Information and Software Technology 52 (2010) 92–109

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Software development effort prediction: A study on the factors impacting the accuracy of fuzzy logic systems

Zeeshan Muzaffar [a,*], Moataz A. Ahmed [b,1]

[a] Department of Computer Science, University of Western Ontario, Canada
[b] LEROS Technologies Corporation, Fairfax, VA 22030, USA

ARTICLE INFO

Article history:
Received 20 January 2009
Received in revised form 2 August 2009
Accepted 8 August 2009
Available online 14 August 2009

Keywords:
Effort prediction
Fuzzy logic
COCOMO
Imprecision
Uncertainty

ABSTRACT

Reliable effort prediction remains an ongoing challenge to software engineers. Traditional approaches to effort prediction such as the use of models derived from historical data, or the use of expert opinion are plagued with issues pertaining to their effectiveness and robustness. These issues are more pronounced when the effort prediction is used during the early phases of the software development lifecycle. Recent works have demonstrated promising results obtained with the use of fuzzy logic. Fuzzy logic based effort prediction systems can deal better with imprecision, which characterizes the early phases of most software development projects, for example requirements development, whose effort predictors along with their relationships to effort are characterized as being even more imprecise and uncertain than those of later development phases, for example design. Fuzzy logic based prediction systems could produce further better estimates provided that various parameters and factors pertaining to fuzzy logic are carefully set. In this paper, we present an empirical study, which shows that the prediction accuracy of a fuzzy logic based effort prediction system is highly dependent on the system architecture, the corresponding parameters, and the training algorithms.

1. Introduction

Software cost estimation, generally, refers to the prediction of the likely amount of effort, staffing level, time, and materials necessary to accomplish the required development task [15]. The effort prediction aspect of software cost estimation is concerned with the prediction of the person-hour required to accomplish the tasks. The development of effective software effort prediction models has been a research target for quite a long time [5]. The traditional parametric approaches at building effort models involve one or more mathematical formulae that, typically, have been derived through statistical data analysis (e.g., regression analysis) of historical data. These approaches identify key contributors to effort and use historical organizational projects data to generate a set of mathematical formulae that relates these contributors to effort. Such a set of mathematical formulae are often referred to as *parametric model* because alternative scenarios can be defined by changing the assumed values of a set of fixed coefficients (parameters). Typically, the accuracy of these parametric effort prediction models is improved via calibration to the target development environment. The calibration activity involves adjusting the parameters of the formulae [3,8,30]. Examples of popular effort prediction models include Constructive Cost Model (COCOMO) [7], Constructive Cost Model II (COCOMO II) [6], and Software Life Cycle Management (SLIM) [22,24]. These models are centered on using the future software size as the major determinant of effort. Detailed critical surveys on traditional effort prediction models are presented in [5,9]. However, such models have been shown to have a number of problems:

1. Difficulty handling categorical data (data that are defined by a range of values) [26].
2. Difficulty drawing conclusions or making judgments based on available data (that is, a lack of reasoning capabilities) [25,27].
3. Difficulty adapting to new environments as the models (i.e., formulae) are typically company specific [27].
4. Use of historical data has some limitations because predictors and relationships used to predict software development effort can vary over time, and/or differ among software development environments [29].
5. Difficulty capturing complex sets of relationships that are present in many software development environments (e.g., the effect of each variable in the model on the overall estimation

* Corresponding author. Tel.: +1 703 691 8122x225; fax: +1 703 691 8125.
E-mail addresses: smuzaff2@csd.uwo.ca (Z. Muzaffar), mahmed5@gmu.edu (M.A. Ahmed).

[1] This research was conducted while Dr. Ahmed was at King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia. He is currently CTO with LEROS Technologies Corporation and Adjunct Professor with George Mason University, Fairfax, VA 22030, USA.

0950-5849/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2009.08.001

made using the model). There are many factors, i.e., cost drivers [7] that can play a vital role in estimating software development effort. It is very difficult (if not impossible) for a model to consider all such factors within a mathematical model [17].
6. Difficulty incorporating experts' input; i.e., lack of transparency [1].
7. Difficulty handling uncertain data, i.e., data that contains noisy measurements [22].

The problems with traditional parametric models have emphasized the need for other techniques, such as Price-to-Win [7], Parkinson [7], expert judgment [3,7], and models based on machine learning (ML) approaches [26,28]. The ML approaches include: fuzzy logic (FL), analogy, regression trees, rule induction, Bayesian belief networks (BBN) [4], evolutionary computation (EC), and artificial neural networks (ANN). For a detailed discussion and evaluation of prominent ML-based effort prediction models, the reader can consult the survey work by Saliu and Ahmed [26].

The accuracy of any ML-based software effort prediction system depends on the setting of the parameters of the underlying technique. For example, with ANN, the topology as well as the neuron functions affect the network performance (e.g., its accuracy). Similarly, with EC (e.g., genetic algorithms), the selection of the operators as well as the corresponding probability values has a great impact on the performance. Software effort prediction systems using FL are no exceptions. In this paper, we present a study of the impact of major factors on the prediction accuracy.

The paper is organized as follows. In Section 2, we give a literature review of FLS based effort prediction approaches. Section 3 provides a brief background on FL and FL systems. Section 4 presents different factors impacting the accuracy of FL-based effort prediction systems. Section 5 presents experiments and results. Section 6 concludes the paper and points out possible directions for future research.

2. Effort prediction systems

Effort prediction systems involve a *prediction model* (with associated prediction procedures) and *measures*. An effort prediction model allows one to predict the effort as a function of several other variables. These variables represent key contributors to effort; they are typically called predictors—they are mainly characteristics of the future software and the corresponding development environment. Measures are used to assess such predictors for a given development project; the resulting assessments are fed into the model to predict the effort for that project. There are at least two important sources of information for building the prediction model: *historical data* and *human experts*. Historical data provide numerical *quantitative* measurements from past projects regarding the predictors and effort. Human experts use their past experience to provide *qualitative* descriptions of the correlation between predictors and effort.

Historical data are typically analyzed using means of regression analysis to build a parametric prediction model. Regression analysis assumes that all predictors are quantitative so that arithmetic operations such as addition and multiplication are meaningful [13]. Accordingly, regression-based models can only make use of quantitative measurements and have difficulties incorporating qualitative judgments.

As an example of a parametric quantitative prediction model, let us consider Intermediate COCOMO II [6]. It takes the following as input: (1) the estimated size of the software product in thousands of Delivered Source Instructions (KDSI) adjusted for code reuse [8], (2) the project development mode given as a constant value B (also called the scaling factor), and (3) seventeen cost drivers that have impact on the overall project effort. The development mode reflects one of three categories of software development modes: organic, semi-detached, and embedded. Respectively, B takes three values, {1.05, 1.12, 1.20}, which reflect the difficulty of the development. The cost drivers have up to six levels of rating: very low, low, nominal, high, very high, and extra high. Each rating has a corresponding real number (effort multiplier), based on the degree to which the cost driver can influence productivity.

The estimated effort in person-months (PM) for the Intermediate COCOMO is given as:

$$Effort = A \times [size]^B \times \prod_{i=1}^{17} EM_i \quad (1)$$

where $A \times [size]^B$ is termed as Nominal Effort, $\prod_{i=1}^{17} EM_i$ is called Effort Adjustment Factor (EAF) and EM_i is effort multiplier corresponding to a particular cost driver. The constant A is known as productivity coefficient. It takes the values 3.2, 3.0, and 2.8 for the three different modes, respectively.

To describe the nominal effort prediction model, experts may give a similar model but in a qualitative form such as [1]:

IF mode is Organic AND size is High THEN effort is Medium.
IF mode is Semi-detached AND size is High THEN effort is a Little-High.
IF mode is Embedded AND size is High THEN effort is High.
IF mode is Organic AND size is Medium THEN effort is Low.
...
Or in general,
IF mode is m_j *AND size is* s_i *THEN effort is* E_{ji} $(1 \leqslant i \leqslant n, 1 \leqslant j \leqslant 3)$

where $m_j (1 \leqslant j \leqslant 3)$ are qualitative values for the variable *mode*, $(1 \leqslant i \leqslant n)$ are qualitative values for the variable *size*, and E_{ji} $(1 \leqslant i \leqslant n, 1 \leqslant j \leqslant 3)$ are qualitative values for the variable *effort*.

Similarly, predictors' assessments come in two different forms: qualitative and quantitative. For example, one may ask an expert about his/her perception regarding the future size of the software based on information available during requirements specification. On the other hand, one may use mathematical models to predict the future size of the software based on information available from requirements development. The former assessment would be in a qualitative form whereas the latter would be in a quantitative form.

According to a recent study by Jørgensen and Shepperd [14], the experts' judgment based software effort prediction approaches are the most common used approaches by the software industry. Moreover, a study by Hodgkinson and Garratt claims that, in certain situations, prediction by expert judgment was better than all regression-based models [10]. The problem with expert judgment approaches, though, is that the experts are not always available, and accordingly a systematic consistent effort prediction is not possible. It would be highly beneficial if the software industry could be able to document and reuse experts' knowledge in different projects. This raised the need for prediction models that allow incorporating knowledge provided by experts in a systematic and efficient manner. The problem is that experts provide knowledge in a qualitative and imprecise form; human beings like to use linguistic terms (e.g., *small* size, *low* complexity, and *highly* safety critical, each of which represents a range of values) that depict their interval of confidence regarding a correlation or a perception. Therefore, the desirable models should be able to handle both quantitative and qualitative information simultaneously.

With its powerful linguistic representation, FL can represent imprecision in qualitative inputs and outputs [27]. Such represen-