

Chapter 3

Neural Network Based Software Reliability Prediction with the Feed of Testing Process Knowledge

Tian Jie, Zhou Yong and Wang Lina

Abstract Software reliability is an important factor for evaluating software quality in the domain of safety-critical software. Traditional software reliability growth models (SRGMs) only uses the failure data produced in a testing process to evaluate the software reliability and its growth. However, the number and severity of the failures uncovered are determined by the effectiveness and efficiency of testing process. Therefore, an unbiased reliability prediction has to take test process knowledge into account. In this paper, we proposed a neural network based reliability prediction method to utilize the testing process metrics to correlate testing process knowledge with failure prediction. The metrics designed in this paper cover information from the system under test (SUT), design of testing, software failure and repair process. The method is validated through the testing process data collected from a real embedded operating system. And the results show a fairly accurate prediction capability.

Keywords Software reliability · Testing process · Neural network

T. Jie (✉)
Bei Jing Command College of CPAPF, Beijing, China
e-mail: happyjie717@126.com

Z. Yong
China Machinery TIDI Geotechnical Engineering Co.,Ltd, Wuhan, China

W. Lina
National Key Laboratory of Science and Technology on Aerospace Intelligent Control,
Beijing Aerospace Automatic Control Institute, Beijing, China

3.1 Introduction

Software plays a very important role in our modern life and is used in many applications. It has become the core of many safe-critical systems: communications, monitoring systems, automobiles, airplanes, and so on. All of these applications demand high-quality software. As a critical factor for software quality, software reliability has become more and more important in computer system. Software reliability is defined in ANSI, which is *the probability of failure-free software operation for a specified period of time in a specified environment* [1]. During the last 3 decades, there are many researches around this area, and many software reliability growth models (SRGMs) have been proposed for evaluating software reliability. The basic principle of SRGMs is that computes software reliability through the observed failure data and failure time, and then checks whether the reliability meets the desired requirement. SRGMs can be generally divided into two categories: one is the parametric models, such as nonhomogeneous Poisson process (NHPP) models, Schneidewind models and Musa's Basic execution time model; the other is non-parametric models, which is based on neural network, support vector machine (SVM) and so on. All these proposed models work on software failure data. Through the analysis and test on the collected failure data, we can choose appropriate model to evaluate and predict software reliability.

Failure data is one of the productions of testing process. The number and seriousness of the failures uncovered are determined by the effectiveness and efficiency of testing process. Therefore, we cannot get reliable estimation of the practical reliability only with the failure data. The estimation could be biased toward the low effective or efficient testing. Therefore, an unbiased reliability prediction approach has to take test process knowledge into account. At present, there are some SRGMs have been taken into account relevant factors, such as test coverage. We focus on how to measure and evaluate the testing process to identify more practical factors to feed into the software reliability model to get reliable estimations.

In this paper, we propose a machine learning method to correlate test process knowledge with failure prediction based on neural network. The test metrics used include the information from system under test (SUT), design of testing, software failure and repair process, etc. This method can improve the confidence of software reliability.

The rest of this paper is organized as follows: [Sect. 3.2](#) introduce some related work about software reliability prediction. [Section 3.3](#) describe the proposed method about multi-view analysis of software testing process. In [Sect. 3.4](#), we design the method to predict software reliability based on neural network. [Section 3.5](#) shows the experimental results and finally we conclude this paper in [Sect. 3.6](#).

3.2 Related Works

In this section, we first briefly introduce the related works about software testing process in software reliability, and then we introduce some research about neural network based software reliability.

Many studies have indicated that during the software testing process, fault detection rate depends on the test team skills, the program size and software testability factors. Kapur et al. [2] proposed that the fault repair rate can be expressed as a function of testing effort and testing time. Huang et al. [3] pointed out when the testing effort metrics spent on fault detection is not constant, the traditional S-curve may not be suitable for the software failure data. In order to solve this problem, they first evaluated the testing effort function (TEF), and combined the logic testing effort function with the exponential and S-shape software reliability model.

Meanwhile, there are some discussions about software reliability focus on that the execution time cannot be the only measure factors for software failure behavior. In order to improve the accuracy of the prediction, many researchers have taken more factors into account and improved the software reliability models. As the important factor to display the completeness and effectiveness of the testing process, test coverage became a software reliability metrics. Malaiya et al. [4] proposed a logarithm-exponent model, Cai [5] also proposed a test coverage-reliability model (TC-SRM). Their model estimated the failure intensity function based on the testing calendar time and test coverage. However, these models only considered a few classic of test coverage, i.e. code coverage. An and Zhu [6] proposed the concept of integrated test coverage (ITC). They estimated the software reliability though combining multi kinds of test coverage metrics to an integrated parameter.

There are also some researches related to the fault repair process. Schneidewind [7, 8] first modeled the failure repair process. Based on this model, many researchers extended the works.

Since the approaches mentioned above considered the testing process for predict the software reliability. We found that many models depend on a priori assumptions and they have not considered the relationship between the different metrics of the process. In recent years, neural network approach has been used to predict software reliability. This method is more flexible and usable, so we introduce some work about neural network based software reliability.

Karunanithi [9] first used some kind of neural networks to estimate the software reliability. Cai et al. [10] proposed using the recent 50 inter-failure times as the multiple-delayed inputs to predict the next failure time. Tian and Noore [1] proposed a software reliability prediction method based on multiple-delayed-input and single-output.

From existing research, we found the testing process has become an important factor for software reliability, and the neural network can provide a flexible application for predict software reliability. And the multiple-delayed-input and

single-output method can help us considering complex factor to prediction. In this paper, we proposed Neural Network based Software Reliability Prediction with the Feed of Testing Process Knowledge. In order to give an accurate prediction, we first analyze the testing process. Furthermore, we will show how to link the testing process factor with the software reliability prediction.

3.3 Multi-Perspective Testing Process Analyses

Safety-critical software requires a high reliability. However, in the actual project, we found that using the failure data and failure time to estimate the software reliability is not accuracy. The most important reason is that the testing process directly determines the veracity of the failure data, in addition, the complexity of the system under test (SUT) determine the testability and the number of the potential failure numbers. Testing process is the last stage of software development; we can collect the process data as many as possible. We propose a method which through analyze different perspective on the whole testing process to create a complete comprehensive for software test.

Based on the above understanding, we combine with the characteristic of the software reliability and proposed a method to analyzing the software reliability based on the three perspectives: SUT, Testing process and the Failure data.

In order to analyze the impact of different perspectives on the software reliability, we are following three clues to do the process analysis and the information extraction. For every clue, we first give the information which needs collection, then we give the effect analysis of these information.

3.3.1 System Under Test

We choose the SUT as the first perspective because SUT reflect the testability of the system.

As we know, the complexity of SUT determines the testability and the numbers of failures. Generally, we think that the bigger numbers of LOC and McCabe, the more failures will be existed in the system. The relativity of the complexity of SUT and numbers of failures is positive. We choose the LOC and McCabe as the measure index.

3.3.2 Testing Process

Testing process is the key factor for the software reliability, because it determines the capability of detecting failures.

In this paper, we take the testing effort, design of the test case and the execution of the test case into account.

In the testing process, we measure the coverage of the test case, including the numbers of the normal test and robust test. Link with the SUT, we also observe the distribution of the test case based on the size and complexity (i.e. McCabe) of the SUT.

In this paper, we choose the number of test cases, execution of test case and test coverage as the measure target. We define the efficiency of the testing as the failure numbers divide the test case numbers. In practice, test case numbers is larger than failure numbers. In order to keep the data consistent, we compute Effective of hundred test cases. Equation 3.1 defines the EoTC:

$$\text{EoTC} = \frac{\text{Number of failure}}{\text{NoTC}/100}. \quad (3.1)$$

3.3.3 Software Failure

Failure is the core factor for the software reliability; we not only collect the failure numbers but also separate the test case which is detecting the failure.

Software failure is the kernel factor of software reliability; it can reflex the situation of the system. In this study, we only concerned about the failure numbers; we will use the knowledge of testing process and SUT to predict failure numbers. In the future work; we will take other factors into account.

Figure 3.1 shows the relationship of the effect factors and software reliability, we divide effects into three perspective based on the three clues. In the figure, blue nodes express the effect factors we will utilize in the software reliability prediction.

3.4 Software Reliability Predictions

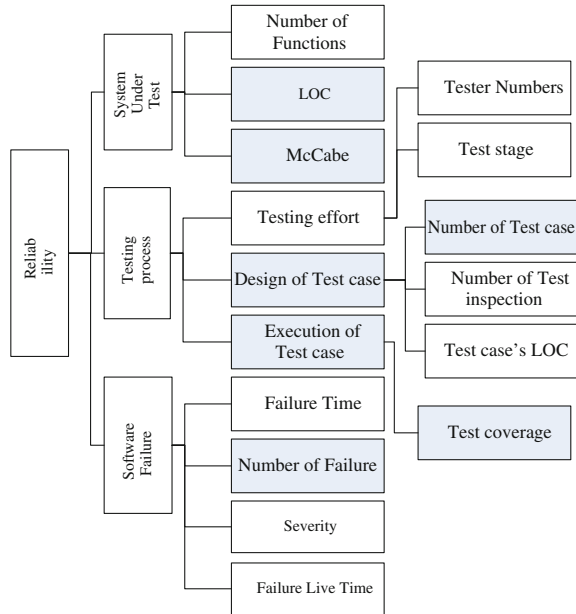
After we collect the data of the SUT and testing process, we can predict the software failure numbers based on neural network.

3.4.1 Neural Network

Neural network can solve following problem: training set contain complex sensor data with noise, and the problems need more symbolic expression.

The characteristics of such problems are: the instance can be expressed as “property-value”, and the output of the objective function may be discrete values,

Fig. 3.1 Multi-perspective of testing process



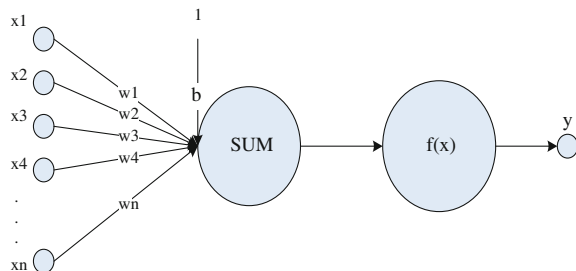
real values or vectors which are contained some discrete property and some real property, the training data may contain errors and so on. These characteristics are very fit for our problems. First, we have a series of process data, and then we use these data to predict the failure numbers. Figure 3.2 give the sketch map of the neurons. w is a real constant, which express the weight of the input. w determines the input x on the neuron of the contribution rate for the output.

Equations 3.2 and 3.3 give the mathematical expression of the neuron.

$$y = f(WA' + b) \tag{3.2}$$

$$y = \sum_{i=1}^n w_{ji}x_i + b \tag{3.3}$$

Fig. 3.2 Neurons: $x_1 \sim x_n$ individual component of the input vector, $w_1 \sim w_n$ weight of each input, b bias, f transfer function, y neuron output



We choose the most commonly used sigmoid function as the activation function. Equation 3.4 give the expression of the sigmoid function.

$$f(a) = \frac{1}{1 + e^{-a}}. \quad (3.4)$$

3.4.2 Design Neural Network

We choose a most common type of neural network architecture which named feed-forward network.

We use the data obtained from the previous section which contains the knowledge of testing process as the input of the neural network. During the training process, we use the Levenberg–Marquardt (LM) algorithm and Gaussian-Newton algorithm to achieve fast convergence. Our work supposes that the ability of testers is stable, and the quality of different test case is equality.

3.5 Experiments

3.5.1 Data Sets

Our datasets come from a real operating system. We choose kernel module of system as the research data, and the description of the data sets is shown in Table 3.1.

In the experiment, we will partitions the data sets as different modules. In dataset DS1, we use the first 25 modules which cover the 84 % of all modules to training the system. For the dataset DS2, the training data are the first 10 modules coving the 76 % of all modules in DS2. Both of two dataset, the remaining module are used for prediction failure numbers. Through predict the number of failures, we predict the software reliability.

Table 3.1 Real data sets

Module	Data		
	Description	Number of test case	Number of failure
DS1	Kernel module	976	25
DS2	Library functions	725	186

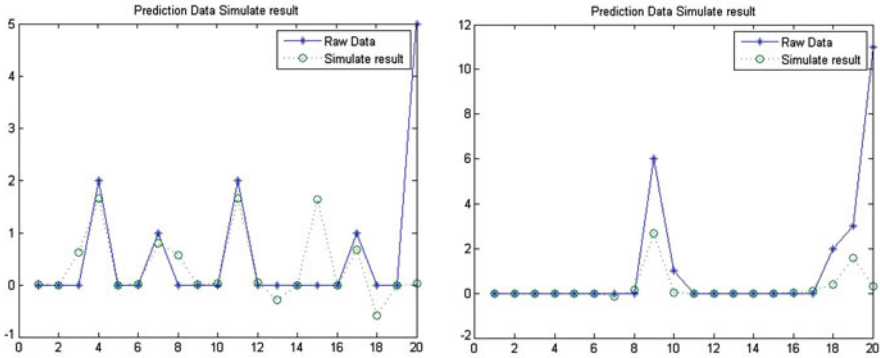
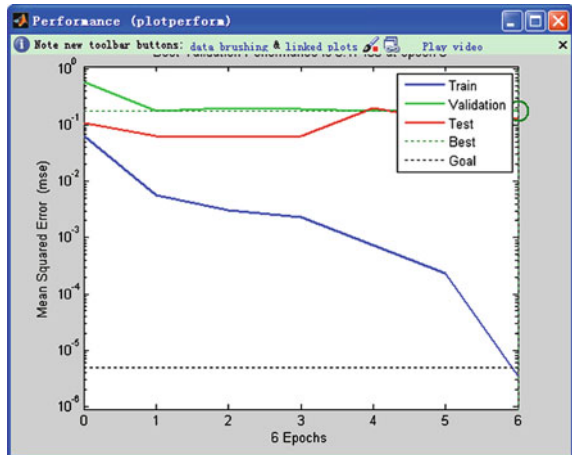


Fig. 3.3 a Predict result of DS1. b Predict result of DS2

Fig. 3.4 Performance of network



3.5.2 Prediction Result

Table 3.1 gives the number of test case and failure numbers in DS1 and DS2. We use first 20 data to train the network, and use last 20 data to predict failure numbers. We compare the reality data and the simulate data, and then compute the mean squared error.

Figure 3.3a shows the predict result of DS1 and Fig. 3.3b give the predict result of DS2. X axis describe the different modules, and y axis give the predict failure numbers of corresponding module.

Figure 3.4 give the performance of neural network. In figure, we can see that when the training reaches the 6th epoch, mean squared error (MSE) is 10^{-1} . In our experiment, we only predict the failure numbers of the module. In fact, based on the raw data, we can calculate the cumulative of the failure numbers.

3.6 Conclusion

In this paper, we proposed a neural network based software reliability method which is depend on the knowledge of software testing process. We first introduce the method, and then we give a real example. This method depends on adequate knowledge of testing process, so we need collect raw data go with the whole testing process. But it can give us a more close to reality result. In the future, we will extract more refined process data, and based on this work, we will predict the cumulative failure numbers.

References

1. Tian L, Noore A (2005) Evolutionary neural network modeling for software cumulative failure time prediction. *Reliab Eng Sys* 77:45–51
2. Kapur PK, Goswami DN, Bardhan A, Singh O (2008) Flexible software reliability growth model with testing effort dependent learning process. *Appl Math Model* 32(7):1298–1307
3. Huang CY, Kuo SY, Lyu MR (2007) An assessment of testing-effort dependent software reliability growth models. *IEEE Trans Reliab* 56(2):198–211
4. Malaiya YK, Li N, Bieman JM, Karcich R (2002) Software reliability growth with test coverage. *IEEE Trans Reliab* 51(4):420–426
5. Cai X (2006) Coverage-based testing strategies and reliability modeling for fault-tolerant software systems. Ph.D. thesis, The Chinese University of Hong Kong
6. An J, Zhu J, (2010) Software reliability modeling with integrated test coverage. In: Fourth IEEE international conference on secure software integration and reliability improvement, Singapore
7. Schneidewind NF (2001) Modeling the fault correction process. In: Proceedings of the 12th international symposium on software reliability engineering (ISSRE), Hong Kong, China, 185–190
8. Schneidewind NF (2003) Fault correction profiles. In: Proceedings of the 14th international symposium on software reliability engineering (ISSRE), Denver, USA, 257–267
9. Karunanithi N, Malaiya YK (1992) The scaling problem in neural networks for software reliability prediction. In: Proceedings of the third international IEEE symposium of software reliability engineering, Los Alamitos, CA, 76–82
10. Cai KY, Cai L, Wang WD, Yu ZY, Zhang D (2001) On the neural network approach in software reliability modeling. *J Syst Softw* 58(1):47–62