A Model Study for Software Development Effort and Cost Estimation by Adaptive Neural Fuzzy Inference 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

Several algorithmic models have been proposed to estimate software cost and other management parameters. In particular, early prediction of completion time is absolutely essential for proper advance planning and a version of the possible ruin of a project.

However, estimation is difficult because of its similarity to export judgment approaches and for its potential as an expert assistant in support of human judgment.

Especially, the nature of the Norden/Rayleigh curve used by Putnam, renders it unreliable during the initial phases of the project, in projects involving a fast manpower buildup, as is the case with most software projects.

Estimating software development effort is more complexity, because of infrastructure software related to target-machines hardware and process characteristics should be considered in software development for DCS (Distributed Control System).

In this paper, we propose software development effort estimation technique using adaptive neural fuzzy inference system. The methods is applied to case-based projects and discussed.

2. Introduction

Several algorithmic models have been proposed to estimate software cost and other management parameters:
- Putnam's SLIM model
- Jensen model
- Checkpoint
- RCA Price-S
- COCOMO, COCOMO-II
- etc.

These methods are models by crisp approaches. Approaches by intelligent method (fuzzy, neural, Hybrid method) are a few things among thing I found until now. So, I guess it is a valuable one if I study approaches by intelligence technique step by step. There are few method S/W cost estimation model until now. There are many kinds of characteristics of influence function to cost estimation.

Estimating technology of software costs is important because of the overall magnitude of these costs and the fundamental influence software will have on our future as well as present of life. That is, underestimated costs may convince manager to approve proposed projects that then exceed their budgets and thus fail to produce their expected advantages.

On the other hand, overruns can damage software management's credibility and stile future general management support. If general management is cancelled before completion because of overruns or underestimated methods, it might be wasted the resources invested in them [1, 2].

Software cost estimations include system size and complexity, personnel capabilities and experience, hardware constraints, the use of modern software tools and practices. So, it is more of an art or engineering than a science [3].

The proposed models estimation are many, but usually too restrictive to apply across a wide range of projects even though considerable analysis of empirical data collected over long periods of time have done, since most models for software cost estimation are definable in terms of a mathematical algorithm and can thus be termed algorithmic models.

We can classify these algorithmic models used for software cost estimation as follows [4]:
- Linear models or mathematical models that try and fit a simple algorithm to the observed data;
- Multiplicative models that express effort as a product of constants with various cost drivers as their exponents;
- Analytic models that usually express effort as a function that is neither linear nor multiplicative;
- Tabular models that represent the relationship between cost drivers and development effort in a matrix form;
- Composite models that use a combination of all or some of the aforementioned approaches.

During the last two decades, many software estimation models have been developed to accurately predict the cost of a software project model such as, SLIM (Software Life cycle Model), Jensen model, Checkpoint, RCA Price-S, COCOMO, COCOMO-II because their precise mathematical definition makes it easy to implement them on a computer [2~3].

Two such composite models among them widely used in industry are the Price-S model and Putnam's SLIM model. Besides the IBM model, Doty model, the Boeing model, the Estimacs model has been provided.

Since the methodologies of the Price-S model are largely unpublished, it has some unavailable things for scrutiny and subsequent improvement.

The Putnam model can be used in process analysis to assess the impact of a tightened schedule, and to predict long term software costs and portable completion times through straightforward curve-fitting techniques. However, its capability in predicting development time and total manpower requirements at an early stage is not satisfactory.

The COCOMO model proposed by Boehm can provides a combination of various functional forms made to the user in a structured manner.

Both the Putnam and the COCOMO models use the Rayleigh distribution based on the observation by Norden as an approximation to the smoothed labor distribution curve for cost estimation.

The Rayleigh distribution provides a good approximation of the manpower curve for various hardware development processes. However, the slow manpower buildup and the long tail-off time curve is not in accordance with the labor curves of most organic-mode of software cost estimations.

Software technologies generally have a faster buildup rate than hardware technologies, and this is a deviation from the Rayleigh curve. To compensate for this the COCOMO model uses only the central portion of the Rayleigh curve to arrive at the labor estimating equation. An alternative model to the Rayleigh curve was proposed by Parr[ ].

Many of the previous articles on software cost estimating have largely focused on estimating techniques. These techniques may suggest the view that the only important consideration in estimating accurately is the basis of the