International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:11, No:9, 2017

# Neural Network Based Approach of Software Maintenance Prediction for Laboratory Information System

Vuk M. Popovic, Dunja D. Popovic

***Abstract*****—**Software maintenance phase is started once a software project has been developed and delivered. After that, any modification to it corresponds to maintenance. Software maintenance involves modifications to keep a software project usable in a changed or a changing environment, to correct discovered faults, and modifications, and to improve performance or maintainability. Software maintenance and management of software maintenance are recognized as two most important and most expensive processes in a life of a software product. This research is basing the prediction of maintenance, on risks and time evaluation, and using them as data sets for working with neural networks. The aim of this paper is to provide support to project maintenance managers. They will be able to pass the issues planned for the next software-service-patch to the experts, for risk and working time evaluation, and afterward to put all data to neural networks in order to get software maintenance prediction. This process will lead to the more accurate prediction of the working hours needed for the software-service-patch, which will eventually lead to better planning of budget for the software maintenance projects.

***Keywords*****—**Laboratory information system, maintenance engineering, neural networks, software maintenance, software maintenance costs.

## I. Introduction

SOFTWARE maintenance plays a more and more important role in planning and executing new software products. Therefore any new input which can better describe or propose a solution for faster and most cost effective resolution of issues in software maintenance is very useful.

According to ISO/IEC 12207 Software Life Cycle Processes Standard, the maintenance process is defined as a primary process in software life cycle [1]. Therefore evaluation of software maintenance tasks, which is recognized as the most significant and most costly part of the software life cycle [2], can significantly contribute to the efficiency of maintenance management. Those maintenance costs are commonly regarded as the cost of the software developers in maintenance because those developers have to be not only experienced in software development but also to have deep knowledge about the area of the particular software product. Also, software maintenance includes a process of issue investigation and issue verification and validation which could be more costly than in usual software development.

Maintainability is mainly influenced by two project factors: Maintenance task to be performed and people that will perform the task. Those two factors should always be taken into consideration [3].

Typical software development life-cycle consists of following phases: requirement gathering and analysis, design, implementation (coding), testing, deployment, maintenance. And it is stated in [4] that only software maintenance costs are taking between 67 and 80% of the overall software life-cycle costs. To increase the efficiency of services provided to clients and to improve the quality of final software product, it is important to have the knowledge about costs and also a time estimation of maintenance process. This knowledge is giving us an opportunity to improve planning activities.

Software maintenance could be defined as any further work on a software product between the two major software releases. The diversity of maintenance activities depends not only on the domain where software is used but also on software size and frequency of changes [5]. According to Parikh [2], software maintenance includes understanding and documenting existing systems, extending existing functionality, adding new functionality, finding and correcting bugs, answering questions for users and operations staff, training new systems staff, rewriting, restructuring, converting and purging software.

In software solution covered in this paper, there are parts of it which are used at the customer site for more than a decade. Therefore the implementation details concerning the initial development are usually unknown to the developers responsible for the maintenance of this product. This causes the estimation of the software-service-patch even more difficult.

Prediction model, presented in this paper, is based on Backpropagation Neural Network (BPN). This model is proposed to make an estimation of the time needed to develop and deliver one maintenance release of the observed software solution.

The specific records regarding the implementation, business risk, and product risk are chosen as the input data of the prediction model. Important input data are also the original time estimation (time needed for particular maintenance according to developers) and the actually spent time in the past maintenance phase.

Vuk Popovic is a Principal Software Engineer in Roche Diagnostics Berlin and a PhD Student in Faculty of Technical Sciences Novi Sad Serbia, Department for Computing and Control Engineering (e-mail: vuk.popovic.vp1@roche.com).

Dunja Popovic is a PhD Student in Faculty of Natural Sciences Novi Sad Serbia, Department for Biology and Genetics (e-mail: dunja.popovic@dbe.uns.ac.rs).

The estimation time should always take into consideration that the quality of software is an important concern because once solved issue should not be opening another issue. That means that software should be reliable. Software reliability could be defined as the probability that the software will work without failure in a specified environment and for a specified period of time. There are many approaches which can be implemented to improve the software reliability, however in each of those approaches it is important to have knowledge about development time and planed budget for improving the software reliability.

Parts of software that require maintenance need to be delivered in proposed time and with high-quality. Quality is achieved if the repeated maintenance of the same software part is avoided. For a high-quality software system in the medical branch, complicated as the one observed, maintenance can be costly and therefore all possible estimation techniques should be taken into consideration, including the one presented in this paper.

The rest of the paper is structured as follows. The second section provides background on data used in the paper. Data are based on real data from company internal repository of maintenance requests. The third section covers the methodology describing the neural networks. The fourth section presents the case study. The last section contains conclusions and some remarks for further research.

## II. Used Data

### A. Idea of Data Usage

Prediction of maintenance, presented in this paper, is based on the data set which is composed from risk evaluation and time evaluation variables.

Every issue in maintenance phase has its measurable parameters and those are complexity, business and product risk. Idea is to make prediction based on these parameters, which will predict the time to resolve the issue. Classification of tasks complexity can be based on structural measures (number of code lines, the number of procedures, the number of modules) or on a subjective assessment of software experts. In observed software product the complexity is calculated mostly on the base of subjective assessment of software experts. Only in a small number of cases, the first assessment is adjusted after the extraction of (above mentioned) technical data. The complexity of maintenance tasks is influenced by many factors from human behavior domain (personal characteristics, knowledge, and experience) and technical domain (maintainability). Also, the complexity of maintenance tasks influences the company internal organization and the efficiency of services provided to the clients. If those are well organized it brings the benefits for both, software organization and its clients. In observed case, there is a new team of developers formed exactly to fulfill issue fixing tasks. This approach simplifies the software maintenance estimation and estimation of the software maintenance costs because the same pool of developers is always available for evaluation and resolving the issues in the specific parts of the maintenance.

Other variables are three different types of risk evaluation, for every issue in the maintenance phase. Those are product risk, business risk, and implementation risk. Each of those risks is evaluated by the dedicated engineers for the field of risk. Each of these risks can have one of the possible values: low, medium, or high. In this paper, low respond to value 1, medium to value 2 and high is represented by a value 3. All those data are prepared for a training of the neural networks and extracted from the known data of old issues, which are implemented in previous software service patches.

In general, the life-cycle of software maintenance includes a requirements phase, investigation phase, implementation phase, test phase, installation and checkout phase. The maintenance phase is defined as the process of modifying a software system or component, after delivery, to correct faults, improve the performance or other attributes, or adapt to a changing environment [4]. All named parts of the life-cycle are also taken into consideration. Apart from evaluated risks, there is also a number of planned and a number of actually used implementation time, which is taken into account. This implementation time is based on the expertise of the software development engineers who give the prognosis how much time each issue can take. It is one of the key variables used in research. The output of neural networks is also implementation time – only this time it is predicted implementation time.

With all this information we are trying to provide the conclusion which will lead the decisions regarding the length of a time needed for a software maintenance period, so called software-service-patch, in future. Also, this information will hopefully lead to better adjustment of the planned working hours, implementation time, with the actually used ones.

### B. Construction of Data

In this case, there is a repository of various maintenance tasks conducted by a company, and it is possible to extract some information from historical data. Data are extracted from the company maintenance repository which includes all maintenance issues raised by customers or test engineers. It was done with JIRA software ticketing system and GIT code repository [5]-[7]. In this case, the construction of dataset is done with the amount of data which accords to 6 months of software maintenance work.

Maintenance covered with used dataset was delivered by a new established team whose only task was to take care of software product maintenance. This establishing was the initial idea of this paper because among other conclusions it should be proven that establishing such a department in one software company is of importance for the organization. It is stated that many software organizations failed to define and establish procedures for software maintenance activities, because they are missing the maintenance process management models [8]. As a consequence, there is an evident crisis of management and lack of planning in software maintenance [9].

Software development and maintenance activities are organized in the way that one or more programmers are assigned to each software application, or to each part of the software product. Once a maintenance request (MR) is received from a client, it is forwarded to a selected programmer from a set of assigned programmers in the product care team. Each request may be solved by one programmer, or by a set of programmers assigned to a software application. In this case, as already stated, there is a team of developers who are all experts in their areas. Also apart from one expert developer in each programming language that is used in the observed software, there is also one software testing engineer.

## III. Methodology

In this paper, the Artificial Neural Networks methodology is used to predict the software maintenance. Artificial Neural Networks are mathematical models which are derived from examinations of the human brain system, and based on the human brain processes. Those networks have the capability to learn complex and nonlinear behavior for given data set, and that is why are they an excellent choice for modeling a software maintenance prediction model.

ANN (Artificial Neural Networks) has been commonly used in engineering (such as [10]-[12]). ANN prediction approach has also been used in software prediction models, for predicting software reliability [13]-[16]. They are widely used due to relative simplicity, together with its universal approximation capacity [17].

Neural networks used in this research are standard BPNs, and their architecture is composed of one input layer, one hidden layer and one output layer. Backpropagation networks are the most widely used neural network, and they are based on the backpropagation algorithm. Backpropagation is calculating the error contribution of each neuron after processing a batch of data. In this process each neuron is able to individually introduce the corrections. It is done by changing the values of the weight coefficients on all its inputs, and this change is based on a set error value. The weight of each synapse is connecting a series of tuning neurons and an iterative process to achieve proper tuning (training data) is conducted. In each iteration there is additional fine tuning, conducted with the back-propagation algorithm, to adjust to the desired level. Eventually the network is tuned and when it is used for prediction it will predict with an acceptably low error rates. Also the selected ANN is further composed of two neurons in the input layer, 8 neurons in the hidden layer, and one neuron in the output layer. The ANN model is developed using the functionality of the MATLAB Neural Network Toolbox [18]. The used training function is the MATLAB *trainlm* function, because it is generally the fastest training function in MATLAB software package. Backpropagation algorithm of the function used for network training is based on Levenberg-Marquardt approximation. This function, as a network training function, actually updates weight and bias values according to Levenberg-Marquardt optimization. The performance function for used backpropagation networks is mean square error function. It is the average squared error between the network outputs and the target outputs. Afterwards, the model performance is evaluated and verified with the test datasets. In this process is also evaluated the performance of network by adjusting the numbers of neurons and hidden layers. Once the ANN prediction models are trained to a satisfactory level, and error rates are acceptable, they are used for prediction on other data.

### A. Model

There are several possible prediction models, regarding the input data which will be used. The model used in this paper is shown in Fig. 1.

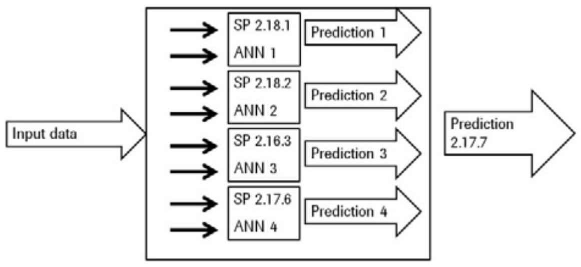

Fig. 1 Preview of used model

Big amount of data, from the constructed dataset, is composing the input for the used model. That is why the idea was to split the main model and to make four separate sub-models. Each of those sub-models has a specific part of input data, which is tailored regarding the maintenance service patch. We took four service patches into consideration with following names 2.16.3, 2.17.6, 2.18.1 and 2.18.2. Each sub-model has unique data input from particular software service-patch. Also, each sub-model has an own neural network and output. For each of that sub-models calculation is done separately. Afterwards, all output results, described in next chapter, from all sub-models are making an output (prediction) of the main model.

## IV. Results

Data from each service patch are used for training the networks. As stated in this case, one service patch responds to one network of one sub-model.

In the trained networks we inputted the data of the service patch (so called future service patch) for which we wanted to make a prediction. Each network gave an output result for this future specific service patch. Those results are listed in Tables I and II, and also visually presented in Figs. 3-8. All network outputs for each service patch along with actually spent time for all service patches are presented. It can be clearly seen how the prediction varies for each service patch. Those variations could be explained by numerous factors such as a number of input data, the accuracy of engineer valuation, or actual complexity of an issue.

The results shown in Tables I and II are presenting the results from executed prediction. These results are after the service patch was done, compared with the actually spent time.

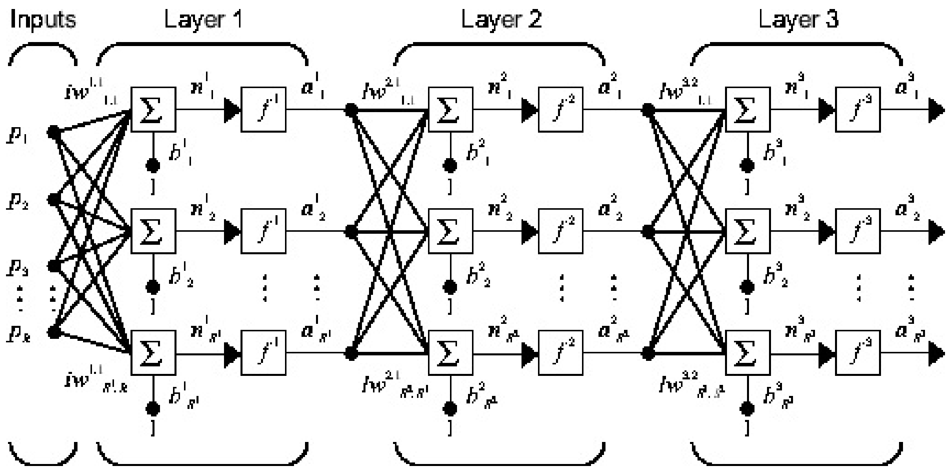

Fig. 2 Preview of used sub-model

TABLE I
RESULT OF PREDICTION FOR EACH SERVICE PATCH

| Time spent | Output 2.16.3 | Output 2.17.6 | Output 2.18.1 | Output 2.18.2 |
|---|---|---|---|---|
| 30 | 555 | 2266 | 276,05 | 345,64 |
| 1410 | 555 | 2266 | 291,44 | 376,63 |
| 180 | 1080 | 2266 | 291,94 | 292,74 |
| 2580 | 1080 | 2266 | 291,44 | 328,42 |
| 45 | 555,02 | 2266 | 275,67 | 292,74 |
| 45 | 555 | 2266 | 276,37 | 376,63 |
| 1500 | 1080 | 2266 | 291,44 | 376,63 |
| 165 | 555 | 2266 | 276,37 | 376,63 |
| 165 | 555 | 2266 | 276,37 | 376,63 |
| 30 | 555,49 | 2266 | 321,84 | 376,63 |

TABLE II
RESULT OF PREDICTION FOR EACH SERVICE PATCH

| Time spent | Output 2.16.3 | Output 2.17.6 | Output 2.18.1 | Output 2.18.2 |
|---|---|---|---|---|
| 300 | 555 | 2266 | 276,05 | 345,64 |
| 120 | 555 | 2266 | 276,05 | 345,64 |
| 480 | 555 | 2266 | 276,36 | 328,42 |
| 2700 | 1080 | 2266 | 291,44 | 328,42 |
| 690 | 555 | 2266 | 2914,3 | 345,64 |

TABLE III
RESULT OF PREDICTION - ADJUSTED RESULTS

| Actually spent time | Prediction - best fitting | Prediction - average |
|---|---|---|
| 30 | 276,05 | 489,17 |
| 1410 | 2266 | 500,77 |
| 180 | 291,94 | 611,17 |
| 2580 | 2266 | 724,96 |
| 45 | 275,67 | 520,86 |
| 45 | 276,37 | 519,5 |
| 1500 | 1080 | 632,02 |
| 165 | 276,37 | 497 |
| 165 | 276,37 | 497 |
| 30 | 321,84 | 638,49 |
| 300 | 276,05 | 658,17 |
| 120 | 276,05 | 516,67 |
| 480 | 555 | 484,94 |
| 2700 | 2266 | 619,96 |
| 690 | 555 | 1148,735 |

The comparison is done in two ways. The first one, also shown in Fig. 7, is done using the calculation of the average value from all sub-model outputs with appropriate MATLAB functionality [17].

The other calculation is done with overlapping the diagrams of predicted data and actual achieved data and taking the best fitting results from predicted data into consideration. After this comparison the result got from neural network prediction could be better compared with the actually achieved results.

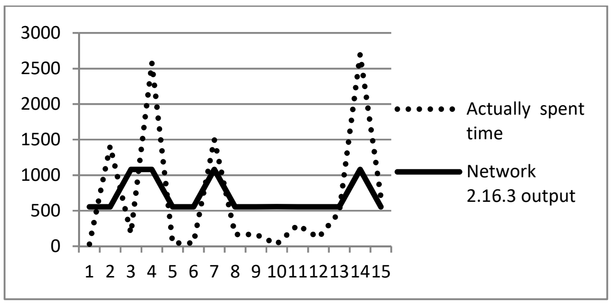

Fig. 3 Network output for service patch 2.16.3

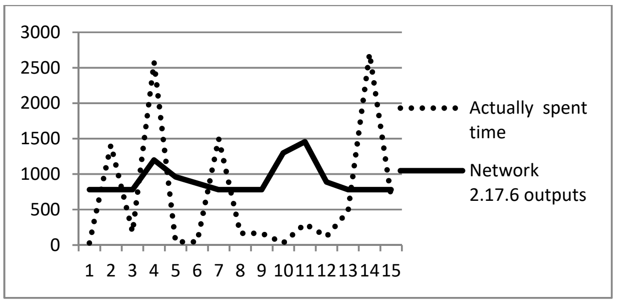

Fig. 4 Network output for service patch 2.17.6

## V. CONCLUSION

This paper is based on research made in one of the world biggest medical diagnostic company. Data used in this publication represent research on maintenance of one of the most used laboratory software in Europe. Therefore conclusions found in this paper may serve as guidance to other colleagues who are dealing with the same issues regarding the efficient software maintenance.

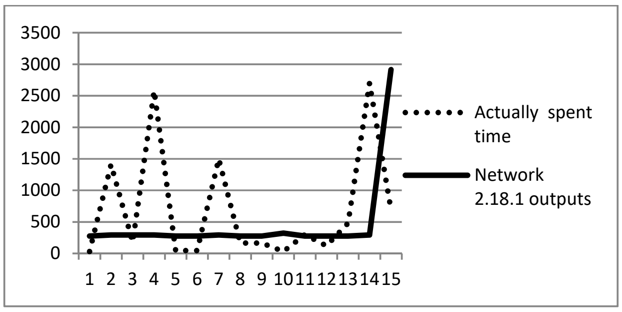

Fig. 5 Network output for service patch 2.18.1

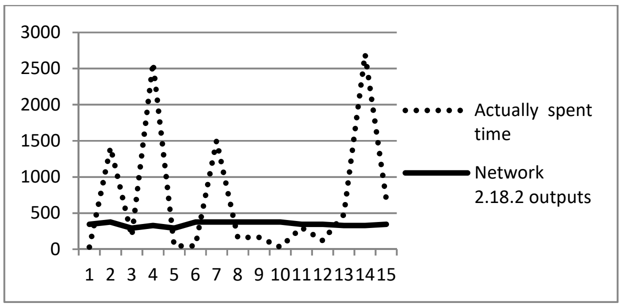

Fig. 6 Network output for service patch 2.18.2

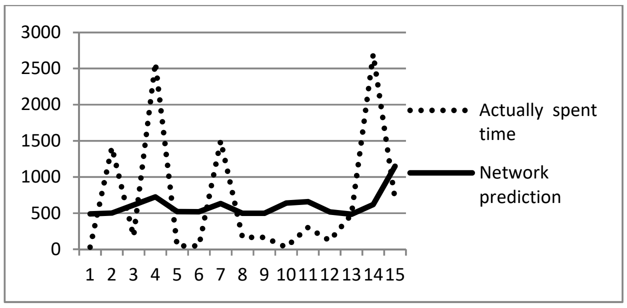

Fig. 7 Network prediction for the average predicted results

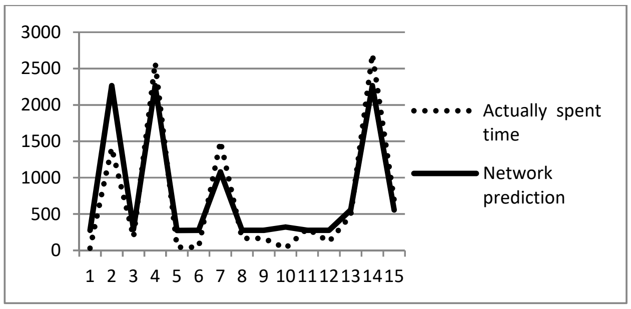

Fig. 8 Network prediction for best fitted results

Systems based on the neural networks have been already used in different fields for making decisions, controlling systems or forecasting. According to the results of data analysis presented in this paper, maintenance prediction based on neural network provides easy to use and also reliable approach for evaluating the maintenance processes. These approaches of maintenance prediction defintely deserve more attention in planning maintenance activities.

Many promising directions for further work exist. The first direction is related to the inclusion of more real data about software complexity (number of code lines, a number of modules) and about maintenance staff skills (experience, familiarity with software products and familiarity with technologies) in the analysis. Also, one direction is related to the comparison of this approach with other commonly used AI approaches (genetics algorithm, fuzzy logic) on the same data sets. And finally, one direction could be the modification of the presented approach through the introduction of modern neural networks based techniques such as Theano, Keras or Tensorflow and comparison of results with this approach.

## References

[1] R. Singh, “International Standard ISO/IEC 12207 Software Life Cycle Processes”, Software Process: Improvement and Practice, vol. 2, 1996.
[2] G. Parikh, “Exploring the world of software maintenance: what is software maintenance?”, ACM SIGSOFT Software Engineering Notes, vol. 11, 1986.
[3] A. April, J. H. Hayes and A. Abran, “Software Maintenance Maturity Model (SMmm): the software maintenance process model”, Journal of Software Maintenance: Research and Practice, vol. 17, issue 3, 2005.
[4] Cycle Processes”, Software Process: Improvement and Practice, vol. 2, 1996.
[5] Patrick Li, “JIRA 7 Essentials”, Packt Publishing Ltd, Apr 2015
[6] Matthew Doar, “Practical JIRA Administration”, O'Reilly Media, Inc., May 2011
[7] Ravi Sagar, “Mastering JIRA”, Packt Publishing Ltd, May 2015
[8] F. J. Pino, F. Ruiz, F. García and M. Piattini, “A software maintenance methodology for small organizations: Agile_MANTEMA”, *Journal of Software: Evolution and Process*, vol. 24, 2012.
[9] A. April and A. Abran, “A Software Maintenance Maturity Model (S3M): Measurement Practices at Maturity Levels 3 and 4”, Electronic Notes in Theoretical Computer Science, Volume 233, 27 March 2009.
[10] K. Xu, M. Xie, LC. Tang, SL. Ho, “Application of neural network in forecasting engine systems reliability”, Applied Soft Computing, vol. 2, 2003.
[11] P.S. Rajpal, K.S. Shishodia, G.S. Sekhon, “An artificial neural network for modeling reliability, availability and maintainability of a repairable system”, Reliability Engineering and System Safety, vol. 91, 2006.
[12] Y. Takada, K. Matsumoto and K. Torii, “A softwarereliability prediction model using a neural-network”, Systems Comput Japan., vol. 25, 1994.
[13] T.M. Khoshgoftaar and R.M. Szabo, “Using neural networks to predict software faults during testing”, IEEE Trans Reliab., vol. 45, 1996.
[14] K.Y. Cai, L. Cai, W.D. Wang, Z.Y. Yu and D. Zhang, “On the neural network approach in software reliability modeling”, J Systems Software, vol. 58, 2001.
[15] L. Tian and A. Noore, Evolutionary neural network modeling for software cumulative failure time prediction”, Reliab Eng Syst Saf., vol. 87, 2005.
[16] D. Srinivasan, Neurocomputing, vol. 23, 1998.
[17] M. H. Beale, M. T. Hagan, H. B. Demuth, “MATLAB Neural Network Toolbox User’s Guide”, The MathWorks, Inc. , 2004
[18] PerOlof Bengtsson and Jan Bosch, Architecture Level Prediction of Software Maintenance, 1999