Embedded Software Reliability Prediction Based on Software Life Cycle

Ting Dong[1*], Hui Shi[2], Yajie Zhu[1], Kai Li[1], Fengping Chai[1], Yan Wang[1]
[1]*Beijing Institute of Space Mechanics & Electricity, Beijing Key Laboratory of Advanced Optical Remote Sensing Technology*
Beijing,China
[2]*Computer and Information Technology College, Liaoning Normal University*
Dalian, China
dongting1129@163.com, shihui_jiayou@lnnu.edu.cn, 570383726@qq.com, lkbjcn@sina.com, 2661865604@qq.com, wysister@163.com

***Abstract*—In order to guarantee the quality of embedded software, based on the software life cycle, a BP neural network is proposed to predict the software reliability. First analyze the various factors that affect the reliability of the software, and then select the metrics that affect the reliability of the software based on relevant standards and engineering practices. The software reliability measurement data in the actual project was collected, and the established software reliability prediction model is used to predict the software module defects, and the prediction results are compared with the real results. The comparison results show that the model can effectively predict the number of software module defects and effectively indicate the test key module for the software unit test work.**

***Index Terms*—software reliability prediction, BP neural network, software life cycle, embedded software**

I. INTRODUCTION

At present, embedded systems have been widely used in information appliances, intelligent control, military electronics and other fields, embedded software has played an increasingly important role. Its features include high real-time, strict timing, high security, high reliability, harsh environment, high precision, large scale, etc. [1]. In some important areas, the loss caused by software failure is huge, so reliability becomes the most important indicator for evaluating a software quality, and software reliability prediction is particularly important.

Key factors in the implementation of software reliability engineering include reliability definition, operational profile determination, reliability modeling and measurement and reliability certification [2]. Software reliability early prediction can make development and test personnel have a general understanding of the reliability of the software before testing, can seize the opportunity in time, help to evaluate the software design, optimize the design strategy, reduce the huge design changes, improve the software process, and it is very important for cost evaluation, resource planning, program validation and quality prediction in software management. The quality of testing work directly affects the quality of embedded products [3,4]. The prediction of software reliability is mainly to predict the defects of the software.

According to the characteristics of embedded software and software life cycle, this paper introduces the principle and method of metric element selection for software reliability prediction. Based on this, a software reliability prediction model based on BP neural network is proposed. The metric data in the actual project is collected, and the software reliability prediction is performed using this method.

This work is supported by the National Youth Science Foundation of China (61601214)
* is corresponding author

978-1-7281-2348-6/19/$31.00 ©2019 IEEE

II. SELECTION OF METRICS

Static defect prediction technology is a technique for predicting software defects by collecting product metric data which is based on the development process and the measurable characteristics of the products produced in the process. Therefore, the selection of metrics is the first step in software reliability prediction.

A. Principle of Reliability Metrics Selection

- Focus on reliability. There are many factors in the software life cycle that can affect software reliability behaviors, such as design methods, developer quality, management level, development environment, test strategy, maintenance activities, and so on. Software reliability is also related to development costs and deadlines for delivery. Factors that are closely related to software reliability should be selected as reliability indicators in the project.
- Integrity. The selected software reliability metrics should reflect the activities and levels of the software development process as completely and completely as possible. Therefore, metrics should be used throughout the development process and include technical metrics and management metrics.
- Applicability. The metric applies to a certain period of the software life cycle; the metric is consistent with the characteristics of the software being developed; the metric data can be collected through the products of this phase and the data is unambiguous.
- Ease of use. Metrics can easily be used to reflect the level of reliability, and metrics should be easy to be collected.

B. Selection of Reliability Metrics

High reliability and long life embedded software uses a V-shaped software cycle model [5]. The main idea of the V-shaped model is that development and testing are equally important. The left side represents development activities, while the right side represents test activities. For each stage of development, there is a test level corresponding to it. This model is suitable for situations where requirements are clear and requirements are changed infrequently. The V-shaped model is shown in Fig. 1.

The V-shaped model divides the software life cycle into nine stages, including system requirements analysis, software requirements analysis, summary design, detailed design, software coding, unit testing, assembly testing, confirmation testing, and system testing.

Authorized licensed use limited to: Carleton University. Downloaded on September 20,2020 at 10:20:21 UTC from IEEE Xplore. Restrictions apply.

System requirements analysis stage and software requirements analysis stage: It is to analyze the feasibility of

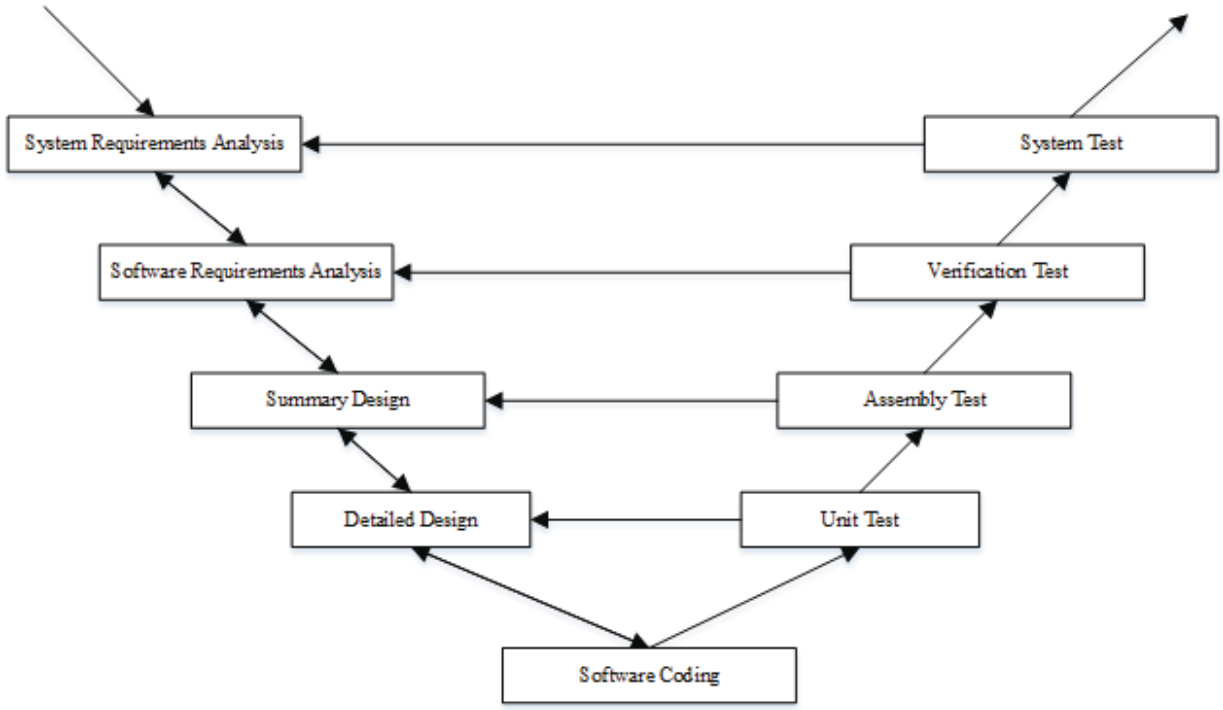

Fig.1 V-shaped life cycle model

software development, and analyze the various functions that the software needs to implement, and select the corresponding software development team to carry out software development. The project management level of the software development team in the software development process It is an important factor in determining software reliability.

The Capability Maturity Model for Software is a standard used to assess the maturity of an organization's software development capabilities. It defines the main management processes and engineering processes throughout the software lifecycle. According to the activities in the software life cycle, the software development team's software development capability maturity is divided into five levels, namely, initial level, repeatable level, defined level, managed level and optimization level [6]. The development team of the embedded software is generally certified by the corresponding level, which is generally managed or defined. At this stage, the engineering management level of the software development team is selected as the metric of reliability. According to the different levels of management, the metric values are defined for the maturity of the corresponding level, as shown in TABLE I.

Software design stage: including summary design and detailed design, the main task of the summary design is to divide the module of a complex system by function, establish the hierarchical structure of the module and call relation, determine the interface between the modules, etc. Detailed design is a refinement of the summary design, a detailed description of the functions performed by each module, transforming the functional description into a precise, structured process description. Card and Glass define three software design complexity metrics [7]: structural complexity, data complexity and system complexity. Henrry and Kafura also propose architectural design metrics. The above metrics are closely related to the module's average fan-in number and the module's average fan-out number. Therefore, the fan-in and fan-out of the software are selected as the metrics of this stage. The most widely discussed complexity measure of computer software is cyclomatic complexity, which was first proposed by Thomas McCabe. Therefore, in this stage, the fan -in number of the module, the fan-out number of the module, and cyclomatic complexity of the module are selected as the metric elements of the design stage.

Software coding stage: This stage converts the detailed design content of the software into specific program code. The effective length of the code is an important factor affecting the reliability of the software. Therefore, at this stage, the effective length of the code is selected as the metric element.

In summary, the metric elements in the early stages of the software life cycle are selected as shown in TABLE II.

TABLE I. SOFTWARE DEVELOPMENT CAPABILITY MATURITY AND METRICS

Maturity Level	Manage Metrics
(1)Initial	0
(2) Repeatable	0.4
(3) Defined	0.8
(4) Managed	0.9
(5) Optimizing	1

TABLE II. SOFTWARE RELIABILITY METRIC

Software Life Cycle Stages	Reliability Metrics	Metrics Identifier
Requirements analysis	Development team maturity level	I
Software design	Fan-in number of the module Fan-out number of the module Cyclomatic complexity of the module	II III IV
Software coding	Code effective length	V

Authorized licensed use limited to: Carleton University. Downloaded on September 20,2020 at 10:20:21 UTC from IEEE Xplore. Restrictions apply.

III. RELIABILITY PREDICTION MODEL BASED ON BP NEURAL NETWORK

A. BP Neural Network

The artificial neural network model is a mathematical model that simulates the working principle of the human brain neural network. It is composed of a large number of neural network processing units through different connections, and is a large-scale nonlinear adaptive system with powerful processing capabilities. The Back Propagation neural network, is a widely used neural network model. The learning process of BP neural network consists of two processes: forward propagation of input information and back propagation of error. First, the input sample data is input from the input layer to the network, and after being processed by each hidden layer, it is transmitted to the output layer to obtain a network output. Then calculate the error value between the actual output of the network and the expected output, and go to the back propagation phase. After many times of forward propagation of information and back propagation of errors, the weights are continuously adjusted until the error satisfies certain accuracy conditions, which is the training process of BP neural network [8].

The BP neural network has the following characteristics: First, the neural network contains one or more hidden layers. These hidden neurons are connected to the input layer and the output layer, which can extract more pattern features from the input data, which is conducive to BP neural network learning complex tasks. Second, BP neural network has strong nonlinear mapping ability, which is suitable for solving complex problems. Third, BP neural network has strong generalization ability and can learn from sample data, abstract the features contained in it, and apply it to the new sample data.

B. Reliability Prediction Model

Software reliability prediction is implemented before the unit testing and after the software coding, which facilitates timely feedback of information into the software development process. Software reliability prediction is based on the development process and the metric elements of the products produced in the early stages of the software lifecycle. The metric data is collected and used to predict the number of defects in the software module.

In order to determine the entire BP neural network model, the model parameters need to be determined from the following aspects.

- Network topology design. The topology of the BP neural network is a feed forward network with three or more layers and no feedback, and the inner layers are not interconnected. These include the input layer, the hidden layer, and the output layer. The theory has proved that the three layers feed forward neural network can approximate any continuous function with arbitrary precision [9]. The BP neural network used in this paper is a three layers structure. When the training samples of the BP neural network are determined, the number of neurons in the input layer and the output layer of the network is also determined. In this model, the number of neurons in the input layer is five, and the number of neurons in the output layer is one. The number of training samples, the influence of noise data, and the complexity of the patterns contained in the samples all affect the number of neurons in the hidden layer, so the number of hidden layer neurons is continuously adjusted according to the performance of the model. The number of hidden layer nodes is generally selected according to the following empirical formula as in (1). In the formula, n=5 is the number of input nodes; m=1 is the number of output nodes; a is a constant between 1 and 10. Therefore,n_1 ranges from 3 to 13.

$$n_1 = \sqrt{n+m} + a \tag{1}$$

- Determination of each layer function. In the single hidden layer BP network training, the hidden layer transfer function generally uses the tansig function, and the output layer transfer function uses the purelin function.
- Design of initial weight of network. The initial weight of the network should be small enough that the network training does not start from a local minimum on the error surface. So in this model, the initial value of the weight matrix is assigned using a random value between (0, 1).
- Design of network training algorithm. The training algorithm selects trainlm by default. The input data is provided to the network for the pair of metrics and the number of defects, and the error signal between the actual number of defects output and the number of defects expected to be output is calculated according to the BP model. Propagation adjustment weights make the error between the actual number of defects and the number of defects expected to be output smaller and smaller.
- Conditions for network termination training. First, the accuracy of the model, that is, the size of the error. In this model, the maximum error is set to 0.18, that is, when the error of the model is less than 0.18 the training network can be stopped. Second, the maximum number of cycles, set to 1000 in this model, that is, when the number of times the model is trained is more than 1000 times, the network training can be stopped. Third, the learning rate setting should be set to a small value. Although setting large value will speed up the convergence at the beginning. However, when it comes to the best point, it will cause turbulence, which will make it impossible to converge, so the learning rate is set to 0.01.

According to the above analysis, a software reliability prediction model is established, as shown in the Fig.2.

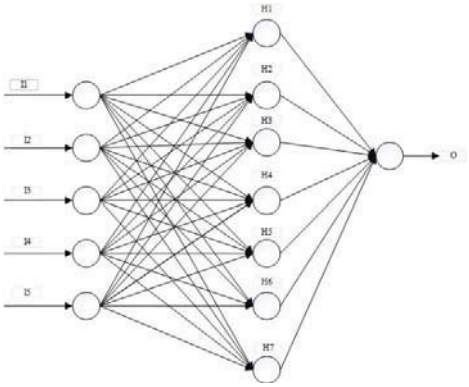

Fig.2 Software reliability prediction model

Authorized licensed use limited to: Carleton University. Downloaded on September 20,2020 at 10:20:21 UTC from IEEE Xplore. Restrictions apply.

C. Sample Training

This paper collects metric data of a total of 200 modules of embedded software, of which 190 modules are used as training samples for neural networks and 10 modules are used as modules to be tested. TABLE III shows an example of a training sample.

In this paper, MATLAB [10] is used as the simulation prediction platform. After determining the parameters and input data, BP network training is carried out. The main code is as follows.

```
[P1]=xlsread('test',1,'A2:E191');
% Read training sample input data
[T1]=xlsread('test',1,'F2:F191');
% Read training sample output data
P1=P1'; % Transpose the training sample input data
T1=T1'; % Transpose the training sample output data
[P1,ps1] = mapminmax(P1);
% Normalize training sample input data
[T1,ts1] = mapminmax(T1);
 % Normalize training sample output data
[P2]=xlsread('test',1,'A192:E201');
 % Read test sample input data
[T2]=xlsread('test',1,'F192:F201');
% Read test sample output data
P2=P2'; % Transpose test sample input data
T2=T2'; % Transpose test sample output data
P2 = mapminmax('apply',P2,ps1);
% Normalize test sample input data according to training sample input data
TF1 = 'tansig';TF2 = 'purelin';% Set training function
net=newff(P1,T1,[7],{TF1,TF2},'trainlm');
% Create a network
net.trainParam.epochs = 1000;
% Set the number of trainings
net.trainParam.goal=0.018;
 % Set convergence error
net.trainParam.lr=0.01;
 % Learning rate setting
net.iw{1,1}= rands(7,5);
% Generate initialization weights, threshold matrix, (0,1) random matrix
net.lw{2,1}= rands(1,7);
net.b{1}= rands(7,1);
net.b{2}= rands(1,1)
net.divideFcn = ""
```

```
[net,tr]=train(net,P1,T1); % Training network
Ttrain2=sim(net,P2); % Test sample prediction
Ttrain2=mapminmax('reverse',Ttrain2,ts1);
 % Predictive result reduction
err2=Ttrain2-T2; % Calculation error
save('D:\matlab work\BPYC.mat','net');
 % Save training results
```

After training the BP neural network, the weights w1, w2 and the thresholds b1, b2 are obtained as shown as follows.

$$w1 = \begin{bmatrix} -1.1334 & -0.6204 & -2.1896 & -0.2405 & 0.4984 \\ -18.8038 & 5.0792 & 27.4074 & -84.1827 & -70.6009 \\ -0.7289 & 1.8734 & -4.8728 & -4.2448 & -2.1506 \\ -1.1402 & 1.8381 & -0.8006 & -2.5753 & 0.8447 \\ -1.4421 & 1.678 & 0.997 & -2.424 & 0.8098 \\ 1.0253 & 0.6247 & 2.2218 & 0.2967 & -0.5321 \\ -0.1406 & -3.9914 & -0.0276 & 2.148 & -1.1491 \end{bmatrix}$$

$$w2^T = \begin{bmatrix} 91.5917 \\ -0.5745 \\ -1.5435 \\ 45.4765 \\ -48.9170 \\ 88.4864 \\ -151.7277 \end{bmatrix} \quad b1 = \begin{bmatrix} -1.3295 \\ -29.9512 \\ -1.168 \\ -1.0724 \\ -1.1472 \\ 1.4685 \\ -5.7858 \end{bmatrix} \quad b2 = [-153.4345]$$

It can be seen from the training results in FIG.3 that in the first 30 trainings of the model, the rate of error decline is obvious, but after 50 trainings, the decreasing amplitude of the error is gradually reduced, that is, the learning efficiency is gradually reduced, and the minimum mean square error 0.018 is achieved after 90 trainings.

D. Experimental Analysis and Verification

In order to verify the validity of the prediction model, this paper uses 10 software modules to verify the number of predicted defects and the number of real defects as shown in the TABLE IV. It can be seen from TABLE IV that the

TABLE III. TRAINING SAMPLE EXAMPLE

Software Module	1	2	3	4	5	6	7	8
Metric I	0.8	0.4	0.8	0.8	0.8	0.4	0.8	0.8
Metric II	13	3	5	7	3	4	3	9
Metric III	1	1	1	1	1	2	2	1
Metric IV	1	1	1	5	1	1	1	5
Metric V	453	64	102	439	86	81	93	350
Defects Number	7	5	6	8	6	7	4	8

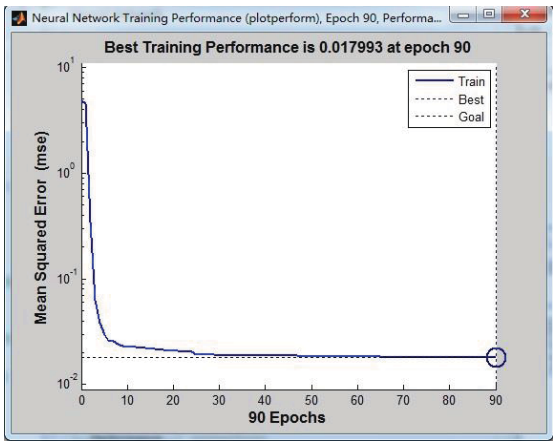

Fig.3 Training Result

Authorized licensed use limited to: Carleton University. Downloaded on September 20,2020 at 10:20:21 UTC from IEEE Xplore. Restrictions apply.

TABLE IV. COMPARISON OF PREDICTED AND ACTUAL VALUES

Module	1	2	3	4	5	6	7	8	9	10
Predicted value	5	5	4	4	3	5	4	6	3	4
Actual value	4	6	5	3	2	5	6	5	3	3

predicted defect value is basically consistent with the true value, and the number of defects of the software module can be predicted, which serves as a reference for the software unit test.

IV. CONCLUSION

Based on the life cycle of embedded software, this paper introduces the selection principle and method of software reliability metrics, and proposes a method to establish software reliability model by BP neural network. The metric data of each stage of the specific software was collected and model training was carried out, and the reliability prediction of the software was carried out. Comparing the error between the real value and the predicted value, it is proved that the prediction model is effective.

However, BP neural network also has some shortcomings. For example, the network learning speed is slow. Because the neural network needs to extract complex relationships and patterns from the sample data set, it requires a lot of calculations and for different problems, the learning speed will vary according to the learning algorithm. Due to conditions and time constraints, the method proposed in this paper needs further research and improvement. The aspects that need to be improved mainly include: First, reduce the correlation of variables by principal component analysis of metric elements. Second, establish an empirical database to provide rich and objective basis and benchmark for neural network training and software defect prediction.

REFERENCES

[1] Zhu Shi, Xincheng Yuan, Weihua Ma, Yi You, "Reliability Metrics Suitable for Aerospace Software Development",Aerospace Control, Vol.22, pp.87-92, Jun. 2004.

[2] MICHANEL R.LYU, Handbook Of Software Reliabilty Engineering, Publishing House of Electronics Industry, 1997, pp. 123-135.

[3] Chao Wu, Jianping Xu andLirong Chen. "Lifecycle-based software defect prediction technology". Computer Engineering and Design Vol 30, pp.2956-2959, Dec. 2009.

[4] Cuihong Shi, "Static Analysis for Embedded Assembly Program", Spacecraft Recovery & Remote Sensing, Vol. 29, pp.59-62, Jan. 2008.

[5] Haicheng Yang, Aerospace Software Engineering, Beijing: China Astronautic Publishing House, 2009, pp.23-25.

[6] Xingui He, "FRAME AND CONTENT OF SEI'S CAPBILITY MATURTY MODEL", Computer Applications, Vol.21, pp.1-5, Mar. 2001

[7] Roger S Pressman, Software Engineering, Beijing: China Machine Press, 2004, pp.377.

[8] Zhongzhi Shi, Neural Network.Beijing: Higher Education Press , 2009, pp.50-60.

[9] Zhexue Ge, Zhiqiang Sun, Neural Network Theory and the Implementation of MATLAB R2007, Beijing: Publishing House of Electronics Industry, 2007, pp.108-116.

[10] Jun Gao, Artificial Neural Network Principle and Simulation Example , Beijing: China Machine Press,2003, pp.151-153.

Authorized licensed use limited to: Carleton University. Downloaded on September 20,2020 at 10:20:21 UTC from IEEE Xplore. Restrictions apply.