SOFTWARE MAINTENANCE: A CONVENTIONAL APPROACH USING NEURAL NETWORKS

Neena Chaudhary[1], Aditi Kumar[2]

[1,2] *Department of Computer Science, Baddi University, Baddi, Solan, Himachal Pradesh, (India)*

ABSTRACT

Software Maintenance (SM) is vital task for any software product throughout the entire software life cycle. Many methods have been developed in this context and their results are very beneficial. Today's modern technology environment, intelligent computing methods play an important and significant role towards software maintenance. In this paper we present various techniques and problem faced by software maintenances phase. Further, paper presents several ways to reduce cost and efforts involved in software maintenance. Software maintenance costs can be reduced significantly if the software architecture is well defined, clearly documented, and creates an environment that promotes design consistency through the use of guidelines and testing quality.

Keywords: Software Maintenancet, Neural Networks, Effort Prediction.

I INTRODUCTION

Software maintenance is a recurring process of any software product during its life cycle. When a scratch of the new software products comes into the human mind, the maintenance aspects should also come with its significance. As we know that any new software product goes from the different level of software product life cycle and each stage of cycle has its own importance and role. Our opinion is that maintenance is required at each stage of the cycle. There are many ideas and definition for the software maintenance but as according to the IEEE Std. 1219- 1993, it is a modification of a software product after delivery in order to correct faults, to improve performance or other attributes, to adapt a product to a changed environment, or to improve the product maintainability.

There are many methods classical as well as modern intelligent computing methods for the software maintenance and one of that we have chosen here is an intelligent computing method. The ICMs have broad area for analysis and gives the very significance results for software maintenance. In this paper, we have gone through the different types of software maintenance such as adaptive, corrective, perfective and preventive and application of ICM methods in these. Also, we try to correlate the different ICMs with software maintenance types and also collect the information about the source code analysis for its comprehension

II SOFTWARE MAINTENANCE

Software maintenance includes correcting the discovered faults, adapting the system to changes in the environment, and improving the system's reliability and performance. Maintenance is the last stage of the software life cycle. After the product has been released, the maintenance phase keeps the software up to date with environment changes and changing user requirements. Swanson [1] defines software maintenance as a "process of correcting, perfecting, or adapting existing programs so their performance more closely meets the needs of the organization using the software". Ammar [2] have focused on the technical side of the software maintenance (e.g., development of models and tools) and completely ignore the managerial side. In this paper they review the software maintenance management literature and identify management issues related specifically to adaptive and perfective maintenance. Two type of maintenance activities are selected based on the empirical evidence drawn from Martin [3], Nosek, and Palvia [4], and Van Vliet [5] that indicates 75% of maintenance costs are spent on providing enhancement (adaptive and perfective). Software maintenance is categorized in the following types provided by the IEEE Standard [6] as shown in table 1.

Table 1

Software Maintenance Categories [22]

	Correction	**Enhancement**
Proactive	Preventive	Perfective
Reactive	Corrective	Adaptive

2.1 General Problem in Software Maintenance

The major issues that can slow down the maintenance process are as follows:

- Unstructured code.
- Maintenance programmers having insufficient knowledge of the system.
- Documentation being absent, out of date, or at best insufficient.
- Software maintenance is having a bad image.

Apart from the above personal problem the categorization of the software maintenance is based on requirement type and logical aspects of the software product. The following types of SM are given in bit detail.

Adaptive Software Maintenance: Adaptive maintenance deals with adapting the software to new environments. System adaptation in general identified as the modification in the process when there is an external and internal change in the process (system), so as the performance of the process (system) does not degrade up to certain limit or remain as before. In context of software, the external changes may be taken as the changes in the environment of an external agent as a user who ask for the modification in the system such as its performance remain intact with the changing environment. The environment may consist of another platform on system or user's requirement. It may require reusable of particular module or set of modules with reference.

Corrective Software Maintenance: Corrective maintenance deals with fixing bugs in the code (traditional maintenance). It is determining or locating the fault (bugs) in the program or source code and making corrective measure for the same. This is also prone to predictions of fault for the system so that corrective measures can be taken in hand before its occurrence. Major works in corrective maintenance have been done.

Perfective Software Maintenance: Perfective maintenance deals with updating the software according to changes in user requirements. It is more or less in the time of adaptive maintenance which deals with specifically the user requirement on priority rather the environment as a whole.

Preventive Software Maintenance: Preventive maintenance deals with updating documentation and making the software more maintainable. It deals primarily a prediction and caution measure so that the process causing it may not be executed.

2.2 Techniques for Maintenance

• Effective software maintenance is performed using techniques specific to maintenance. The following provides some of the best practice techniques used by maintainers [7], [8].

• Program Comprehension: Programmers spend considerable time in reading and comprehending programs in order to implement changes. Code browsers are a key tool in program comprehension. Clear and concise documentation can aid in program comprehension.

1) Re-engineering: Re-engineering means to restructure and examine the system to restructure it in a new form. Reengineering is often not undertaken to improve Software maintainability but is used to replace aging legacy systems.

2) Reverse engineering: Reverse engineering is the process of Re-engineering and Re-Structuring analyzing a system to identify the system components, system documentations and their interrelationships. Reverse engineering is passive; it does not change the system, or result in a new one.

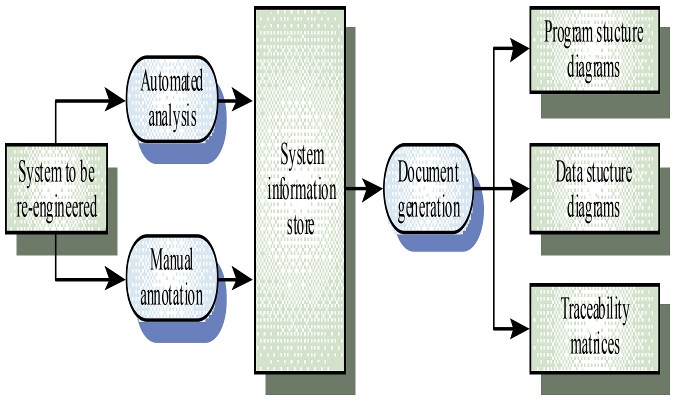

Figure 1

3) Impact Analysis: Impact analysis is identifies all systems process and products affected by a change request and develop an estimate of the resources needed to accomplish the change. It is performed after a change request enters the configuration management process.

III LITERATURE SURVEY

E.Praynlin and P.Latha (2013) [13] uses Adaptive Neuro fuzzy Inference system (ANFIS) model for software effort estimation. The advantage of both the neural network and fuzzy logic can be combined by using the neuro fuzzy model. ANFIS used in this paper is of sugeno type Fuzzy inference system. It applies the combination of Least square method and the back propagation gradient descent method. The functioning of ANFIS is a five-layered feed-forward neural structure and various type of membership functions are used. ANFIS Uses two set of datasets. One set of dataset consists of 63 projects and other consists of 93 datasets both are from the historic projects of NASA. ANFIS generates FIS by two main methods i.e. Grid partitioning and Clustering. The objective of E.Praynlin and P.Latha is to provide which membership function is to be used in ANFIS which gives accurate predictive effort. It was concluded that the proposed ANFIS model using Trapezoidal membership function has low MMRE than the above mentioned membership functions.

Jyoti G. Borade and Vikas R. Khalkar (2013) [14] describes several existing methods for software project effort and cost estimation [10].Software cost estimation is the process of predicting the amount effort required to build a software system. Estimation is a complex activity that requires knowledge of a number of key attributes. At the initial stage of a project, there is high uncertainty about these project attributes. Conventional estimation techniques focus only on the actual development effort furthermore; this paper also described test effort estimation. Testing activities make up 40% total software development effort.

Olga Fedotova et al. (2013) [15] presents common methods used in the software effort estimation (SEE) and them study performed in a software development organization (SDO) that is implementing the software development process improvement framework Capability Maturity Model Integrated(CMMI). The aim of this paper is to present the main software effort estimation methods, their advantages and disadvantages and to apply a formal method based on the Multiple Linear Regression technique to the historical data of a medium-sized multinational software company. The stepwise Multiple Linear Regression (MLR) technique was selected and used for the software development and software testing processes. The results achieved with MLR were compared with the estimates provided by the area expert. The model obtained for the testing team performed better results than the expert judgments.

Jyoti G. Borade [16] in Software Project Effort and Cost Estimation Techniques states that no model can estimate the cost of software with high degree of accuracy. There are various techniques used in software cost estimation. By following Boehm's classification system, these Methods are summarizing into three categories expert judgment, algorithmic estimation, and analogy based estimation.

IV TECHNIQUES TO REDUCE SOFTWARE MAINTENANCE EFFORTS

There are several ways to reduce cost and efforts involved in software maintenance. Software maintenance costs can be reduced significantly if the software architecture is well defined, clearly documented, and creates an environment that promotes design consistency through the use of guidelines and testing quality are describe as:

4.1 (Re) Documentation

Maintenance cost can be reduced by (Re) documentation. Without documentation programmer spent 21.5% understanding of code. With documentation, we could save 12% of the cost of maintenance .Without documentation the good and bad programmer skill level disappeared. During the past decade, CARE has developed by Omnext, a concept that helps to cost reduction. CARE Stands for "Computer Aided (Re) documentation and evaluation. This concept can be used to improve the principal [9]. CARE may generate up to 50% of saving on the cost, impact analysis, documentation and testing. It also used to increases the quality of software. This concept comprises the technical and functional documentation. CARE helps to monitoring the software system's size and quality.

Intake: A CARE environment is depending on the environments that have a software documentation and software evaluation [9]. The result, generated by the CARE must be analyzed, the standards, and the method of documentation and evaluation.

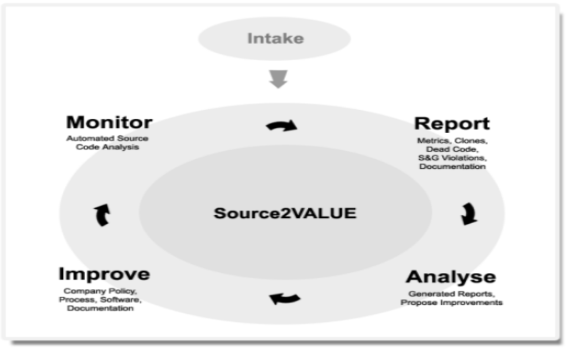
Intake
Monitor
Automated Source Code Analysis
Report
Metrics, Clones, Dead Code, S&G Violations, Documentation
Source2VALUE
Improve
Company Policy, Process, Software, Documentation
Analyse
Generated Reports, Propose Improvements

Figure 2: CARE [9]

Monitor: CARE based on monitoring, in order to analyze the sources automatically or manually. The result generated by the CARE with quality and size. For documentation purposes, it may contain structure and relationship between application sources [9].

Report: The different types of reports can be generated from the object. These reports may contain the information about quality, size, and development [9]. For documentation purposes, technical documentation was developed and functional documentation that used to adapt the changes in the application.

Analyze: Analysis is used to carry out signals, quality, and productivity, and results generated for improvement. For documentation purposes, the functional documentation is also analyzed from the signals [9].

Improve: Improvement can be occurred by implementing and updating the functional documentation [9]. The updating can be carried out manually or by automatically that used for tools. If required, the CARE definition can be updated on the basis of advanced insight.

4.2 Decreasing Turnovers

Maintenance cost can be reducing turnovers by internal and external. Internal turnovers result from moving internally and external turnovers are the results of people moving out. To reduce maintenance cost through work scheduling and reducing number of internal moves through strict policy.

4.3 Eliminating Dead Code

Dead code means unnecessary code that can be removed without affecting the program that are never called.30% of the software in an older system can be dead code by eliminating dead code, we reduce the code size [9].

4.4 Reduce Complexity

Saving can be achieved by monitoring and improving the technical quality of a system. Maintenance costs affected by the existing software complexcity.25% of maintenance cost should be total life-cycle costs.

4.5 Testing Quality

The number of errors can be reduced by applying an effective testing strategy. With reduced errors, maintenance effort can be quite low. So better testing quality reduces maintenance effort.

4.6 Eliminating Bugs

Bugs in software are costly and difficult to find and fix. Techniques and tools have been developed for automatically finding bugs by analyzing source code [10].

4.7 Eliminating Cloned Code

Large software systems typically contain large redundant code. When applying new functionality, many programmers cannot copy and customize existing pieces of code [10]. Large software systems contain 10-25% redundant code. If redundant code were removed, spend the budget savings on new problems and enhancing the efficiency of the organization.

V SOFTWARE MAINTENANCE HELPFUL IN A NEURAL NETWORK

Neural network software offers reliable, scalable, distributed processing of large data process across clusters of machines to create highly accurate predictive models for data mining and data analysis. It is designed to scale up from a single computer to thousands of machines, each offering local computation. The easy-to-use interface allows you to set minimal conditions for preprocessing and neural network learning. The specific conditions to have total control over data preprocessing, training termination rules and neural network architectures. Perform **sales like data forecasting, sports predictions, and weathers prediction, fault prediction, rain prediction, medical classification, and various disaster predictions.**

A neural network is a most popular data modeling tool

and techniques that is able to capture and represent complex input/output relationships. It is very difficult for most people to understand the complex math involved with neural networks and another's challenging to figure out how to organize your data into meaningful information for the neural network to understand. NeuroSolutions Infinity is designed to make powerful neural network technology easy to use for both novice and advanced developers.

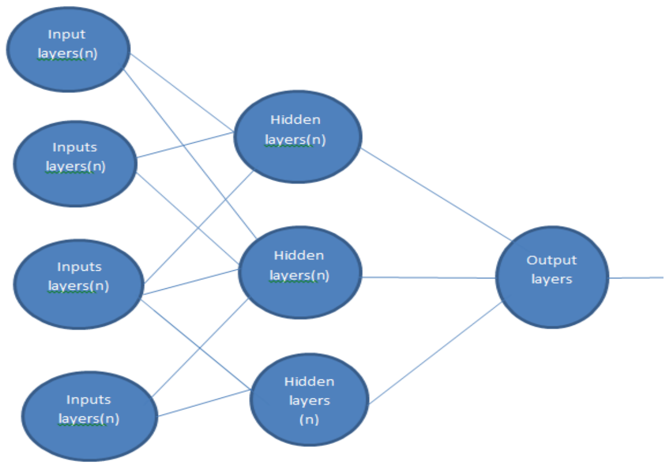

Figure 3: Neural networks works

VI CONCLUSION

Software maintenance is an integral part of a software life cycle. Software maintenance is defined as the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment. Effort estimation at the early stage of software development is very difficult because of lot of uncertainty in input parameters which decides the software effort. This paper represents different problems and their solutions for software maintenances using neural networks. For future, we are going to propose a model to enhance performance of software maintenance.

REFERENCE

1. E. Burton Swanson, “The Dimensions of Maintenance”, Proceedings of the 2nd International Conference on Software Engineering IEEE Computer Society Press: Los Alamitos CA, pp. 492 497, 1976.
2. Ammar Rashid, William Y.C. wing, and Dan Dorner, “Gauging the Differences Between Expectation and Systems Support: The Managerial Approach of Adaptive and Perfective Software Maintenance”, Fourth International Conference on Cooperation and Promotion of Information Resources in Science and Technology, 2009.
3. J. Martin, “Software Maintenance: The Problem and Its Solutions”. Prentice Hall, Englewood Cliffs, NJ, pp. 400-472, 1983.
4. J.Nosek, and P. Palvia, “Software Maintenance Management: Changes in Last Decades”. Journal of Software Maintenance Research and Practice, 2(3), pp. 157-174, 1990.
5. H. Van Vliet, “Software Engineering: Principles and Practice”. Wiley, 2001.
6. IEEE Std. 1219-1998, IEEE Standard for Software Maintenance, 1998.
7. Thomas M. Pigoski, Technical Software Services (TECHSOFT), Inc. Software Maintenance. IEEE – Trial Version 1.00 – May 2001.
8. Rajib Mall. Software Engineering: Fundamentals of Software Engineering. PHI, third edition, 2009.

9. Omnext white paper, "How to save on software maintenance costs", March 2010.
10. "Software Maintenance and Evolution: a Roadmap" K. H. Bennett V.T Rajlich Research Institute for Software Evolution Department of Computer Science University of Durham Wayne State University UK Detroit, MI 48202 DH1 3LE USA.
11. Top ten ways to reduce software costs PRACTICAL TIPS FOR SOFTWARE ASSET MANAGEMENT Top ten ways to reduce software costs PRACTICAL TIPS FOR SOFTWARE ASSET MANAGEMENT.
12. Software Maintenance Gerardo Canfora and Aniello Cimitile, cimitile@unisannio.it University of Sannio, Faculty of Engineering at Benevento Palazzo Bosco Lucarelli, Piazza Roma 82100, Benevento Italy
13. E.Praynlin and P.Latha. Estimating Development Effort of Software Projects using ANFIS. ICON3C 2012 , IJCA.
14. Jyoti G. Borade and Vikas R. Khalkar. Software Project Effort and Cost Estimation Techniques, IJARCSSE, Volume 3, Issue 8, August 2013.
15. Olga Fedotova, Leonor Teixeira and Helena Alvelos Software Effort Estimation with Multiple Linear Regression: Review and Practical Application. Journal of Information Science And Engineering 29, 925-945 (2013).
16. Yunsik Ahn, Jungseok Suh, Seungryeol Kim and Hyunsoo Kim. The software maintenance project effort estimation model based on function points. 2003; 15:71–85.