Neueural-Network-Based Approach on Reliability Prediction of Software in the Maintenance Phase

Yung-Chung Chen, PhD* and Xiao–Wei Wang

Abstract—Maintenance of software involves debugging of errors and implementations of enhancement requested by users, these both cause the reliability of software decreased. For the systems that have been used for a considerably long period of time, the various details concerning the initial development phase are usually not known to the users who are responsible for the maintenance of these systems. These cause the estimation of software reliability more difficult. In this paper, a prediction model based on Back-Propagation Neural Network (BPN) is proposed to estimate the failures of the software system in the maintaining phase. The “failure correction” records and the “enhancement” records are chosen as the input data of the prediction model, the future failure time is the output. A numerical example of a commercial Shop Floor Control system (SFC) is used to illustrate the validation and application of the proposed method.

Keywords — Software reliability, Maintenance phase, Back-Propagation Neural Network.

I. INTRODUCTION

In this knowledge-based economy era, not only traditional industries but also high-tech industries are related to the software systems. The construction of software system can raise the competition in many industries. When the demand for software systems increases, the possibility of crises from the software failure will also increase. Therefore, emphasizing on the quality of software has become an important concern in this knowledge-based economy era. However, a high quality software system becomes more difficult to obtain when the software system becomes more and more complicated.

Software reliability is an important facet of software quality. It is defined as “the probability of failure-free software operation for a specified period of time in a specified environment” [1]. Software reliability is concerned with the time between failures or its reciprocal, the failure rate.

There are a great many models and methods for predicting faults and failures in software. Pham's book provides an excellent introduction on Software Reliability Modeling and gives detailed mathematical descriptions of the models used in the book [2]. Non-Homogeneous Poisson Process (NHPP) SRGMs is one of the important analytical software reliability growth models (SRGMs). ANN (Artificial Neural Network) software reliability models are studied more these years [3–11]. In this research, we present the proposed models for the software failures prediction in the maintenance phase based on the Artificial Neural Networks approaches.

II. SOFTWARE FAILURES IN THE MAINTENANCE PHASE

In general, the life-cycle of software system typically includes a requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase. Maintenance phase is defined as the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment [12]. Software maintenance costs continue long after development ends, typically costing between 67 percent and 80 percent of the overall life-cycle cost [13]. Therefore, software reliability in maintenance phase is one of important evaluating subject in the enterprise management arena.

Software maintenance is, of course, far more than “fixing mistakes”, it may be defined by four activities [14]:
1) Corrective maintenance,
2) Adaptive maintenance,
3) Perfective maintenance, and
4) Preventive maintenance.

The last three categories (adaptive maintenance, perfective maintenance, and preventive maintenance, usually called as “non-collective maintenance” or “planned maintenance”) can be planned for with reasonable certainty, while the first (corrective maintenance) is stochastic in nature and is not known with certainty in advance.

In this paper, the term “enhancement” will used for all activities of non-corrective maintenance, and the term “failure
correction” for the corrective maintenance activities. Corrective maintenance needs immediate attention to minimize serious consequences. Many researches, such as [15], show that only about 20 percent of all maintenance work is spent on “failure correction” (corrective maintenance), while the remaining 80 percent is spent on “enhancement” (non-corrective maintenance, including: adapting existing systems to changes in their external environment, making enhancements requested by users, and reengineering an application for future use).

There have already existed several papers considering the difference between the testing and the maintenance phases, such as References [16-21]. In these researches, they describe the fault detection process in the maintenance phase is completely different from the stochastic process in the testing phase, and suppose that the software fault detection process in the maintenance phase followed an SRGM based on non homogeneous Poisson process (NHPP). An environmental coefficient k is introduced, which represents the differences between the environment in the testing phase (effort, test method, and so on) and the software maintenance environment such as the frequency of use.

Unfortunately, in most situations, the software systems are maintained by the system users themselves and such systems may have been used for a considerably long period of time. Various details concerning the development stages are usually not known to the users who are responsible for the maintenance of these systems. In such a situation, the statistical information collected in the testing phase is not available to estimate the software reliability in the maintenance phase.

Moreover, maintenance of software involves all activities of “failure correction” and “enhancement”. Many researchers report that newly developed software has a high failure rate until the errors are worked out, and after this point, failure rates usually drop to a low level [22-23]. However, the software system will continue to have errors and reliability problems, since the software continues to get maintained [24].

In order to overcome such natural complexities, in this work, the Artificial Neural Networks methodology is used to predict the software reliability in the maintenance phase. Artificial Neural Networks are mathematical models based on the processes of the human brain. They are capable of 'learning' complex non-linear behavior for sample data. This makes them an excellent choice for modeling of software reliability in the maintenance phase.

A numerical example of a commercial Shop Floor Control system (SFC) also is presented to illustrate the validation and application of the proposed method.

III. METHODOLOGY

ANN (Artificial Neural Networks) has been used in reliability engineering (such as [25-27]). More theses years, ANN approaches also been studied in software reliability models for predicting software reliability, such as [3-11]. The research results show that the neural network approaches provide accurate forecasting results.

In this section, we present the proposed models for the software failures prediction in the maintenance phase based on the Artificial Neural Networks approaches.

A. Back-Propagation Networks

The most widely used neural network is the Back-Propagation algorithm. This is due to it’s relatively simplicity, together with its universal approximation capacity (such as [28-30]).

Back-propagation neural networks (BPN) are typically composed of three or more layers of nodes: inputs, one or many hidden layers, and an output layer. Hidden and output neuron layers include the combination of weights, biases, and transfer functions. The weights are connections between neurons while the transfer functions are linear or nonlinear algebraic functions. When a data pattern is presented to the network, weights and biases are adjusted so that a particular output is obtained. BPN provide a learning rule for modifying their weights and biases. Once a neural network is trained to a satisfactory level, it may be used as an analytical tool on other data.

B. Software Reliability Data

The inputs to the ANN prediction models consist of “failure correction” records (corrective maintenance, especially for the emergency maintenance) and the “enhancement” records (including adaptive maintenance, perfective maintenance, and preventive maintenance; which we called as “non-collective maintenance” and especially for the planned or predictable maintenance). The “failure correction” records and “enhancement” records are all available in maintenance records of the system. The future failure time is taken as the targets. The inputs and outputs are normalized between –1 and 1.

The global set of experimental data is divided into two randomly selected groups: the training data set, corresponding to 2/3 of the data, and the validation data set, corresponding to the rest 1/3 of the data. Therefore, the generalization capacity of the network could be checked after the training phase.

The aim of software reliability prediction is to accurately estimate the execution time of future failures in based on the software fault history.

C. Model Assumptions

The software failure process is illustrated in Fig. 1, where fi is the execution time of the ith “failure correction” and ej is the execution time of the jth “enhancement” requested by user.

Basic assumptions for the developed software reliability estimate model are stated below:
1) The time for correction is not that trivial to be ignored.
2) The software is operated in a similar manner as that in which reliability predictions are to be made.
3) The maintenance activities (“failure correction” and “enhancement” by users) will contribute to the future failure of the software system.
4) The software system will continue to have errors and reliability problems, since the software continues to get maintained [24].
D. Proposed Prediction Model

Fig. 2 shows the basic architecture of BPN prediction model. Based on the assumptions that the maintenance activities (“failure correction” and “enhancement” by users) will contribute to the future failure of the software system, the past, lagged observations of “failure correction” and “enhancement” data are used as the inputs to the ANN prediction models. The future failure time is taken as the targets. The mapping function g can be described as below:

\[
f_i = g(f_{i-1}, f_{i-2}, \ldots, f_{i-a}, e_{j-1}, e_{j-2}, \ldots, e_{j-b})
\]

where, \(f_i\) is the execution time of the ith “failure correction”, a is the number of the past “failure correction” data related to the ith “failure correction”, and b is the number of the past “enhancement” data related to the ith “failure correction”.

Each set of input patterns is composed of any moving fixed-number “failure correction” data and “enhancement” data within the data series.

Before applying the back-propagation, there are several things needed to define as follows:
1) The ANN architecture selected consists of one input layer, one hidden layer and one output layer.
2) The structure of the ANN consists of 2 neurons in the input layer, 2~8 neurons in the hidden layer, and one neuron in the output layer. It is observed during the preliminary trials that 2~8 neurons in the hidden layer enhances the performance of the ANN.
3) The network consists of hyperbolic tangent activation functions in the hidden layers and linear activation function in the output layer.
4) The ANN model is developed using the procedure of the MATLAB Neural Network Toolbox (Demuth and Beale, 2004).
5) The back-propagation algorithm of the Levenberg-Marquardt approximation is used for ANN training.
6) The criterion that terminates the algorithm (Learning epochs = 1,000)
7) The test datasets are used independently for the evaluation of model performance.

Moreover, we adjust the number of neurons in the hidden layer of neural networks and utilize the variant network structures and algorithms to analyze and modify the results. Therefore, an optimum prediction system is acquired.

Once the ANN prediction models is trained to a satisfactory level by the “failure correction” and “enhancement” record, it may be used as an analytical tool on other data.

E. Measurements of Neural Network Training

Mean average percentage error (MAPE) is applied as a standard performance measure for software reliability prediction in this research. The definition of MAPE is:

\[
MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{l_i - \hat{l}_i}{l_i} \right| \times 100%
\]

where \(n\) represents the number of the test samples, \(l_i\) is the actual failure time of the rth failure, \(\hat{l}_i\) represents the estimated failure time of the rth failure.

IV. NUMERICAL EXAMPLE

We show numerical results of applying the proposed software reliability growth model for maintenance environment. The data is a real data collected from a commercial Shop Floor Control system (SFC) in an electronics works.

Fig. 3. Maintenance records of a Shop Floor Control system.

The data set includes both the “failure correction” data and the “enhancement” data. As shown in Fig. 3, the number of “enhancement” is 183, and 344 for the “failure correction” within 750 days.

The global set of experimental data is divided into two randomly selected groups: the training data set, corresponding to 2/3 of the data, and the validation data set, corresponding to the rest 1/3 of the data. Therefore, the generalization capacity of the network could be checked after the training phase.

Neural Network ToolBox of Matlab7 is used to train and test BPN. Input for the network is a list of 4~16 variables, the past failure time and the past enhancement time related to the subsequently failure date. The output value is the estimated subsequently failure time. Levenberg-Marquardt is chosen as training function, and purelin function is selected as the BPN transfer function. We set hidden layer as one layer, the number of hidden neurons is set as 2~8 respectively, and the training cycles is 1,000 cycles. We performed 10 times iterations per BPN architecture pattern during training and testing to measure
the neural network training.

In the training and testing stage, a lower MAPE would indicate a better performing model. In this example, through the combinations of different learning parameters, the MAPE of training reach the lowest, 0.47%. From the performance level defined by Lewis [31], our model locates at the very good model range.

The comparison of the observed and simulated results (predicted by 8 enhancements and 5 failures, predicted only by 2 failures, predicted only by 5 failures, and predicted only by 8 failures,) of the ANN prediction models for the commercial Shop Floor Control system is depicted in Fig. 4. The performance of the models is assessed using the Mean average percentage error (MAPE). The results of the statistical analysis are summarized in Table I. It is to be noted that the predicted failure time with “failure correction” data and “enhancement” data are much better than the predicted failure time with “failure correction” data only. As shown in Fig. 4, the proposed prediction model based on Back-Propagation Neural Network gets better estimation for software failures.

V. CONCLUSION

In this study we presented a simple and quick method to evaluate the failure for software system in maintenance phase through the use of prediction model based on Back-Propagation Neural Network.

Numerical results show that the predicted failure time with “failure correction” data and “enhancement” data are much better than the predicted failure time with “failure correction” data only.

The proposed prediction model based on Back-Propagation Neural Network gets better estimation for software failures in maintenance phase.

Fig. 4. Observed and predicted failure time for the rest 1/3 of the maintenance records of a Shop Floor Control system.

<table>
<thead>
<tr>
<th>Table I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Statistical Performance of the Model Results for a Shop Floor Control System</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input data</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhancement (requested by users)</td>
<td>8</td>
</tr>
<tr>
<td>Failure correction</td>
<td>5</td>
</tr>
<tr>
<td>-</td>
<td>2</td>
</tr>
</tbody>
</table>

VI. ACKNOWLEDGMENTS

This work was supported in part by a Research Grant from Shu-Te University of Taiwan (ROC), and National Science Council, Taiwan (NSC 95–2815-C-366–005 –E).

REFERENCES


Yung-Chung Chen is currently an Associate Professor at the Department of Logistics Management, Shu-Te University. He was born in Kaohsiung, Taiwan, R.O.C., in 1956. He received his B.S. degree in Mechanical Engineering from Tatung University, Taipei, Taiwan, in 1978, followed by MEng degree in Material Science from National Sun Yat-Sen University in 1990, and PhD degree in Material Science from The University of Tokyo, Tokyo, Japan in 1996. His research and teaching interests are in the area of applications of artificial intelligence, risk analysis & management, logistics, and reliability engineering.

Xiao-Wei Wang is currently an Industrial Engineer, DYACO. He was born in Tainan, Taiwan, R.O.C., in 1984. He received his industrial management from Shu-Te University, Kaohsiung, Taiwan, in 2005, followed by AMBA degree from Shu-Te University 2008. His research interests are in the area of applications of standardization, Process Improvement & Facilities Planning, logistics, and reliability engineering.