Fuzzy Systems and Neural Networks in Software Engineering Project Management

SATISH KUMAR
Department of Physics and Computer Science, Faculty of Science, Dayalbagh Educational Institute, Dayalbagh, Agra UP 282005, India

B. ANANDA KRISHNA
M. Tech. student, Regional Engineering College, Surathkal, Karnataka, India

PREM S. SATSANGI
Professor, Department of Electrical Engineering, Indian Institute of Technology, Hauz Khas, New Delhi 110016, India.

Received October 20, 1992; revised May 3, 1993

Abstract. To make reasonable estimates of resources, costs, and schedules, software project managers need to be provided with models that furnish the essential framework for software project planning and control by supplying important “management numbers” concerning the state and parameters of the project that are critical for resource allocation. Understanding that software development is not a “mechanistic” process brings about the realization that parameters that characterize the development of software possess an inherent “fuzziness,” thus providing the rationale for the development of realistic models based on fuzzy set or neural theories.

Fuzzy and neural approaches offer a key advantage over traditional modeling approaches in that they are model-free estimators. This article opens up the possibility of applying fuzzy estimation theory and neural networks for the purpose of software engineering project management and control, using Putnam’s manpower buildup index (MBI) estimation model as an example. It is shown that the MBI selection process can be based upon 64 different fuzzy associative memory (FAM) rules. The same rules are used to generate 64 training patterns for a feedforward neural network. The fuzzy associative memory and neural network approaches are compared qualitatively through estimation surfaces. The FAM estimation surfaces are stepped, whereas those from the neural system are smooth. Also, the FAM system sets up much faster than the neural system. FAM rules obtained from logical antecedent–consequent pairs are maintained distinct, giving the user the ability to determine which FAM rule contributed how much membership activation to a “concluded” output.

Key words: Fuzzy associative memory, feedforward neural network, software engineering, resource estimation, manpower buildup index.

1. Introduction

It is now well known that large-scale system projects, once started, get surreptitiously out of hand, and it is not unusual for the project to double its cost by delivery time [1]. Schedule slippages and escalating costs, coupled with the ever increasing demand for software, its poor reliability, and the eventual inability of managers to meet project goals or schedules are factors that have led to the software crisis, where projects slip imperceptibly but inexorably out of control [2, 3]. For the manager, managing on quicksand has become a way of life; for the customer, getting progressively accustomed to an ever increasing budget is commonplace. It is therefore important to devise techniques to help the project manager keep the project under control.

One of the conditions to make this possible is for the software development plan to be based on
realistic estimates. The software estimating process requires the basic understanding that software development is not a “mechanistic” process, in that the tasks are not all visible or measurable, as in the case of a deterministic quantity. Software development comprises a large number of tasks with strongly coupled interactions of considerable complexity, each possessing a degree of uncertainty. The relationships between controlling variables are best modeled empirically due to the inherent “fuzziness” in the estimation of these variables. Thus, these tasks are not all capable of being objectively measured, and at best they are assessed through group consensus of experts in the area. Software cost and development time prediction will remain no better than a probabilistic estimating method until we accept the inherent “fuzziness” of variables and develop more realistic models for estimation based on fuzzy set theory [4-6].

A very strong case for effective modeling of the dynamics inherent in the system development methodology, and for understanding factors that management can control and factors that are limited by the process itself, is thus made, and one can resort to powerful modeling methodologies such as system dynamics [7] for this purpose, with the objective of providing the manager with a framework to make reasonable estimates of resource, cost, and schedule [8-10]. Apart from this, sound measurement techniques must be devised and substantiated by available project data. Such estimation models not only do the job of estimating the schedule, effort, cost, etc., but also provide an essential framework for software project planning and control. They provide important “management numbers” concerning the state and parameters of the project, which are critical for resource allocation and which help progress towards a more organized software product. Software development estimation models are “fuzzy information systems” providing information to a decision making unit.

2. Software Measurement Techniques

Measurements on software can be broadly classified into two major categories: direct measures and indirect measures. Direct measures such as cost, effort, source lines of code (SLOC), speed, memory utilization, and size are made objectively. Indirect measures are more subjective or fuzzy in nature that they are based on an individual’s estimation or on compromised group consensus. Typical indirect measures would include functionality, quality, efficiency, and reliability.

Other broad classifications of software metrics have been proposed in the literature [11]. For example, one could classify metrics into those pertaining to the technicality, productivity, or quality of software. Alternatively, one could have size-oriented metrics based on SLOC estimates, and metrics that bypass SLOC.

Size-oriented metrics are controversial and not universally accepted as the best way to measure software development productivity [12]. Some of the more popular size-oriented models are macroscopic in nature: they deal with estimation techniques for software project resources such as cost and effort. These models are empirical and are dependent to a large extent on historical data. Empirical resource models could be static single-variable, static multi-variable, dynamic multi-variable, or theoretical [13]. More notable examples of empirical models are the Constructive Cost Model (COCOMO) [14] for project cost estimation, and the Putnam Estimation Model [15, 16] for project effort and development time estimation.

The Putnam Estimation Model is a dynamic multi-variable model that assumes a specific distribution of effort over the development lifecycle. The model has been derived from manpower distribution encountered on large projects (with efforts greater than 30 person-years, 1-10,000 KLOC (kilo lines of code)). Extrapolation to smaller projects is also possible. Assuming a Rayleigh–Norden [17] manpower loading over the development life cycle, a fundamental equation called the software equation has been derived, which relates the effort, development time (time to full operational capability (FOC) of the software), and size of the project, using an overall productivity index (PI) for the organization based on historical data. Estimates for the size of the current project are typically done using delphi polling of experts in the application domain. The software equation is used in con-