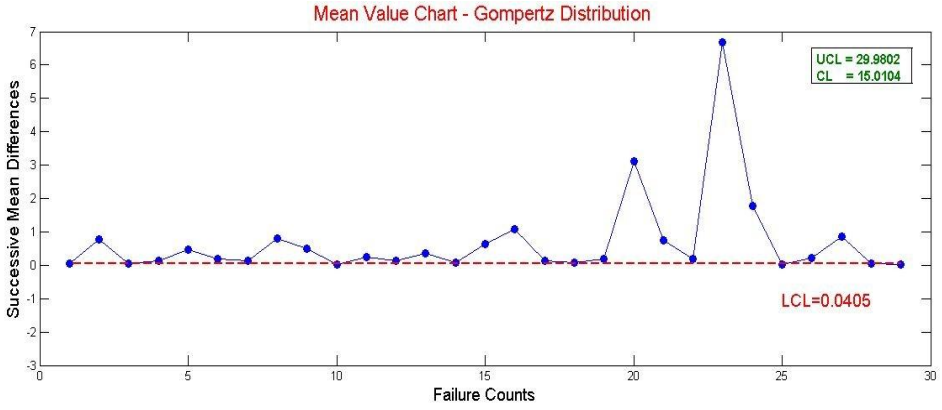

Fig.3.3 Mean Value Chart For Gompertz Distribution

In Fig.3.3 X axis is number of failures and Y axis is successive mean differences. The Fig.3.3 shows that 10, 25th and 29th failure data as fallen below □(□_□) which indicates the failure process is identified. It is significantly early detection of failure using mean value chart. The software quality is determined by detecting failures at an earlier stage. The remaining failure data are in stable.

IV. Conclusion

In this paper the parameter estimation is carried out by Newton-Raphson iterative method. This method of estimation and the control chart giving a positive recommendation for finding out preferable control process are desirable out of control signal. By observing the mean value chart we identify that the failure is detected at 10th, 25th and 29th point. This early detection of software failure will prove software reliability. Gompertz distribution also we can use for software reliability monitoring instead of Weibull distribution.

V. References

1. C. B. Read, Gompertz Distribution, Encyclopedia of Statistical Sciences, Wiley, New York, 1983.
2. G. Krishnamohan, R. Satya Prasad, "Interval Domain based Software Process Control using Weibull Mean Value Function". IRACST- International Journal of Computer Science and Information Technology and Security, Vol. 1, No. 2, December 2011.
3. Musa J D. "Software Reliability Engineering: More reliable software, faster and cheaper.", Tata McGraw-Hill Education, (2004).
4. Musa, J. D. "A Theory of Software Reliability and its Application". IEEE Trans. On Software Eng., Vol. SH-1, pp 312-327, 1979.
5. Musa, J. D. Iannino, A., Okumoto, K., "Software Reliability: Measurment prediction Application". McGraw-Hill, New York, 1987.
6. M. R. Lyu, "Handbook of Software Reliability Engineering". McGraw-Hill, 1996.
7. Pham H, "Hand Book of Reliability Engineering", Springer, 2003.
8. Pham H, "System Software Reliability", Springer, 2006.
9. Pham H, "Software Reliability Data. Report available from Data and Analysis Centre for software, Rome Air Development Centre, Rome, New York, USA, 1979.

10. Xie. M., Goh. T. N., Ranjan P., "Some Effective Control chart procedures for Reliability monitoring", Reliability Engineering and System safety, 77, 2002, PP 143-150.

Corresponding Author:

V. Vallinayagam*,

Email: *vngam05@yahoo.com*