Software Reliability Nonlinear Modeling and Its Fuzzy Evaluation

GUO JUNHONG[1,2], YANG XIAOZONG[1], LIU HONGWEI[1]
School of Computer Science and Engineering
[1]Harbin Institute of Technology
Harbin, Xidazhi street No.92 150001
School of Computer Science and Technology
[2]Heilongjiang University
Harbin, Xuefu road No.74 150080
CHINA

Abstract: - Based on the research of the present developing situation in software reliability models, this paper found that it was difficult to satisfy the hypothesis condition of models in actual projects, and these hypothesis restrict the universality of those models. However, the testing data satisfies the properties of time series. This paper presents a nonlinear model of software reliability based on time series and gives corresponding algorithms. The simulation experiments show the accuracy and efficiency of this new model. The newly proposed model suits the requirement of software engineering better and its parameters can reflect the changing of software reliability. The new model can have more accurate analysis and forecast to software reliability issue without any strict assumption. Furthermore, the fuzzy theory is introduced to evaluate software reliability. Using the fuzzy theory that integrated the fuzziness and randomness, the software reliability evaluation method is more approaching to the real system and the accuracy of software reliability evaluation is promoted greatly.

Keywords: - Software reliability nonlinear model; Time series analysis; Software reliability growth model; Software reliability evaluation

1 Introduction

Due to the complexity of computer software systems and the increasing of development cost, it is of utmost importance to develop high quality software systems. The quality of the software system is described by many metrics such as: complexity, maintainability, availability, reliability, etc. As tragedies of unreliable software often take place, people recognize the importance of developing reliable software. Now, the reliability problem in software systems is a well-known research field.Therefore, designing reliable software and evaluation software reliability accurately are the most important issues [1].

The standard definition of reliability for software (Musa, Iannino, and Okomoto, 1987) is the probability of execution without failure for some specified interval of natural units or time [2]. Software failures are the manifestation of software errors which are introduced into the software by software engineers during the phases of software development cycle. In the literatures, researchers establish software reliability growth model (SRGM) to measure software reliability. To apply these models, it is necessary to know how well the models suit an actual observation failure data set. In order to obtain accurate software reliability estimation, it requires a large number of failure data which are not usually available until the system has been tested for a long relative period. Many software reliability engineers are more interested in estimating the software reliability as early as possible [3].

This paper proposes a method for early and more accurate software reliability prediction by time series analysis. Its calculation methods are simple and without any strict assumption. This paper is organized as follows. Section 2 presents the state of software reliability modeling. Section 3 shows the feasibility of software reliability nonlinear modeling based on time series. The implemented algorithm is given in section 4. Section 5 provides the simulation analysis of testing data. In section 6, fuzzy theory are used to evaluate software reliability. Finally, a brief conclusion is presented.

2 The State of Software Reliability Modeling

SRGM is defined as the mathematical relationship between the number of software errors removed and testing time. Classical SRGMs have great influence on software reliability modeling research. In spite of the fact that many software reliability growth models have been proposed in the software reliability literature since the first one appeared in 1972 [4], no one of them can deal properly with all possible situations.

According to the software error removal process, Jelinski and Moranda proposed the first model. The model has a simple structure and assumptions. Then Musa proposed the Basic Execution Time Model which has similar assumptions to those of Jelinski and Moranda(J-M) model [5]. J-M model made a major contribution to the understanding the relation of error removal and software execution time. Goel-Okumoto model is the first nonhomogeneous Poisson process(NHPP) SRGM. Goel-Okumoto model assumed that the error removal process follows NHPP. The assumption of this model is similar to the Basic Execution Time model of Musa [6].

Those models are mainly based on some assumption conditions and believe that SRGM has been established according to a certain probability process. The assumptions are the key factors of establishing SRGM. There is a relation between assumptions chosen and modeling success. But in practical application, quite a few assumptions are unfit to software developing process, which can not be accepted by most people, thus limiting the application of the models. Using this kind of model can even get the unimaginable parameter under some conditions. Furthermore, a software failure data set being evaluated can get different results by using different models. So it has reached a common viewpoint in software reliability evaluation field, in that there isn't a "good model" that can fit all failures data very well [7].

J-M model assumes that all errors that have been checked in the test can be eliminated and removes one error each time. Removing one error does not affect the remaining errors. But in fact, software development and error removal are all human behaviors, which are unpredictable, so it can not avoid of introducing new errors during the process of error removal. Goel-Okumoto model suppose that each error is independent, each error has the same probability of leading system failure and each failure interval time is independent. Sometimes, modules in software program have some relations, so it is impossible to have complete independence, and the probability of each failure that causes system failure is not the same [8,9].

In these three models, the cumulative numbers of errors removal grow exponentially with the testing time. The exponential growth curve is due to the assumption that the error removal intensity is linearly related to the remaining number of software errors [10]. In many software development projects it was observed that the relation between the cumulative numbers of errors removed and the testing time is not linear. So we should establish nonlinear model to evaluate the software reliability.

3 The Feasibility of Software Reliability Nonlinear Modeling Based on Time Series

Time series analysis theory is a method of describing statistics character of dynamic data, which can set up time series model from limited sample data, its advantage is convenience and practicality. There are many literatures on the development of estimation and prediction in autoregression time series models. Time series analysis method is well studied in some statistical literatures. However, its use in software reliability engineering is rather limited [11].

Time series prediction can be stated as follows: given a finite sequence $X_1, X_2, X_3, \ldots, X_t$, predicting the continued sequence $X_{t+1}, X_{t+2}, \ldots$. For example, $\{X_t\}$ can be viewed as the stochastic failure intervals or the number of failures per time interval [12].

Based on software reliability analysis, input data are cumulative number of software failures or failure intervals mainly. That is to say, software reliability failure data are discrete data sequence, whether it is steady or not, we can use the data to modeling and evaluate software reliability by applying proper time series method [13].

During the process of software reliability evaluation, it can be seen that the nonlinear phenomena exist very commonly and we can not deal these data with linear model. The relevant software reliability time series nonlinear model are deduced in the following section. This modeling method proves that software reliability evaluation also can be presented in the way of time series nonlinear model.

4 The Implemented Algorithms

The cumulative number of failures $M(k)$ is increasing and trend to a fixed value which is defined as the desired cumulative number of failures.

Assume cumulative failures of software are stochastic variable. We can find that all total failures in one time interval are exponential decreasing according to experience. Then we have software reliability nonlinear model

$$M(k) = b^k M(k-1) + M(k-1) \tag{1}$$

Let $\theta(k) = b^k$, then the software reliability nonlinear model can be transformed to time series model as follow:

$$M(k) = (\theta(k)+1))M(k-1) + \varepsilon(k) \tag{2}$$

where $\varepsilon(k)$ is zero-average white noise. Obviously, we can get

$$\theta(k) = b\theta(k-1) \tag{3}$$

that's to say, parameter $\theta(k)$ can be described by the following AR time series model

$$\theta(k) = b\theta(k-1) + \xi(k) \tag{4}$$

where $\xi(k)$ is zero-average white noise.

As testing time is based on the unit of day, the cumulative numbers of software failures fluctuate greatly. Considering the series $\{\theta(k)\}$ got from observed testing data exist much disturbance noise, we apply a smoothing filter to testing data as following steps: first, we get series $\{\eta(k)\}$ simply as following equation

$$\eta(k) = M(k)/M(k-1) - 1, k = 2,...N, \tag{5}$$

and let $\eta(1) = \eta(2)$; then we can get time series $\{\theta(k)\}$ from $\{\eta(k)\}$ by applying smoothing filter:

$$\theta(k) = \lambda\theta(k-1) + (1-\lambda)\eta(k), \tag{6}$$

where the initial is $\theta(1) = \eta(1)$ and $\lambda = 0.7$ in the simulation.

Now we can use exponential weighted Least Squares method [14] to estimate the parameter b from equation (4). Formulas can be shown as the following equations:

$$\hat{b}(k+1) = \hat{b}(k) + k(k+1) [\theta(k+1) - \hat{b}(k)\theta(k)] \tag{7}$$

$$K(k+1) = \frac{P(k)\theta(k)}{\omega + \theta^2(k)P(k)} \tag{8}$$

$$P(k+1) = \frac{1}{\omega}[1 - K(k+1)\theta(k)]P(k) \tag{9}$$

where ω is forgetting factor, $0 < \omega < 1$, commonly is about 0.9~0.99. The initial values are given by:

$$\hat{b}(0) = 1, P(0) = 10^4.$$

Then we can calculate the estimation as follows:

$$\hat{\theta}(k) = \hat{b}(k-1)\theta(k-1) \tag{10}$$

$$\hat{M}(k \mid k-1) = (\hat{\theta}(k)+1)M(k-1) \tag{11}$$

Based on these results, we can get the prediction from the following function:

$$\hat{\theta}(N+p \mid N) = \hat{b}(N)\theta(N+p-1 \mid N) \quad p = 1,2,... \tag{12}$$

$$\hat{M}(N+p \mid N) = (\hat{\theta}(N+p \mid N)+1) \hat{M}(N+p-1 \mid N) \tag{13}$$

where $\hat{\theta}(N \mid N) = \hat{\theta}(N), \hat{M}(N \mid N) = M(N)$.

5 Simulations and Analysis

Software failure data in Table 1 comes from Data8 in chapter 17 of Handbook of Software Reliability Engineering [15], where Day is the test time and CF is cumulative number of software failures. To view the veracity and accuracy of prediction, N is assumed as 98 and p is 10 in the simulation.

Table 1 A set of software failure data

Day	CF	Day	CF	Day	CF	Day	CF
0	5	**52**	211	**72**	346	**92**	460
3	10	**54**	217	**73**	367	**93**	463
6	15	**55**	230	**74**	375	**94**	464
9	20	**56**	234	**75**	381	**95**	465
13	26	**57**	236	**76**	401	**96**	466
22	43	**58**	240	**77**	411	**97**	467
29	36	**59**	243	**78**	414	**98**	468
31	43	**60**	252	**79**	417	**99**	469
32	47	**61**	254	**80**	425	**100**	470
36	49	**62**	259	**81**	430	**101**	472
37	80	**63**	263	**82**	431	**102**	473
40	84	**64**	264	**83**	433	**103**	475
41	108	**65**	268	**84**	435	**104**	476
44	157	**66**	271	**85**	437	**105**	477
46	171	**67**	277	**86**	444	**106**	478
48	183	**68**	290	**88**	446	**107**	479
49	191	**69**	309	**89**	448	**108**	480
50	200	**70**	324	**90**	451		
51	204	**71**	331	**91**	453		

Table 2 Comparison of observed data and estimated data of nonlinear model

Day	Observed Data	Estimated Data	Estimated Error	Relative Error
5	10	11.6828	1.6828	0.1683
15	26	27.2344	1.2344	0.0475
25	34	34.1372	0.1372	0.0040
35	47	47.0364	0.0364	0.0008
45	157	157.0992	0.0992	0.0006
55	230	217.0308	-12.9692	-0.0564
65	268	264.0089	-3.9911	-0.0149
75	381	375.0463	-5.9537	-0.0156
85	437	435.0170	-1.9830	-0.0045
95	465	464.0065	-0.9935	-0.0021

$$\text{Relative estimated error} = \frac{\hat{M}(k) - M(k)}{M(k)}$$

Table 3 Comparison of observed data and predicted data of nonlinear model

Day	Observed Data	Predicted Data	Predicted Error	Relative Error
99	467	467.8342	0.8342	0.0018
100	468	469.4710	1.4710	0.0031
101	469	470.9310	1.9310	0.0041
102	470	472.2329	2.2329	0.0048
103	472	473.3934	1.3934	0.0030
104	473	474.4274	1.4274	0.0030
105	475	475.3486	0.3486	0.0007
106	476	476.1690	0.1690	0.0004
107	477	476.8995	-0.1005	-0.0002
108	478	477.5499	-0.4501	-0.0009

$$\text{Relative predicted error} = \frac{\hat{M}(N+p\mid N) - M(N+p)}{M(N+p)}$$

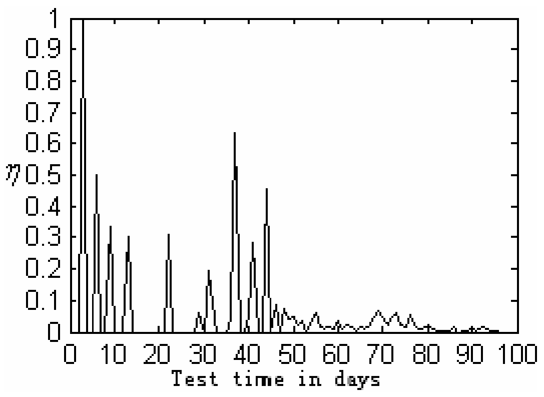

Fig.1 $\eta(k)$

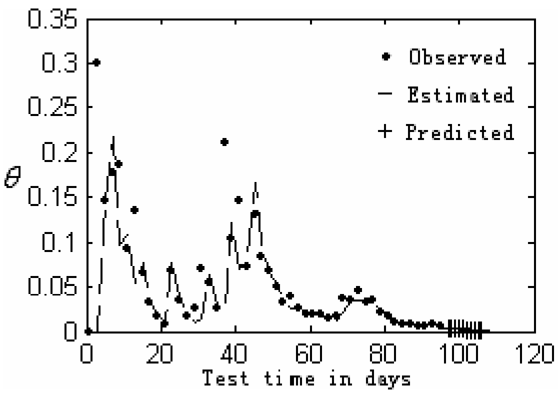

Fig.2 $\theta(k)$, $\hat{\theta}(k)$ **and** $\hat{\theta}(N+p\mid N)$

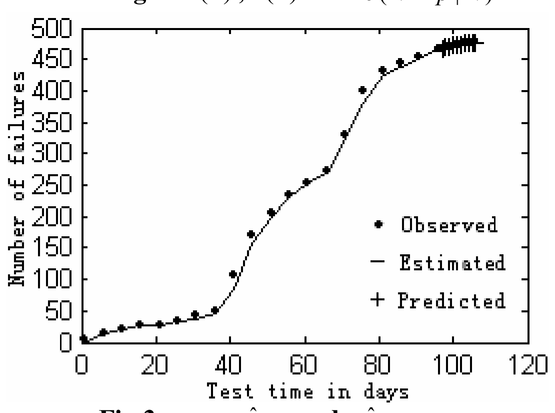

Fig.3 $M(k)$, $\hat{M}(k)$ **and** $\hat{M}(N+p\mid N)$

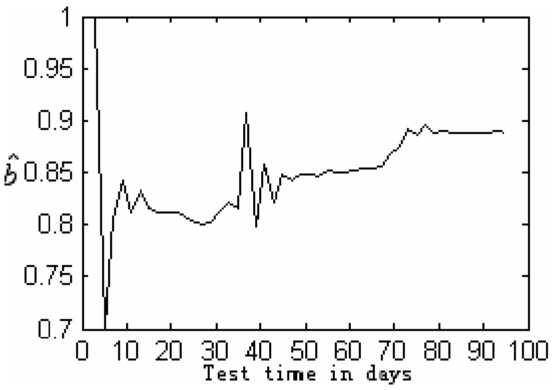

Fig.4 Parameter b

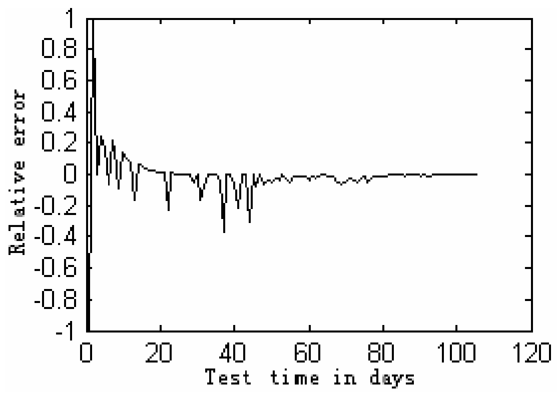

Fig.5 Relative error

Table 1 and Table 2 show the estimation and prediction of cumulative number of software failures. We can find that the relative errors are very small. Fig.1 illustrates the effect of time series $\eta(k)$. Fig.2 gives the curve of parameter $\theta(k)$ and its estimation and prediction, $\theta(k)$ denotes the changing of direction and speed. In this Figure, besides lacking testing data in the beginning of estimation, all estimations of $\theta(k)$ are less than 0.15. We can have the result that the trending of software reliability is increasing step by step. Fig.3 shows the observed, estimated and predicted result of time series nonlinear model $M(k)$.

In Fig.4, we get the curve of parameter b. In the processing of error removal, b is not a constant value and it changes continuously. We can see that this estimation accords with the characteristic of software testing error removal. Parameter b can show the speed and efficiency of error removal. In the beginning of test, the degressive speed of parameter b is very fast, it shows that the efficiency of error removal is very high and the software reliability increases rapidly. In the middle phase, b is stable relatively which indicates the efficiency of error removal is unchangeable. After stated periods, b increases and it shows that the efficiency of software error removal is lower and the growth of software reliability is slow down. In that new errors can be introduced into the software during the error removal process and the error numbers are more than that of removal, we must find out the reason and get a solution. At last, b stabilizes to 0.0014 and it shows that the efficiency of error removal has no change and software reliability is increasing steadily.

Fig.5 illustrates the relative errors and shows good evaluation results of the new nonlinear model.

6 Software Reliability and Its Fuzzy Evaluation

From Fig.2, we can see that when $\theta(k)$ is closer to zero, the number of remaining errors is less and software reliability is higher. So the objective of software reliability can be designed

$$R(k) = -\log(\hat{\theta}(k)) \tag{14}$$

Simulation result is showed in Fig.6 and Fig.7. From the figures, we can see that software reliability rises on the whole trending. Strictly speaking, $R(k)$ is a fuzzy variable. So we can use fuzzy method to evaluate software reliability after analyzing and forecast reliability based on testing data. We compartmentalize some grades to $R(k)$ and use fuzzy variable to describe the software reliability. The results of fuzzy evaluation can be used to determine the covering range of the linguistic variables and confirm the fuzzy membership grade. Considering the fuzzy of variable $R(k)$, we can establish relevant fuzzy linguistic variables and fuzzy rule to estimate software reliability as tabulated in Table 4.

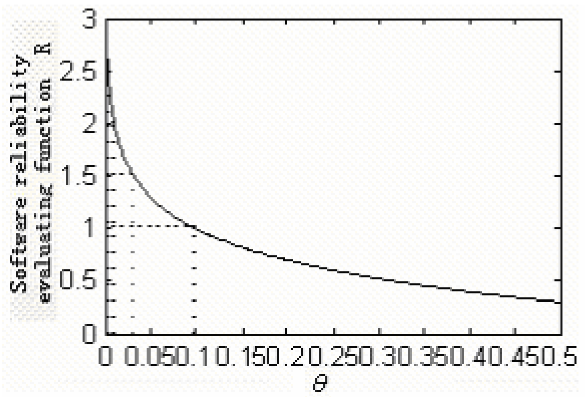

Fig.6 Evaluation function

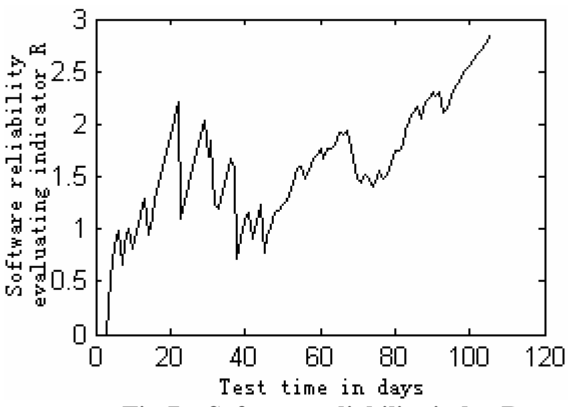

Fig.7 Software reliability index R

Table 4 Fuzzy evaluation of software reliability index

R	$(-\infty,1)$	[1,1.5)	[1.5, 2)	[2,2.5)	[2.5,3)	$[3,+\infty)$
Software reliability	extremely low	Very low	lower	higher	very high	extremely high

Through Table 4, we can get more practical results than former evaluation. For example, one possible result of software reliability fuzzy evaluation is "software reliability is very high and can use soon" or "software reliability is general and the software should be tested further and improve reliability" and so on. In this example, $R(108) = 2.8653$, the evaluation of software reliability is "very high" from Table 4.

7 Conclusion

Although many SRGMs have been proposed since the first one appeared in 1972, no one model can suit all projects. While new projects arose, new SRGMs were created to suit them. It is difficult to propose a universal model to estimate and forecast software reliability.

Through studying, we can find that testing data accord with the properties of time series. So we can apply proper time series model to evaluate software reliability. In the testing and modeling, we can find that nonlinear phenomenon exist in many situations and we can not neglect and deal with as linear model. This paper proposes a new method to establish software reliability.

The new nonlinear forecasting model of software reliability has been proposed. The new model consider the disturb noise of observed data and have no need for strict hypothesis condition and the parameters are changing with time. All these accord with the characteristic of real projects. This method has demonstrated good results in terms of accumulative failures. The nonlinear modeling technique can be used for nonlinear and non-stationary processes of the data. Finally, we introduce fuzzy theory and unify the fuzziness and randomness to evaluate software reliability. The results of simulation example show that the new method presented in this paper is very useful to evaluate software reliability.

References:

[1] FL. Popentiu, D.N. Boros, Software Reliability Growth Supermodels, *Microelectronics and Reliability*, 1996, Vol 36, No. 4, pp. 485-491.

[2] Takamasa Nara, Masahiro Nakata, Akihiro Ooishi, Software Reliability Growth Analysis-Application of NHPP Models and Its Evaluation, *IEEE Transactions on Reliability*, HASE 1996, pp. 222-227.

[3] Huang Xizi, *Software Reliability、Security and Quality Assurance,* Publishing House of Electronics Industry, 2002.

[4] J.Musa, A.Jannino, and K.Okumoto, *Software Reliability- Measurement, Prediction, Application*, McGraw Hill, New York,1987.

[5] J.D. Musa, A theory of software reliability and its application, *IEEE Transactions on Software Engineering*, SE-1(3), September, 1975, pp. 312-327.

[6] AL Goel, K. Okumoto, Time-dependent error detection rate model for software reliability and other performance measures, *IEEE Transactions on Reliability*, R-28(3), August, 1979, pp. 206-211.

[7] PK Kapur, S. Younes, Modelling an imperfect debugging pheonmenon in software reliability, *Microelectronics and Reliability*, 1996, vol.36, pp. 645-650.

[8] Z. Jelinski and PB Moranda, Software Reliability Research, *Statistical Computer Performance Evaluation*, Academic, New York, 1972, pp. 465-485.

[9] Amerit L Goel., Software reliability models: Assumptions,limitations and applicability, *IEEE transaction on software engineering*, 1985, SE11 (12), pp. 1411~1425.

[10] Zhang, X.; Pham, H., An analysis of factors affecting software reliability, *Journal of Systems and Software,* 2000, pp. 43-56.

[11] Wang ZL., *Time series Analysis,* Publishing House of Chinese Statistics, 2000.

[12] Granger Clive WJ, Teräsvirta Timo, A simple nonlinear time series model with misleading linear properties, *Economics Letters*, 62 ,1999, pp. 161-165.

[13] Stephen HKan, *Metrics and Models in Software Quality Engineering* Second Edition, Publishing House of Electronics Industry, 2004.

[14] Dengzili, *Theory and Application of Optimal Filtering: Modern Time Series Analysis Method,* Publishing House of Harbin Institute of Technology, 2000.

[15] M. R. Lyu, Ed., *Handbook of Software Reliability Engineering*, McGraw Hill, 1996.