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-

ware development effort from use case diagrams that aims to improve on Karner's method by implementing multiple linear regression models. Furthermore, none of the existing works proposed a suitability analysis of UCP variables to investigate their effects on predicting software size.

2. Problem statement

In this study we analyse the UCP variables that are used in UCP models. We evaluate the UAW, UUCW, TCF and ECF to determine their contributions to software size estimation. The correlations to real project size and correlations among independent variables are measured. This study discusses the selection of MLR models based on UCP variables, which should improve the UCP method and make estimations less sensitive to the introduction of errors. We assume that MLR handles any unsystematic noise not addressed in the UCP equation. Therefore, the analysis of several types of MLR models (Table 8) is presented. Moreover, the best performing model will indicate which UCP variables contribute to size estimation. Consequently, the first research question concerns the UCP variables and the second research question concerns MLR model complexity. Here, the number of model terms is treated as a measure of model complexity.

2.1. Research questions

The research questions answered by this study are as follows:

RQ1: Are all UCP variables significant in estimation?
RQ2: Does MLR complexity improve estimation model accuracy?

2.2. Evaluation criteria

All the tested models were evaluated according to (1) the adjusted coefficient of determination (R^2), (2) the residual sum of squares (RSS), (3) mean squared error (MSE), (4) root mean squared error (RMSE), and (5) the Akaike information criterion for finite sample size (AICc). Their equations are given as follows:

$$R^2 = 1 - \left[\frac{\sum_i^n (y_i - \widehat{y_i})^2}{\sum_i^n (y_i - \bar{y})^2} \right], \tag{1}$$

$$RSS = \sum_{i=1}^{n} \varepsilon_i^2, \tag{2}$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \varepsilon_i^2, \tag{3}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^n \varepsilon_i^2}{n}}, \tag{4}$$

$$AICc = 2k - 2\ln(L) + \frac{2k \times (k+1)}{n-k-1}, \tag{5}$$

where R^2 is the coefficient of determination, which illustrates model variability, n is number of observations, k is the number of independent predictors, L is a maximum value of the likelihood function and ε is a residual error value.

3. Methods used

3.1. Use Case Points

The basic UCP method is based on assigning weights to clustered actors and use cases. It employs three cluster types: simple, average, and complex. The sum of the weighted actors creates a value called UAW; the UUCW value is calculated similarly. Two coefficients, technical factors and environmental factors, are used to describe the project, related information, and the experience level of the development team.

Actors play roles in the UAW variables (Azzeh et al., 2015a; Silhavy et al., 2015a). A simple actor typically represents an application programming interface and a complex actor represents a human using a graphical user interface. The actor groups and weighting factors for Eq. (6) are summarised in Table 1.

Table 1
Actor classification and weighting factors.

Actor classification (AC)	Weighting factor (WFa)
Simple	1
Average	2
Complex	3

Table 2
Use case classification and weighting factors.

Use case classification (UCC)	Number of steps	Weighting factor (WFb)
Simple	⟨0,4⟩	5
Average	⟨4,7⟩	10
Complex	⟨7, ∞⟩	15

Table 3
Technical factors.

Factor ID	Description	Weighting factor (WFc)	Significance (SIa)
T1	Distributed System	2	⟨0,5⟩
T2	Response adjectives	2	⟨0,5⟩
T3	End-User Efficiency	1	⟨0,5⟩
T4	Complex Processing	1	⟨0,5⟩
T5	Reusable Code	1	⟨0,5⟩
T6	Easy to Install	0.5	⟨0,5⟩
T7	Ease to Use	0.5	⟨0,5⟩
T8	Portable	2	⟨0,5⟩
T9	Easy to Change	1	⟨0,5⟩
T10	Concurrent	1	⟨0,5⟩
T11	Security Feature	1	⟨0,5⟩
T12	Access for Third Parties	1	⟨0,5⟩
T13	Special Training Required	1	⟨0,5⟩

The UAW are calculated according to the following formula:

$$UAW = \sum^{A} C \times WFa. \tag{6}$$

Use cases are classified in a similar manner (see Table 2). The complexity of a use case is based on the number of scenario steps or, sometimes, on the number of transactions it contains (Ochodek et al., 2011a). However, a transaction typically refers to a set of activities, not a simple step in a structured scenario. Therefore, the term "step" which is used here can be used in the meaning of step or transaction. The absolute number of scenario steps are used in counting, if the use case is extended by (or included in) another use case, those steps are not counted; in other words, such nested use cases are counted as separate scenarios.

The UUCW are calculated according to the following formula:

$$UUCW = \sum^{U} CC \times WFb. \tag{7}$$

The unadjusted UCP (UUCP) is then calculated by summing the UAW and the UUCW.

$$UUCP = UAW + UUCW. \tag{8}$$

Technical Complexity Factors (TCF) and Environmental Complexity Factors (ECF) later correct UUCP. The TCF is considered as a correction factor that describes a set of important factors for the project. Table 3 presents the technical factors.

Table 4
Environmental factors.

Factor ID	Description	Weighting factor (WFd)	Significance (SIb)
E1	Familiar with the rational unified process (RUP)	1.5	⟨0,5⟩
E2	Application Experience	0.5	⟨0,5⟩
E3	Object-oriented Experience	1	⟨0,5⟩
E4	Lead Analyst Capability	0.5	⟨0,5⟩
E5	Motivation	1	⟨0,5⟩
E6	Stable Requirements	2	⟨0,5⟩
E7	Part-Time Workers	−1	⟨0,5⟩
E8	Difficult Programming Language	2	⟨0,5⟩

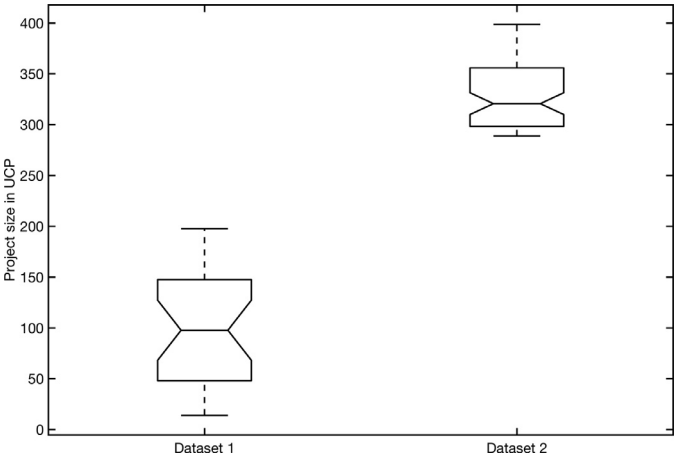

Fig. 1. Boxplots of projects in UCP.

The TCF can be calculated according to the following formula (Robiolo et al., 2009):

$$TCF = 0.6 + \left(0.01 \times \sum_{T13}^{T1} WFc \times SIa\right). \quad (9)$$

The second correction value is based on the ECF, which describes non-functional requirements. Table 4 presents the ECF as defined in the UCP. The ECF is calculated as follows:

$$ECF = 1.4 + \left(-0.03 \times \sum_{E8}^{E1} WFd \times SIb\right). \quad (10)$$

Factors T1–T13 and E1–E8 have fixed weights (WFc, WFd). Moreover, for each factor, the significance (SIa, SIb) can be set to a value between 0 and 5, where 0 indicates no impact, 3 indicates an average impact, and 5 indicates a strong impact.

The final size estimation is called an adjusted UCP (AUCP) and represents the project (system or software) size in points. To obtain the AUCP the UUCP, TCF and ECF are multiplied using the following formula (Karner, 1993; Ochodek et al., 2011b; Robiolo and Orosco, 2008; Wang et al., 2009):

$$UCP = UUCP \times TCF \times ECF. \quad (11)$$

3.2. Linear regression models

Linear regression models describe the relationship between a dependent variable and one or more independent variables. The goal is to find the best fit straight line that minimizes the sum of squared residuals of the linear regression model. The least squares method is the most common method used to fit a regression line. The case when a linear regression has only one independent variable is called simple linear regression (Bardsiri et al., 2014; Jorgensen, 2004; Montgomery et al., 2012; Shepperd and MacDonell, 2012), whereas multiple linear regression (Bardsiri et al., 2014; Jorgensen, 2004; Montgomery et al., 2012; Shepperd and MacDonell, 2012) involves more than one independent variable.

The multiple linear regression model is defined as follows:

$$y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_p X_{ip} + \varepsilon_i, \; i = 1, \ldots n \quad (12)$$

where y_i is the dependent variable, $X_{i1} \ldots X_{ip}$ are independent variables (predictors), β_0 is an intercept, and $\beta_1 \ldots \beta_n$ are regression coefficients. The value of ε_{ii} represents the error residuals. The model is designed as a matrix, where each row represents a data point.

Another type of linear regression is polynomial regression (Bardsiri et al., 2014; Jorgensen, 2004; Shepperd and MacDonell, 2012) in which the relationship between the dependent variable and the independent variables is modelled as an m^{th} degree polynomial:

$$y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2}^2 + \ldots + \beta_p X_{ip}^m + \varepsilon_i, \; i = 1, \ldots n \quad (13)$$

In matrix notation Eqs. (12) and (13) could be written as follows

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \quad (14)$$

Using ordinary least squares estimation, the vector of estimated regression coefficients is

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \quad (15)$$

In the case of multiple independent variables it is appropriate to use stepwise regression (Bardsiri et al., 2014; Jorgensen, 2004; Shepperd and MacDonell, 2012). The aim of the stepwise regression technique is to maximize the estimation power using the minimum number of independent variables. Stepwise regression is a combination of forward and backward selection that involves an automatic process for selecting independent variables and can be briefly described as follows:

(1) Set a starting model, which contains predefined terms;
(2) Set limits for the final model—what type of model is needed, whether linear terms are used, squared terms, or vice versa;
(3) Set an evaluation threshold (in our case this is whether the Sum of Squared Errors (SSE) is significantly decreased);
(4) After adding or removing terms, retest the model;
(5) Stepwise regression halts when no further improvement in estimation occurs.

There is a modification of forward selection such that after each step in which a variable is added, all the candidate variables in the model are checked to see whether their significance has been reduced below a specified tolerance threshold. Forward selection starts without any variables in the model and then iterates to add each variable. When a non-significant variable is found, it is removed from the model. Backward selection works in a similar manner, but removes variables when they are found to be non-significant. Therefore, stepwise regression requires two significance levels: the first for adding variables and the second for removing variables.

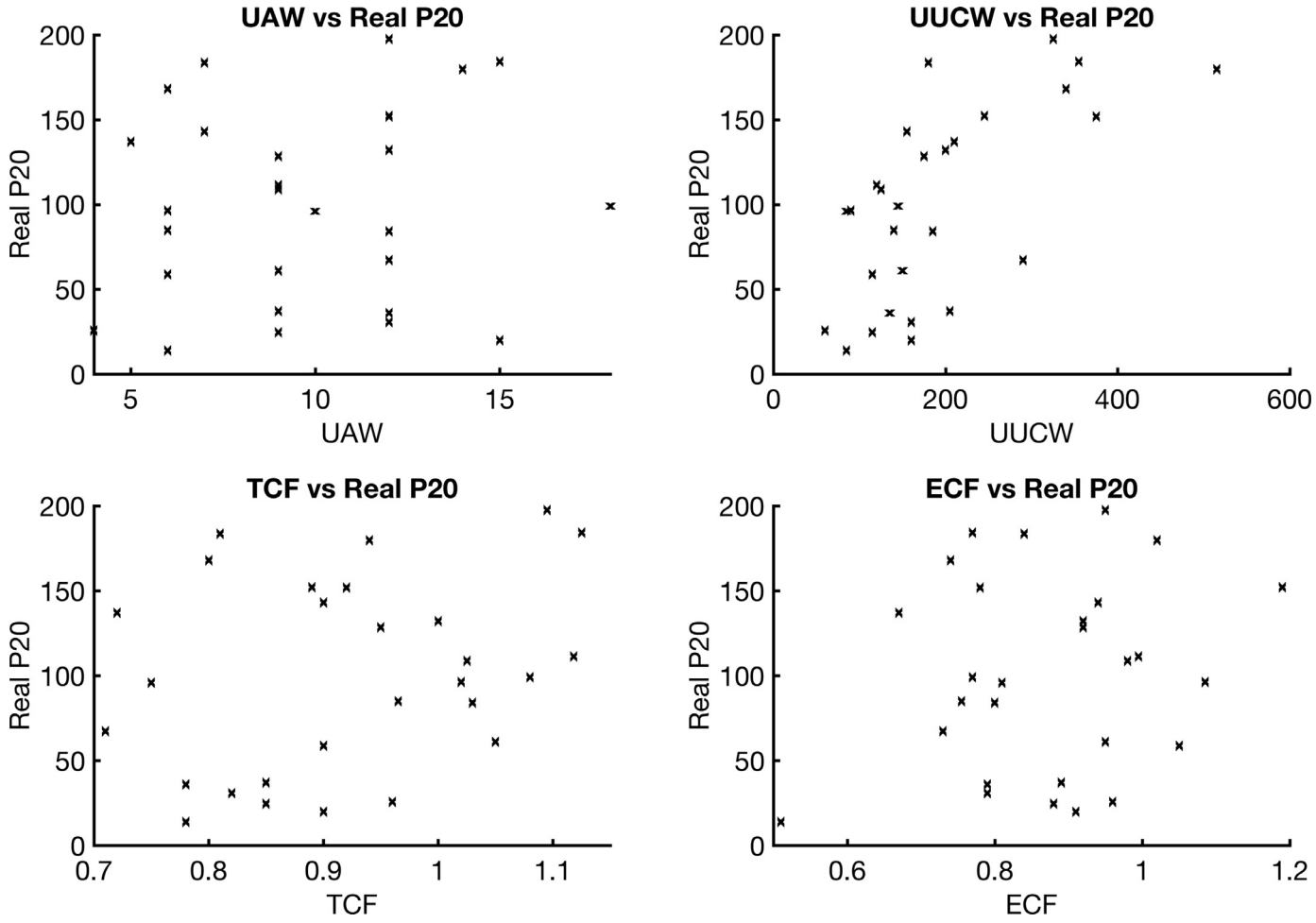

Fig. 2. Scatter plots for Pearson's correlation (independent vs. dependent Real_P20) for dataset 1.

4. Experiment design

The research process consists of the following steps:

(1) Obtaining data for experiments,
(2) Analysing the assumptions for linear regression,
(3) Analysing the correlations between the independent and dependent variables,
(4) Creating a multiple regression model, in which each variable/term is added and the entire model is retested,
(5) Comparing models and selecting the best performing model,
(6) Comparing the selected model and the UCP model.

MLR models are designed as a matrix in which each row represents a data point in Eq. (12) or Eq. (13). In this study, the regression models require historical data to evaluate the effort required in a new project, where the dependent variable y_i equals the Real_P20 vector and the UCP attributes (UAW, UUCW, TCF, and ECF) are used as independent variables (X_i). Real_P20 describes the real project size in points (UCP) obtained as person-hours divided by 20 (constant which represents productivity factor). Each model should contain a sufficient number of independent variables because of the stepwise approach to model construction. The principle of the stepwise approach was discussed in Section 3 of this paper. Therefore, interaction variables or squared independent variables can be added to the models.

The proposed models were constructed using standardized complexity and were selected based on how well they represent different approaches to linear regression. All the models were programmed in computational software and the stepwise principle was followed. Each step was evaluated using a significance level of $p = 0.05$ for the SSE criterion as the threshold for adding or removing variables in the model. The tested models are summarized in Table 5.

The comparative analysis of regression models is a basic research experiment. Statistical linear regression analysis was performed for all the models. The aim of this analysis was to identify the model that is best able to predict complexity and achieve the best values of the evaluation measures. The models are compared according to Eqs. (1)–(5). Predictors were used as variables, as described in the following section, and all of them were obtained by UCP. This approach makes replication experiments simple using any other dataset or using only a set of use case models, because all the variables were prepared according to the UCP method (see Eqs. (6), (7), (9), and (10)).

5. Experiment evaluation

5.1. Project datasets

The experiment described above was evaluated using two datasets. Dataset 1 was obtained from Silhavy et al., (2015a), in which the dataset was based on Ochodek et al., (2011b) and Subriadi and Ningrum (2014). The UAW, UUCW, TCF, and ECF are known from the UCP method. Dataset 2 was collected by the authors from three data donators (D1, D2 and D3) and are based on

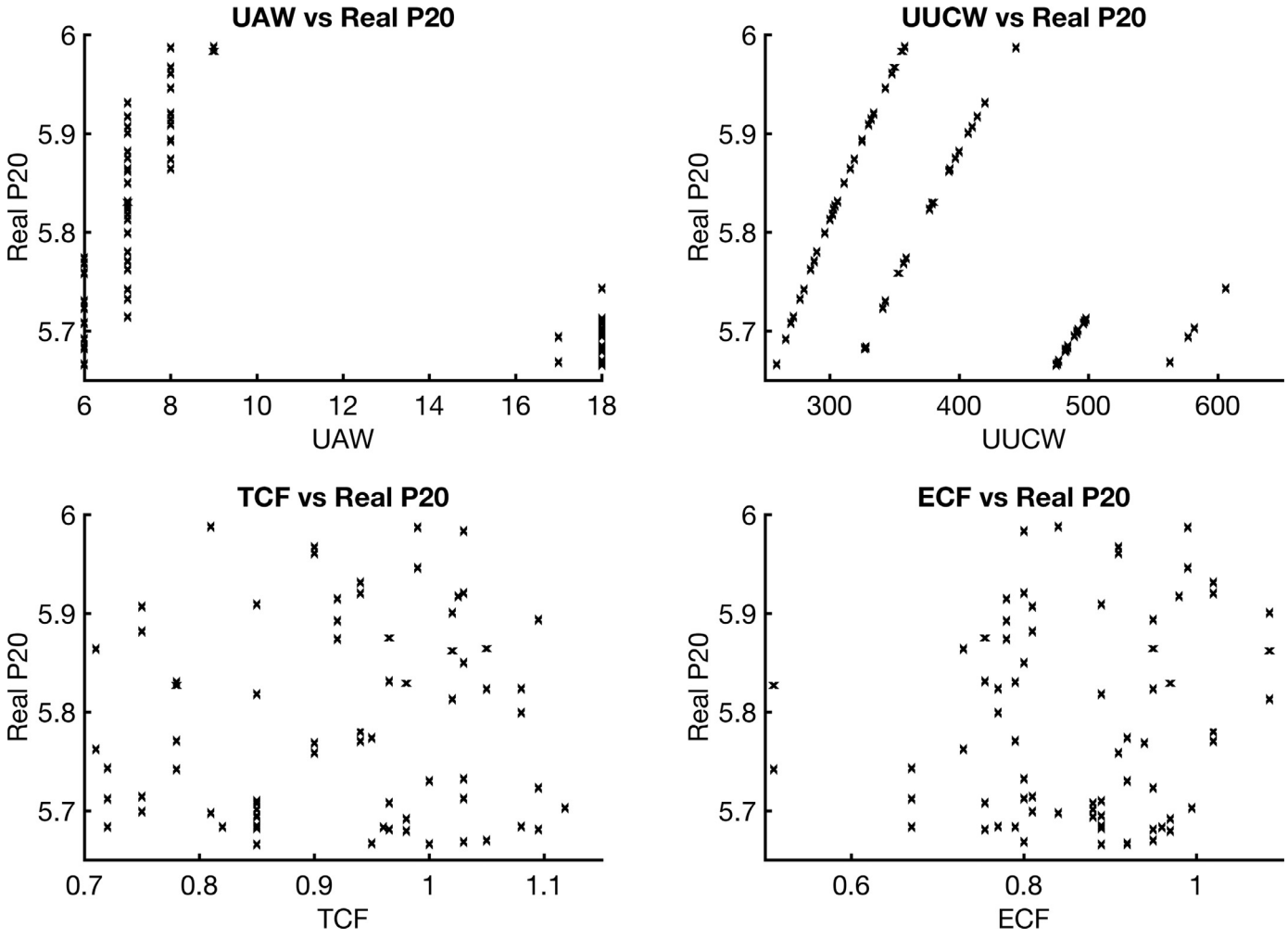

Fig. 3. Scatter plots for Pearson's correlation (independent vs. dependent Real_P20) for dataset 2.

Table 5
Regression model list.

Model A	The model contains a constant (intercept) term only.
Model B	The model contains an intercept and linear terms for each predictor.
Model C	The model contains an intercept, linear terms, and all products of pairs of distinct predictors (no squared terms).
Model D	The model contains an intercept, linear terms, and squared terms.
Model E	The model contains an intercept, linear terms, interactions, and squared terms.
Model F	The model contains no intercept; the terms are used as they appear in UCP equation. Therefore the regression formula was specified as $Real_{P20} \sim (UAW + UUCW) \times TCF \times ECF$.

Table 6
Dataset characteristics.

	Median person-hours	Median Real_P20	Range Real_P20	Standard deviation	Minimum Real_P20	Maximum Real_P20	n
Dataset 1	1952.500	97.625	183.650	57.063	13.850	197.500	28
Dataset 2	7012.000	320.600	109.750	33.394	288.750	398.500	71

following problem domains: Insurance, Government, Banking and other domain (see Attachments for more details).

For purpose of this study we do not aggregate data by data donators or problem domains. Fig. 1 shows a boxplot of the datasets. As shown, the two datasets represent groupings of different project sizes. Both datasets are available as attachments.

Table 6 shows the characteristics of the datasets used in the experiments. As shown, both datasets have a similar standard deviation, but Dataset 2 contains larger projects. All the values in Table 6 are based on the Real_P20, which describes the real project size in points (UCP).

5.2. Analysis of UCP variables

The correlations between the variables for Dataset 1 are illustrated in Fig. 2 and for Dataset 2 in Fig. 3. Dataset 1 has a high positive correlation only between UUCW and Real_P20, which shows that the number of use cases is the most important variable in estimating the real project size. The other three parameters are only weakly correlated. Dataset 2 correlations (Fig. 3) should be caused by using historical data points from different software companies across different problem domain. This phenomenon will be under further investigation in future research.

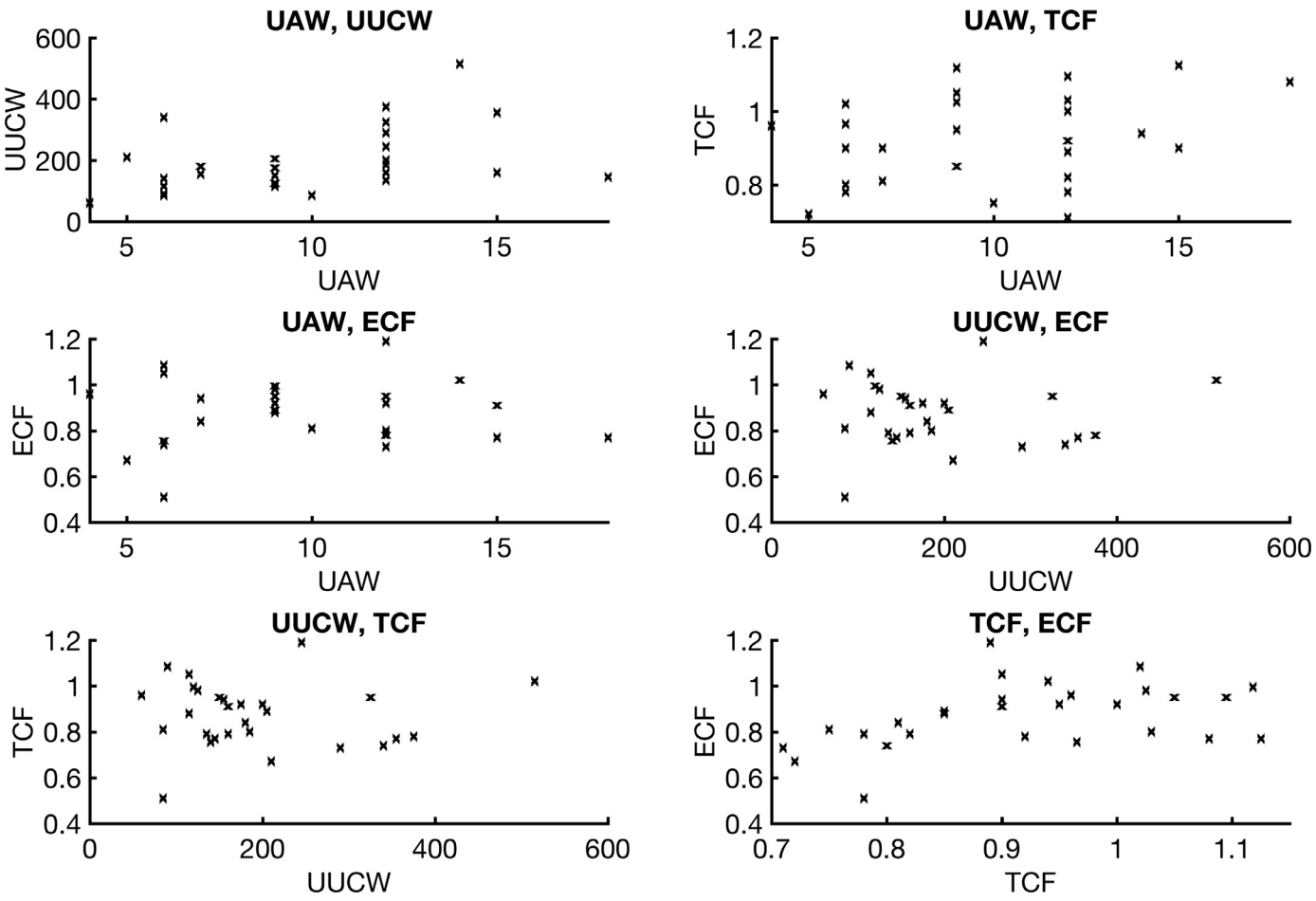

Fig. 4. Scatter plots for the correlation analysis of independent variables using dataset 1.

Table 7
Correlation coefficients between independent variables and the dependent variable.

Variable	Dataset 1 Real_P20	Dataset 2 Real_P20
UAW	0.142	−0.531
sig	0.471	<0.001
n	28	71
UUCW	0.656	−0.338
sig	<0.001	0.004
n	28	71
TCF	0.277	0.084
sig	0.153	0.486
n	28	71
ECF	0.189	0.126
sig	0.357	0.295
n	28	71

As Table 7 shows, all the independent variables are correlated to dependent variables for both datasets. The correlation values range from 0.142 to 0.656 for Dataset 1, in which the highest correlation is for UUCW. The Dataset 2 correlations range from −0.531 to 0.126. Weak correlations can be seen for both the TCF and ECF factors. The correlation between the UUCW and Real_P20 is −0.338, while interestingly, the correlation between the UAW and Real_P20 is −0.531.

The independent variables correlation analysis is presented in Fig. 4 (Dataset 1) and in Fig. 5 (Dataset 2). Table 8 shows that the variables UAW and UUCW are correlated (0.423, resp. 0.869), which is expected, because when there are more actors in the use case models, more use cases can be expected. This should have an impact on model variability. Contrastingly, UAW or UUCW are important variables for application of method in problem domain, therefore UAW or UUCW were not omitted. Moreover, both variables were found significant for estimation by stepwise approach. TCF and ECF show signs of correlation too, but there is no obvious reason for the correlation; we will investigate this aspect further in future research.

5.3. Regression model evaluations

The regression models obtained from stepwise regression were set according to Table 5. For each model, all four variables were used. Tables 9 and 10 show the regression formulas obtained for each model. All the models are presented in Wilkinson Notation. Models A, B, C and F are linear models, while Models D and E are classified as polynomial models.

Further, Fig. 6 shows the histograms of the residuals for Models A–F, using Dataset 1, and Fig. 7 shows the histograms of the residuals using Dataset 2.

Tables 11 and 12 list the selected evaluation measures of Models A–F for Datasets 1 and 2, respectively. The R^2 values show the coefficients of determination, which are calculated according to Eq. (1). The MSE was calculated based on the MLR models for Dataset 1 and Dataset 2. A similar approach was used to calculate SSE, RMSE, AICc and the *p*-values. All the models except Model E on Dataset 1 are statistically significant. The Mean $SSE_{10\text{-Fold}}$ is based on 10-fold cross validation as one of the factors for selecting the best performing model.

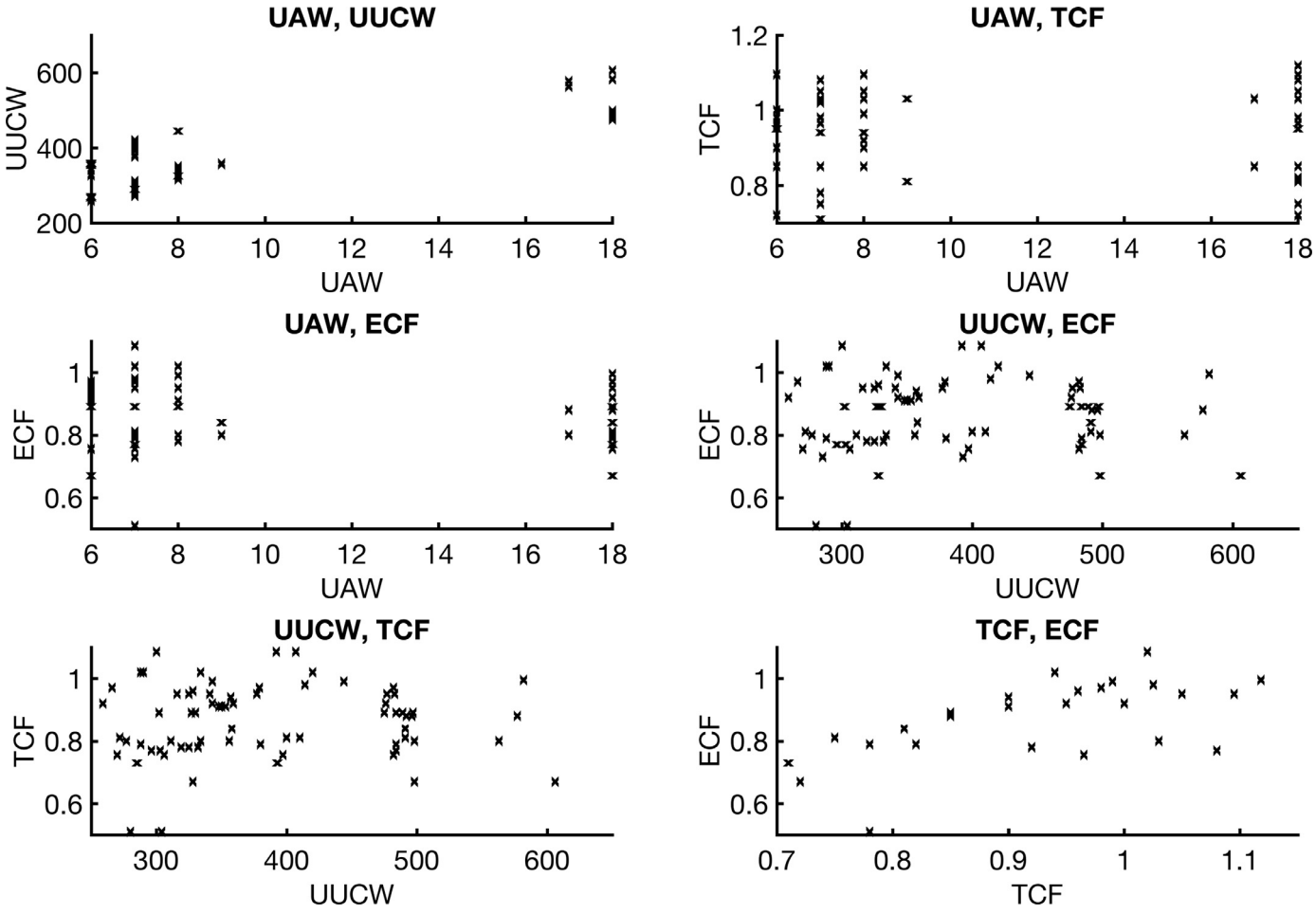

Fig. 5. Scatter plots for the correlation analysis of independent variables using dataset 2.

Table 8
Correlation coefficients for the independent variables.

	Dataset 1				Dataset 2			
	UAW	UUCW	TCF	ECF	UAW	UUCW	TCF	ECF
UAW sig n	–	–	–	–	–	–	–	–
UUCW	0.423	–	–	–	0.869	–	–	–
sig	0.025	–	–	–	<0.001	–	–	–
n	28	–	–	–	71	–	–	–
TCF	0.299	0.0575	–	–	−0.075	−0.076	–	–
sig	0.121	0.771	–	–	0.533	0.527	–	–
n	28	28	–	–	71	71	–	–
ECF	0.021	−0.005	0.424	–	−0.068	0.039	0.494	–
sig	0.021	0.980	0.025	–	0.575	0.749	<0.001	–
n	28	28	28	–	71	71	71	–

5.4. Selecting the best performing model

The regression models are able to explain from 43% to 70% (R^2) of the model uncertainty for Dataset 1 and from 56% to 91% (R^2) of the model uncertainty for Dataset 2. Model D outperforms the tested models with respect to SSE and Mean of $SSE_{10\text{-Fold}}$ as listed in Tables 11 and 12. Model D's SSE and Mean $SSE_{10\text{-Fold}}$ are the lowest in both tested datasets. Moreover, its adjusted R^2 values (0.712 and 0.907) are also better than those of most of the other models. An analysis of the AICc results leads to the same conclusion. Therefore, we can conclude that Model D is the best performing model. The linear models (Models A, B, C and F) are unable to match the performances of the polynomial models (Model D and E) according to all the evaluation measurements. Fig. 8 shows plots of the residuals for Model D vs. the fitted values, where the left side shows the results from Dataset 1 and the right side shows the results from Dataset 2.

This study performed a comparative analysis of regression models with respect to predicting software size. The results show that more complex models tend to predict the value of software size more accurately. Tables 11 and 12 support these results, listing all the performance measures. All the variables used in the final regression equation can be found in Model D. Model D contains an intercept, linear terms, and squared terms.

The effects of predictors on Model D can be seen in Fig. 9. The dots show the magnitude of the effect and the lines show the upper and lower confidence limits for the main effect. The obtained results for Dataset 1 show that an increase in the UAW value from

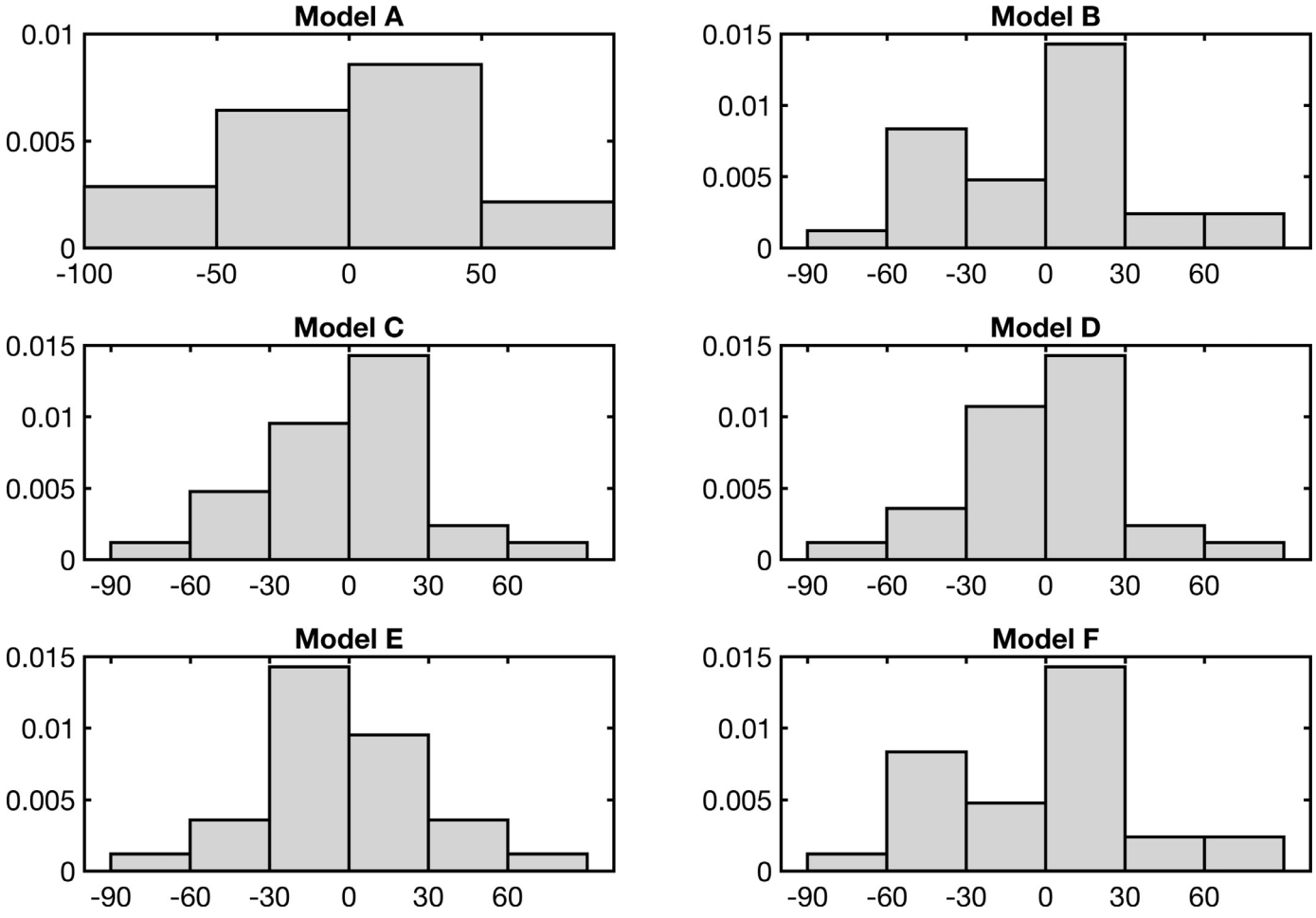

Fig. 6. Histograms of residuals for dataset 1: models A–F.

Table 9
Regression formulas for dataset 1: models A–F.

Model A	$Real_{P20} \sim 1 + UUCW$
Model B	$Real_{P20} \sim 1 + UUCW + ECF + UAW * UUCW$
Model C	$Real_{P20} \sim 1 + UAW * UUCW + UAW * TCF + UAW * ECF + UUCW * TCF + UUCW * ECF + TCF * ECF$
Model D	$Real_{P20} \sim 1 + UUCW + ECF + UAW * TCF + UAW^2 + UUCW^2 + TCF^2 + ECF^2$
Model E	$Real_{P20} \sim 1 + UAW * UUCW + UAW * TCF + UAW * ECF + UUCW * TCF + UUCW * ECF + TCF * ECF + UAW^2 + UUCW^2 + TCF^2 + ECF^2$
Model F	$Real_{P20} \sim 1 + UUCW + TCF + UAW * TCF$

Table 10
Regression formulas for dataset 2: models A–F.

Model A	$Real_{P20} \sim 1 + UAW * UUCW$
Model B	$Real_{P20} \sim 1 + TCF + ECF + UAW * UUCW$
Model C	$Real_{P20} \sim 1 + UAW * UUCW + UAW * TCF + UAW * ECF + UUCW * TCF + UUCW * ECF + TCF * ECF$
Model D	$Real_{P20} \sim 1 + TCF + ECF + UAW * UUCW + UAW^2 + UUCW^2 + TCF^2 + ECF^2$
Model E	$Real_{P20} \sim 1 + UAW * UUCW + UAW * TCF + UAW * ECF + UUCW * TCF + UUCW * ECF + TCF * ECF + UAW^2 + UUCW^2 + TCF^2 + ECF^2$
Model F	$Real_{P20} \sim 1 + UAW * UUCW + UAW * ECF$

Table 11
Selected evaluation measures for dataset 1′ models A–F.

Model	R^2	MSE	SSE	RMSE	AICc	Mean $SSE_{10\text{-Fold}}$	p-value
Model A	0.430	1928.171	50,132.449	43.911	293.667	19,778.993	<0.001
Model B	0.646	1413.128	31,088.807	37.592	291.808	11,184.311	<0.001
Model C	0.700	1549.912	26,348.508	39.369	309.676	9216.102	<0.001
Model D	0.712	1406.300	25,313.402	37.501	302.995	8987.093	<0.001
Model E	0.727	1843.550	23,966.154	42.937	338.522	8356.717	0.055
Model F	0.646	1413.128	31,088.807	37.592	291.808	11,184.311	<0.001

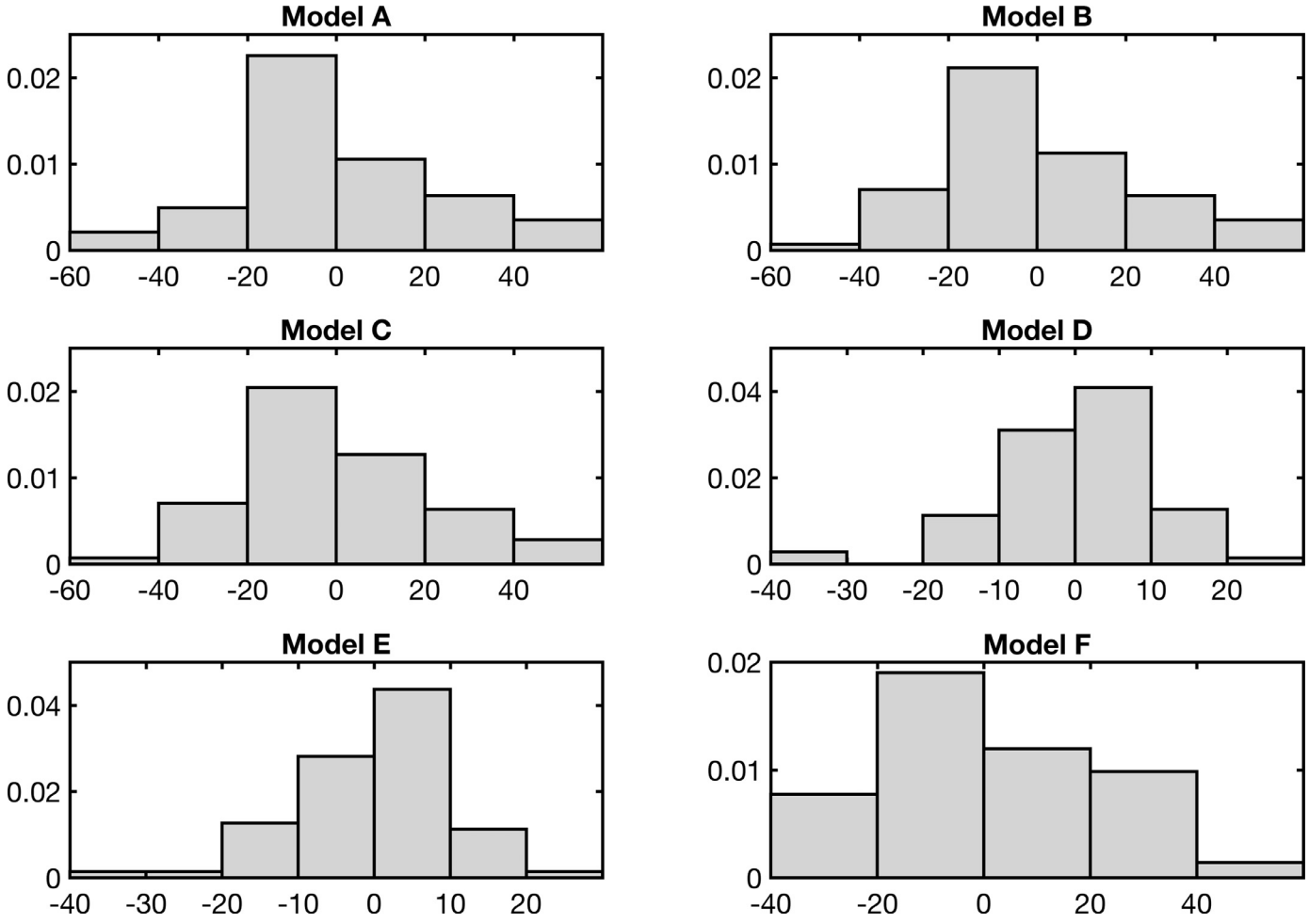

Fig. 7. Histograms of residuals for dataset 2: models A–F.

Table 12
Selected evaluation measures for dataset 2′ models A–F.

Model	R^2	MSE	SSE	RMSE	AICc	Mean $SSE_{10\text{-Fold}}$	*p*-value
Model A	0.566	505.717	33,883.042	22.488	648.023	4830.691	<0.001
Model B	0.572	513.684	33,389.459	22.665	651.687	4790.819	<0.001
Model C	0.586	539.266	32,355.934	23.222	662.617	4686.665	<0.001
Model D	0.907	118.44	7224.813	10.883	553.36	1165.248	<0.001
Model E	0.909	126.235	7069.161	11.235	566.875	1131.333	<0.001
Model F	0.623	482.745	29,447.444	21.971	653.121	4261.34	<0.001

Table 13
Comparison of model D and UCP for datasets 1 and 2.

	Dataset 1 – model D	Dataset 1 – UCP	Dataset 2 – model D	Dataset 2 - UCP
SSE	25, 313	268, 620	7224	87,055
n	28	28	71	71
Median SSE	312	3134	37.26	7312
Wilcoxon Rank Sum Test	$P < 0.01$ at alpha = 0.05.		$P < 0.01$ at alpha = 0.05.	

6.38 to 18 caused a 10-unit reduction in Real_P20. In comparison, a change in UUCW from 60 to 515 caused a 120-unit increase in Real_P20. Both examples have the expectation that all other variables are held constant.

5.5. Best performing model and use case points comparison

In the previous section we selected Model D as the best performing model for both datasets. In this section, we compare Model D to UCP, where UCP represents the Karner's UCP method (see Eq. (11)). As can be seen in Table 13 Model D is demonstrably better then UCP with respect to SSE, and attains a significantly lower SSE median on both datasets. A two-tailed Wilcoxon's rank sum test was used to evaluate the medians.

5.6. Threats to validity

The threats to validity in this study can be summarized as follows. The major risk to the validity of these evaluations lies in the quality of the datasets. The Dataset 1 used in this study was col-

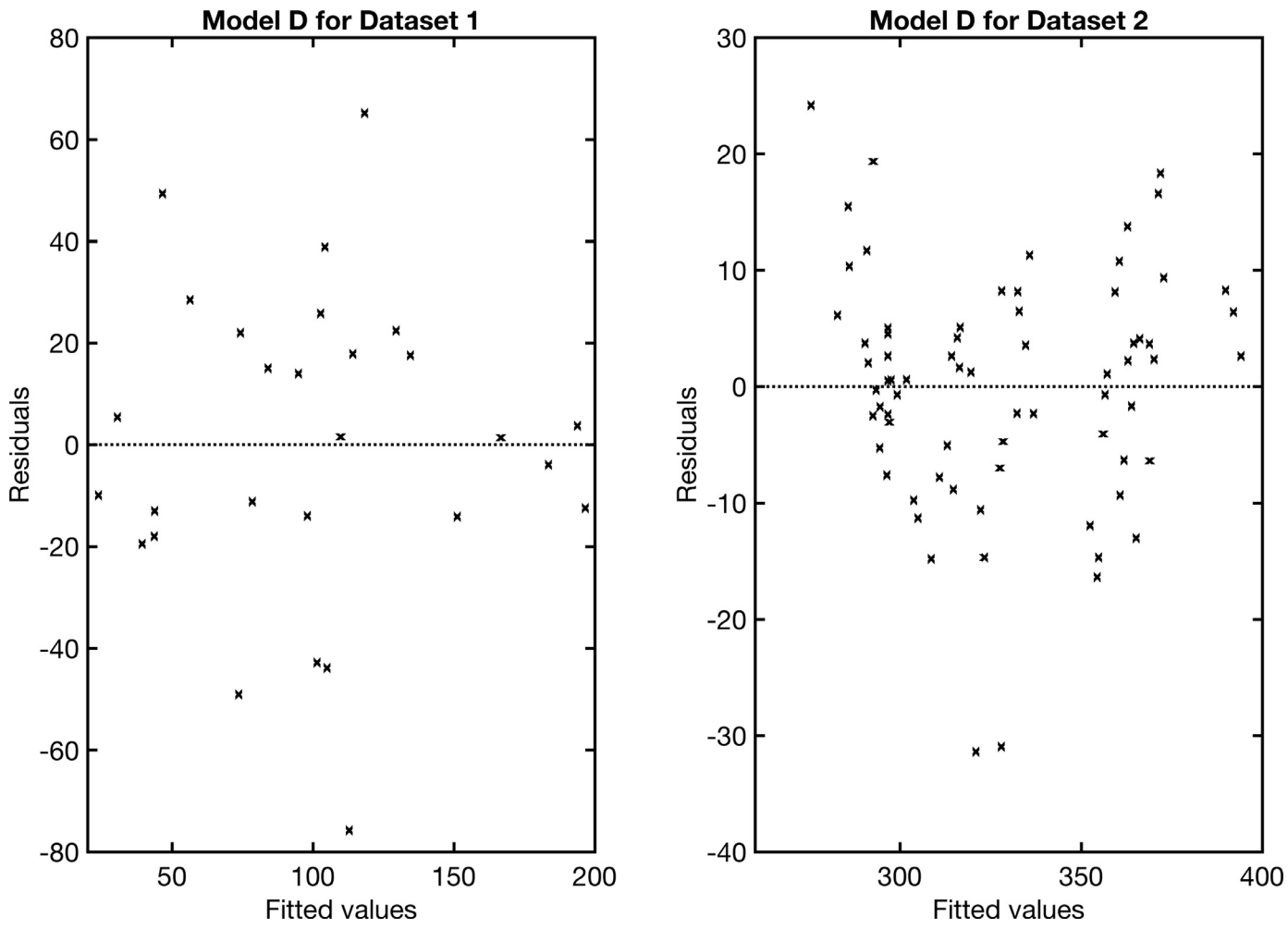

Fig. 8. Residuals vs. fitted values of model D.

lected from publicly available sources. We selected this dataset to achieve an appropriate level of replicability and comparability to previously published results. The dataset contains 28 data points, which represent relatively small projects (the median person-hours is 1952). The second dataset, Dataset 2, was collected by the authors (the median person-hours is 7012). The construct validity includes a question regarding the PFs. The collected data also includes the total effort in person-hours. This value represents real development time, but because the PF factor of each project is unknown (for Dataset 1), it could not be included in our experiment. For Dataset 2, team productivity is based on the difference between the starting date for a project work and the acceptance date (at which time each project was formally completed). Therefore, Real_P20 was used for all calculations and predictions. The Real_P20 value was obtained by dividing the total effort by a constant value for PF (20). During the data collection process for Dataset 2, authors obtained copies of the use case diagrams for only some of the projects—the rest of the participants considered their use case models proprietary. Therefore, we were forced to rely on the information provided by those who were involved in preparing the data for the survey in the software companies. For the set of projects for which use case models were available, there is still a risk that the use case scenarios and actors were improperly designed by the data donator.

All the models were constructed according to the stepwise method for MLR. As listed in Tables 9 and 10, the MLR obtained a slightly different set of models for Dataset 1 vs. Dataset 2; however, typologically, the best performing models are the same. Therefore, we believe our research is generally repeatable and that the same approach can be used for projects of different sizes, although the model formula cannot be replicated as-is with new datasets.

MLR models depend on assumptions, and when those assumptions are violated, data transformation (Christensen, 2006) may be required. However, such transformations also change the method of prediction because the predicted value of the dependent variable does not represent the project size.

6. Conclusion

Based on the experimental results detailed in this study, the research questions that all MLRs are equal can be rejected. All the UCP variables are significant, although some have only slight correlations. Still, all the UCP variables are correlated with the dependent variable and are therefore significant as predictors.

We can conclude that the sizes of a software development projects depend mainly on the value of UUCW, which represents a number of use cases. Moreover, all the other variables (UAW, TCF, ECF) from the UCP method have an impact on software size estimation, and therefore cannot be omitted from regression model. The best performing model (Model D) contains an intercept, linear terms, and squared terms.

RQ1: All UCP parameters are significant for estimation. Correlation was found between UAW and UUCW (Table 8), which was expected, because in the tested datasets, there is at least one use case

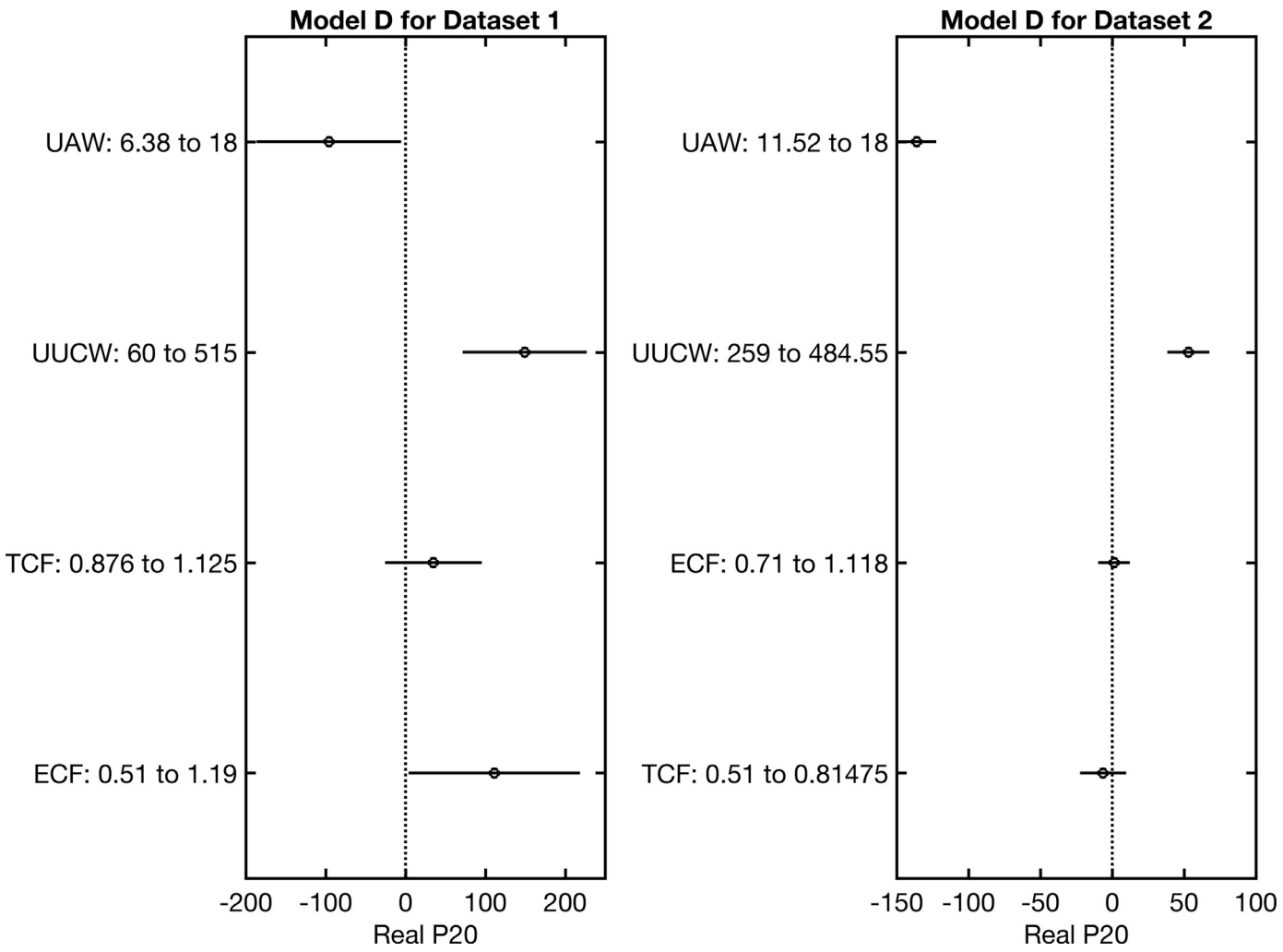

Fig. 9. Predictor effects plots for model D.

for each actor. The most important variable is the UUCW, as can be seen from the effect plots in Fig. 9. The effects plots show that all predictors (UAW, UUCW, TCF, and ECF) are valuable for estimation; however, TCF was not as valuable as the rest of the variables in estimating the sizes of the projects in Dataset 2. We can conclude that the UAW also has an effect on size estimation, which is different than the findings published previously in Ochodek et al., (2011b). Here, the UAW values were valuable because the number of actors helps determine the number of interfaces required in the project, which, in turn, impacts software product construction.

Both TCF and ECF have a low impact but cannot be omitted. They are not only significant for the regression model because they have an effect on size estimation; they also have an impact on project size. However, we may find them to be more valuable in project management.

RQ2: Model D is a complex model. The evaluation measurements show that it has a better estimation ability than the other models. A notable difference can be seen between Model D and Model A, the latter of which is based only on the intercept and the UUCW variable.

Model D obtained the best R^2 values (0.712 and 0.907) on both datasets and the lowest AICc scores (302.995 and 553.36). The SSE scores of Model D (25,313 and 7224) are slightly worse than those of Model E (23,966.154 and 7069.161). If we compare the AICc scores of Model D (302.995 and 553.36) to those of Model E (338.522 and 566.875) we can see that Model D is better. Moreover, Model E was not statistically significant for Dataset 1.

By comparing Model D and the simplest Model A we can conclude that more complex models (more terms) obtain better results. Model A achieved R^2 values of 0.430 and 0.556 while the R^2 values obtained by Model D were 0.712 and 0.907. Similarly, Model A had SSE scores of 50,132.449 and 33,883.042 while Model D achieved SSE scores of 25,313.402 and 7224.813

Finally, Model D performed better compared to UCP. The SSE value for UCP was 268 620 points for Dataset 1 and 87 055 for Dataset 2. Therefore, Model D achieved a greater than 90% decrease in SSE (25,313) compared to UCP on Dataset 1 and a greater than 91% decrease of SSE (7224) on Dataset 2. The median SSE for both are significantly different at 95% confidence level ($p < 0.01$), and the medians for Model D (312 and 37.26) are lower than UCP (3134 and 3712).

In future work, we plan to address two major areas. First, as can be seen from the results obtained here, regression models are sensitive to the data range. Therefore, cluster analysis will be performed to determine the optimal variables for clustering. A proper clustering method will also be investigated. Second, other forms of regression will be studied and, moreover, the principles of recursive regression will be addressed. The recursive method leads to approaches such as partial, exponential, and directed forgetting, which should be valuable when building a historical dataset and lead to an optimal estimation ability.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.jss.2016.11.029.

References

Amasaki, S., Lokan, C., 2015. On the effectiveness of weighted moving windows: experiment on linear regression based software effort estimation. J. Softw. Evol. Proc. 27, 488–507.

Anandhi, V., Chezian, R.M., 2014. Regression techniques in software effort estimation using COCOMO dataset. In: 2014 International Conference on Intelligent Computing Applications. IEEE, pp. 353–357.

Attarzadeh, I., Ow, S.H., 2011. Improving estimation accuracy of the COCOMO II using an adaptive fuzzy logic model. In: IEEE International Conference on Fuzzy Systems. IEEE, pp. 2458–2464.

Azevedo, S., Machado, R.J., Bragança, A., Ribeiro, H., 2011. On the refinement of use case models with variability support. Innov. Syst. Softw. Eng. 8, 51–64.

Azzeh, M., Nassif, A., Banitaan, S., Almasalha, F., 2015a. Pareto efficient multi-objective optimization for local tuning of analogy-based estimation. Neural Comput. Appl 1–25.

Azzeh, M., Nassif, A.B., 2016. A hybrid model for estimating software project effort from use case points. Appl. Soft Comput.

Azzeh, M., Nassif, A.B., Minku, L.L., 2015b. An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation. J. Syst. Softw. 103, 36–52.

Bardsiri, V.K., Jawawi, D.N.A., Hashim, S.Z.M., Khatibi, E., 2014. A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons. Empir. Softw. Eng. 19, 857–884.

Borandag, E., Yucalar, F., Erdogan, S.Z., 2016. A case study for the software size estimation through MK II FPA and FP methods. Int. J. Comput. Appl. Technol. 53, 309–314.

Braz, M.R., Vergilio, S.R., 2006. Software effort estimation based on use cases, computer software and applications conference, 2006. In: COMPSAC'06. 30th Annual International. IEEE, pp. 221–228.

Clark, B.K., 1996. Cost modeling process maturity - COCOMO 2.0. In: 1996 IEEE Aerospace Applications Conference. IEEE, pp. 347–360.

Diev, S., 2006. Use cases modeling and software estimation. ACM SIGSOFT Softw. Eng. Notes 31, 1.

Christensen, R., 2006. Log-Linear Models and Logistic Regression. Springer Science & Business Media.

Idri, A., Amazal, F.A., Abran, A., 2015. Analogy-based software development effort estimation: a systematic mapping and review. Inf. Softw. Technol. 58, 206–230.

Jorgensen, M., 2004. Regression models of software development effort estimation accuracy and bias. Empir. Softw. Eng. 9, 297–314.

Jurkiewicz, J., Nawrocki, J., Ochodek, M., Glowacki, T., 2015. HAZOP-based identification of events in use cases an empirical study. Empir Softw Eng 20, 82–109.

Karner, G., 1993. Metrics for objectory', December 1993, Diploma, University of Linkoping, Sweden, No. LiTH-IDA-Ex-9344 21.

Kazemifard, M., Zaeri, A., Ghasem-Aghaee, N., Nematbakhsh, M.A., Mardukhi, F., 2011. Fuzzy emotional COCOMO II software cost estimation (FECSCE) using multi-agent systems. Appl. Soft Comput. 11, 2260–2270.

López-Martín, C., 2015. Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects. Appl. Soft Comput. 27, 434–449.

Manalif, E., Capretz, L.F., Ho, D., 2014. Fuzzy rules for risk assessment and contingency estimation within COCOMO software project planning model. In: Masegosa, A.D. (Ed.), Advances in Computational Intelligence Robotics. Information Science Reference, Hershey, PA, pp. 88–111.

Mohagheghi, P., Anda, B., Conradi, R., 2005. Effort estimation of use cases for incremental large-scale software development. In: 27th International Conference on Software Engineering. IEEE, pp. 303–311.

Montgomery, D.C., Peck, E.A., Vining, G.G., 2012. Introduction to Linear Regression Analysis, 5th ed. Wiley, Hoboken, NJ.

Nageswaran, S., 2001. Test effort estimation using use case points. Quality Week 1–6.

Nassif, A., Azzeh, M., Capretz, L., Ho, D., 2015. Neural network models for software development effort estimation: a comparative study. Neural Comput. Appl 1–13.

Nassif, A.B., Capretz, L.F., Ho, D., 2011. Estimating software effort based on use case point model using sugeno fuzzy inference system. In: 23rd IEEE International Conference on Tools with Artificial Intelligence. IEEE, pp. 393–398.

Nassif, A.B., Capretz, L.F., Ho, D., 2012. Estimating software effort using an ANN model based on use case points. In: 11th International Conference on Machine Learning and Applications. IEEE, pp. 42–47.

Nassif, A.B., Ho, D., Capretz, L.F., 2013. Towards an early software estimation using log-linear regression and a multilayer perceptron model. J. Syst. Softw. 86, 144–160.

Ochodek, M., Alchimowicz, B., Jurkiewicz, J., Nawrocki, J., 2011a. Improving the reliability of transaction identification in use cases. Inf. Softw. Technol. 53, 885–897.

Ochodek, M., Nawrocki, J., Kwarciak, K., 2011b. Simplifying effort estimation based on use case points. Inf. Softw. Technol. 53, 200–213.

Oplatkova, Z.K., Senkerik, R., Zelinka, I., Pluhacek, M., 2013. Analytic programming in the task of evolutionary synthesis of a controller for high order oscillations stabilization of discrete chaotic systems. Comput. Math. Appl. 66, 177–189.

Robiolo, G., Badano, C., Orosco, R., 2009. Transactions and paths: two use case based metrics which improve the early effort estimation. In: 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE, pp. 422–425.

Robiolo, G., Orosco, R., 2008. Employing use cases to early estimate effort with simpler metrics. Innov. Syst. Softw. Eng. 4, 31–43.

Rosa, W., Jones, C., McGarry, J., Madachy, R., Dean, J., Boehm, B., Clark, B., 2014. Improved Method for Predicting Software Effort and Schedule. International Cost Estimating and Analysis Association (ICEAA).

Senkerik, R., Oplatkova, Z.K., Pluhacek, M., Zelinka, I., 2014. Analytic programming-a new tool for synthesis of controller for discrete chaotic lozi map. Comput. Probl. Eng. 307, 137–151.

Shepperd, M., MacDonell, S., 2012. Evaluating prediction systems in software project estimation. Inf. Softw. Technol. 54, 820–827.

Silhavy, R., Silhavy, P., Prokopova, Z., 2015a. Algorithmic optimisation method for improving use case points estimation. PLoS ONE 10, e0141887.

Silhavy, R., Silhavy, P., Prokopova, Z., 2015b. Applied Least Square Regression in Use Case Estimation Precision Tuning, Software Engineering in Intelligent Systems. Springer International Publishing, pp. 11–17.

Subriadi, A., Ningrum, P., 2014. Critical review of the effort rate value in use case point method for estimating software development effort. J. Theror. Appl. Inf. Technol. 59, 735–744.

Tadayon, N., 2004. Adaptive dynamic COCOMO II in cost estimation. In: Serp'04: Proceedings of the International Conference on Software Engineering Research and Practice, 1 and 2, pp. 559–563.

Urbanek, T., Prokopova, Z., Silhavy, R., 2015a. On the value of parameters of use case points method. In: Silhavy, R., Senkerik, R., Oplatkova, Z.K., Prokopova, Z., Silhavy, P. (Eds.), Artificial Intelligence Perspectives and Applications. Springer International Publishing, Cham, pp. 309–319.

Urbanek, T., Prokopova, Z., Silhavy, R., Vesela, V., 2015b. Prediction accuracy measurements as a fitness function for software effort estimation. Springerplus 4, 17.

Wang, F., Yang, X., Zhu, X., Chen, L., 2009. Extended use case points method for software cost estimation. In: International Conference on Computational Intelligence and Software Engineering. IEEE, pp. 1–5.

Yang, D., Wan, Y.X., Tang, Z.A., Wu, S.J., He, M., Li, M.S., 2006. COCOMO-U: an extension of COCOMO II for cost estimation with uncertainty. Softw. Process Change 3966, 132–141.

Radek Silhavy was born in Vsetin in 1980. He received a B.Sc. (2004), M.Sc. (2006), and Ph.D. (2009) in Engineering Informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlin. He is a Senior Lecturer and Researcher in the Computer and Communication Systems Department. His Ph.D. research was on The Verification of the Distributed Schema for the Electronic Voting System. His major research interests are empirical software engineering, software size estimation, effort estimaion and system engineering.

Petr Silhavy was born in Vsetin in 1980. He received a B.Sc. (2004), M.Sc. (2006), and Ph.D. (2009) in Engineering Informatics from the Faculty of Applied Informatics, Tomas Bata University in Zlin. He is a Senior Lecturer and Researcher in the Computer and Communication Systems Department. His Ph.D. research was on Electronic Communication and Services in Medical Information Systems. His major research interests are data mining, database systems and web-based services.

Zdenka Prokopova was born in Rimavska Sobota, Slovak Republic in 1965. She graduated from the Slovak Technical University in 1988, with a Master's degree in Automatic Control Theory. She received her Technical Cybernetics Doctoral degree in 1993 from the same university. She worked as an Assistant at the Slovak Technical University from 1988 to 1993. During 1993–1995, she worked as a programmer of database systems in the Datalock business firm. From 1995 to 2000, she worked as a Lecturer at Brno University of Technology. Since 2001, she has been at Tomas Bata University in Zlin, in the Faculty of Applied Informatics. She presently holds the position of Associate Professor at the Department of Computer and Communication Systems. Her research activities include programming and applications of database systems, mathematical modelling, computer simulation and the control of technological systems.