Effective Parametric Model for System Engineering Project Estimationysis

Ho Le Thi Kim Nhung, Ph.D.

Doctoral Thesis Summary

Tomas Bata University in Zlín

Tomas Bata University in Zlín
Faculty of Applied Informatics

Doctoral Thesis Summary

Effective Parametric Model for System Engineering Project Estimation

Efektivní parametrický model pro odhad projektu systémového inženýrství

Author:	**Ho Le Thi Kim Nhung, Ph.D.**
Degree programme:	Engineering Informatics / P3902
Degree course:	Engineering Informatics / 3902V023
Supervisor:	Assoc. Prof. Ing. Zdenka Prokopová, CSc.
Consulting supervisor:	Assoc. Prof. Ing. Radek Šilhavý, Ph. D.
External examiners:	Prof. RNDr. Ing. Miloš Šeda, Ph.D. Assoc. Prof. Ing. Petr Čermák, Ph.D. Assoc. Prof. Ing. Zuzana Komínková Oplatková, Ph.D.

Zlín, November 2022

Published by **Tomas Bata University in Zlín** in the Edition **Doctoral Thesis Summary.**
The publication was issued in the year 2022.

Key words in Czech: *Odhad úsilí vývoje softwaru, Body případů užití, Optimalizace korekčních faktorů*

Key words: *Software development effort estimation, Use Case Points, Optimising correction factors*

Full text of the doctoral thesis is available in the Library of TBU in Zlín.

ISBN 978-80-7678-130-6

ABSTRAKT

V předkládané disertační práci jsou představeny návrhy nových způsobů odhadů složitosti projektů založených na metodě Use Case Points, která se používá v raných fázích vývoje softwaru. Navržené metody jsou vyvinuty tak, aby zvládaly nepřesnosti při odhadování a zahrnovaly expertní posudky pro vytvoření přesných a spolehlivých odhadů úsilí. Každý přístup má své výhody a vzájemně se doplňují. Cílem je, aby jednotlivé metody vytvořily kompletní proces a podporovaly efektivitu odhadu úsilí, tj. aby se ve všech situacích účinněji minimalizovala chyba v odhadu. Výsledky ukazují, že navržené metody Software Development Effort Estimation (SDEE) jsou konkurenceschopné ve srovnání s jinými alternativami, na základě sedmi hodnotících kritérií a statistických párových srovnání t-testů.

ABSTRACT

In the presented doctoral thesis, proposals for new methods of estimating the complexity of projects based on the Use Case Points method, which is used in the early stages of software development, are presented. The proposed methods are developed to handle estimation inaccuracies and incorporate expert judgments to produce accurate and reliable effort estimates. Each approach has its advantages, and they complement each other. The goal is for them to create a complete process and support the efficiency of effort estimation, i.e., to minimize estimation error more effectively in all situations. The results show that the proposed Software Development Effort Estimation (SDEE) methods are competitive compared to other alternatives, based on seven evaluation criteria and statistical pairwise t-test comparisons.

CONTENTS OF THE THESIS

1 CURRENT STATE OF THE ART

Software Project Development has evolved into a dynamic and competitive industry requiring high-level human resources. Software products are becoming more complicated, unpredictable, and challenging to control. Many research projects in the software field have been conducted in recent decades with the goal of steering software development processes into more regulated, manageable, and predictable paths. Project managers must estimate the cost of the software product as well as the resources, effort, and time required to complete a project on time and within budget [1]. Software measurement problems, such as project duration prediction or defect density, receive special attention. These issues demonstrate that the project management role has significantly increased.

Software Development Effort Estimation (SDEE) is critical to the overall success of solution delivery. Early SDEE in the first phase of the software development lifecycle is essential to avoiding project failures. The project manager's role is to look at software products to help with budgeting, scheduling, planning, project bidding, human resource allocation, and risk mitigation. The SDEE is vital for some reasons [2]. First, it is beneficial to make informed decisions about resource management before the project begins. The project plan is then used to make informed decisions about managing and planning the project. It is critical to allocate appropriate effort to the various activities in managing project development. As a result, this has led many researchers to study software estimation to obtain a more accurate SDEE [3], [4], [5]. However, based on the requirement specifications, the SDEE cannot be expected to produce correct results [6]. The issue of accurate effort estimation remains unresolved. An effort estimation method is used to reduce the risk of surprises during the project to the lowest possible value. It provides project managers with good control decisions to ensure that reasonable effort is allocated to the various activities throughout the project's development life cycle. When inaccurate models are used, such estimation decisions can have disastrous consequences. The most visible example of problems in managing complex, distributed software systems is the failure of many software projects [7]. The results show that actual effort and schedule are exceeded for most projects compared to estimates. If the software cost is underestimated, the project will be inefficient, and the actual price will undoubtedly be surpassed. Finally, even if completed on time, these overestimated projects usually become more extensive and costly than planned. In contrast, the functionality and quality of these underestimated projects are reduced to meet the plan's requirements. This can result in losing the bid or wasting time, money, personnel, and other resources, resulting in financial loss or even bankruptcy.

Use Cases can be helpful to measure the estimated effort at an early stage of a software project before the essential information is obtained during the requirements phase of the software lifecycle [8]. Neil et al. [9] surveyed the

techniques used in the requirements elicitation, description, and modeling phases and found that the use cases were used in the early stages by more than half of the software projects. This has sparked the interest of numerous researchers in using use cases-based SDEE approaches and their initial applicability for greater accuracy. Karner [10] introduced the Use Case Points (UCP) method as a metric for sizing object-oriented software projects based on a structured scenario and actor analysis of the Use Case Model (UCM). Most studies focus on evaluating UCP as a potential early SDEE method that could be used to estimate software development effort and show its suitability for the software industry [11], [12], [13], [14]. The UCP is a promising method for effort estimation in the early stages of software development that offers numerous benefits to the software industry [15], [16], [17]. Using machine learning to build SDEE models based on the original UCP formula could be a solution to improve its accuracy. Some approaches [18], [19], [20], [21], [22], [23] have also addressed variant models, especially regression models, to improve estimation accuracy based on historical data. The main drawback of the methods described above is that none of them is comprehensive or provides better accuracy in estimating software effort in all situations. There are still known problems in using UCP methods.

- The first problem is a particular uncertainty in evaluating technical complexity factors (TCF) and environmental complexity factors (ECF), as it depends on the experience of experts [24], [25], [26], [27], [28]. In particular, assigning an appropriate value to an ECF is difficult due to the lack of relevant information. This is because an ECF is associated with the level of information and experience of a particular software development team. Similar problems exist in assigning a value to a TCF. These correction factors affect the estimation accuracy of UCP, so they need to be refined [29], [30]. Therefore, we will examine the close relationship between technical and environmental factors and prediction error to identify the best factors that significantly affect the estimation accuracy of the UCP method. This issue will be discussed in Chapter 3.1, as we have proposed a new formula for calculating the correction factors in the UCP method.
- The second problem is that potentially unsuitable variables are not considered in the UCP equation. In particular, use cases are written in natural language, and there is no rigorous process for assessing the quality or fragmentation of use cases. As a result, the number of steps in a use case may vary, affecting the estimate's accuracy. In addition, the estimate's accuracy may suffer if a use case contains multiple scenarios. Almost all previous methods for estimating software effort based on UCP have focused on developing the method by evaluating the complexity of the use case model and complexity weights [31], [32], [33], [34], [35], [36], [37]. However, we believe the regression approach based on UCP elements can solve this problem. Specifically, we will explore the implementation of multiple linear regression (MLR) models to select new formulas and regression coefficient values to reduce the impact of human error

in evaluating actors or use cases. As shown in Chapter 3.2, this new formula outperforms the estimation accuracy of UCP.

- Moreover, given the complexity of today's software development projects, effort estimation requires the support of statistics and machine learning (ML). According to Kumar et al. [38], the overall estimation accuracies of SDEE methods based on statistical and machine learning techniques are almost acceptable as they are within 25% of the percent error (PRED (0.25)). The techniques are used to model the relationship between effort and software variables, which is particularly useful when the relationship is non-linear. However, one question is how to select unbiased approaches and appropriate algorithms. We note that single statistical and machine learning methods are unreliable, and the accuracy of a single method depends on its parameter configurations [39]. According to Thiago et al. [40], using a single model does not lead to optimal SDEE results. Priya et al. [41] also found that combining multiple models improves reliability. For all datasets, almost all ensemble SDEE approaches use the same learning parameter settings. With the above analysis, the thesis aims to reduce the bias and variability errors of the single models. In Chapter 3.3, we present the ensemble approach, which integrates seven well-known statistical and machine learning methods and fine-tunes the parameters of all the single methods to create a new and more comprehensive method in the early stages of software development.
- The fourth focus is the difficulty of converting software size into the corresponding effort. Many researchers consider the software productivity factor, or the amount of software produced per effort, critical to estimating effort [42], [43], [44], [45]. This term also refers to the ratio of effort to size, also known as the productivity factor (PF). Most accepted values for the productivity factor have been suggested by project managers or use predetermined values for software productivity [46], [47]. However, we believe each software project takes place in a unique environment. Therefore, the question of whether to impose a fixed PF on all software projects has not been adequately addressed. This issue was discussed in Chapter 3.4 when we developed the software productivity model using the ensemble approach with historical correction factors. According to the findings, learning productivity values for each project is more useful and efficient than using predetermined values for all projects.

2 AIMS OF THE THESIS

The thesis focuses on developing SDEE methods for estimating software size and effort from UCM. Our methods can be used during the requirements phase of the software lifecycle. We aim to develop methods to handle imprecision and incorporate expert opinions to produce accurate and reliable effort estimates. With this objective, the thesis analyzes and proposes SDEE methods to reduce the impact of human error in UCM analysis and simplify the original principles of UCP. Each approach has its advantages, and they complement each other to form a complete process and promote significant efficiency to minimize the estimation error more efficiently in all situations.

With each problem statement presented in Chapter 1, we have made the following objectives:

- Determining software complexity factors that significantly affect estimation accuracy and proposing a new formula for the calculating of the correction factors.
- Designing a comprehensive approach for determining software size in the early stages of software development.
- Validation of the proposed methods and procedures using various evaluation criteria and their comparison with the UCP reference method and other approaches.

3 THE PROPOSED METHODS

First, a proposal for increasing the estimation accuracy of the existing UCP method was called Optimization Correction Factors (OCF) [48]. We analyze correction factors to identify the best technical and environmental complexity factors that significantly affect the estimation accuracy of the UCP method in regression analysis. To put this idea into practice, we propose a new formula to calculate the correction factors in the UCP method. Then, to obtain more accurate estimates, we aim to apply the MLR models to improve the ability of the OCF method to estimate the software size and minimize the prediction error. This is referred to as the Extension of Optimizing Correction Factors (ExOCF) [49]. The OCF variables are used in this method to determine the software size. The MLR formulation was created to estimate the software size values. Following the proposed ExOCF is another alternative framework for effort prediction to improve the overall performance of the regression. A novel Stacked SVR-MLR-MLP-DT-RF-KNN-GB on the OCF (SOCF) model is proposed to improve the overall performance of the regression. The model includes seven statistical and ML techniques: MLR, KNN, SVR, MLP, RF, GB, and DT. Finally, the calculations of the effective productivity factor (PF_{CFE}) are proposed in conjunction with the OCF as predictors of effort [50]. The summary of the four proposed methods is shown in Figure 3-1.

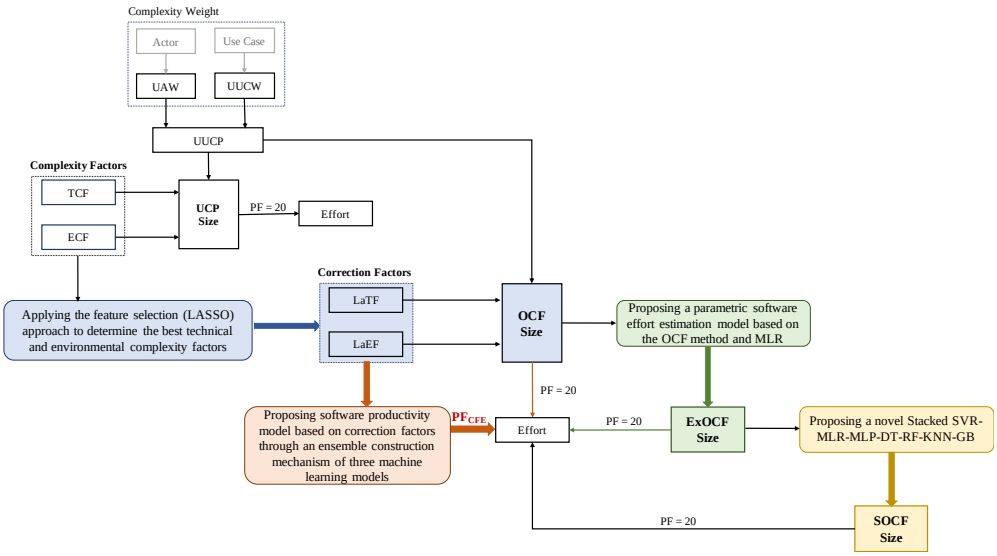

Figure 3-1. The proposed methods

3.1 The proposed Optimization Correction Factors method

A detailed illustration of the OCF method is shown in Figure 3-2. The LASSO-based Selection Phase (Phase I), applies the LASSO regression with the determined regularisation parameter λ to extract a selected variable set; as shown in Eq. (3.1).

$$\hat{\beta}(\lambda) = \underset{\beta}{\operatorname{argmin}} \left(\frac{\| Y - X\beta \|_2^2}{n} + \lambda \| \beta \|_1 \right) \quad (3.1)$$

$$\text{subject to} \sum_{j=1}^{k} |\beta_j| < t$$

where

$$\| Y - X\beta \|_2^2 = \sum_{i=0}^{n} (Y_i - (X\beta)_i)^2 \quad (3.2)$$

$$\| \beta \|_1 = \sum_{j=1}^{k} |\beta_j| \quad (3.3)$$

where $\lambda \geq 0$ is the lasso parameter that controls the strength of the penalty determined by the Leave One Out Cross-Validation (LOOCV) method. The choice of the lasso parameter is adjusted based on the lowest possible estimation error and a lack of bias against the correction factors of the observations in the training set. The lasso parameter relates directly to the number of correction factors selected via non-zero β.

LASSO regression is used to obtain the TCF and ECF correction coefficients - as described in Eq. (3.4) and Eq. (3.5), respectively.

$$y\_TCF_i = \alpha_0 + \sum_{i=1}^{13} \alpha_i \times t_i \times fw_i \quad (3.4)$$

$$y\_ECF_i = \beta_0 + \sum_{i=1}^{8} \beta_i \times e_i \times ew_i \quad (3.5)$$

where, y_TCF_i be $\left(\frac{Real\_P20}{(UAW+UUCW)\times ECF} - 0.6\right) \times \frac{1}{0.01}$, y_ECF_i be $\left(\frac{Real\_P20}{(UAW+UUCW)\times TCF} - 1.4\right) \times \frac{-1}{0.03}$, and $\alpha_0, \alpha_i\ \beta_0, \beta_i$ are the regression coefficient parameters obtained from the LASSO regression; Real_P20 is the real size of software projects from historical datasets. The LASSO-based selected variables in TCF and ECF are designated as LaTF and LaEF respectively.

Then, Least Squares Regression (LSR) is used to obtain the coefficients for LaTF and LaEF in Eq. (3.6) and Eq. (3.7), respectively. LaTF and LaEF values represent the final technical and environmental complexity factors - (correction factors), in the OCF method.

$$y\_LaTF_i = \alpha_0 + \sum_{i=1}^{n} \alpha_i \times LaT_i \times WLt_i \quad (3.6)$$

$$y\_LaEF_i = \beta_0 + \sum_{i=1}^{m} \beta_i \times LaE_i \times WLe_i \quad (3.7)$$

where, let y_LaTF_i and y_LaEF_i be the TCF and ECF from the historical dataset; n is the number of LaTF; m is the number of LaEF; and $\alpha_0, \alpha_i\ \beta_0, \beta_i$ are regression coefficient parameters obtained from LSR.

LaTF and LaEF are obtained according to Eq. (3.8) and Eq. (3.9).

$$\mathrm{LaTF} = \alpha_0 + \sum_{i=1}^{n} \alpha_i \times \mathrm{LaT_i} \times \mathrm{WLt_i} \tag{3.8}$$

$$\mathrm{LaEF} = \beta_0 + \sum_{i=1}^{m} \beta_i \times \mathrm{LaE_i} \times \mathrm{WLe_i} \tag{3.9}$$

The model fitting phase - (Phase II) is determined. The effort estimation final result of the proposed OCF method is described by Eq. (3.10).

$$\mathrm{UCP_{OCF}} = (\mathrm{UAW} + \mathrm{UUCW}) \times \mathrm{LaTF} \times \mathrm{LaEF} \tag{3.10}$$

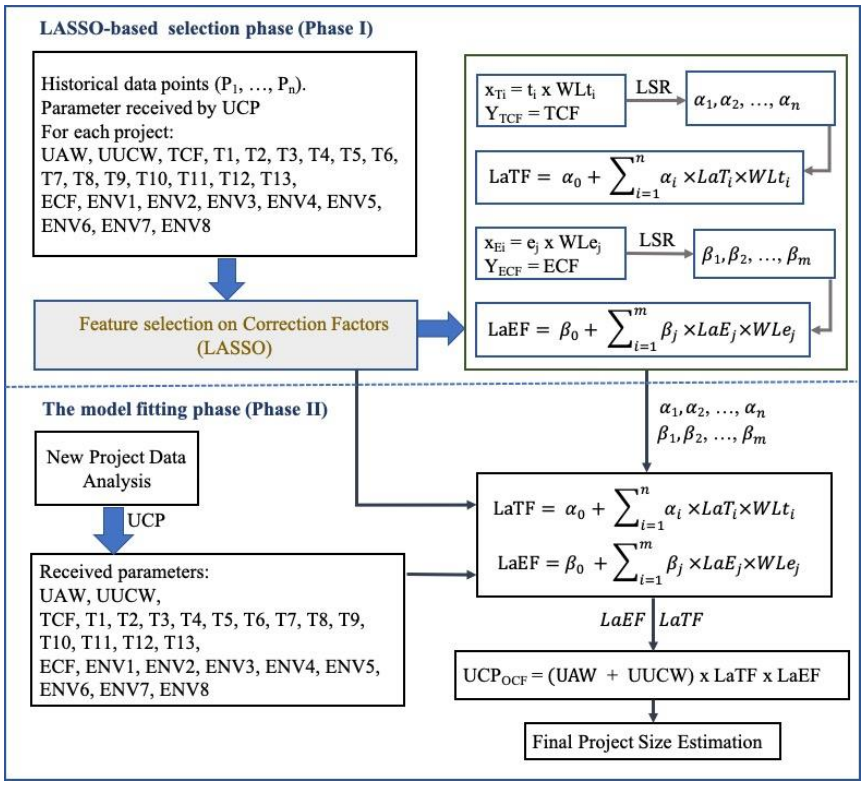

Figure 3-2. The proposed Optimizing Correction Factors method

3.2 The proposed approach based on Optimization Correction Factors and Multiple Linear Regression

The OCF approach can help project managers reduce the risks associated with evaluating correction factors. The results show that the method improves the average SSE by more than 53.6% compared to the UCP method. The detailed results are presented in Chapter 5.1. We further develop the OCF method and propose an extension of OCF (ExOCF) that applies MLR models to the OCF elements to reduce the estimation error and the influence of the unsystematic noise of the OCF technique. A detailed illustration of the ExOCF method is shown in Figure 3-3.

The proposed model is built using MLR as follows:

$$\begin{aligned} UCP_{ExOCF} = \gamma_1(UAW \times LaTF \times LaEF) \\ + \gamma_2(UUCW \times LaTF \times LaEF) \end{aligned} \quad (3.11)$$

where γ_1, γ_2 are obtained according to two steps. First, the historical data points $(P_1, \dots, P_2)$ are collected. The UAW, UUAW, LaTF, and LaEF elements for each project are identified. The result of this step is the collection of values $(x_{i1}, x_{i2}, y_i), i = \overline{1 \dots n}$, where y_i is the actual size (Real_P20 values) of the software project from a historical dataset.

$$x_{i1} = (UAW_i \times LaTF_i \times ECF_i) \quad (3.12)$$

$$x_{i2} = (UUCW_i \times LaEF_i \times ECF_i) \quad (3.13)$$

The LSR model is then used for obtaining the regression coefficients γ_1, γ_2 as followings.

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} \times \begin{pmatrix} X_{11} & X_{12} \\ \vdots & \vdots \\ X_{n1} & X_{n2} \end{pmatrix} \quad (3.14)$$

$$\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = (X^T X)^{-1} X^T y \quad (3.15)$$

Because y_i is a real value from a historical dataset, the regression coefficient values of γ_1, γ_2 can vary from each dataset. This means that when a historical dataset changes, this phase needs to be performed again to obtain new regression coefficient values. The second step of this phase will calculate the UAW, UUCW, LaTF, and LaEF of the current project, and Eq. (3.11) is applied with values γ_1, γ_2 to estimate the UCP_{ExOCF}.

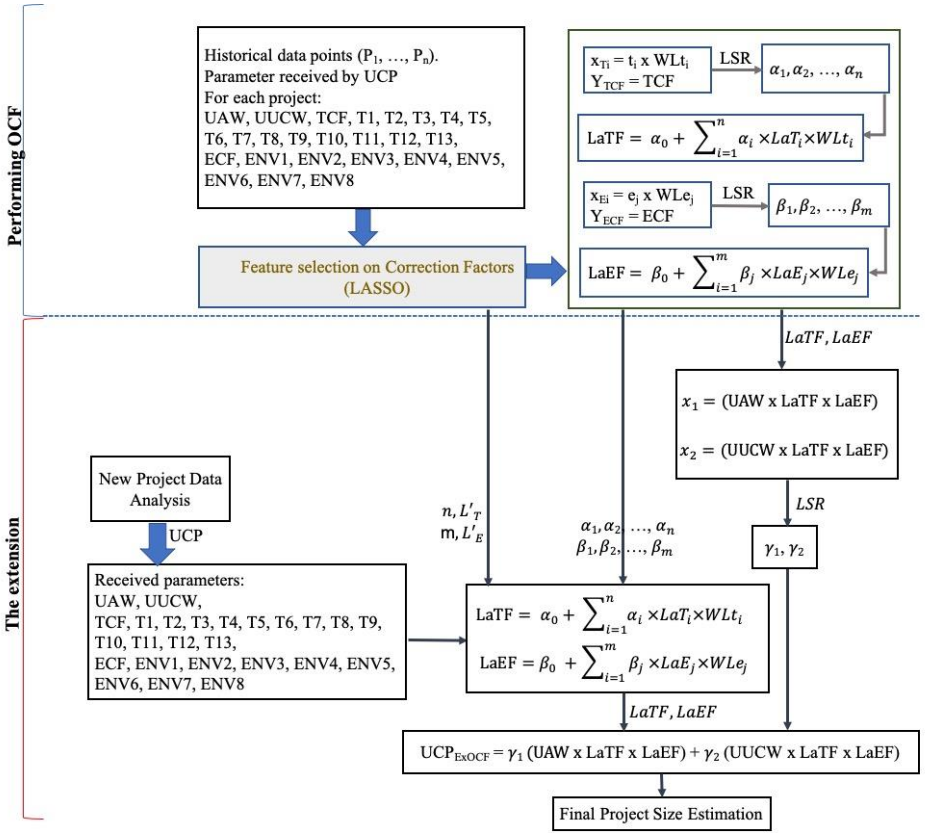

Figure 3-3. Detailed illustration of the proposed ExOCF method

3.3 The proposed Stacking ensemble model based on Optimizing Correction Factors

Based on the literature review, we believe the ensemble approach can provide an unbiased estimate of the effort required for a new software project. The ensemble approach combines at least two different single models through a unique aggregation mechanism and generates the final solution through weighted voting on their solutions [51]. As a result, this section aims to investigate the effect of the ensemble approach in predicting the software size early in the project development using the OCF method. The SOCF model is proposed that incorporating seven statistical and ML techniques MLR, KNN, SVR, MLP, RF, GB, and DT.

The detailed SOCF architecture is shown in Figure 3-4, which consists of cleaning the data, dividing it into training and test sets, and applying the stacking model to estimate the OCF-based size.

The following methodology was used:

1. LASSO regression is used to determine the best correction factors.
2. The input and output vectors are determined.
3. The data is split into a training set $S^{(-j)}$ and a test set S_j. $S^{(-j)}$ is used to create the level 0 models (regressors) via seven ML techniques, SVM, KNN, DT, MLP, MLR, GB, and RF.

4. The configuration parameters for the seven regression models (level 0 models) SVM, KNN, DT, MLP, MLR, GB, and RF are tuned on the validation set (30% of the training set) to produce their optimal settings.
5. Create an ensemble model with the stacking approach. The estimator's predictions are stacked and fed into a final estimator, which computes the final estimation. More precisely, each of the level 0 models in the first stage undergo five-fold cross-validation in $S^{(-j)}$ to output its prediction and generate a prediction for S_j by taking the average of the seven estimation results generated by the five CV models in the training phase. Then these level 0 models create a vector of predictions to input into the level 1 model (in the second stage). RF was selected as the meta-regressor to train a new model for the final project size estimation.

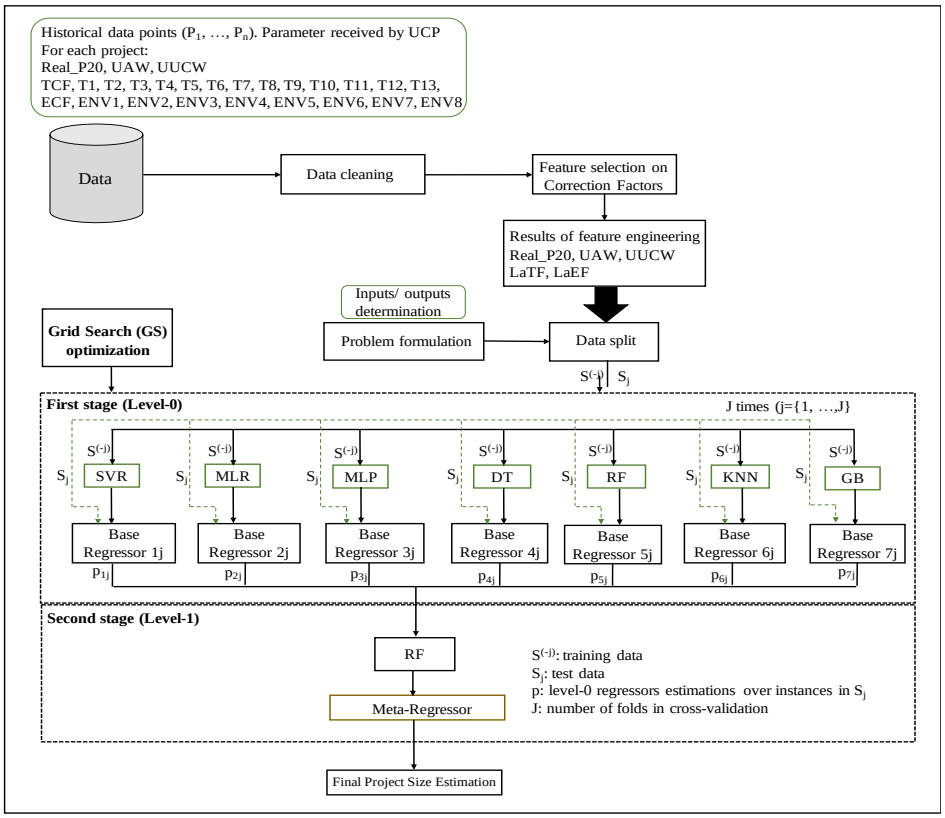

Figure 3-4. The architecture of the proposed SOCF model

3.4 The proposed software productivity model based on ensemble approach

Our primary goal is to research and confirm the role of software productivity in generating early effort estimates from UCP. To address the fourth problem in Chapter 1, we proposed effective productivity factor calculations in conjunction with UCP as predictors for effort. The approach employs an ensemble construction mechanism from ML techniques (OCF(PF_{CFE})) such as Support

Vector Regression (SVR), Multiple Linear Regression (MLR), and Decision Tree (DT). The voting ensemble is used as an ensemble model ML.

Figure 3-5 shows the proposed software productivity mode. The methodology was used: (1) Correction factors from OCF are used as input. (2) Built the voting regressor algorithm [52] consisting of three base estimators, such as Support Vector Regression (sklearn.svm.SVR), Multiple Linear Regression (sklearn.linear_model.LinearRegression), and Decision Tree Regression (sklearn.tree.DecisionTreeRegressor).

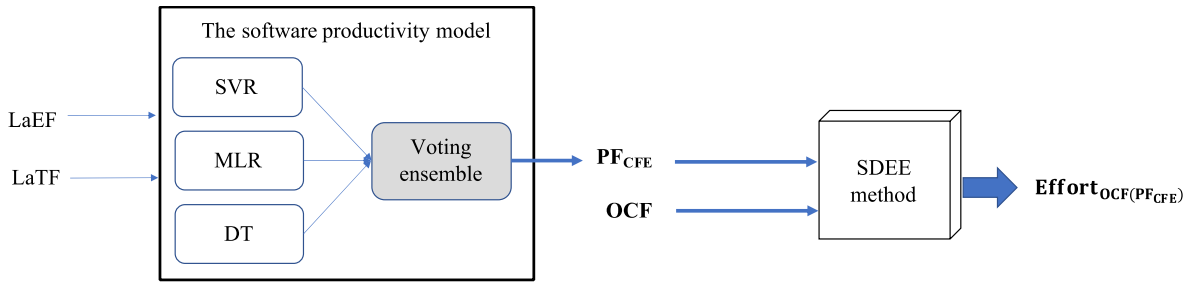

Figure 3-5. The proposed software productivity model

Estimated effort is obtained by multiplying OCF by PF_{CFE}, as follows Eq. (3.21).

$$\text{Effort}_{\text{OCF}(\text{PF}_{\text{CFE}})} = \text{OCF} \times \text{PF}_{\text{CFE}} \quad (3.21)$$

4 RESEARCH METHODOLOGY

4.1 Experiment 1 (EX1)

EX1 is performed to evaluate the proposed OCF method with other related methods, such as the baseline UCP [10] and OTF - a variant of the UCP model that omits the technical factors [53]. These methods are summarized in Table 4-1.

Table 4-1. Methods implemented for EX1

No.	SDEE methods	Summary	Notation
1	Use Case Points	- Size is measured by UCP variables (UAW, UUCW, TCF, and ECF).	UCP
2	UCP (omitting technical factors)	- Size is measured from UCP variables (UAW, UUAW, and ECF) except for the technical factors.	OTF

3	Optimization Correction Factors (proposed in Chapter 3.1)	- Correction factors are determined in regression analysis by the LASSO regression model. - Size is measured in UCP variables (UAW and UUCW) and correction factors (LaTF and LaEF).	OCF

The statistical hypothesis was tested to determine whether the proposed OCF approach provides a better estimate.

- H_0: $\mu_{\text{the proposed OCF method}} = \mu_{\text{the other tested methods}}$. The estimation ability of the proposed OCF method is not significantly different from the estimation abilities of the other tested methods.
- H_1: $\mu_{\text{the proposed OCF method}} > \mu_{\text{the other tested methods}}$. The estimation ability of the proposed OCF method is significantly different from the estimation abilities of the other tested methods.

4.2 Experiment 2 (EX2)

EX2 is performed to evaluate the proposed ExOCF method with the related software size estimation models from the literature. The selected models are the baseline UCP model [10], the OCF model, and the AOM model [23]. We also developed two models that establish a linear relationship between software and UCP factors (UAW, UUC, TCF, and EF). These models are SVR, and DT. All methods are summarized in Table 4-2.

Table 4-2. Methods implemented for EX2

No.	SDEE method	Summary	Notation
1	Use Case Points	- Size is measured by UCP variables (UAW, UUCW, TCF, and ECF).	UCP
2	Optimization Correction Factors (proposed in Chapter 2.1)	- Correction factors are determined in regression analysis by the LASSO regression model. - Size is measured in UCP size variables (UAW and UUCW) and correction factors (LaTF and LaEF).	OCF

3	Algorithmic Optimization Method	- Size is measured based on linear regression on UCP variables (UAW, UUC, TCF, and EF).	AOM
4	Use Case Points using SVR	- SVR is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&SVR
5	Use Case Points using DT	- DT is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&DT
6	Extension of Optimization Correction Factors (proposed in Chapter 3.2)	- Correction factors are determined in regression analysis by the LASSO regression model. - Size is based on linear regression on OCF variables (UAW, UUCW, LaTF, and LaEF).	ExOCF

The statistical hypothesis was tested to determine whether the proposed ExOCF approach provides a better estimate.

- H_0: $\mu_{\text{the proposed ExOCF method}} = \mu_{\text{the other tested methods}}$. The estimation ability of the proposed ExOCF method is not significantly different from the estimation abilities of the other tested methods.
- H_1: $\mu_{\text{the proposed ExOCF method}} > \mu_{\text{the other tested methods}}$. The estimation ability of the proposed ExOCF method is significantly different from the estimation abilities of the other tested methods.

4.3 Experiment 3 (EX3)

EX3 is conducted to compare the proposed SOCF method with the related SDEE methods, such as UCP-based single methods (described in Table 4-3), OCF-based single methods (described in Table 4-4), and ensemble methods (described in Table 4-5). In addition, we experimented with baseline SDEE methods (UCP and OCF).

Table 4-3. UCP-based single methods implemented for EX3

No.	ML technique	Summary	Notation
1	SVR	- SVR is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&SVR
2	KNN	- KNN is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&KNN
3	DT	- DT is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&DT
4	GRNN	- GRNN is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&GRNN
5	MLP	- MLP is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&MLP
6	RF	- RF is used to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	UCP&RF

Table 4-4. OCF-based single methods implemented for EX3

No.	ML technique	Summary	Notation
1	SVR	- SVR is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&SVR
2	MLP	- MLP is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&MLP
3	GB	- GB is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&GB

4	MLR	- MLR is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&MLR
5	KNN	- KNN is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&KNN
6	DT	- DT is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&DT
7	RF	- RF is used to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	OCF&RF

Table 4-5. Ensemble methods implemented for EX3

No.	ML technique	Summary	Notation
1	Majority voting ensemble	- Majority voting ensemble with MLR, SVR, MLP models to estimate the software size based on UCP variables (UAW, UUCW, TCF, and ECF).	VUCP
2	Stacked Generalization Ensemble	- Stacked generalization ensemble with SVM, KNN, DT, MLP, MLR, GB, RF models to estimate the software size based on OCF variables (UAW, UUCW, LaTF, and LaEF).	SOCF (proposed in Chapter 3.3)

The statistical hypothesis was tested to determine whether the proposed SOCF approach provides a better estimate.

- H_0: $\mu_{\text{the proposed SOCF method}} = \mu_{\text{the other tested methods}}$. The estimation ability of the proposed SOCF method is not significantly different from the estimation abilities of the other tested methods.
- H_1: $\mu_{\text{the proposed SOCF method}} > \mu_{\text{the other tested methods}}$. The estimation ability of the proposed SOCF method is significantly different from the estimation abilities of the other tested methods.

4.4 Experiment 4 (EX4)

EX4 is conducted to compare the proposed OCF(PF_{CFE}) method with the previous SDEE methods (UCP, SW [54], OCF), as summarized in Table 4-6.

Table 4-6. Methods implemented for EX4

No.	SDEE method	Summary	Notation
1	Use Case Points	- Size is measured by UCP variables (UAW, UUCW, TCF, and ECF). - 20 person-hours to develop each UCP (PF=20). - The effort is computed by multiplying Size by the PF.	UCP
2	Schneider and Winter (SW)	- Size is measured by UCP variables (UAW, UUCW, TCF, and ECF). - PF is computed from the UCP environmental complexity factors. - The effort is computed by multiplying Size by the PF.	SW
3	Optimization Correction Factors	- Correction factors are determined in regression analysis by the LASSO regression model. - Size is measured in UCP size variables (UAW and UUCW) and correction factors (LaTF and LaEF). - 20 person-hours to develop each UCP (PF=20). - The effort is computed by multiplying Size by the PF.	OCF
4	Software Productivity Model based on Ensemble Construction Mechanism (proposed in Chapter 3.4)	- Size is measured in UCP size variables (UAW and UUCW) and correction factors (LaTF and LaEF). - A proposed PF_{CFE} model is constructed based on correction factors through an ensemble construction mechanism of three ML models (SVR, MLR, and DT).	OCF(PF_{CFE})

	- The effort is computed by multiplying Size by the $PF_{CFE.}$

The statistical hypothesis was tested to determine whether the proposed OCF(PF_{CFE}) approach provides a better estimate.

- H_0: $\mu_{\text{the proposed OCF(PFCFE) method}} = \mu_{\text{the other tested methods}}$. The estimation ability of the proposed SOCF method is not significantly different from the estimation abilities of the other tested methods.
- H_1: $\mu_{\text{the proposed OCF(PFCFE) method}} > \mu_{\text{the other tested methods}}$. The estimation ability of the proposed SOCF method is significantly different from the estimation abilities of the other tested methods.

5 MAIN RESULTS

This section presents the solutions to the four problem statements given above. The purpose of the results is to minimize the SSE, MdMRE, MAE, MBRE, MIBRE, and RMSE and maximize the PRED (0.25). Specifically, low values for the SSE, MdMRE, MAE, MBRE, MIBRE, and RMSE show good results. In contrast, high values for the PRED (0.25) show good results. Besides that, the results of SSE, MAE, MdMRE, MBRE, MIBRE, and RMSE in the four experimental datasets were used for the paired t-test statistical comparisons. After five runs on different random training- testing had split, we obtained the average p-value of the t-test.

5.1 EX1

In the EX1, we will compare the proposed OCF method as well as the UCP and OTF methods based on the four experimental datasets. Figure 5-1 shows the average estimation results of the proposed OCF and other methods.

The percentage improvements of the proposed OCF over the UCP and OTF methods averaged on all datasets in Table 5-1. These results show that the proposed OCF method outperformed the UCP and OTF methods when the SSE, MAE, MBRE, MIBRE, MdMRE, and RMSE criteria were used. The OCF method also gave good results when PRED (0.25) was used.

Table 5-1. The percentage improvements of the OCF over the UCP and OTF methods averaged on all datasets

Methods	SSE	PRED	MAE	MdMRE	MBRE	MIBRE	RMSE
OCF vs.UCP	53.6%	33.6%	35.4%	42.9%	36.7%	30.2%	34.1%
OCF vs.OTF	37.7%	21.1%	19.2%	36.4%	20.8%	17.8%	23.3%

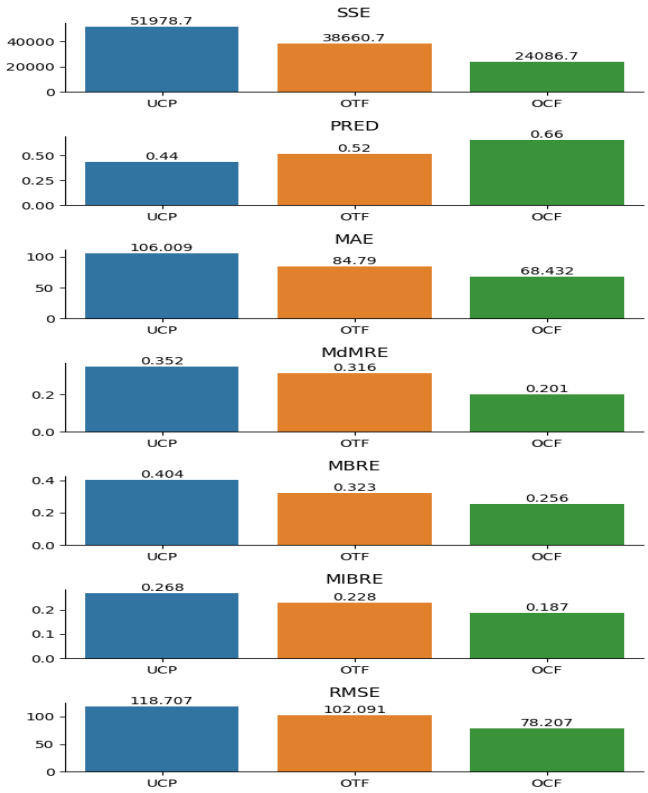

Figure 5-1. The average estimation results of the proposed OCF method and other methods on all datasets

Moreover, we use the SSE, MAE, and RMSE results for all the experimental methods for statistical comparisons, i.e., to draw the most accurate conclusions by comparing estimation methods. The t-test, a parametric statistical comparison test, is used in this study. For a less than, the two statistical methods involved in the comparison are significantly different. As shown in Table 5-2, our proposed OCF method is statistically superior to the baseline UCP method and the OTF method. A>> B means that A is statistically superior to B. Therefore, we accept the alternative hypothesis H1.

Table 5-2. The t-test results for five different runs of the proposed OCF method in comparison with the other methods.

Pairs of methods		OCF vs. UCP	OCF vs. OTF
SSE	Avg. SSE	24,086.736 vs. 51978.747	24,086.736 vs. 38660.720
	Avg. p-value	0.00000	0.00000
	Statistical conclusion	>>	>>
MAE	Avg. MAE	68.432 vs. 106.009	68.432 vs. 84.790

	Avg. p-value	0.00000	0.00001
	Statistical conclusion	>>	>>
RMSE	Avg. RMSE	78.207 vs. 118.707	78.207 vs. 102.091
	Avg. p-value	0.00000	0.00000
	Statistical conclusion	>>	>>

5.2 EX2

In this section, we will evaluate the proposed ExOCF method and five other methods across the four experiment datasets. The average estimation results of methods are shown in Figure 5-2.

The first observation from these results is that the proposed ExOCF method produces the best SSE, MdMRE, MAE, MBRE, MIBRE, RMSE, and PRED (0.25) values, suggesting that it is possible to modify the OCF method to improve its estimation accuracy. From the results obtained, we believe that applying the MLR model to the OCF variables has proven its effectiveness.

The second observation from these results is that the proposed ExOCF method improved accuracy over the baseline UCP method and other tested methods such as AOM, UCP&DT, and UCP&SVR. Table 5-3 presents the percentage improvement of the proposed ExOCF over the AOM, UCP&DT, and UCP&SVR methods averaged on all datasets. Based on this comparison, we can confidently confirm that the proposed method outperforms all other methods with superior accuracy in the evaluation criteria.

Table 5-3. The percentage improvements of the ExOCF over the other methods averaged on all datasets

Methods	SSE	MAE	RMSE	MdMRE	MBRE	MIBRE
ExOCF vs. UCP&DT	46.16%	31.13%	32.17%	32.35%	30.08%	29.97%
ExOCF vs. UCP&SVR	44.11%	31.35%	40.71%	32.28%	31.32%	23.17%
ExOCF vs. AOM	16.73%	13.39%	18.10%	13.06%	12.89%	7.84%

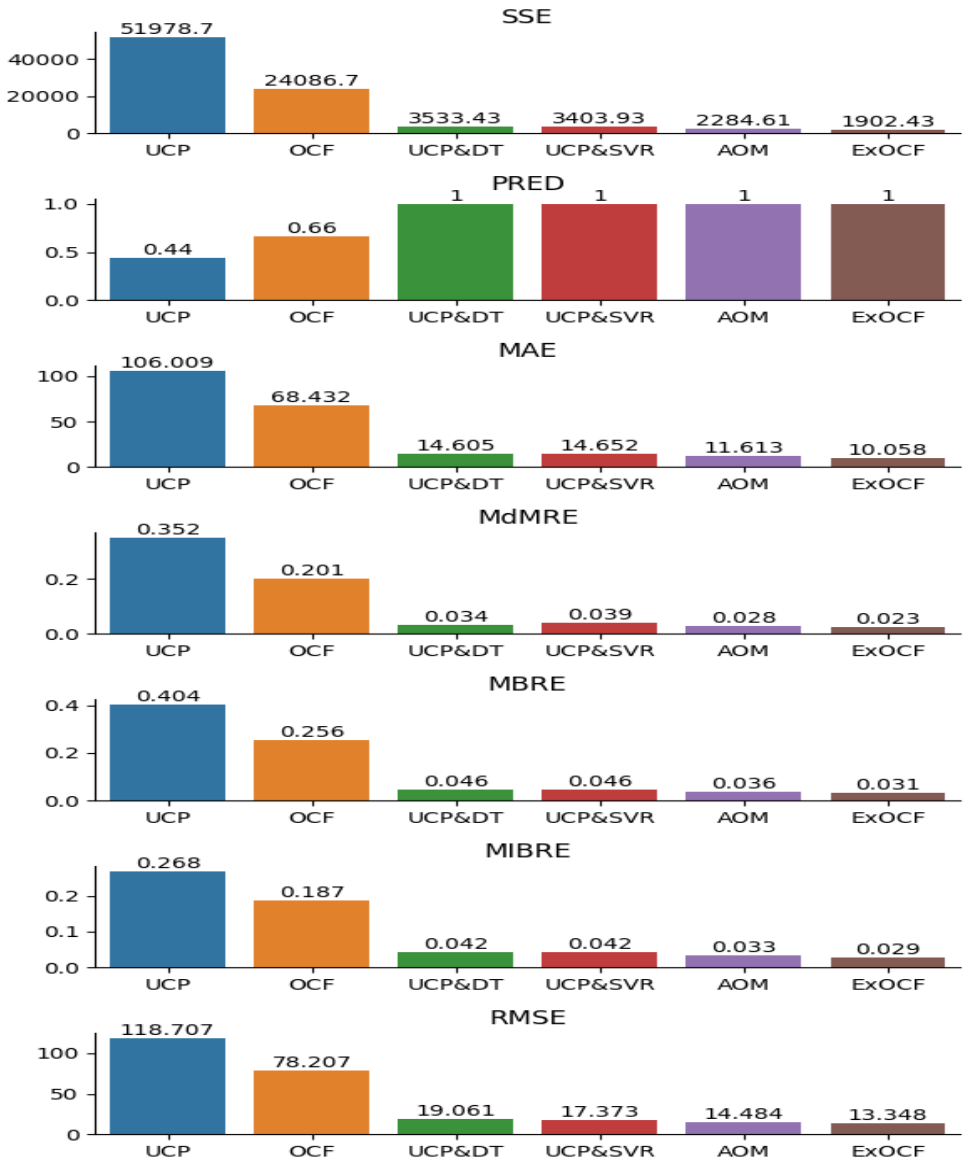

Figure 5-2. The average estimation results of the proposed ExOCF method and other methods on all datasets

Furthermore, the results confirm that the ExOCF method is statistically significant at the 95% confidence level compared to the other five methods, as shown in Table 5-4. As a result, we are inclined to accept the alternative hypothesis (H1), which is also consistent with the results presented above. A>>B means that A is statistically superior to B.

Table 5-4. The t-test results for five different runs of the proposed ExOCF method in comparison with the other methods

Pairs of methods		ExOCF vs. UCP	ExOCF vs. OCF	ExOCF vs. UCP&DT	ExOCF vs. UCP&SVR	ExOCF vs. AOM
SSE	Avg. SSE	1902.4 vs. 51,978.7	1902.4 vs. 24,086.7	1902.4 vs. 3533.4	1902.4 vs. 3403.9	1902.4 vs. 2284.6
	Avg. p-value	0.00000	0.00001	0.00267	0.00316	0.00388
	St. conc.	>>	>>	>>	>>	>>
MAE	Avg. MAE	10.058 vs. 106.009	10.058 vs. 68.432	10.058 vs. 14.605	10.058 vs. 14.651	10.058 vs. 11.613
	Avg. p-value	0.00000	0.00000	0.00001	0.00000	0.00005
	St. conc.	>>	>>	>>	>>	>>
RMSE	Avg. RMSE	13.348 vs. 118.707	13.348 vs. 78.207	13.348 vs. 19.060	13.348 vs. 17.372	13.348 vs. 14.484
	Avg. p-value	0.00000	0.00000	0.00000	0.00001	0.00007
	St. conc.	>>	>>	>>	>>	>>

5.3 EX3

The comparison between the OCF-based and UCP-based single methods is illustrated in Figure 5-3. The first finding from these results is that the experimental results suggest that OCF-based methods reduce estimation errors more effectively than UCP model-based methods. Table 5-5 show the percentage improvements of OCF&SVR, OCF&MLP, OCF&KNN, OCF&DT, and OCF&RF over UCP&SVR, UCP &MLP, UCP &KNN, UCP &DT, and UCP &RF. Based on this finding, we can conclude that approaches that use OCF variables outperform those that use UCP variables.

Table 5-5. The percentage improvements of the OCF-based single methods averaged on all datasets

Methods	SSE	MAE	RMSE	MdMRE	MBRE	MIBRE
OCF&SVR vs. UCP&SVR	35.80%	19.43%	28.17%	20.23%	19.18%	15.82%
OCF&MLP vs. UCP&MLP	39.25%	19.53%	18.50%	19.83%	18.21%	15.80%
OCF&KNN vs. UCP&KNN	44.62%	33.65%	40.18%	34.06%	32.00%	27.47%
OCF&DT vs. UCP&DT	6.63%	6.49%	9.28%	7.23%	6.52%	7.70%
OCF&RF vs. UCP&RF	52.22%	26.56%	25.50%	27.71%	25.30%	24.08%

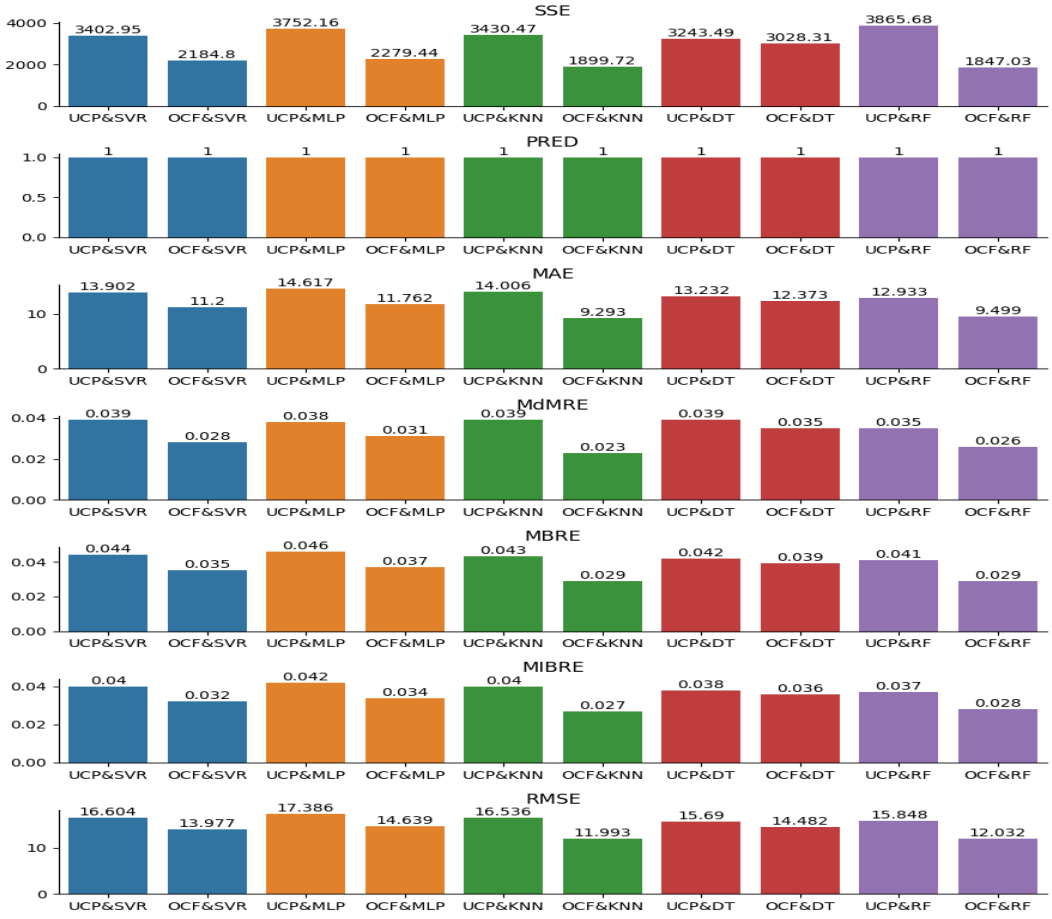

Figure 5-3. The average estimation results of the UCP-based and OCF-based single methods

The comparison between between the ensemble methods and their single approaches is shown in Figure 5-4 and Figure 5-5. Based on these results, we can conclude that the ensemble methods outperform their single methods, and the proposed SOCF approach surpasses the VUCP method. Moreover, the results confirm that the SOCF method is statistically significant at the 95% confidence level compared to the other methods, as shown in Table 5-6, Table 5-7, and Table 5-8. A>> B means that A is statistically superior to B. Therefore, we accept the alternative hypothesis H1.

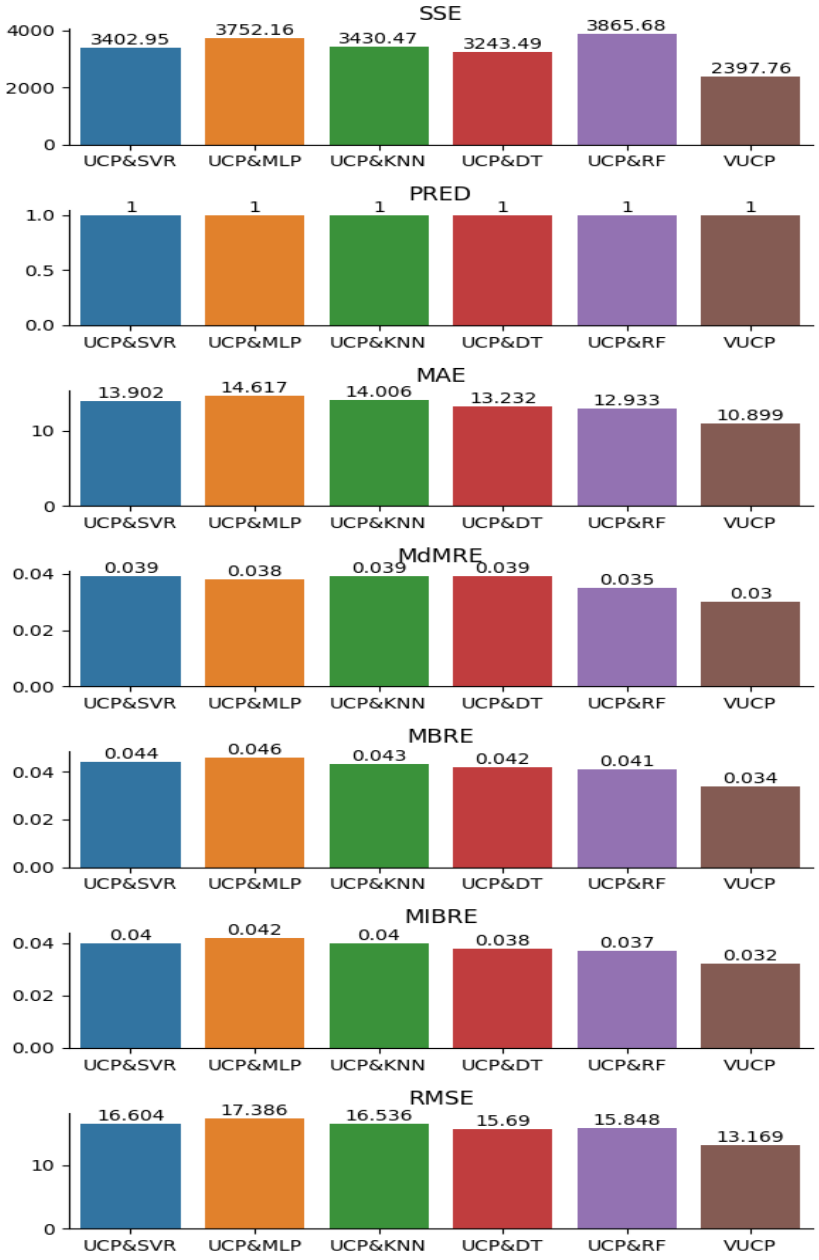

Figure 5-4. The comparison between the ensemble method VUCP and its single approaches

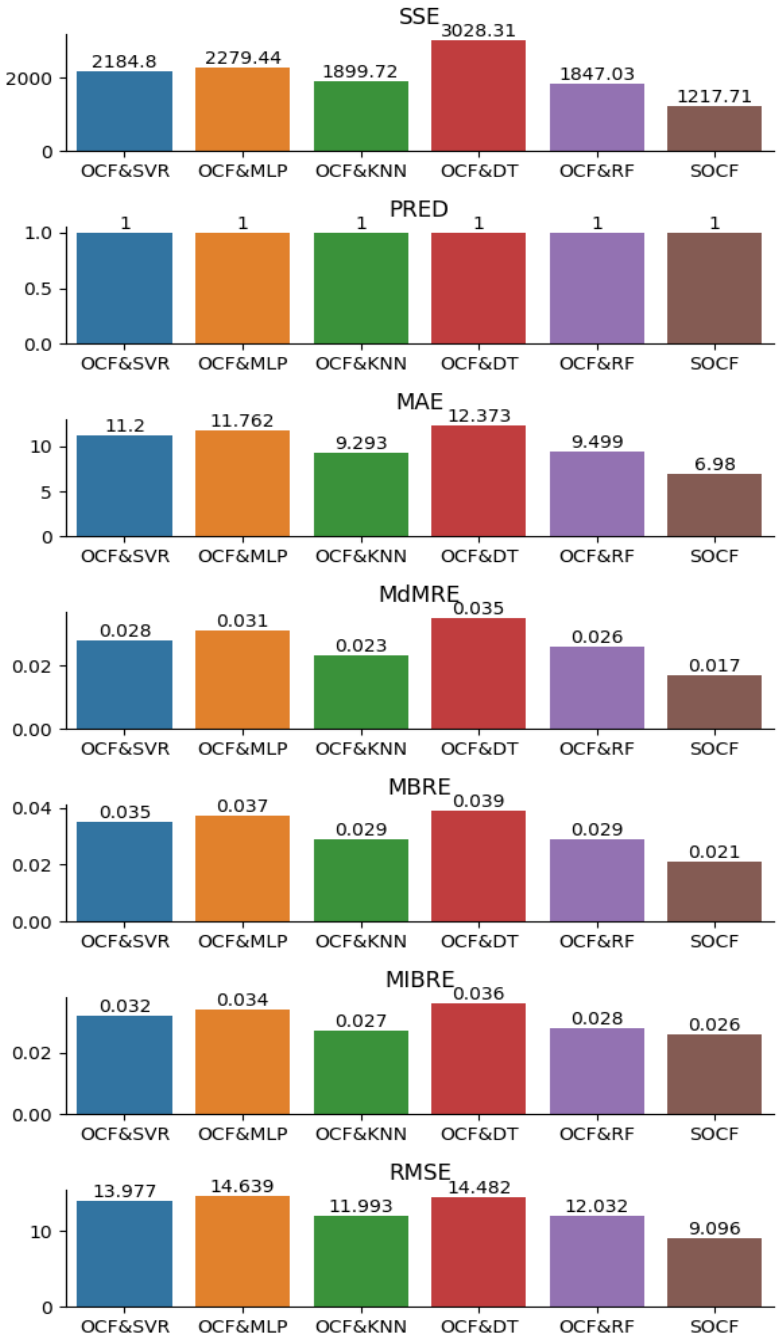

Figure 5-5. The comparison between the ensemble method SOCF and its single approaches

Table 5-6. The t-test results for five different runs of the proposed SOCF method in comparison with the other methods

Pairs of methods		SOCF vs. UCP	SOCF vs. OCF&MLP	SOCF vs. OCF&DT	SOCF vs. OCF&SVR	SOCF vs. OCF&MLR
SSE	Avg. SSE	1217.7 vs. 54838.9	1217.71 vs. 2279.44	1217.71 vs 3028.31	1217.71 vs. 2184.80	1217.71 vs. 2117.07
	Avg. value	0.00000	0.00076	0.00440	0.00190	0.00514
	St. conc.	>>	>>	>>	>>	>>
MAE	Avg. MAE	6.980 vs. 95.615	6.980 vs. 11.762	6.980 vs. 12.373	6.980 vs. 11.200	6.980 vs. 10.874
	Avg. value	0.00000	0.00000	0.00005	0.00000	0.00001
	St. conc.	>>	>>	>>	>>	>>
RMSE	Avg. RMSE	9.096 vs. 104.339	9.096 vs. 14.639	9.096 vs. 14.482	9.096 vs. 13.977	9.096 vs. 13.197
	Avg. value	0.00000	0.00002	0.00006	0.00005	0.00000
	St. conc.	>>	>>	>>	>>	>>

Table 5-7. The t-test results for five different runs of the proposed SOCF method in comparison with the other methods

Pairs of methods		SOCF vs. OCF&GB	SOCF vs. OCF&RF	SOCF vs. UCP&KNN	SOCF vs. UCP&SVR	SOCF vs. UCP&MLP
SSE	Avg. SSE	1217.7 vs. 3069.7	1217.7 vs. 1847.0	1217.71 vs. 1899.03	1217.71 vs. 3402.95	1217.7 vs. 2117.07
	Avg. value	0.00460	0.00583	0.01199	0.00195	0.00514
	St. conc.	>>	>>	>>	>>	>>
MAE	Avg. MAE	6.980 vs. 12.441	6.980 vs. 9.499	6.980 vs. 9.293	6.980 vs. 13.902	6.980 vs. 10.874
	Avg. value	0.00005	0.00000	0.00005	0.00005	0.00001
	St. conc.	>>	>>	>>	>>	>>

RMSE	Avg. RMSE	9.096 vs. 14.530	9.096 vs. 12.032	9.096 vs. 11.993	9.096 vs. 16.604	9.096 vs.
	Avg. value	0.00006	0.00005	0.00010	0.00000	13.197
	St. conc.	>>	>>	>>	>>	>>

Table 5-8. The t-test results for five different runs of the proposed SOCF method in comparison with the other methods

Pairs of methods		SOCF vs. UCP&GB	SOCF vs. OCF&KN	SOCF vs. UCP&DT	SOCF vs. UCP&RF	SOCF vs. VUCP
SSE	Avg. SSE	1217.7 vs. 3069.7	1217.7 vs. 1847.0	1217.71 vs. 1899.03	1217.71 vs. 3402.95	1217.71 vs. 2397.77
	Avg. value	0.00460	0.00583	0.01199	0.00195	0.00764
	St. conc.	>>	>>	>>	>>	>>
MAE	Avg. MAE	6.980 vs. 12.441	6.980 vs. 9.499	6.980 vs. 9.293	6.980 vs. 13.902	6.980 vs. 10.899
	Avg. value	0.00005	0.00000	0.00005	0.00005	0.00003
	St. conc.	>>	>>	>>	>>	>>
RMSE	Avg. RMSE	9.096 vs. 14.530	9.096 vs. 12.032	9.096 vs. 11.993	9.096 vs. 16.604	9.096 vs. 13.169
	Avg. value	0.00006	0.00005	0.00010	0.00000	0.00007
	St. conc.	>>	>>	>>	>>	>>

5.4 EX4

In the EX4, we will compare the proposed OCF(PFCFE) method as well as the previous related methods (UCP, SW, and OCF) based on the four experimental datasets. The comparison between the OCF(PF_{CFE}) method and three related methods is illustrated in Figure 5-6. The obtained results allow us to confidently conclude that the OCF(PF_{CFE}) using the proposed software productivity approach achieves better improvements than the previous related methods using fixed productivity metrics, concerning all accuracy measures.

(3) The percentage improvements of the proposed OCF(PFCFE) over the other methods are presented in Table 5-9. This conclusion is confirmed by statistical t-test comparisons for each corresponding method (see

Table 5-10). A>>B refers to A statistical superiority to B. The OCF(PFCFE) using the proposed software productivity approach is statistically better than other methods, as the obtained p-values are all below 0.05.

Table 5-9. The percentage improvements of the proposed OCF(PF_{CFE}) method averaged on all datasets

Methods	SSE	PRED	MAE	RMSE	MdMRE	MBRE	MIBRE
OCF(PF_{CFE}) vs. UCP	58.6%	43.4%	43.7%	40.7%	51.7%	51.1%	41.6%
OCF(PF_{CFE}) vs. SW	62.2%	37.1%	45.2%	43.5%	54.8%	47.0%	39.4%
OCF(PF_{CFE}) vs. OCF	31.3%	30.1%	31.1%	23.2%	44.0%	41.9%	35.0%
OCF(PF_{CFE}) vs. AOM	58.6%	43.4%	43.7%	40.7%	51.7%	51.1%	41.6%
OCF(PF_{CFE}) vs. UCP	62.2%	37.1%	45.2%	43.5%	54.8%	47.0%	39.4%

Table 5-10 The t-test results of five different runs for statistical comparison of our proposed OCF(PF_{CFE}) methods with other tested methods

Pairs of methods		OCF(PF_{CFE}) vs. UCP	OCF(PF_{CFE}) vs. SW	OCF(PF_{CFE}) vs. OCF	OCF(PF_{CFE}) vs. AOM
SSE	Avg. SSE	2.16E+07 vs. 5.21E+07	2.16E+07 vs. 5.72E+07	2.16E+07 vs. 3.15E+07	2.16E+07 vs. 3.75E+07
	Avg. p-value	0.00000	0.00000	0.00000	0.00011
	St. conc.	>>	>>	>>	>>
MAE	Avg. MAE	1194.141 vs. 2171.213	1194.141 vs. 2228.636	1194.141 vs. 1773.276	1194.141 vs. 1756.148
	Avg. p-value	0.00000	0.00000	0.00000	0.00000

	St. conc.	>>	>>	>>	>>
RMSE	Avg. RMSE	1554.136 vs. 2620.587	1554.136 vs. 2748.516	1554.136 vs. 2024.497	1554.136 vs. 2126.666
	Avg. p-value	0.00000	0.00000	0.00000	0.00000
	St. conc.	>>	>>	>>	>>

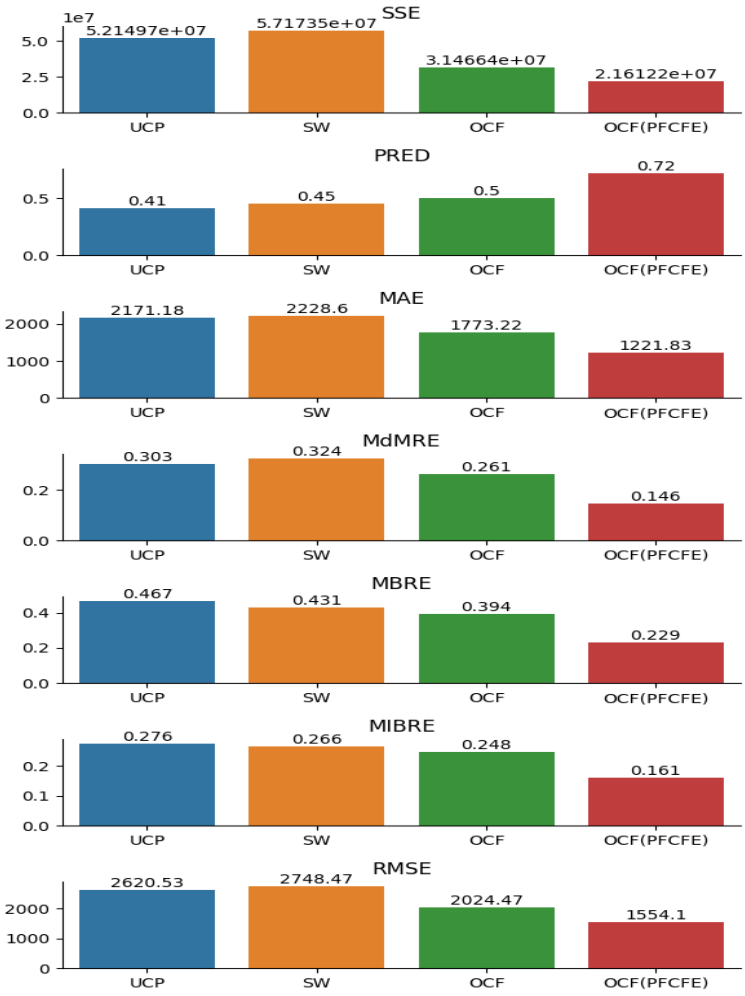

Figure 5-6. The average estimation results of the proposed OCF(PF_{CFE}) method and other methods on all dataset

6 CONTRIBUTIONS OF THE THESIS TO SCIENCE AND PRACTICE

The main benefit of this work is the introduction of a new approach to complex algorithms based on engineering requirements research for a more accurate estimation of software effort. The new algorithms are inspired by the possibilities of using a standardized estimation procedure to address the impact of human error in UCM analysis and to simplify the original UCP principles.

The main benefits of this work can be summarized as follows:

- Proposed procedures can help project managers reduce risks in evaluating correction factors and obtain effort estimates.
- An algorithm for calculating productivity based on correction factors has been proposed through a voting set approach consisting of three ML techniques.
- Proposed a comprehensive approach to improve estimation accuracy and minimize project risks in the early stages of software development.
- Experiments have shown that the use of the proposed new algorithms minimizes the estimation error compared to the selected methods.

In summary, the results obtained can be considered beneficial for industrial applications, as they show that the proposed algorithms lead to more accurate estimates of the size and complexity of the software.

7 CONCLUSIONS

The presented doctoral thesis is proposed UCP-based estimation methods in the early stages of software development. Our methods can help project managers estimate costs early and efficiently, avoiding project overestimation and late delivery, among other issues. Each approach has its advantages, and they complement each other to form a complete process and promote significant efficiency to minimize estimation error more efficiently in all situations. The results show that our proposed SDEE method outperforms other related methods.

One of our future works is to calibrate the weighting values of the correction factors to reflect the latest trend in the software development industry and improve the accuracy of the proposed methods. Therefore, an approach to calibrate the weights of the correction factors using an artificial neural network will be performed in the future. Another concern relates to a key aspect of the heterogeneity of the historical data. This could lead to an increase in the estimation error for SDEE. The use of clustering approaches is considered a solution to improve the method's estimation accuracy in our future work.

LITERATURE

[1] B. W. Boehm, "Software Engineering Economics," *IEEE Transactions on Software Engineering,* vol. SE-10, *(1),* 1984.

[2] B. Boehm *et al*, "Software Cost Estimation with COCOMO II. Prentice Hall," *Upper Saddle River, NJ,* 2000.

[3] M. Jørgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering,* vol. 33, *(1),* 2007.

[4] A. Trendowicz, J. Münch and R. Jeffery, "State of the practice in software effort estimation: A survey and literature review," in *IFIP Central and East European Conference on Software Engineering Techniques,* 2008, pp. 232-245.

[5] Nhung, Ho Le Thi Kim, H. T. Hoc and V. V. Hai, "A review of use case-based development effort estimation methods in the system development context," *Proceedings of the Computational Methods in Systems and Software,* pp. 484-499, 2019.

[6] B. Boehm, C. Abts and S. Chulani, "Software development cost estimation approaches — A survey," *Annals of Software Engineering,* vol. 10, *(1/4),* pp. 177-205, 2000.

[7] R. N. Charette, "Why Software Fails," *IEEE Spectrum,* vol. 42, *(9),* 2005.

[8] Arlene Minkiewicz, "Use Case Sizing," *PRICE Systems, L.L.C,* 2015.

[9] C. J. Neill and P. A. Laplante, "Requirements Engineering: The State of the Practice," *IEEE Software,* vol. 20, *(6),* 2003.

[10] Gustav Karner, "Resource Estimation for Objector Projects." , 1993.

[11] M. Azzeh, A. Bou Nassif and I. B. Attili, "Predicting software effort from use case points: A systematic review," *Science of Computer Programming,* vol. 204, 2021.

[12] V. Khatibi and D. N. a. Jawawi, "Software Cost Estimation Methods : A Review," *Journal of Emerging Trends in Computing and Information Sciences,* vol. 2, *(1),* 2010.

[13] B. Marapelli, A. Carie and S. M. Islam, "Software effort estimation with use case points using ensemble machine learning models," in *2021 International Conference on Electrical, Computer and Energy Technologies (ICECET),* 2021.

[14] R. Silhavy, P. Silhavy and Z. Prokopova, "Using actors and use cases for software size estimation," *Electronics (Switzerland),* vol. 10, *(5),* 2021.

[15] M. Manzoor and A. Wahid, "Revised Use Case Point (Re-UCP) Model for Software Effort Estimation," *International Journal of Advanced Computer Science and Applications,* vol. 6, *(3),* 2015.

[16] F. Wang *et al*, "Extended use case points method for software cost estimation," in *International Conference on Computational Intelligence and Software Engineering,* 2009.

[17] K. Periyasamy and A. Ghode, "Cost estimation using extended use case point (e-UCP) model," in *2009 International Conference on Computational Intelligence and Software Engineering*, 2009.

[18] M. Jørgensen, "Regression models of software development effort estimation accuracy and bias," *Empirical Software Engineering*, vol. 9, *(4)*, 2004.

[19] S. Humpage, "An introduction to regression analysis," *Sensors (Peterborough, NH)*, vol. 17, *(9)*, 2000.

[20] V. Khatibi Bardsiri *et al*, "A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons," *Empirical Software Engineering*, vol. 19, *(4)*, 2014.

[21] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Information and Software Technology*, vol. 54, *(8)*, 2012.

[22] M. Azzeh *et al*, "Pareto efficient multi-objective optimization for local tuning of analogy-based estimation," *Neural Computing and Applications*, vol. 27, *(8)*, 2016.

[23] R. Silhavy, P. Silhavy and Z. Prokopova, "Algorithmic optimisation method for improving use case points estimation," *PLoS ONE*, vol. 10, *(11)*, 2015.

[24] N. Nunes, L. Constantine and R. Kazman, "IUCP: Estimating interactive-software project size with enhanced use-case points," *IEEE Software*, vol. 28, *(4)*, 2011.

[25] A. R. Gray and S. G. MacDonell, "A comparison of techniques for developing predictive models of software metrics," *Information and Software Technology*, vol. 39, *(6)*, 1997.

[26] R. Alves, P. Valente and N. J. Nunes, "Improving software effort estimation with human-centric models: A comparison of UCP and iUCP accuracy," in *EICS 2013 - Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 2013.

[27] P. Jovan *et al*, "Enhancing use case point estimation method using fuzzy algorithms," in *2015 23rd Telecommunications Forum Telfor (TELFOR)*, 2015.

[28] M. Saroha and S. Sahu, "Software effort estimation using enhanced use case point model," in *Nternational Conference on Computing, Communication and Automation, ICCCA 2015*, 2015.

[29] L. M. Huanca and S. B. Oré, "Factors affecting the accuracy of use case points," in *International Conference on Software Process Improvement*, 2016.

[30] A. B. Nassif, D. Ho and L. F. Capretz, "Towards an early software estimation using log-linear regression and a multilayer perceptron model," *Journal of Systems and Software*, vol. 86, *(1)*, 2013.

[31] Sholiq, R. S. Dewi and A. P. Subriadi, "A comparative study of software development size estimation method: UCPabc vs function points," in *Procedia Computer Science*, 2017.

[32] P. Mohagheghi, B. Anda and R. Conradi, "Effort estimation of use cases for incremental large-scale software development," in *Proceedings - 27th International Conference on Software Engineering, ICSE05*, 2005.

[33] M. R. Braz and S. R. Vergilio, "Software effort estimation based on use cases," in *Proceedings - International Computer Software and Applications Conference*, 2006.

[34] K. Qi *et al*, "Calibrating use case points using bayesian analysis," in *International Symposium on Empirical Software Engineering and Measurement*, 2018.

[35] K. Rak, Ž Car and I. Lovrek, "Effort estimation model for software development projects based on use case reuse," *Journal of Software: Evolution and Process*, vol. 31, *(2)*, 2019.

[36] G. Robiolo, C. Badano and R. Orosco, "Transactions and paths: Two use case based metrics which improve the early effort estimation," in *3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*, 2009.

[37] L. Lavazza and G. Robiolo, "The role of the measure of functional complexity in effort estimation," in *ACM International Conference Proceeding Series*, 2010.

[38] P. S. Kumar *et al*, "Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades," *Computer Science Review*, vol. 38, 2020.

[39] T. M Kiran Kumar and M. A. Jayaram, "Comparison of hard limiting and soft computing methods for predicting software effort estimation: In reference to Small Scale Visualization Projects," *International Journal of Engineering & Technology*, vol. 7, *(4.6)*, 2018.

[40] Jose Thiago, Jose Thiago H. and A. L. I. Oliveira, "Ensemble Effort Estimation using dynamic selection," *Journal of Systems and Software*, vol. 175, 2021.

[41] A. G. Priya Varshini, K. Anitha Kumari and V. Varadarajan, "Estimating software development efforts using a random forest-based stacked ensemble approach," *Electronics (Switzerland)*, vol. 10, *(10)*, 2021.

[42] M. Jørgensen, U. Indahl and D. Sjøberg, "Software effort estimation by analogy and "regression toward the mean"," in *Journal of Systems and Software*, 2003.

[43] J. Heidrich, M. Oivo and A. Jedlitschka, "Software productivity and effort estimation," *Journal of Software: Evolution and Process*, vol. 27, *(7)*, 2015.

[44] D. Rodríguez *et al*, "Empirical findings on team size and productivity in software development," *Journal of Systems and Software*, vol. 85, *(3)*, 2012.

[45] K. Petersen, "Measuring and predicting software productivity: A systematic map and review," *Information and Software Technology*, vol. 53, *(4)*, 2011.

[46] B. Kitchenham and E. Mendes, "Software productivity measurement using multiple size measures," *IEEE Transactions on Software Engineering*, vol. 30, *(12)*, 2004.

[47] L. M. Alves *et al*, "An empirical study on the estimation of software development effort with use case points," in *Proceedings - Frontiers in Education Conference, FIE*, 2013.

[48] Le Thi Kim Nhung, Ho, H. T. Hoc and V. Van Hai, "An evaluation of technical and environmental complexity factors for improving use case points estimation," in *Advances in Intelligent Systems and Computing*, 2020.

[49] Nhung, Ho Le Thi Kim *et al*, "Parametric Software Effort Estimation Based on Optimizing Correction Factors and Multiple Linear Regression," *IEEE Access*, vol. 10, 2022.

[50] Nhung, Ho Le Thi Kim, V. Van Hai and R. Jašek, "Towards a correction factors-based software productivity using ensemble approach for early software development effort estimation," in *Computer Science on-Line Conference*, 2022.

[51] A. Chandra and X. Yao, "Ensemble learning using multi-objective evolutionary algorithms," *Journal of Mathematical Modelling and Algorithms*, vol. 5, *(4)*, 2006.

[52] K. An and J. Meng, "Voting-averaged combination method for regressor ensemble," in *International Conference on Intelligent Computing*, 2010.

[53] B. Anda, E. Angelvik and K. Ribu, "Improving estimation practices by applying use case models," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2002.

[54] W. J. Schneider. G, "Applied use cases," in *A Practical Guide, Second Edition, Addison-Wesley,* 2001.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS USED

Abbreviations	Description
SDEE	Software development effort estimation
UCP	Use Case Points
UCM	Use Case Model
TCF	Technical complexity factors
ECF	Environmental complexity factors
MLR	Multiple linear regression
ML	Machine learning
PF	Productivity factor
LASSO	Least Absolute Shrinkage and Selection Operator
OCF	Optimization Correction Factors
LSR	Least square regression
ExOCF	Extension of Optimizing Correction Factors
KNN	K-nearest neighbor
RF	Random forest
SVR	Support vector regression
MLP	Multi-layer perceptron
GB	Gradient Boosting
DT	Decision tree
SOCF	Stacked OCF
UAW	Unadjusted actor weight
UUCW	Unadjusted use case weight
GS	Grid search
MAE	Mean Absolute Error
MMRE	Mean magnitude of relative error
MBRE	Mean balance relative error
MIBRE	Inverted balance relative error
MdMRE	Median magnitude of relative error
RMSE	Root mean square error
SSE	Sum of squares errors
PRED(x)	Percentage of prediction within x%

SA	Standardized accuracy
SLOC	Source lines of code
FPA	Function points analysis
COCOMO	Constructive cost model
SLIM	Software life cycle management
AOM	Algorithmic Optimisation Method
LOOCV	Leave on out cross-validation
OCF(PF_{CFE})	Effective productivity factor calculations

LIST OF PUBLICATIONS OF THE AUTHOR

Journal papers:

1. H.LT.K. Nhung, V.V. Hai, R. Silhavy, Z. Prokopova, and P. Silhavy, "Parametric Software Effort Estimation Based on Optimizing Correction Factors and Multiple Linear Regression, " IEEE Access, vol. 10, pp. 2963-2986, DOI: 10.1109/ACCESS.2021.3139183, 2022.
2. V.V. Hai, H.L.T.K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, "A New Approach to Calibrating Functional Complexity Weight in Software Development Effort Estimation," Computers 11, no. 2: 15, DOI: 10.3390/computers11020015, 2022.
3. V.V. Hai, H.L.T.K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, "Towards improving the efficiency of software development effort estimation via clustering analysis," IEEE Access, vol. 10, pp. 83249-83264, DOI: 10.1109/ACCESS.2022.3185393, 2022.

Conference papers:

4. H.LT.K. Nhung, V.V. Hai, and R. Jasek, "Towards a Correction Factors-based Software Productivity using Ensemble approach for Early Software Development Effort Estimation," Lecture Notes in Networks ans Systems, vol. 501 LNNS, pp. 413-425, DOI: 10.1007/978-3-031-09070-7_35, 2022.
5. H.LT.K. Nhung, V.V. Hai, and H.T. Hoc, "Analyzing Correlation of the relationship between Technical Complexity Factors and Environmental Complexity Factors for Software Development Effort Estimation", Lecture Notes in Networks and Systems, 232 LNNS, pp. 835-848, DOI: 10.1007/978-3-030-90318-3_65, 2021.
6. H.LT.K. Nhung, V.V. Hai, and H.T. Hoc, "Evaluation of Technical and Environmental Complexity Factors for Improving Use Case Points Estimation," Advances in Intelligent Systems and Computing Springer, 1294, pp. 757–768, DOI: 10.1007/978-3-030-63322-6_64, 2020.
7. H.L.T.K. Nhung, H.T. Hoc, and V.V. Hai, "A Review of Use Case-Based Development Effort Estimation Methods in the System Development Context," Advances in Intelligent Systems and Computing, 1046, pp. 484-499, DOI: 10.1007/978-3-030-30329-7_44, 2019.

8. V.V. Hai, H.L.T.K. Nhung, and R. Jasek, "Toward appying agglomerative hierarchical clustering in improving the software development effort estimation," Lecture Notes in Networks and Systems, vol. 501 LNNS, pp. 353-371, DOI: 10.1007/978-3-031-09070-7_30, 2022.
9. V.V. Hai, H.L.T.K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, “Analyzing the effectiveness of the Gaussian Mixture Model clustering algorithm in Software Enhancement Effort Estimation,” ACIIDS 2022.
10. V.V. Hai, H.L.T.K. Nhung, H.T. Hoc, "Calibrating Function Complexity in Enhancement Project for Improving Function Points Analysis Estimation," Lecture Notes in Networks and Systems, 232 LNNS, pp. 857-869, DOI: 10.1007/978-3-030-90318-3_67, 2021.
11. V.V. Hai, H.L.T.K. Nhung, H.T. Hoc, "Empirical Evidence in Early Stage Software Effort Estimation Using Data Flow Diagram," Lecture Notes in Networks and Systems, 230, pp. 632-644, DOI: 10.1007/978-3-030-77442-4_53, 2021.
12. V.V. Hai, H.L.T.K. Nhung, H.T. Hoc, "A Productivity Optimising Model for Improving Software Effort Estimation," Advances in Intelligent Systems and Computing, 1294, pp. 735-746, DOI: 10.1007/978-3-030-63322-6_62, 2020.
13. V.V. Hai, H.L.T.K. Nhung, H.T. Hoc, "A Review of Software Effort Estimation by Using Functional Points Analysis," Advances in Intelligent Systems and Computing, 1047, pp. 408-422, DOI: 10.1007/978-3-030-31362-3_40, 2019.
14. H.T. Hoc, V.V. Hai, H.L.T.K. Nhung, "An Approach to Adjust Effort Estimation of Function Point Analysis," Lecture Notes in Networks and Systems, 230, pp. 522-537, DOI: 10.1007/978-3-030-77442-4_45, 2021.
15. H.T. Hoc, V.V. Hai, H.L.T.K. Nhung, "AdamOptimizer for the Optimisation of Use Case Points Estimation," Advances in Intelligent Systems and Computing, 1294, pp. 747-756, 2020.
16. H.T. Hoc, V.V. Hai, H.L.T.K. Nhung, " A Review of the Regression Models Applicable to Software Project Effort Estimation," Advances in Intelligent Systems and Computing, 1047, pp. 399-407, 10.1007/978-3-030-31362-3_39, 2019.

CURRICULUM VITAE AUTHOR

Name:	Ho Le Thi Kim Nhung
Education and degrees:	Tomas Bata University in Zlin, Czech Republic PhD student, Software Engineering 12/2018-Now University of Science, Vietnam (HCMUS-VNU) Master of Information Systems 2011-2014 University of Science, Vietnam (HCMUS-VNU) Bachelor of Information Systems 2007-2010
Related Work Experience:	University of Science, Vietnam (HCMUS-VNU) Lecturer, Faculty of Information Technology 2011-2018 Institute of International Management (IIMBA), National Cheng Kung University, Tainan, Taiwan Visiting research scientist 8/2016-8/2017 National Institute of Informatics, Japan Visiting research scientist 8/2016-1/2017
Honors and Awards:	Outstanding Young Lecturer at University of Science HCMUS-VNU (2015, 2017) Top 10 graduate of Honors degree at University of Science (HCMUS-VNU), Certificate of Merit on being the best student of graduation (7/500 students) (2010)

Efektivní parametrický model pro odhad projektu systémového inženýrství

Effective Parametric Model for System Engineering Project Estimation

Doctoral Thesis Summary

Published by: Tomas Bata University in Zlín,

nám. T. G. Masaryka 5555, 760 01 Zlín, Czech Republic

Edition: published electronically

Typesetting by: Ho Le Thi Kim Nhung, Ph.D.

This publication has not undergone any proofreading or editorial review.

First Edition

Publication year: 2022

ISBN 978-80-7678-130-6