December 1998

A Neural Network Decision Method for Software Maintenance Life Cycle Identification

Hsiang-Jui Kung
Rensselaer Polytechnic Institute

Hui-Chung Chu
HuaFan University

Cheng Hsu
Rensselaer Polytechnic Institute

James Lin
HuaFan University

A Neural Network Decision Method for Software Maintenance Life Cycle Identification

Hsiang-Jui Kung
Cheng Hsu
Rensselaer Polytechnic Institute

Huei-Chung Chu
James Quo-ping Lin
HuaFan Unviersity

Abstract

The software maintenance life cycle concept is a powerful model in helping software maintenance planning. The operationalization of the life cycle concept requires a heuristic decision method. Although the heuristic decision method works most of the time, the method requires integration of different tools and sometimes leads to errors. In this paper, we propose a neural network decision method, which combines data smoothing and maintenance stage identification into one unit.

Introduction

Software maintenance has taken on an increasing importance as the Information Systems (IS) community recognizes its economic impact. The cost of maintaining application software is currently estimated to double the original development cost [4]. One of the most important factors attributed to the high cost of software maintenance is the lack of close and effective management of the maintenance process. A large proportion of this work consists of trying to respond rapidly to changing requests due to the direct impact on the customers, so the maintenance activity takes on a fire fighting role.

Cooper and Munro [5] suggest that software maintenance planning is very important. An important prerequisite for planning is a system of estimation of efforts, resources and time required. Software maintenance planning is a difficult problem because the demand for maintenance is difficult to predict. The accuracy of forecasting is determined by how much patterns and relationships change, and by how much human interactions can influence future event [10]. The critical assumption for accurate forecasting is the patterns or relationships remain constant. The greater human influence on future event leads to less accurate forecasts. The traditional forecasting methods cannot forecast software maintenance requests accurately because these requests are non-stationary, and these requests are the results of the interactions among users, IS staff, and application software [9]. This paper introduces a neural network decision method to provide the critical decision information for software maintenance planning.

Background

Software maintenance is modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment [2]. It also includes training and other services for users. Required knowledge and skills in maintenance can be partitioned into three domains: the user, the software and the application. The user domain is that within which user support staff works to solve user support requests. The software domain is that within which the programmer works to fix system faults and to perform the detailed design for enhancement requests. The application domain is that within which the system analyst works to study system feasibility, develops plans and requirements, and produces a set of system specifications for enhancement projects [1, 11]. IS researchers have assumed that maintenance requests have the same distribution after implementation. Gefen and Schneberger [6] have challenged this assumption and reported three different maintenance periods in a case study where different types of maintenance surged in different periods. Burch and Kung [3] discover the fourth period which all types of maintenance requests remain low.

Kung [8] extended the findings of Genfen and Schneberger, integrated the life cycle concept and maintenance classification, developed the software maintenance life cycle (SMLC) model and also provided a heuristic decision method to render timely maintenance stage changes information for software maintenance planning. The SMLC model has four stages (Figure 1). The changes of maintenance regimes correspond to the changes of the software maintenance life cycle stages. A regime of user support occurs in the introduction stage, a regime of repair in the growth stage, a regime of enhancement in the maturity, and a regime of decline in the decline stage.

The heuristic decision method (Figure 2) consists of the following elements: request classification, data smoothing and stage identification. Maintenance requests can be classified into user support, repair and enhancement based on the consensus between

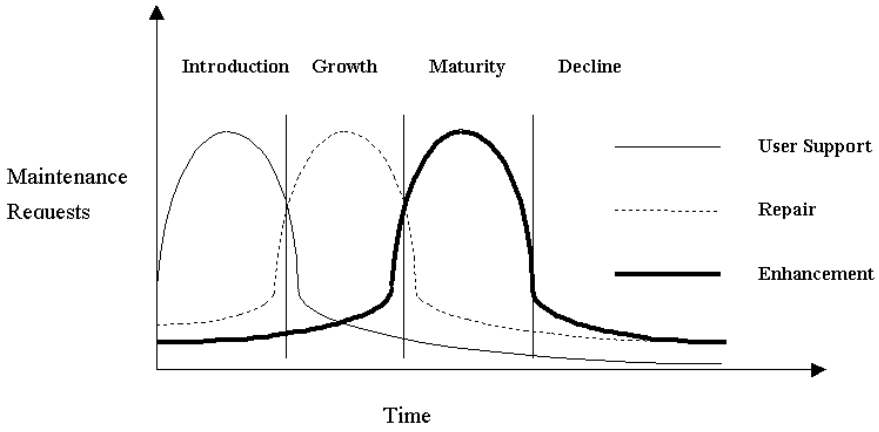

Figure 1: Software Maintenance Life Cycle

users and IS staff. Classified maintenance request data is a point process, which is a unique type of time series when a series of events occurs randomly in time. Usually, it is difficult to see the trend in the non-smoothed time plot. The kernel estimator is the most well known non-parametric density estimation to smooth out the variation of a given set of data [7]. The sequence of the surges of maintenance requests is user support, repair and enhancement. There is a set of heuristic decision rules, which determines the changes of software maintenance life cycle stages.

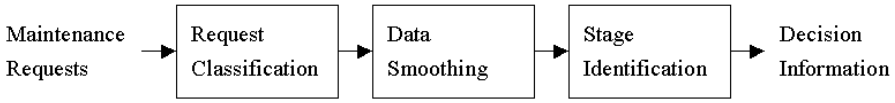

Figure 2: Heuristic Decision Method

A Neural Network Decision Method

Although the Kung's heuristic decision method is a breaking-ground step in applying the life cycle concept to software maintenance planning, it is not the optimal method that there are situations in which the method fails to identify the software maintenance stage changes [8]. The identification of software maintenance life cycle stage is about classification and pattern recognition, which is the strength of neural networks. Neural networks are especially useful for classification and pattern recognition problems with lots of training data available but without hard and fast rules.

Artificial neural networks are physical cellular systems, which can acquire, store, and utilize experiential knowledge [13]. Neural networks have some sort of training rule whereby the weights of neural cells are adjusted on the training data. In other words, neural networks learn from examples (training data) and create information that is contained in the training data. There is overlap between neural networks and statistics. Most neural networks that learn to generalize from the training data are similar to statistical inference [12].

The neural network decision method will use an unsupervised learning neural network. This method looks at the data, finds out about the properties of the data and learns to reflect these properties in its output. A case is the input vector of the neural network decision method which includes not only input variables (user support, repair, enhancement and time), but also a target variable (life cycle stage change). The neural net adjusts its weights when feeds training data to the net. Despite training datasets, we need another two datasets: validation and test. Validation datasets will be used to fine-tune the weights, and test datasets will be used to assess the performance of the neural network decision method.

Discussion and Future Research

This paper proposes a fully automated rule-free decision method to provide the real time decision information of software maintenance life cycle stage changes. With the neural network decision method, we don't need to apply any statistical tool to smooth out the maintenance requests. Another contribution of this paper is to extend the neural networks application to software maintenance management.

The decision method proposed by this paper still needs data to validate. One limitation of the proposed future study is that neural networks can not magically create information, which is not contained in the training data. Neural networks require lots of training datasets. Collecting empirical maintenance data is too costly in terms of time, and simulated data is a feasible alternative.

References

1. Abrand, A. and Nguyenkim, H. (1991) Analysis of Maintenance Work Categories Through Measurement, *Proceedings of IEEE Conference on Software Maintenance*, Sorrento, Italy: IEEE, 104-113.
2. An American National Standard IEEE Standard Glossary of Software Engineering terminology, *ANSI/IEEE Standard 729*, 1983.
3. Burch, E. and Kung, H. (1997) Modeling Software Maintenance Requests: A Case Study, *Proceedings of IEEE Conference on Software Maintenance*, Bari, Italy: IEEE, 40-47.

4. Calow, H. (1991) Maintenance Productivity Factors: A Case Study, *Proceedings of IEEE Conference on Software Maintenance*, Sorrento, Italy: IEEE, 250-253.
5. Cooper, S. D. and Munro, M. (1989) Software Change Information for Maintenance management, *Proceedings of IEEE Conference on Software Maintenance*, Miami, FL: IEEE, 279-282.
6. Gefen, D. and Schneberger, S. L. (1996) The non-homogeneous maintenance periods: A case study of software modifications, *Proceedings of Conference on Software Maintenance*, Monterey, CA: IEEE, 134-141.
7. Hardle, W. (1991) *Smoothing Techniques with Implementation in S*, New York, NY: Springer-Verlag.
8. Kung, H. (1997) *A Life Cycle Model for Software Maintenance Management*, Unpublished PhD Thesis: Rensselaer Polytechnic Institute.
9. Lehner, F. (1991) Some Quantitative Aspects of Software Maintenance Management, *The Economics of Information Systems and Software*, London, UK: Butterworth-Heinemann, 162-180.
10. Makridakis, P. G. (1990) *Forecasting, Planning, and Strategy for the 21st Century*, New York, NY: Collier Macmillan.
11. Swanson, E. B. and Beath, C. M. (1989) Reconstructing the Systems Development Organization, *MIS Quarterly*, 13, 293-305.
12. White, H. (1992) *Artificial Neural Networks: Approximation and Learning Theory*, Blackwell.
13. Zurada, J. M. (1992) *Introduction to Artificial Neural Systems*, Boston, MA: PWS Publishing.