Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

A hybrid model for estimating software project effort from Use Case Points

Q1 Mohammad Azzeh[a], Ali Bou Nassif[b,*]

Q2 [a] Department of Software Engineering, Applied Science University, P.O. BOX 166, Amman, Jordan
Q3 [b] Department of Electrical and Computer Engineering, University of Sharjah, Sharjah, United Arab Emirates

ARTICLE INFO

Article history:
Received 21 December 2015
Accepted 6 May 2016
Available online xxx

Keywords:
Effort estimation
Use Case Points
Radial basis neural networks
Support vector machine

ABSTRACT

Early software effort estimation is a hallmark of successful software project management. Building a reliable effort estimation model usually requires historical data. Unfortunately, since the information available at early stages of software development is scarce, it is recommended to use software size metrics as key cost factor of effort estimation. Use Case Points (UCP) is a prominent size measure designed mainly for object-oriented projects. Nevertheless, there are no established models that can translate UCP into its corresponding effort; therefore, most models use productivity as a second cost driver. The productivity in those models is usually guessed by experts and does not depend on historical data, which makes it subject to uncertainty. Thus, these models were not well examined using a large number of historical data. In this paper, we designed a hybrid model that consists of classification and prediction stages using a support vector machine and radial basis neural networks. The proposed model was constructed over a large number of observations collected from industrial and student projects. The proposed model was compared against previous UCP prediction models. The validation and empirical results demonstrated that the proposed model significantly surpasses these models on all datasets. The main conclusion is that the environmental factors of UCP can be used to classify and estimate productivity.

1. Introduction

Use Case Points (UCP) is a well-established software sizing technique that utilizes a UML use case diagram to estimate the size of object-oriented projects at the early stages of software development [2,16]. The basic idea of UCP was mainly inspired from another software sizing technique that depends mainly on functional requirements, called Function Points [1,24]. Karner [16] established a well-defined procedure to convert the use case diagram elements into a set of metrics that reflect the work effort needed to accomplish software projects. The translation procedure requires a standard of writing use case descriptions. Thus, it is recommended to avoid the free style that often depends on natural language and follow a common guideline [24]. The first version of UCP lacks validation and examination about its reliability for software organizations. The major challenge in UCP is the arbitrary numbers involved in calculating the software size. In fact, there is no justification as to how these numbers were found. In addition, there is no efficient method that can convert the UCP size into its corresponding software effort in terms of person-hours or person-months. Therefore, it was hard to build an effort prediction model because of the limitation in the number of collected projects. The first version of effort estimation based on UCP suggests the use of productivity as a second cost driver, as shown in Eqs. (1) and (2). This approach has long been used in many studies conducted on UCP, but the validity of this approach has not been well examined over a large number of observations.

Q5 Software productivity is defined as a ratio between effort and size [12,14]. This relationship has two contradicting interpretations. On one hand it can be defined as project productivity when it is measured as effort/size. On the other hand it can be defined as team productivity when it is measured as size/effort. Both definitions are used within the software engineering community, but the first one is preferable. The effort estimation model is usually constructed based on the productivity interpretation. Eq. (1) is used to compute effort when the productivity ratio is interpreted as project productivity. In contrast, Eq. (2) is used when the productivity ratio is interpreted as team productivity [2]. Nevertheless, computing the software productivity must be made before the effort can be estimated. This may depend on many variables such as: reuse percentage, type of software process, team communications,

Q4 * Corresponding author.
E-mail addresses: m.y.azzeh@asu.edu.jo (M. Azzeh), anassif@sharjah.ac.ae, ali.bounassif@gmail.com (A.B. Nassif).

http://dx.doi.org/10.1016/j.asoc.2016.05.008
1568-4946/

and the number of deliverables. Although adopting good development practices may increase the productivity, it does not always do so because of circumstances outside the control of the software development team.

$$\text{Effort} = \text{Productivity} \times \text{UCP} \quad (1)$$

$$\text{Effort} = \frac{\alpha}{Productivity} \times \text{UCP}^{\beta} \quad (2)$$

Estimating effort from UCP can fall into one of three approaches. The first approach presumes that there are no historical projects available within the software organization so the project manager must pre-determine the productivity ratio for the software project. In this case, the decision is heavily dependent on the estimator and subject to a large degree of bias. Typical examples that have followed this approach are the studies conducted by Karner [16] and Schneider and Winters [18]. Karner [16] proposed a fixed productivity ratio (=20 person-hours/UCP) for all software projects. This approach is not practical because it does not take into consideration the type, complexity, domain, and environments of a software project. In contrast, Schneider and Winters [18] proposed three levels of productivity (fair = 20, low = 28, and very low = 36) person-hours/UCP based upon analysing the environmental factors of a software project. The second approach uses machine learning and data mining techniques to build regression models that exhibit the relationship between effort and UCP. This approach does not need to pre-determine the productivity but needs historical data to build the regression model. Nevertheless, this approach is affected by the number of projects in the training set, setup parameters, and validation procedure. The third approach attempts to use both of the above approaches in one model. An example of this scenario is the work proposed by Nassif et al. [2], who proposed four levels of the productivity ratio based on the weighted sum of the environmental factor and using an expert-based fuzzy model. Nassif et al. [2] built a log-linear regression model that uses UCP and productivity.

Above all, we can see that all previous models were constructed using a very limited number of observations. In addition, the assumptions made about using productivity ratios have not been well examined. Using fixed or limited productivity ratios did not contribute well to improving prediction accuracy. No previous studies attempted to study the relationship between productivity and environmental factors when historical data was available. In fact, the productivity prediction should be flexible and adjustable when historical data is present. The flexibility means that the productivity must be affected by the UCP factors assessment. The adjustability means the productivity of one project should be adjusted based on the productivity from the historical projects. Finally, there is no study that has attempted to examine the effect of using UCP components with productivity to predict effort.

Stimulated by this situation, we proposed a new effort estimation model that can support management decisions during the feasibility study and project inception. The proposed model consists of two stages. In the first stage, the historical productivity is clustered to create fine-grained productivity labels and then classified based on environmental factors. For that purpose we used the bisecting k-medoids clustering technique [19,22] and support vector machine [11]. The predicted productivity, computed during the test phase, is based on the centre of the predicted productivity label. The studies conducted by Nassif et al. [2] and Schneider and Winters [18] showed that the environmental factors can work as good indicators for software productivity since they reflect the team workload within the software project. In the second stage, the effort estimation model is built using Radial Basis Neural Network (RBFNN) [15]. The model is trained using historical UCP and productivity variables. Then during the estimation process, the predicted productivity from stage one is entered with the UCP of the new project as input to RBFNN to predict effort. The proposed model has advantages over previous models in that it can learn productivity from environmental factors using classification and decomposition techniques. So the number of productivity levels in each training set depends on the structure of that set. It also offers a non-linear learning mechanism to mimic the relationship between effort and two other predictors (UCP and productivity)

The rest of this paper is organized as follows: Section 2 gives an introduction to Use Case Points. Section 3 presents related work. Section 4 introduces the proposed model. Section 5 presents research methodology. Section 6 shows the empirical results and discussion. Section 7 presents threats to validity and, finally, Section 8 presents conclusions.

Table 1
Types of actors.

Type	Description
Simple	Actor interacts using API
Average	Actor interacts using text-based interface
Complex	Actor interacts using Graphical User Interface

Table 2
Types of use cases.

Type	#transactions
Simple	≤3
Average	4–7
Complex	>7

2. An overview of Use Case Points

The UCP estimation method was first introduced by Karner [16] to predict the size of object-oriented software projects. The UCP is computed by converting the elements of UML use case diagram to size metrics through a well-defined procedure. In the first step, the estimator must classify the actors in the use case diagram into three categories according to their difficulties: simple, average, and complex, as shown in Table 1. Based on that the Unadjusted Actor Weights (UAW) is computed, as shown in Eq. (3). Similarly, the use cases are also classified into three classes (simple, average, and complex) based on the number of transactions mentioned in the use case descriptions, shown in Table 2. A transaction is defined as a stimulus and response occurrence between the actor and the system [21]. Based on that, the UUC is calculated as shown in Eq. (4). The Unadjusted Use Case Points (UUCP) is computed based on the summation of UAW and UUC.

$$\text{UAW} = 1 \times \text{sa} + 2 \times \text{aa} + 3 \times \text{ca} \quad (3)$$

where sa, aa, ca are the numbers of simple, average, and complex actors respectively.

$$\text{UUC} = 5 \times \text{suc} + 10 \times \text{auc} + 15 \times \text{cuc} \quad (4)$$

where suc, auc, cuc are the numbers of simple, average, and complex use cases respectively.

$$\text{UUCP} = \text{UAW} + \text{UUC} \quad (5)$$

Finally, the UUCP should be adjusted by two sets of adjustment factors: Technical Complexity Adjustment Factor (TCF) and Environmental Adjustment Factor (EF). TCF is computed from a set of 13 technical factors ($F_1, F_2, \ldots, F_{13}$) that have great influence on project performance. Similarity, EF is computed from a set of eight environmental factors ($E_1, E_2, \ldots, E_8$) that have great effect on productivity. Each factor in both sets can take an influence value between zero and five and predefined weights that reflect the influence of that factor. Eqs. (6) and (7) demonstrate how TCF and EF are calculated,

Please cite this article in press as: M. Azzeh, A.B. Nassif, A hybrid model for estimating software project effort from Use Case Points, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.05.008