The Journal of Systems and Software 125 (2017) 1–14

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Analysis and selection of a regression model for the Use Case Points method using a stepwise approach

Radek Silhavy*, Petr Silhavy, Zdenka Prokopova

Tomas Bata University in Zlin, Faculty of Applied Infomatics, Nad Stranemi 4511, 76001, Zlin, Czech Republic

ARTICLE INFO

Article history:
Received 3 March 2016
Revised 15 November 2016
Accepted 18 November 2016
Available online 22 November 2016

Keywords:
Software size estimation
Stepwise approach
Multiple linear regression
Use Case Points
Dataset
Variables analysis

ABSTRACT

This study investigates the significance of use case points (UCP) variables and the influence of the complexity of multiple linear regression models on software size estimation and accuracy.

Stepwise multiple linear regression models and residual analysis were used to analyse the impact of model complexity. The impact of each variable was studied using correlation analysis.

The estimated size of software depends mainly on the values of the weights of unadjusted UCP, which represent a number of use cases. Moreover, all other variables (unadjusted actors' weights, technical complexity factors, and environmental complexity factors) from the UCP method also have an impact on software size and therefore cannot be omitted from the regression model. The best performing model (Model D) contains an intercept, linear terms, and squared terms. The results of several evaluation measures show that this model's estimation ability is better than that of the other models tested. Model D also performs better when compared to the UCP model, whose Sum of Squared Error was 268,620 points on Dataset 1 and 87,055 on Dataset 2. Model D achieved a greater than 90% reduction in the Sum of Squared Errors compared to the Use Case Points method on Dataset 1 and a greater than 91% reduction on Dataset 2. The medians of the Sum of Squared Errors for both methods are significantly different at the 95% confidence level ($p < 0.01$), while the medians for Model D (312 and 37.26) are lower than Use Case Points (3134 and 3712) on Datasets 1 and 2, respectively.

1. Introduction

Predicting the effort required to create software has been based on numerous software size models such as the Constructive Cost Model (Anandhi and Chezian, 2014; Clark, 1996; Manalif et al., 2014) and all its alternatives (Attarzadeh and Ow, 2011; Kazemifard et al., 2011; Tadayon, 2004; Yang et al., 2006) as well as on function points (Borandag et al., 2016) and analogy based models (Idri et al., 2015). The main goal of all these approaches is to minimize prediction error. Prediction is needed during the initial phase of software project developments. One significant approach to software size prediction is the Use Case Points (UCP) method, which is a prediction model based on the work of Karner (1993). Azevedo et al. (2011) brings a discussion about influence of extends association in use cases, which helps to count UCP more precisely. Software size prediction through use case analysis addresses object-oriented design; thus, this method is now widely used. As reported in Silhavy et al., (2015a,b) UCP has some important drawbacks. Several approaches help identify the drawbacks of the UCP method and offer solutions, many of which are based on an analogy approach. Analogy based size estimation is commonly used for prediction in all the methods mentioned above (Idri et al., 2015; Shepperd and MacDonell, 2012). Many researchers have addressed effort estimation and, therefore, consider productivity factors (PFs) (Wang et al., 2009), but they do not address the possibility of potentially inappropriate variables in the UCP algorithm itself, which is important for software size estimation. Humans introduce errors when evaluating actors or use cases. Therefore, the goal of this study is to improve size estimation accuracy by minimizing the influence of human errors during Use Case model analysis and other influences that are understood as unsystematic noise. The noise is not addressed in the UCP equation. Multiple Regression Models (MLR) handles any unsystematic noise by selecting new formula and values of regression coefficients. This new formula will achieve better prediction performance than UCP, as will be shown later in the text.

First, the UCP variables and their impacts on size estimation are analysed to determine whether using all the variables is appropriate when predicting software size. Second, this study discusses

* Corresponding author.
E-mail addresses: rsilhavy@fai.utb.cz (R. Silhavy), psilhavy@fai.utb.cz (P. Silhavy), prokopova@fai.utb.cz (Z. Prokopova).

http://dx.doi.org/10.1016/j.jss.2016.11.029

the selection of MLR models based on the UCP variables that can improve the UCP method and make the estimation less sensitive to unsystematic noise.

1.1. Related work

The UCP method is based on use case models, which are commonly used as functional descriptions of proposed systems or software. The method involves assigning weights to groups of actors and use cases. Karner's original UCP method (Karner, 1993) identifies three groups: simple, average and complex. The sum of the weighted actors creates a value called unadjusted actor weights (UAW); the unadjusted use case weights (UUCW) value is defined similarly. Two variables, called technical complexity factors and environmental complexity factors, are used to describe the project, related information and the experience level of the development team. A final UCP score is obtained by summing the UAW and the UUCW and then multiplying the resulting value by the technical and environmental factor coefficients.

A number of use case scenario steps are typically involved in the initial estimation process. There have also been several modifications of the original UCP principles including use case size points (Braz and Vergilio, 2006), extended UCP (Wang et al., 2009), modified UCP (Diev, 2006), adapted UCP (Mohagheghi et al., 2005), and transaction or path analysis (Robiolo et al., 2009).

The use case size points method was evaluated in Braz and Vergilio (2006). The authors emphasised the internal structure of the use case scenario in their method, where the primary actors take on roles and are classified based on an adjustment factor. This approach can lead to better evaluations of actors and use cases. Fuzzy sets are used for the estimations.

Several authors have presented improvements to Karner's method based on the identification of transactions rather than steps in use cases. This approach is based on analysing a scenario, not step by step, but using steps merged logically into so-called transactions in which each transaction should include more than one step. Robiolo et al., (2009) improved transactions by calculating paths by which the complexity of each transaction is based on the number of binary or multiple conditions used in the scenarios. Their approach is based on Robiolo and Orosco (2008), where number of transactions is equal to the number of stimuli. A stimulus is a system entry point, which generates response (transaction) of an actor action in a use case. Ochodek et al., (2011a) discusses a reliability of transaction identification process and Jurkiewicz et al. (2015) discusses event identification in use cases, which should be useful for path identification.

Wang et al., (2009) proposed an extended UCP in that employed fuzzy sets and a Bayesian belief network used to set unadjusted UCP. The result of this approach was a probabilistic effort estimation model.

Diev (2006) noted that when the actors and use cases are precisely defined, unadjusted UCP (the sum of the UAW and the UUCW) can be multiplied by the technical factors. The product of the technical complexity factors (see Table 3) and unadjusted UCP is considered as the coefficient of the base system complexity in Diev (2006). According to Nageswaran (2001), added effort must be taken to consider support activities such as configuration management or testing.

Yet another modification to the UCP is called adapted UCP (Mohagheghi et al., 2005). In this method, the UCP method is adapted to provide incremental development estimations for large-scale projects. Initially, all actors are classified as average (based on the UCP native classifications) and all use cases are classified as complex. Ochodek et al., (2011b) also proposed omitting UAW and the decomposition of use cases into smaller ones, which are then classified into the typical three use case categories.

However, the existing use case-based estimation methods have some well-known issues (Diev, 2006). Use cases are written in natural language; consequently, there is no rigorous approach for comparing use case quality or fragmentation. The number of steps in use case scenarios may vary, which affects the estimation accuracy. Moreover, an individual use case may contain more than one scenario, which also affects estimation accuracy. Thus, although the use case model is critical for system functional or behavioural modelling, use cases can be employed for estimation purposes only if the estimation approach can be adjusted or calibrated. Such calibration methods can minimize estimation errors, mainly in situations when the errors are constant. Furthermore, aspects such as bugs, new requirements or improvements cannot be resolved by UCP estimation.

All these aspects can be solved by UCP improvements based on analogy or regression approaches. Analogy based estimation methods are discussed in Azzeh et al., (2015b), which evaluated 40 variants of the single adjustment method using four performance measures and eight test datasets. However, none of the tested methods were based on UCPs.

Amasaki and Lokan (2015) addressed the problem of selecting projects using a linear regression model by testing the window principle. The window principle involves first selecting a subset of the data. Then, the estimation algorithm works with that subset only. Their results showed that weighted moving windows have a statistically significant effect on estimation accuracy and that various weighting functions influenced estimation accuracy differently (i.e., weighted moving windows have significant advantages when the window is large. Likewise, unweighted moving windows are significantly advantageous when the window is small. Rosa et al., (2014) investigated whether a linear model based on both size and application type was better than a model based on size only; however, this study did not investigate the effects of each variable nor evaluate additional types of regression models.

López-Martín (2015) described linear regression models as less accurate than neural networks, but they provided no description of the regression models studied. Moreover, they did not consider the stepwise principle for model construction nor did they investigate whether all the UCP variables contribute to size estimation. A discussion of variable significance can be found in Urbanek et al., (2015a). Silhavy et al., (2015a, 2015b) offered a linear model obtained by the least squares approach, in which two prediction coefficients were used to adjust the UAW and the UUCW. These studies did not focus on evaluating of variables for use in regression models, nor did they compare linear and polynomial regression models.

Urbanek et al., (2015b) described the number of points in the use case scenario as the most significant factor, but the scope of this paper is analytical programing; therefore, this finding is not applicable to MLR. Instead, the study by Urbanek et al., 2015b is based on artificial intelligence and is an application of the approach proposed by Senkerik et al., (2014) but with theoretical aspects of Oplatkova et al., (2013). Urbanek et al., 2015b used a symbolic regression tool, analytic programming, together with differential evolution.

Several works have attempted to apply various prediction models to UCP. Nassif et al., (2013) presented a linear regression model with a logarithmic transformation that they created to estimate software effort from use case diagrams. In Nassif et al., (2011), a multiple linear regression model was developed to predict the values of the productivity factor. To adjust the values of the productivity factor, they first employed a fuzzy logic approach (Nassif et al., 2011). Then, they created an artificial neural network (multi-layer perceptron) model (Azzeh and Nassif, 2016; Nassif et al., 2015; Nassif et al., 2012, 2013).

The main distinction between this paper and existing approaches is that we propose a novel approach for estimating soft-