A Novel Modification of the Weibull Function Model for Software Reliability Evaluation based on Cloud Model

Peng Cao[1,a], Bin Wen[2,b], Yu Zhang[2,c], Ziqiang Luo[2,d*]
[1]*Dept. of Mathematics and Physics, Qiongtai Teachers College, Haikou, 571100, China*
[2]*College of Information Science and Technology, Hainan Normal University, Haikou, 571158, China*
[a]*lanyuan97@126.com,* [b]*951714238@qq.com,*[c]*344248003@qq.com,* [d]*306003057@qq.com *Corresponding Author*

Abstract

There are many kinds of uncertainties inherent in software reliability. Cloud model is a new cognitive model for uncertain transformation between linguistic concepts and quantitative values. This paper presents a new approach to modification of the Weibull function model based on cloud model theory. Finally, we show the practical application of the novel method by using a group of incomplete software failure data.

Keywords: software reliability, uncertainty, Weibull function model, cloud model.

1. Introduction

The failure mechanism of software is very complex, so the relationship between different variables is often nonlinear. Nonlinear regression method which does not need to make some rigorous and unpractical assumptions is considered a more promising mathematical tool for software reliability evaluation. As far as the method is concerned, according to engineering experience and a great deal of data, the most critical step is to select the fitting curve carefully and correctly. The selection of the fitting curve, such as Weibull function[1,2], exponential function[1], etc., should be flexible on the basis of specific situation.

It should be noted that, based on analysis of a large amount of failure data, the nonlinear regression model only provides a point or interval estimation of a reliability index θ. Unfortunately, it is not likely to collect enough firsthand data in practical engineering. However, we must admit that such estimation values may vary with different samples. Even if it is one sample, point or interval estimation is also different due to different statistics.

In the cloud model [3, 4], which is a new cognitive model for uncertain transformation between linguistic concepts and quantitative values, we employ the expectation *Ex*, the entropy *En*, and the hyper–entropy *He* to represent the concept

as a whole. Especially, the normal cloud model can avoid the flaw of fuzzy sets[5,6] to quantify the membership degree of an element as an accurate value between 0 and 1, therefore, may be more adaptive for the uncertainty description of linguistic concepts. Then cloud model can be utilized to represent software reliability so as to deal with the universal uncertainty in the concept, which contains many kinds of uncertainties, such as randomness, fuzziness and the correlation between them in particular.

The paper is constructed as follows: first introduces mathematical description of the nonlinear regression method to fit the failure data of a software system during the testing phase. Then present a novel approach to modification of the Weibull function model based on cloud model theory. Finally, the methodology developed in this paper is exemplified with a group of incomplete software failure data set.

2. Mathematical description of the Weibull function model [1]

Several test data indicate that, during the process of software testing, sometimes the reliability growth trend did not appear obviously until the middle or late test phase. On the contrary, in the early stage of the testing process reliability degradation phenomena occurred in some programs. It is because that testers' awareness of programs is gradually deepening in the testing process, troubleshooting capability also increases. In order to characterize this phenomenon, the Weibull model can be used to describe the process error occurs, i.e.

$$N(t) = N_0\left(1-e^{-\beta t^{\alpha}}\right). \tag{1}$$

where N_0 (an unknown parameter) is the total number of error inherent in a software, $N(t)$（$t\geq 0$）is the cumulative number of failures (or detected errors) at time t.

Let k_i be the true cumulative number of software failures at time t_i for i =1, 2,…,n, then the least squares function according to Eq. 1,

$$Q\left(N_0,\beta,\alpha\right) = \sum_{i=1}^{n}\left[k_i - N_0\left(1-e^{-\beta t_i^{\alpha}}\right)\right]^2 .\square \tag{2}$$

The point estimations $\hat{N}_{0n}$, $\hat{\beta}_n$ and $\hat{\alpha}_n$ can be got by using the nlinfit() function of Matlab. And furthermore, we can calculate the confidence intervals $\left[N_0^L, N_0^U\right]$, $[\beta^L, \beta^U]$ and $[\alpha^L, \alpha^U]$ by using the nlparci() function with confidence level of 0.95. In addition, the confidence intervals $\left[\hat{k}_i - \delta_i, \hat{k}_i + \delta_i\right]$ can be calculated by using the nlpredci() function.

The error occurrence rate is a valuable parameter to reflect the reliability level, as follows

$$N'(t) = N_0 \beta \alpha t^{\alpha-1} e^{-\beta t^{\alpha}} . \square \qquad \square(3)$$

Owing to the fact that users are usually unable to debug the system, so we can reasonably assume that the failure rate of the system is a constant after release to the user and equal to the error occurrence rate at release time. Given the expected object of software failure rate λ_c, we can calculate the expected delivery time t_c using objective function

$$\min \ t_c$$

$$\text{s.t. } N_0 \beta \alpha t_c^{\alpha-1} e^{-\beta t_c^{\alpha}} \geq \lambda_c . \square \qquad \square(4)$$

Then, the mean time to failure after release to the user can be calculated by

$$MTBF_C = 1/\lambda_c . \qquad \square(5)$$

3. A modification to the Weibull function model based on cloud model theory

A confidence interval $[\theta_L, \theta_U]$ where confidence limits θ_L and θ_U are the lower and upper boundaries, gives an estimated range of values which is likely to include an unknown parameter θ. The confidence level $(1-\alpha)$ is the probability value associated with a confidence interval.

In general, if the confidence interval for a software reliability index θ is achieved, we can calculate the numerical characteristics of virtual cloud $C(Ex, En, He)$, which can be switched to system qualitative evaluation by X condition cloud generator.

In order to depict randomness and fuzziness of the reliability index θ, θ may be expressed by $C(Ex, En, He)$, where $Ex=(\theta_L+\theta_U)/2$, $En=(\theta_U-\theta_L)/6$, $He=0.05 \cdot En$.

Then, the parameters N_0, β, α and k_{n+1} may be expressed by cloud model as follows,

$$\tilde{N}_{0n} = C(Ex_{1n}, En_{1n}, He_{1n}), \quad \tilde{\beta}_n = C(Ex_{2n}, En_{2n}, He_{2n}),$$

$$\tilde{\alpha}_n = C(Ex_{3n}, En_{3n}, He_{3n}), \quad \tilde{k}_{n+1} = C(Ex_{4n}, En_{4n}, He_{4n}), \qquad (6)$$

where

$Ex_{1n} = \hat{N}_{0n}$, $En_{1n}=\left(N_0^U - N_0^L\right)/6$, $He_{1n}=0.05\cdot En_{1n}$;

$Ex_{2n}=\hat{\beta}_n$, $En_{2n}=\left(\beta^U - \beta^L\right)/6$, $He_{2n}=0.05\cdot En_{2n}$;

$Ex_{3n}=\hat{\alpha}_n$, $En_{3n}=\left(\alpha^U - \alpha^L\right)/6$, $He_{3n}=0.05\cdot En_{3n}$;

$Ex_{4n} = \hat{k}_{n+1}$, $En_{4n} = \delta_{n+1}/3$, $He_{4n} =0.05\cdot En_{4n}$.

Let M_{n+1} be the residual error number at time t_{n+1}, then the estimated value $\hat{M}_{n+1} = \hat{N}_{0n} - \hat{k}_{n+1}$.

Based on the algebraic operation rule of cloud model, the residual error number can be expressed by cloud

$$\tilde{M}_{n+1} = \tilde{N}_{0n} - \tilde{k}_{n+1}. \quad (7)$$

4. Example analysis

The following is an example to show the practical application of the novel method. Table 1 is a group of incomplete software failure data, taken from [1]. It is a large distributed software system failure data in 30 weeks.

When the sample number n is 29 or 30, we compare the true cumulative number of software failures with the predicted values at time t_i for i =1,2,…,30, as shown in Table 2.

When n=29, we compute point estimations, confidence intervals at confidence level of 0.95, and clouds of N_0, β, α, k_{30} and M_{30} according to the above formulas, shown in Table 3.

From the point of view of the whole fitting effect, when the sample number n=30, the predicted values of the Weibull function model have basically reflected the changing trends of the cumulative number of failures, as shown in Fig.1 and Fig.2 . Obviously, there exists great distinction between the actual failure times with the theoretical estimate predicted values in some weeks in Fig. 2. It is mainly because that the number of errors during software testing is restricted by various factors, thus can't completely obey the law of a particular distribution. As far as software reliability is concerned, it is impossible for us to get a fixed accurately estimated value. Therefore, it is expressed based on cloud model maybe more actual.

The cloud $\tilde{k}_{15}$ can especially express the possible value of k_{15} under the different determination degree, compared with confidence intervals, thus more in

line with engineering practice, but should define that the cloud drop is bigger than k_{14}.

Table 1: A group of incomplete software failure data

Time (Week)	Failure Number	Cumulative Number of Failures	Time (Week)	Failure Number	Cumulative Number of Failures
1	68	68	16	66	1901
2	107	175	17	73	1974
3	112	287	18	87	2061
4	114	401	19	74	2135
5	139	540	20	41	2176
6	159	699	21	44	2220
7	197	896	22	67	2287
8	157	1053	23	69	2356
9	97	1150	24	12	2368
10	86	1236	25	12	2380
11	212	1448	26	14	2394
12	186	1634	27	22	2416
13	68	1702	28	14	2430
14	56	1758	29	18	2448
15	77	1835	**30**	14	2462

Table 2: Compare the true cumulative number of failures with the predicted results of the Weibull function model with different number of samples

i	k_i	$\hat{k}_i$ (n=29)	$\hat{k}_i$ (n=30)	i	k_i	$\hat{k}_i$ (n=29)	$\hat{k}_i$ (n=30)
1	68	57.71	57.74	16	1901	1936.01	1935.96
2	175	159.29	159.35	17	1974	2011.67	2011.63
3	287	284.31	284.39	18	2061	2079.34	2079.32
4	401	423.60	423.68	19	2135	2139.52	2139.53
5	540	570.93	571.00	20	2176	2192.76	2192.79
6	699	721.67	721.72	21	2220	2239.62	2239.68
7	896	872.24	872.27	22	2287	2280.66	2280.75
8	1053	1019.88	1019.88	23	2356	2316.44	2316.55
9	1150	1162.46	1162.44	24	2368	2347.48	2347.62
10	1236	1298.42	1298.37	25	2380	2374.30	2374.47
11	1448	1426.64	1426.58	26	2394	2397.37	2397.57
12	1634	1546.40	1546.33	27	2416	2417.14	2417.37
13	1702	1657.27	1657.19	28	2430	2434.01	2434.26
14	1758	1759.10	1759.02	29	2448	2448.36	2448.63
15	1835	1851.94	1851.87	30	2462	2460.50	2460.80

When the failure rate of software after release to the user λ_c =0.06 per week is asked, according to Eq. 7, we can get $MTBF_C$=1/ λ_c ≈16.7 week=2800 hours.

Table 3: The point estimations, confidence intervals, and cloud of $N_0, \beta, \alpha, k_{30}$ and M_{30} when n =29

	N_0	β	α	k_{30}	M_{30}
Point Estimation	2520.32	0.0231	1.4948	2460.50	59.82
Confidence Interval	[2469.79, 2570.85]	[0.0201, 0.0262]	[1.4316, 1.5581]	[2429.54, 2491.46]	/
Cloud	(2520.32, 16.8431, 0.8421)	(0.0231, 0.001011, 5.0592×10^{-5})	(1.4948, 0.02108, 0.001054)	(2460.50, 10.3195, 0.5159)	(59.82, 19.75, 0.98)

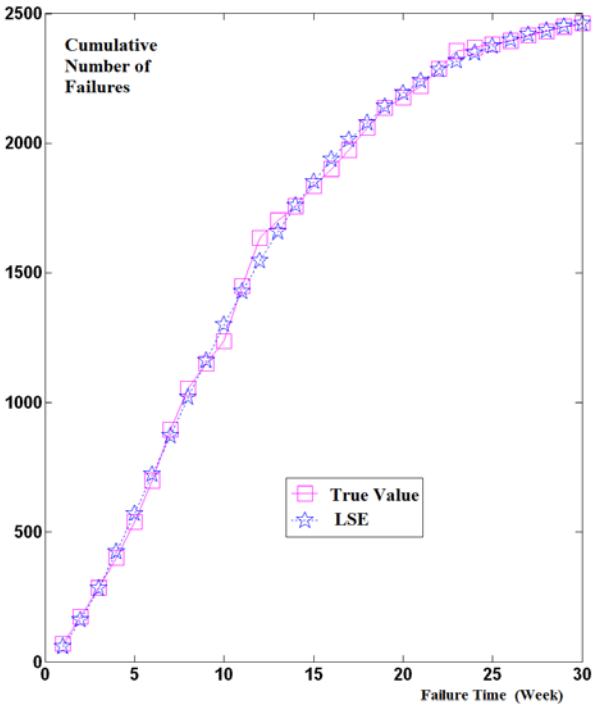

Fig.1: Compare the real cumulative number of failures with the least squares estimations of the exponential functiona model

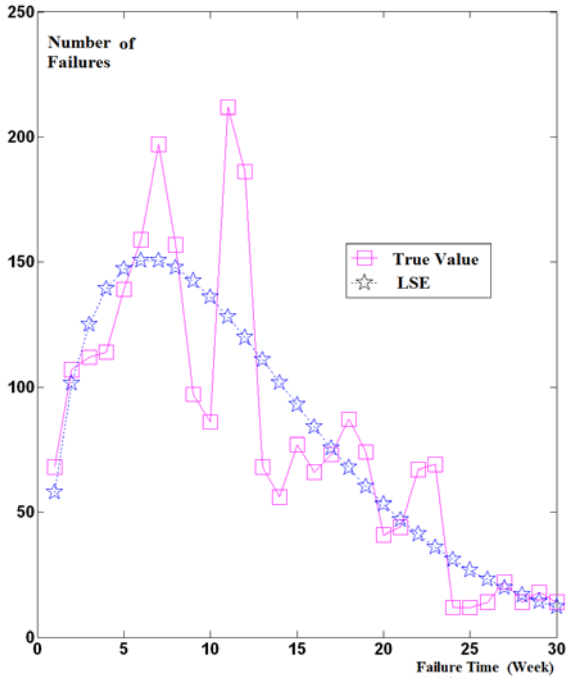

Fig.2: Compare the real number of failures with the least squares estimations of the exponential functiona model

In general, the possibility of failure–free software operation for a specified period of time in a specified environment can be indicated by such 5 linguistic values as very low, low, moderate, high and very high. We then declare that the concept set Re_1 of software reliability evaluation contains the five linguistic values. The five linguistic values of software *MTBF* domain [0, +∞) are expressed by cloud modes as follows:

$$C_{B_1}=\begin{cases}1 & x\in[0,1000]\\ C(1000,300,3) & else\end{cases},\qquad C_{B_2}=C(1\times10^4,3000,30),$$

$$C_{B_3}=C(1\times10^5,3\times10^4,300),\qquad C_{B_4}=C(1\times10^6,3\times10^5,3000),$$

$$C_{B_5}=\begin{cases}C(1\times10^7,3\times10^6,3\times10^4) & else\\ 1 & x>1\times10^7\end{cases},$$

where B_1 denotes very low, B_2 denotes low, B_3 denotes moderate, B_4 denotes high, B_5 denotes very high, especially B_1 and B_5 are half–cloud models which represent the concept with uncertainty on only one side.

Reference to the software reliability conception Re_1, with the input of $MTBF_C$, X condition cloud generators CG_{B_1} and CG_{B_2} may individually output the certain

degrees 1.8×10^{-8} and 0.05 corresponding to the two different concepts. Finally, according to maximum determination principle for membership concept may judge the released software reliability "low".

Acknowledgements

This work is financially supported by National Nature and Science Foundation of China (61463012), Natural Science Foundation of Hainan Province (614233, 114013, 613161, 613162), and Qiongtai Teachers College Scientific Research Project (qtky201304).

References

[1] Huang X. Z. *Software Reliability, Safety and Quality Assurance*. Beijing: Publishing House of Electronics Industry, 2002, in Chinese.

[2] Huang X. Z. Non-linear regression for predicting software reliability. *microelectronics reliability*, Vol. 28, pp.865-868,1988.

[3] Li D. Y. & Du Y. *Artificial Intelligence with Uncertainty*. Beijing: National Defense Industry Press, 2005, in Chinese.

[4] Li D. Y., Liu C.Y. & Gang W.Y. A new cognitive model: cloud model. *Int. J. Intell. Syst.*, Vol. 24, pp.357-375, 2009.

[5] Zadeh L. A. Fuzzy sets. *Information and Control*, Vol.8, pp.338-353, 1965.

[6] Zadeh L. A. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, Vol. 4, pp. 103-111, 1996.

[7] Luo Z. Q. & Cao P. *The Uncertainty Research of Software Reliability*. Wuhan: Hubei Science and Technology Press, 2012, in Chinese.