An Analysis of Cost Driver in Software Cost Model by Neural Network System

Dong Hwa Kim

(Tel : 82-42-821-1170, Fax : 82-42-821-1165 ; E-mail: kimdah@tnut.ac.kr)
Department of Instrumentation & Control Eng., Taejon National University of Technology,
Yusong -gu Taejon, Korea

Abstract

Current software cost estimation models, such as the 1981 COCOMO, its 1987 Ada COCOMO update, is composed of nonlinear models, such as product attributes, computer attributes, personnel attributes, project attributes, effort-multiplier cost drivers, and have been experiencing increasing difficulties in estimating the costs of software developed to new life cycle processes and capabilities.

The COCOMO II is developed for new forms against the current software cost estimation models. This paper provides a case-based analysis result of the cost driver in the software cost models, such as COCOMO and COCOMO 2.0 by fuzzy and neural network.

1. Introduction

Software development involves a number of interrelated factors which affect development effort and productivity. Accurate forecasting has proved difficult since many of these relationships are not well understood. Improving the estimation techniques available to project managers would facilitate more effective control of time and budgets in software development.

Most estimation models in use or proposed in the literature are based on regression techniques[1,2]. A number of these models are discussed in the following section. Statistical regression models estimate software development effort as the dependent variable. Software size (in metrics like lines of code or function points) is used as an independent variable. In some models other parameters such as development programming language or operating system may be used as additional independent variables for a multiple regression model. Regression models have the advantage of a sound mathematical basis as well as measures of goodness of fit, i.e., how well the curve matches the given dataset[2]. This paper examines the potential of two artificial intelligence approaches, i.e., artificial neural networks and case-based reasoning, for providing the basis for development effort estimation models in contrast to regression models. Artificial neural networks (ANNs) adopt a learning approach to deriving a predictive model. The network is designed for the specific set of inputs, e.g., cost driver. The network is presented with a set of known cases which is used to "train" the network, i.e., establish the weights associated with each input in the network. Once the network is trained and stable, development effort for a new case can be predicted by substituting the relevant input values for the specific case. ANNs are recognized for their ability to provide good results when dealing with problems where there are complex relationships between inputs and outputs (effort).

Although the potential for predictive accuracy is good, neural networks lack an explanation capability and do not provide an environment for direct user adaptation of results.

In software development effort estimation, each case could be a previous software development while the current problem is one of extracting a suitable estimate for the current project. Adaptation is based on differences between the stored case and the current problem, e.g., the original case may relate to a system developed with an inexperienced programming team whereas the current system developers may be very familiar with the development environment.

This research compares the effectiveness of back-propagation artificial neural networks in estimating software development effort with the effectiveness of a neural network model and a multiple linear regression model.

2. Software effort estimation models

Numerous models have been proposed for software development effort estimation, e.g., COCOMO, SLIM, Estimacs, Function Point Analysis (FPA) (Albrecht and Gaffney, 1983, Function Point Counting Practices Manual, 1994; Symons, 1991), SPANS, Checkpoint and COSTAR. However no model has proved to be outstandingly successful at effectively and consistently predicting software development effort. The models are based on regression analysis of some set of past cases. Independent variables include an estimate of system size (in lines of code or function points). Matson et al. (1994) review some of the problems associated with regression models, specifically as applied to FPA. Miyazaki et al. (1994) propose a more robust form of least squares calculation for determining parameter values which they consider to be better able to deal with the outliers prevalent in software effort estimation data. All the models presume the ability to estimate system size early in the life cycle which is a major weakness of the lines of code models. FPA estimates can be generated reasonably accurately at an early stage of the development.

The results indicate to what extent models suggested are generalisable to different environments. Most models showed a strong over-estimation bias and large estimation errors with the mean absolute relative error (MARE) ranging from an average of 57 percent to almost 800 percent. Felons and Garner (1992) evaluated three development effort prediction models (SPANS, Checkpoint and COSTAR) using 22 projects from Albrecht's database, and 14 from Kemerer's dataset, The prediction error is large, with the MARE ranging from 46 percent for the Checkpoint model to 105 percent for the COSTAR model.

Jeffery and Low (1990) conducted a study to investigate the