Optimizing complexity weight parameter of use case points estimation using particle swarm optimization

Ardiansyah [a,b,1,*], Ridi Ferdiana [a,2], Adhistya Erna Permanasari [a,3]

[a] Department of Electrical Engineering and Information Technology, Universitas Gadjah Mada, Yogyakarta, Indonesia
[b] Department of Informatics, Faculty of Industrial Technology, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[1] ardiansyah2018@mail.ugm.ac.id, ardiansyah@tif.uad.ac.id; [2] ridi@ugm.ac.id; [3] adhistya@ugm.ac.id
* corresponding author

ARTICLE INFO

Article history
Received March 14, 2022
Revised June 15, 2022
Accepted June 29, 2022
Available online July 31, 2022

Keywords
Use case points
Effort estimation
Particle swarm optimization
Use case complexity
Metaheuristic optimization

ABSTRACT

Among algorithmic-based frameworks for software development effort estimation, Use Case Points I s one of the most used. Use Case Points is a well-known estimation framework designed mainly for object-oriented projects. Use Case Points uses the use case complexity weight as its essential parameter. The parameter is calculated with the number of actors and transactions of the use case. Nevertheless, use case complexity weight is discontinuous, which can sometimes result in inaccurate measurements and abrupt classification of the use case. The objective of this work is to investigate the potential of integrating particle swarm optimization (PSO) with the Use Case Points framework. The optimizer algorithm is utilized to optimize the modified use case complexity weight parameter. We designed and conducted an experiment based on real-life data set from three software houses. The proposed model's accuracy and performance evaluation metric is compared with other published results, which are standardized accuracy, effect size, mean balanced residual error, mean inverted balanced residual error, and mean absolute error. Moreover, the existing models as the benchmark are polynomial regression, multiple linear regression, weighted case-based reasoning with (PSO), fuzzy use case points, and standard Use Case Points. Experimental results show that the proposed model generates the best value of standardized accuracy of 99.27% and an effect size of 1.15 over the benchmark models. The results of our study are promising for researchers and practitioners because the proposed model is actually estimating, not guessing, and generating meaningful estimation with statistically and practically significant.

1. Introduction

Software effort estimation (SEE) is one of the early activities of the software life cycle. SDEE estimates how much effort and cost are required to develop a new software system. Estimating effort is essential because the software organizations must release the software within a given timeframe and cost. Unfortunately, most software projects are delivered over time and budget. Time and cost overruns have been common problems in software projects for many years [1]. As reported by Bloch *et al.* [2], software projects with a budget of more than $15 million run 66% over budget and 33% over time. Hence, estimating effort and cost accuracy is essential in SEE to successfully evade the time and budget overruns for overall software delivery [3], [4].

Software size and productivity factors are two variables in estimating the software development project. Use case points (UCP) is a software sizing and estimation framework introduced by Karner [5]. UCP utilizes use case diagrams to calculate the size of the software and multiply it by the productivity factor. In the second step of UCP, as notated in Eq. (2), the use case (UC) complexity weight level is classified into simple, average, and complex. Each weight complexity level is assigned based on the number of UC transactions. The original UCP framework appointed complexity weight levels as 5 for Simple, 10 for average, and 15 for the complex. Researchers have criticized the origin complexity weight level as the complexity hierarchy level is discontinuous, which can sometimes result in inaccurate measurements [6] and abrupt classification of use case [7], [8]. For example, the use case of 8 transactions has double the weight of the use case with seven transactions. Moreover, they are not considering huge use case transactions. For example, the use case of 25 transactions has the same weight as the use case of 9 transactions.

Several works have been conducted to solve the abrupt classification problem of the use case complexity weight level. For instance, the existing fuzzy approaches are utilized and proposed [6], [7], [9], [10]. Fuzzy logic is always used to discretize the existing complexity weight level. It tries to smoothen the abrupt classification by providing continuous and gradual classification. Most studies prove that fuzzy UCP improved estimation performance compared with the original UCP.

Continuous use case classification weight level allows us to further improve the accuracy of UCP by conducting optimization. A well-known approach to the continuous problem is optimization. Particle Swarm Optimization (PSO) is an appropriate optimization algorithm for this problem due to several considerations. First, PSO is straightforward, easy to implement, and computationally efficient [11], [12]. Second, the performance of PSO outpaces several well-known evolutionary algorithms such as simulated annealing and genetic algorithm. Furthermore, PSO converges quickly and is widely proven for solving various troublesome optimization problems [13].

Search-based software effort estimation research has been used in many studies [14]. Particle swarm optimization (PSO) [13], Genetic programming [15], [16], simulated annealing [17], Bayesian optimization [18], hybrid PSO-SA [19], Genetic Algorithm (GA) [20], and differential evolutionary (DE) [21] optimization are employed for cased-based reasoning effort estimation. COCOMO effort estimation is optimized by using Firefly [22], [23], PSO [24]–[27], GA [28], stochastic gradient descent [29], DE [30], and COA [31], [32]. However, little work has been studied in the UCP framework by utilizing the optimization approach to the best of our knowledge. Hence, there is a gap and challenge to improve the use case classification weight level derived from the modified use case weight level UCP proposed by previous studies. Therefore, this study aims to optimize the use case complexity weight parameter to improve the accuracy of software sizing and effort development using PSO. In contrast, the contribution of this study is an improved Use Case Point estimation method by integrating the PSO algorithm.

The structure of the remaining part of this paper is organized as follows: Section 2 related work; Section 3 the theoretical framework of use case points and particle swarm optimization; Section 4 details the proposed model in this study; Section 5 details of experiment setup; Section 6 presents and discusses the experimental results and Section 7 discusses the conclusion and recommends future works of this study.

2. Related works

There are three primary trends in the study of use case points effort estimation: modification of the UCP sizing technique, simplifying and examining the UCP, and hybridizing the UCP with machine learning and data mining techniques. Several studies proposed the reconstruction of the UCP sizing technique. Braz and Vergilio [33] modified the use case complexity weight using fuzzy theory, while [34] successfully optimized this modified weight. Robiolo and Orosco [35] added two new variables, the size-transactions and entity objects, computed from the use case description. The study of Anda *et al.* [36] has modified the complexity assessment of actors and handled the non-functional requirements. This study made an essential contribution to the adaptability of the UCP for incremental development.

Anda *et al.* [37], [38] examined and simplified the UCP to understand the impacts of technical and environmental complexity factors. The authors suggested that adjusting the environmental factors based on the type of organization will improve estimation precision. Whereas Ochodek *et al.* [39], [40] excluded several parts of UCP to simplify the calculation process of the UCP. The investigator claimed that these parts are insignificant concerning the effort estimation. Recently, Nhung *et al.* [41] optimized the correction factors (ECF and TCF) and multiple regression models to improve the estimation accuracy of the modified UCP.

The utilization of machine learning and data mining techniques to improve UCP performance has been studied in recent years. Nassif [42] built cooperation between effort, UCP, and productivity by introducing a log-linear regression model. This study was followed by a hybrid model that simultaneously predicts productivity factor and effort estimation from historical data [43]. Meanwhile, Nassif *et al.* [44] estimated an effort based on UCP and team productivity using the Treebost model.

The ultimate objective of software effort estimation studies is to minimize an error between actual and estimated effort. The inflexibility of the use case complexity weight level impacts the accuracy of the estimation [10]. Moreover, the original complexity and assigned weight levels might not reflect the actual situations [45]. Fortunately, this was confirmed earlier by Karner [5] that the proposed complexity weight is based on the people's approximation at Objective Systems. Karner [5] also strongly suggests that more data is needed to adjust the model, weights, and parameters. Clearly, the original complexity weight is not the final ideal weighting parameter. In other words, the granularity should be supported to achieve the best weighting scheme and yield the best accuracy in estimation. There are three primary approaches focused on proposing the improvement of use case complexity weight: adding extra complexity weight level, discretizing existing complexity weight level, and calibrating the complexity weight level as described.

Silhavy *et al.* [46] proposed two critical parameters in estimating software effort: actor and use case specification. Meanwhile, technical and environmental complexity factors have been evaluated by Nhung *et al.* [47]. Qi and Boehm [48] measured the project size by automatically calculating the transactions and class diagram using the USIM tool.

Several studies proposed extra complexity weight due to factors affecting the complexity weight level. The type of application and use-case-specific style are two examples that affected the complexity weight level. Hence, the complexity weight should be re-evaluated based on circumstances. Galorath and Evans [49] proposed weight values 10 (simple), 15 (average), and 20 (complex). Periyasamy and Ghode [50] proposed an extra level –"most complex" – to the use case complexity weighting. Manzoor and Wahid [51] proposed the extra level – "critical." Minkiewicz [52] proposed the extra level – "very high" for more

than 14 transactions, and assigned weights are 5, 10, 15, and 20. Whereas Nassif [53] added three more complexity weight levels – 20, 25, and 30.

Fuzzy logic is commonly used for discretizing the existing complexity weight level. It tried to smoothen over the abrupt classification by providing a continuous and gradual classification. Early works in discretizing the current complexity weight level was proposed [7], [6], [10]. Wang *et al.* [7] introduced the complexity weight level from three to five. They proposed the EUCP by integrating the fuzzy set theory and Bayesian Belief Network (BBN). The result showed that EUCP was more effective than UCP for the two projects. Xie *et al.* [6] proposed discretizing complexity weight level that extends the use-case complexity from three to four-level. The result showed that the proposed complexity weight increased by 5.5 (person-hours) with an error rate of 15.45% using four real project data sets. Nassif *et al.* [10] offered ten complexity weight levels according to the number of transactions per use case. The study assumes that the largest use case contains ten transactions, and the complexity factor of the largest use case is fifteen. The result showed that the proposed method improved by 22% in some projects.

The study of UCP calibration is conducted by Nassif *et al.* [8] and followed by Qi *et al.* [45]. Nassif *et al.* [8] introduced a six-level use case complexity weight instead of a three-level as initially proposed by the original UCP. The neural network is used to calibrate the proposed six weight complexity levels. After successfully calibrating the weight, fuzzy logic is applied to smoothen the abrupt change in complexity levels and weights. Unfortunately, this study did not report any experimental results in detail and the model validation. Qi *et al.* [45] explored the Bayesian analysis to calibrate the use case complexity weights. The study collected the use case weight and empirical project data as an input. The a priori use case weights are the source input for calculating a priori mean and variances. At the same time, the empirical project data is the source input for calibrating the use case weight by using multiple linear regression. In the final process, the calibration, mean, and variance results calculate the Bayesian weighted average. Bayesian estimates of the weights are the output of the calculation. The method was evaluated using 105 projects and compared with a priori, original UCP, and regression approaches. The result showed that the Bayesian provides better effort estimation accuracy.

Continuous and gradual classification values provided a broader chance to extend the complexity weight. Moreover, these approaches showed promoting results by expanding the complexity weight in the original UCP method. Despite Nhung *et al.* [41] and Hoc *et al.* [54] conducted the optimization in UCP. However, they do not explore the potential of continuous complexity weight level in terms of optimization function. Metaheuristic-based optimization can smoothen the abrupt change in complexity level because of its ability for continuous function optimization. Thus, this study modified the UCP estimation method by searching for the optimum use case complexity weight parameter to improve estimation performance.

3. Method

3.1. Theoretical Background

3.1.1. Use Case Points

The original UCP framework consists of seven steps.

First, calculating unadjusted actor weighting (UAW) by classifying the actors into three levels of complexity and assigning a weight for each actor based on its complexity, as notated in Eq. (1).

$$UAW = \sum_{i=1}^{3} W_i * A_i \quad (1)$$

where W_i is the weight factor classified as simple for 1, the average for 2, and complex actor for 3. A_i is a number of actors in the use case diagrams based on the same classification as W_i.

Second, calculating unadjusted use case weighting (UUCW) by classifying the use case into three levels of complexity and assigning a weight for each actor based on its level of complexity as formulated in Eq. (2).

$$\mathrm{UUCW} = \sum_{i=1}^{3} \mathrm{W}_i * \mathrm{UC}_i \quad (2)$$

where W_i is a weight factor classified as simple (5), average (10), and complex (15) use case, respectively. UC_i is a number of transactions counted in use case specification diagrams based on the same classification as W_i. Alongside the original weight level, Table 1 presents the complexity weight level derived [9], [10].

Table 1. The original and modified use case complexity weight level

Number of Use Case Transactions	Original weight level	Modified weight level
1-2	5	5.00
3	5	6.45
4	10	7.50
5	10	8.55
6	10	10.00
7	10	11.40
8	15	12.50
9	15	13.60
>10	15	15.00

Third, calculating unadjusted use case points (UUCP) as notated in Eq. (3). UAW in Eq. (1) is added with UUCW in Eq. (2) to obtain UUCP.

$$UUCP = UAW + UUCW \quad (3)$$

Fourth, calculate technical complexity factors (TCF). TCF is formulated in Eq. (4) by grading (G_i) 13 weight factors (W_i) using a score of 0 to 5.

$$TCF = 0.6 + (0.01 * \sum_{i=1}^{13} W_i * G_i) \quad (4)$$

Fifth, calculating environmental complexity factors (ECF) as notated in Eq. (5) by grading (G_i) 8 factors (W_i) using a score of 0 to 5.

$$\mathrm{ECF} = 1.4 + (-0.03 * \sum_{i=1}^{8} \mathrm{W}_i * \mathrm{G}_i) \quad (5)$$

Sixth, calculate the formulae in Eq. (6). UCP is obtained by multiplying UUCP in Eq. (3) by TCF in Eq. (4), and ECF in Eq. (5).

$$\mathrm{UCP} = \mathrm{UUCP} * \mathrm{TCF} * \mathrm{ECF} \quad (6)$$

Seventh, the estimated effort is obtained from UCP in Eq. (6) is multiplied by PF as notated in Eq. (7).

$$Effort = UCP * PF \quad (7)$$

PF is the productivity factor. We can set the number of PF equal to 20 person-hours/UCP, 8.2 person-hours/UCP [55], [56], or using the learning productivity ratio as proposed by [57].

3.1.2. Particle Swarm Optimization

PSO was inspired by the behaviour of bird flocking and fish schooling to find a place with enough food [28]. PSO starts by generating the population according to swarm size parameters randomly. The population itself consists of N particles in which each particle i act as the representation of potential solutions to the given problem. A particle is represented by the vector x_i in the decision space. Each particle has its position (x) and velocity (v). Position means the flying direction, and velocity means the step of the particle.

Optimization is achieved from the cooperation between the particles. The nearest particle to the objective is called the success particle. The successful particles will influence the behaviour of other particles. They will adjust their positions (x_i) toward the global optimum. Two factors affected the position of the particle. First, the best position visited by itself is called personal best ($Pbest_i$); second, the best position visited by the whole particles is called global best ($Gbest_i$).

After the population successfully created, for the subsequent iterations, each particle will apply the following operations:

Update the velocity to define the amount of change applied to the particle described as formulated in Eq. (8).

$$v_i = \omega v_i + C_1 R_1 * (Pbest_i - x_i) + C_2 R_2 * (Gbest_i - x_i) \quad (8)$$

where v_i is the current or initialized velocity by assigning a random number between [0, 1] when the population is generated. C_1 and C_2 represent the constant variables known as cognitive learning and social learning factors. R_1 and R_2 are two random variables in the range of [0, 1]. $Pbest_i$ is the best position visited by particle i. $Gbest_i$ is the best position visited by the overall particles. x_i is the current position of the particle. At the same time, ω is an inertia weight defined as a constant value of 0.9.

Update the position of the particle as notated in Eq. (9), where x_{i+1} is a new position of the particle, x_i is the last position, and v_i is the current velocity of the particle.

$$x_{i+1} = x_i + v_i \quad (9)$$

Each particle will update its personal best solution if $x_i < Pbest_i$ then $Pbest_i = x_i$ and global best will be updated if $x_i < Gbest_i$, then $Gbest_i = x_i$.

3.2. The Proposed Method

This study proposes UCW+PSO for optimizing the use case complexity weight parameters. UCW+PSO is slightly different from UCP. UCP is a pure estimation method without any modification, whereas UCW+PSO is a UCP estimation method integrating PSO. PSO is applied to obtain an optimized weight parameter based on the modified weight classification level submitted by Nassif *et al.* [10] and Hariyanto and Wahono [9], as shown in Table 1. UCW+PSO consists of two-phase which are described as follows.

The first phase is generating the initial population. The population was developed using a random number from Simple, Average, and Complex complexity weights. The number of particles, fitness value,

C_1, C_2, R_1, R_2, maximum iterations, and inertia weight is applied together with random complexity weight. All the parameters and variables are then calculated using UCP methods. This calculation is called effort estimation and will be used in the first iteration of the second phase.

The second phase is calculating initial velocity and positions. The first iteration, initial velocity, and positions for all particles are calculated using Eq. (8) and Eq. (9). Positions x_i itself are acted as same as the complexity weight in the use case. The UCP method was then calculated again to get new estimation values. From this first iteration, the AE value is compared with the fitness value. Two conditions make the iteration stop. The first condition is if the AE value is less than or equal to the fitness value, or the second condition is reached maximum iteration. When the first condition is fulfilled, we get the best optimized UCP and jump to the next iteration. If the AE value is greater than the fitness value, then the best positions and global best value are updated, followed by generating new velocity and positions. The second phase is repeated until all data points in the data set are used as test data. When the second condition is fulfilled, the iteration will stop, and the minimum AE will be assigned as the best optimized UCP. The overall AE will be evaluated using the evaluation metric in Eq. (10) to Eq. (14).

The general description of PSO is pointed out by Algorithm 1 (Fig. 1) whereas our proposed model is shown in Fig 2. It shows the improved Use Case Point estimation model integrating PSO as the weight complexity optimizer. The light blue shape of Weight Optimization denoted the original contribution of this study.

Algorithm 1. Particle Swarm Optimization (PSO).

(1) **Input**: Dataset *X*, Parameters settings in Section 5.4
(2) **Output**: Optimized solutions
(3) **for** *each project* in *X* **do**
(4) generate initial population
(5) **while** (*Gbest* > *stopping value*) **or** (T_{max} > 0) **do**
(6) **for** *i* = 1, 2, 3, …, T_{max} **do**
(7) update velocity using Equation (8)
(8) update positions using Equation (9)
(9) calculate effort estimation
(10) updated particles
(11) **end for**
(12) *Gbest* ← min(*Pbests*)
(13) **if** *Gbest* > *stopping value* **then**
(14) *temps*[] ← *Gbest*
(15) **else**
(16) *Gbest*
(17) **end if**
(18) increment++
(19) **end while**
(20) **if** *temps* ***are not empty*** **then**
(21) *Gbest* ← min(*temps*)
(22) **end if**
(23) **end for**

Fig. 1. Algorithm of Particle Swarm Optimization (PSO)

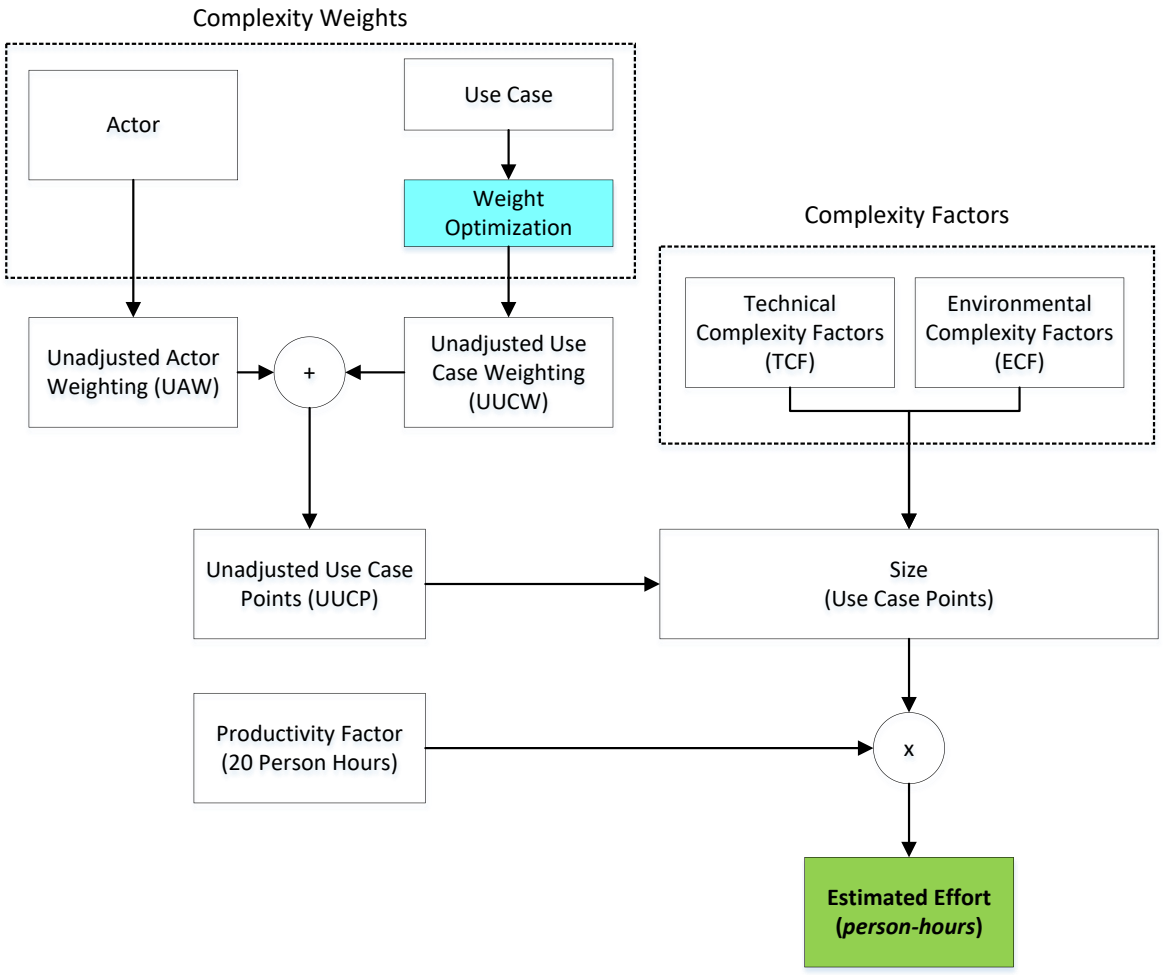

Fig. 2. The proposed model

3.3. Experimental Design

Software effort estimation aims to minimize the estimation error between actual and estimated effort, as notated in Eq. (10). Therefore, the experimental procedure was carried out based on the proposed method described in Section 4. The experimental design consists of four stages: project data set description, data preprocessing, model validation, and evaluation.

3.3.1. Project dataset description

This study employed real-life historical project data set from three software houses. The project data set contains seventy-one projects gathered Silhavy *et al.* [58]. The data set consists of several problem domains such as insurance, government, banking, and others. The data set contains the following thirty-eight (38) variables: Simple Actors, Average Actors, Complex Actors, Simple UC, Complex UC, T1-T13, Env1-Env8, Sector, Language, Methodology, Application Type, UAW, UUCW, TCF, ECF, Real_P20, Real_Effort_Person_Hours, and Data Donator. We employed seven (7) variables for this study and eliminated the rest. The final utilized variables are Simple UC, Average UC, Complex UC, UAW, TCF, ECF, and Actual Effort. We choose these variables because they are the primary variables

of the UCP estimation method, as formulated in Eq. (1)-(7). Most of the projects were written using Java and C# language. The summary statistics for the project data set are demonstrated in Table 2.

Table 2. Descriptive statistics of the project data set (N = 71)

Variable	Mean	StDev	Skewness	Kurtosis	Max	Min
SimpleUC	2.7	2.9	3.29	17.658	20	0.00
AverageUC	15.84	5.37	0.296	0.140	30	3.00
ComplexUC	14.29	4.45	0.191	-0.290	27	5.00
UAW	10.49	5.01	0.803	-1.264	19	6.00
TCF	0.92	0.114	-0.269	-1.019	1.12	0.71
ECF	0.86	0.117	-0.556	0.861	1.09	0.51
Actual Effort	6558.72	664.24	0.574	-0.922	7970	5775

From Table 2, we can observe that AverageUC, ComplexUC, and TCF variables have a normal distribution with skewness very close to zero. In comparison, a simple use case variable has not had a normal distribution with skewness very far away from zero. Interestingly, the complex use case variable tends to widen with a kurtosis value of -0.290, and a simple use case variable formed a leptokurtosis curve with a kurtosis value of 17.658. If a kurtosis value is less than three, the variable is less outlier-prone. Only simple use case variable has a kurtosis value greater than three, which suggests AverageUC, ComplexUC, UAW, TCF, ECF, and Actual Effort variables are outlier-prone. We can also find that most of the project use case is average. It pointed out that the mean and maximum value of the AverageUC variable is more extensive than SimpleUC and ComplexUC.

3.3.2. Model Validation

Model validation is the process where the trained model is evaluated with a testing data set to foresee how good the performance of the estimation method [59]. The testing data set is a separate portion of the same data set from which the training set is derived. This study uses leave-one-out cross-validation (LOOCV) to validate the proposed model [60]. LOOCV takes each project as a test set, while the rest is used as the training set. Each test data entered the prediction model to obtain the predicted effort. The accuracy would be calculated each time the model successfully predicted the effort. The difference between LOOCV and other n-fold cross-validation techniques is that LOOCV uses deterministic procedures that can be easily applied in other studies with various datasets. Moreover, n-fold methods use random selection to build their train and test sets, which introduces the problem of conclusion instability [61]. LOOCV was chosen because it produces lower estimation bias and higher variance values [43], removing the conclusion instability, especially for a relatively small data set [62]. Furthermore, LOOCV ensures that any prediction model is constructed from the same set of training data. The performance of the proposed model is then compared with the existing model, which is polynomial regression [58], multiple linear regression (MLR) [63], WGRA+PSO with setting K = 2 and Mean [13], FUCP [9], [10], and original UCP model proposed by Karner [5].

3.3.3. Model Evaluation

The evaluation of estimation models should be evaluated using reliable accuracy measurement techniques. The measurement results must be unbiased and not produce an asymmetric error distribution [43]. The first evaluation measurement is an absolute error (AE), as formulated in Eq. (10).

$$AE = |y_i - \hat{y}_i| \tag{10}$$

where AE is the estimation or prediction error, y_i is the i^{th} the actual value of the variable being estimated, and $\hat{y}_i$ is the i^{th} estimated value. AE is the fundamental metric because by using this metric, we can measure other metrics such as MAE, MBRE, MIBRE, standardized accuracy (SA), and effect size (Δ). The second evaluation measurement is mean absolute error (MAE), as notated in Eq. (11).

$$MAE = \frac{1}{n}\sum_{i=1}^{n} AE_i \tag{11}$$

where n is the number of projects in the data set and AE_i is the i^{th} absolute error value. The third evaluation measurement is the mean balanced residual error (MBRE) formulated in Eq. (12).

$$MBRE = \frac{1}{n}\sum_{i=1}^{n} \frac{AE_i}{\min(y_i, \hat{y}_i)} \tag{12}$$

where $min(y_i, \hat{y}_i)$ is the minimum value between y_i and $\hat{y}_i$. The fourth evaluation measurement is the mean inverted balanced residual error (MIBRE) formulated in Eq. (13).

$$MIBRE = \frac{1}{n}\sum_{i=1}^{n} \frac{AE_i}{\max(y_i, \hat{y}_i)} \tag{13}$$

where $max(y_i, \hat{y}_i)$ is the maximum value between y_i and $\hat{y}_i$.

MAE, MBRE, and MIBRE accuracy measurements are used because they behave differently from each other. They can be effectively evaluated how well a model performs. The superior model is the one with the minimum value.

Besides the four metrics, we also used the evaluation framework proposed by [64]. The framework consists of two metrics: standardized accuracy (SA) and effect size (Δ) as notated in Eq. (14) and Eq. (15). SA is an accuracy measurement used to evaluate that the estimation model P_i produces meaningful estimation. The value of SA must be better than the baseline estimation model derived from random guessing (P_0).

$$SA_{P_i} = 1 - \left(\frac{MAE_{P_i}}{\overline{MAE}_{P_0}}\right) \text{ x } 100 \tag{14}$$

where MAE_{P_i} is the mean value produced by the proposed or existing estimated model (P_i). $\overline{MAE}_{P_0}$ is the mean value provided by a large number, typically 1000 runs of the random guessing model (P_0). The random guessing model (P_0) is defined as estimating a $\hat{y}$ for the target case t by randomly sampling (with equal probability) over all the remaining $n - 1$ cases and takes$\hat{y}_t = y_r$ where r is drawn randomly from $1 \ldots n \wedge r \neq t$. SA is interested in one direction that how much better P_i is than P_0. If P_i value is greater than P_0, then we can interpret that P_i is predicting, not guessing, and P_i is generating meaningful estimation in this particular study. The SA is discouraging with a value close to zero or even negative. The larger the value yielded by the standardized accuracy metric shows a good estimation model.

Effect size is a metric to interpret the practical or real-world significance of the result [65]. Effect size is used to ensure the results by P_i does not produce by chance as formulated in Eq. (15).

$$\Delta = \frac{MAE_{P_i} - \overline{MAE}_{P_0}}{S_{P_0}} \tag{15}$$

where S_{P_0} is the standard deviation of the random guessing model (P_0). There are three margins we can use to interpret the effect size improvement over the baseline model, small ($\Delta \approx 0.2$), medium ($\Delta \approx 0.5$), and large ($\Delta \approx 0.8$) [64], [66]. For example, if an effect size does not reach a small effect size (e.g., $\Delta = 0.1777$), we can interpret the result is not attractive. In other words, the improvement of the model accuracy is not contributed a good effect in a practical matter. The larger value yielded by the effect size metric shows a good estimation model.

Finally, the significance test was carried out using the Wilcoxon sum ranked at the 95% confidence level. The null hypothesis H_0 is tested based on the absolute errors (AE) of data samples. It then used the p-value to test the hypothesis to decide whether the null hypothesis was accepted or rejected. Hence, there are five hypotheses proposed in this study.

$H_{0(1)}$: There is no difference in absolute error between the proposed and random guessing model.

$H_{0(2)}$: There is no difference in absolute error between the proposed and MLR model

$H_{0(3)}$: There is no difference in absolute error between the proposed and FUCP model

$H_{0(4)}$: There is no difference in absolute error between the proposed and Karner model.

$H_{0(5)}$: There is no difference in absolute error between the proposed and WGRA+PSO model.

3.3.4. Parameter settings and constraints

The objective and fitness functions are derived from Eq. (7) and Eq. (10), respectively. There are three-dimensional variables: simple UC (x_1), average UC (x_2), and complex UC (x_3). The range for these variables then set as x_1 = 5.00 to 7.49, x_2 = 7.50 to 12.49, and x_3 = 12.50 to 15.00, based on the modified use case weight level (see Table 1) proposed [9], [10].

The proposed model was developed using PHP 7.2.28, and the parameters of the PSO are set as particles = 70, C_1 = 2.8, C_2 = 1.3, maximum iterations = 500, fitness value = 50, and inertia factor (ω) = 0.9, respectively. The parameter of C_1, C_2, and maximum iterations was adopted from [13]. The number of particles or swarm size is adopted from [67] as it recommended the size between 70 – 500. For ω itself is linearly decreasing inertia-weight (LDW) [68].

4. Results and Discussion

This section presented empirical results obtained from the experimental, model validation, and evaluation. A reliable and meaningful estimation model is indicated by which received a larger SA value. The estimation model is unlikely to have been generated by chance if the model has a larger effect size value. Thus, we consider three research questions (RQ): RQ1: How much better P_i is over random guessing (P_0)? Q2: How much better are Polynomial, MLR, FUCP, WGRA+PSO, and UCW+PSO than the Karner model? and RQ3: How much better is UCW+PSO than Polynomial, MLR, FUCP, and WGRA+PSO models?

4.1. RQ1: The performance of P_i versus P_0 model

All six models were validated using SA and effect size (Δ) while random guessing as to the baseline model. Table 3 showed that Polynomial, WGRA+PSO, and UCW+PSO obtained better SA values than random guessing, whereas our proposed model (UCW+PSO) yielded the largest value. It is immediately apparent that these models were actually predicting, not guessing since they delivered considerably better accuracy levels than random guessing (P_0). Thus, these models generated meaningful predictions in this

particular study. Surprisingly, FUCP and Karner's model yielded the worst results signed by the large negative values. It is immediately apparent that these two models were not actually predicting since they delivered worse accuracy levels than random guessing (P_0). Thus, these two models did not generate meaningful predictions in this study.

Conversely, all six models yielded considerably better effect size value than random guessing (P_0). Four models produced enormous effect size improvement, and two models (Polynomial and MLR) were in medium effect size improvement. One can find that Δ of 1.499 (Karner), 1.340 (FUCP), 1.083 (UCW+PSO), and 1.059 (WGRA+PSO) were regarded as considerable effect size improvement over random guessing, which was worthwhile at the margin. Hence, we can be confident that these six models were not a chance outcome because the significance test rejected all six null hypotheses ($p < 0.05$). However, due to the negative SA value obtained by FUCP and Karner models, it is immediately clear that these two models were discouraging and perturbing in this study.

Table 3. The results of SA, Δ, and Sig. considering random guessing as to the baseline model.

Method	SA (%)	Δ	Sig.
Polynomial	66.782***	0.737	0.00 ($p < 0.05$)
MLR	49.159	0.573	0.00 ($p < 0.05$)
FUCP	-114.949	1.340**	0.00 ($p < 0.05$)
Karner	-128.541	1.499*	0.00 ($p < 0.05$)
WGRA+PSO	96.021**	1.059	0.00 ($p < 0.05$)
UCW+PSO	98.161*	1.083***	0.00 ($p < 0.05$)

*First best model ** Second best model *** Third best model

To confirm the negative results yielded by FUCP and Karner, we added more historical project data sets from [39], [56], [69], [70]. The new data sets represented various project domains such as educational information systems, social media, ERP, CMS, and CRM. Four unique data sets, D1, D2, D3, and D4, consist of 14, 10, 7, and 9 historical projects, respectively. We merged these four data sets and formed a new data set, MD1. Finally, we joined MD1 with the primary data set used in this study and created a new data set, MD2. Table 4 showed that the D2 and MD2 yielded positive, more considerable SA results. It is immediately apparent that these models were actually predicting, not guessing, since they delivered considerably better accuracy levels than random guessing (P_0). Moreover, in terms of Δ, these results were regarded as medium effect size improvement over random guessing, in other words, worthwhile at the margin. Hence, we can be confident that these two models were not a chance outcome because the significance test rejected both null hypotheses ($p < 0.05$).

Table 4. The accuracy results of UCP, considering random guessing as to the baseline model.

Data set	*n*	SA (%)	Δ	Sig.
D1	10	-41.433	0.597**	0.336 ($p > 0.05$)
D2	14	28.180**	0.474	0.040 ($p < 0.05$)
D3	8	-598.144	8.865*	0.000 ($p < 0.05$)
D4	7	-27.838	0.379	0.647 ($p > 0.05$)
MD1	39	-15.2***	0.204	0.903 ($p > 0.05$)
MD2	109	37.407*	0.485***	0.000 ($p < 0.05$)

n = number of projects *First best **Second best ***Third best

4.2. RQ2: The performance of Polynomial, MLR, FUCP, WGRA+PSO, and UCW+PSO versus Karner model

Karner validated Polynomial, MLR, FUCP, WGRA+PSO, and UCW+PSO as the baseline model. The Karner model was appointed because the model was the center of all UCP-based effort estimation studies. Most of the proposed models compared their results with the Karner model, e.g., [43], [58], [71], [72]. From Table 5, we can observe that all models yielded better SA value over the Karner model, while our proposed method obtained the largest value. It is immediately apparent that these five models were actually predicting, not guessing, since they yielded considerably better accuracy levels than Karner's model.

WGRA+PSO and UCW+PSO yielded more considerable effect size improvement over the Karner model. At the same time, the medium effect size is obtained by Polynomial and MLR. In other words, WGRA+PSO, UCW+PSO, Polynomial, and MLR were worthwhile at the margin. Hence, we can be confident that these four models were not a chance outcome because the significance test rejected all four null hypotheses. In contrast, FUCP did not even reach a small effect size (Δ = 0.054), which suggests that the impact was not significant ($p > 0.05$) and not attractive.

Table 5. The results of SA and Δ considering Karner as the baseline model.

Method	SA (%)	Δ	Sig.
Polynomial	86.802***	0.789***	0.000 ($p < 0.05$)
MLR	77.754	0.707	0.000 ($p < 0.05$)
FUCP	5.9475	0.054	0.614 ($p > 0.05$)
WGRA+PSO	98.419**	0.895**	0.000 ($p < 0.05$)
UCW+PSO	99.269*	0.902*	0.000 ($p < 0.05$)

*First best **Second best ***Third best

4.3. RQ3: The accuracy performance of UCW+PSO versus Polynomial, MLR, FUCP, and WGRA+PSO models

We validated our proposed model by assigning Polynomial, MLR, FUCP, and WGRA+PSO as the benchmark model. We observed that our proposed model yielded better SA and Δ over the four models (see Table 6).

Table 6. The results of SA and Δ considering MLR, FUCP, Polynomial, and WGRA+PSO as the baseline model.

Method	SA (%)	Δ	Sig.
Polynomial as baseline	94.465	0.409***	0.00 ($p < 0.05$)
MLR as baseline	96.716**	0.942**	0.00 ($p < 0.05$)
FUCP as baseline	99.223*	1.283*	0.00 ($p < 0.05$)
WGRA+PSO as baseline	53.790***	0.024	0.00 ($p > 0.05$)

*First best **Second best ***Third best

The accuracy results obtained concerning MAE, MBRE, and MIBRE are presented in Table 7. The results showed that the proposed estimation model performed better than three other models, suggesting significant improvements over these three different models.

The larger SA and Δ value obtained when FUCP and MLR as the baseline models suggest that the proposed model was actually estimated. The result was not by chance and significantly improved the

largest effect size over these two models ($p < 0.05$). In comparison, medium ($\Delta = 0.409$) and very small effect size ($\Delta = 0.024$) yielded by UCW+PSO over Polynomial and WGRA+PSO, which indicated that the result was significant ($p < 0.05$) but not impressive. However, UCW+PSO obtained a remarkable SA value over Polynomial and WGRA+PSO, suggesting the proposed model outperformed Polynomial and WGRA+PSO. Hence, we concluded that our proposed method was actually estimating, generating meaningful estimation by significance in statistics and practical (e.g., MLR and FUCP) over four models in this study.

Table 7. MAE, MBRE, and MIBRE results.

Method	MAE	MBRE	MIBRE
Polynomial	240.19	0.038***	0.035***
MLR	404.85***	0.064	0.058
FUCP	1712	0.589	0.260
Karner	1820	0.378	0.240
WGRA+PSO	28.77**	0.00422**	0.00419**
UCW+PSO	13.29*	0.00210*	0.00212*

*First best **Second best ***Third best

Fig. 3 illustrates the plot of all six models' actual and estimated effort values. *Y*-axis is the project data set, and the *x*-axis is the effort value. The solid green line was representative of the actual effort value, and the dashed red line was representative of the estimated effort value. All models have tried to produce estimated value as closely as possible to the actual effort value. From these figures, we found that when the dashed red line was getting closer and reached the same position over the solid green line, that means the model was estimated accurately. As we can find, Fig. 3e and 3f have the closest between the two-line, suggesting that the two models were estimated with the best accuracy. However, in Fig. 3e, we found that there were still some red dash lines that were separated slightly by the whitespace, suggesting a small gap between the actual and estimated line, whereas, in Fig. 3f, we found the gap itself almost did not exist. Thus, it was immediately apparent that the proposed model showed the best accurate estimation of these five models.

Table 8 shows the results of the Friedman test in ranked using SPSS tools. The Friedman test was conducted for multiple comparisons to infer the difference among various estimation models. The estimation models deducted a sample of AE found for the given data set. Friedman's test considered a null hypothesis that all estimations are equivalent (H_0). From Table 8, we can observe that all models were significant differences ($p < 0.05$), and UCW+PSO was the best performing estimation model.

Table 8. Represents the ranks evaluated through the Friedman test ($p < 0.05$)

Models	Friedman ranks
Polynomial	3.39
MLR	3.93
FUCP	5.19
Karner	5.26
WGRA+PSO	2.01
UCW+PSO	1.23

All results yielded by the proposed model confirmed that the chosen number of particles or swarm sizes 70-500 particles has contributed to better performance [67]. This number of particles is different from the initial suggestion from [73] and other recommended population sizes of 20-50. Parameter settings also influenced the efficiency and effectiveness of the search [74]. For example, when R_1 and R_2 values are high, the particles are improved the current solution by moving toward the best and global positions, whereas the inertia weight parameter (ω) controls the global and local search process to avoid premature convergence and the poor global search ability. These results confirmed that optimization could be applied right across the spectrum of software engineering, especially for effort estimation studies [75]. This also implies that we can establish the automation solution to software engineering problems using search-based optimization algorithms.

Fig. 3. Estimation results for (a) polynomial; (b) MLR; (c) FUCP; (d) Karner; (e) WGRA+PSO; (f) UCW+PSO model

5. Conclusion

This paper presented a metaheuristic approach to address the abrupt classification issues in the use case complexity weight level. We proposed UCW+PSO to search for an optimal value of complexity weight level to obtain the minimum absolute error (MAE). Thus, the objective of this study is generally to improve the accuracy performance of the UCP-based software development effort estimation. Our experimental studies demonstrated that the version of the proposed UCW+PSO model was promising and showed significant improvements over other baseline models. The best performance evaluation for standardized accuracy, effect size, MAE, MBRE, and MIBRE are 99.223, 1.283, 13.29, 0.0021, and 0.00212, respectively. Interesting observation of UCW+PSO is the choice of the number of particles in the parameter settings, which contributed to the performance. This study focused on a fixed value of the control parameter C_1, C_2, and ω. Some objective functions are susceptible to the inappropriate selection of control parameters. Hence, further research is needed to determine whether the objective function in the proposed method is sensitive or not by applying adaptive inertia weight and automating the parameter settings by using a GA. Furthermore, more data is needed to validate the negative results obtained by Karner and FUCP model, as described in Section 6.

Based on these conclusions, practitioners whose organization already has the historical project data set should consider UCW+PSO, WGRA+PSO, or MLR model to estimate a new project. Because using their historical project data set will help make better predictions. Software organizations with no historical project data set would be better using cross-company data set under some circumstances.

Declarations

Author contribution. All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding statement. None of the authors have received any funding or grants from any institution or funding body for the research.

Conflict of interest. The authors declare no conflict of interest.

Additional information. No additional information is available for this paper.

References

[1] M. Choetkiertikul, H. K. Dam, T. Tran, A. Ghose, and J. Grundy, "Predicting Delivery Capability in Iterative Software Development," *IEEE Trans. Softw. Eng.*, vol. 44, no. 6, pp. 551–573, Jun. 2018, doi: 10.1109/TSE.2017.2693989.

[2] M. Bloch, S. Blumberg, and J. Laartz, "Delivering large-scale IT projects on time, on budget, and on value," *McKinsey Digital*, no. 5. pp. 1–7, 2012. Available at: Google Scholar

[3] A. Kaur and K. Kaur, "A COSMIC function points based test effort estimation model for mobile applications," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 3, pp. 946–963, Mar. 2022, doi: 10.1016/j.jksuci.2019.03.001.

[4] K. Rak, Ž. Car, and I. Lovrek, "Effort estimation model for software development projects based on use case reuse," *J. Softw. Evol. Process*, vol. 31, no. 2, pp. 1–17, Feb. 2019, doi: 10.1002/smr.2119.

[5] G. Karner, "Resource Estimation for Objectory Projects." pp. 1–9, 1993. Available at: Google Scholar

[6] Y. Xie, J. Guo, and A. Shen, "Use Case Points Method of Software Size Measurement Based on Fuzzy Inference," in *Proceedings of the 4th International Conference on Computer Engineering and Networks*, vol. 355, W. E. Wong, Ed. Cham: Springer International Publishing, 2015, pp. 11–18. doi: 10.1007/978-3-319-11104-9_2

[7] F. Wang, X. Yang, X. Zhu, and L. Chen, "Extended Use Case Points Method for Software Cost Estimation," in *2009 International Conference on Computational Intelligence and Software Engineering*, 2009, pp. 1–5, doi: 10.1109/CISE.2009.5364706.

[8] A. B. Nassif, L. F. Capretz, and D. Ho, "Calibrating use case points," in *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*, 2014, pp. 612–613, doi: 10.1145/2591062.2591141.

[9] M. Hariyanto and R. S. Wahono, "Estimasi Proyek Pengembangan Perangkat Lunak Dengan Fuzzy Use Case Points," *J. Softw. Eng.*, vol. 1, no. 1, pp. 54–63, 2015. Available at: Google Scholar

[10] A. B. Nassif, L. F. Capretz, and D. Ho, "Enhancing Use Case Points Estimation Method Using Soft Computing Techniques," *J. Glob. Res. Comput. Sci.*, vol. 1, no. 4, pp. 12–21, Dec. 2016. Available at: Google Scholar

[11] V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "A PSO-based model to increase the accuracy of software development effort estimation," *Softw. Qual. J.*, vol. 21, no. 3, pp. 501–526, Sep. 2013, doi: 10.1007/s11219-012-9183-x.

[12] M. Azzeh, A. B. Nassif, S. Banitaan, and F. Almasalha, "Pareto efficient multi-objective optimization for local tuning of analogy-based estimation," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2241–2265, Nov. 2016, doi: 10.1007/s00521-015-2004-y.

[13] D. Wu, J. Li, and C. Bao, "Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation," *Soft Comput.*, vol. 22, no. 16, pp. 5299–5310, Aug. 2018, doi: 10.1007/s00500-017-2985-9.

[14] L. Brezočnik, I. Fister, and V. Podgorelec, "Solving Agile Software Development Problems with Swarm Intelligence Algorithms," in *Lecture Notes in Networks and Systems*, vol. 76, E. Karabegovi, Ed. Sarajevo, Bosnia and Herzegovina: Springer, Cham, 2020, pp. 298–309. doi: 10.1007/978-3-030-18072-0_35

[15] F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro, "Genetic Programming for Effort Estimation: An Analysis of the Impact of Different Fitness Functions," in *2nd International Symposium on Search Based Software Engineering*, 2010, no. 25, pp. 89–98, doi: 10.1109/SSBSE.2010.20.

[16] J. Murillo-Morera, C. Quesada-López, C. Castro-Herrera, and M. Jenkins, "A genetic algorithm based framework for software effort prediction," *J. Softw. Eng. Res. Dev.*, vol. 5, no. 1, pp. 1–33, Dec. 2017, doi: 10.1186/s40411-017-0037-x.

[17] Z. Shahpar, V. K. Bardsiri, and A. K. Bardsiri, "Polynomial analogy-based software development effort estimation using combined particle swarm optimization and simulated annealing," *Concurr. Comput. Pract. Exp.*, vol. 33, no. 20, p. e6358, Oct. 2021, doi: 10.1002/cpe.6358.

[18] P. Phannachitta, "On an optimal analogy-based software effort estimation," *Inf. Softw. Technol.*, vol. 125, no. April, p. 106330, Sep. 2020, doi: 10.1016/j.infsof.2020.106330.

[19] Z. Shahpar, V. Khatibi, and A. Khatibi Bardsiri, "Hybrid PSO-SA Approach for Feature Weighting in Analogy-Based Software Project Effort Estimation," *J. AI Data Min.*, vol. 9, no. 3, pp. 329–340, 2021, doi: 10.22044/jadm.2021.10119.2152.

[20] Z. Shahpar, V. K. Bardsiri, and A. K. Bardsiri, "An evolutionary ensemble analogy-based software effort estimation," *Softw. Pract. Exp.*, vol. 52, no. 4, pp. 929–946, Apr. 2022, doi: 10.1002/spe.3040.

[21] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation," *Swarm Evol. Comput.*, vol. 38, pp. 158–172, Feb. 2018, doi: 10.1016/j.swevo.2017.07.009.

[22] N. Ghatasheh, H. Faris, I. Aljarah, and R. M. H. Al-Sayyed, "Optimizing Software Effort Estimation Models Using Firefly Algorithm," *J. Softw. Eng. Appl.*, vol. 08, no. 03, pp. 133–142, 2015, doi: 10.4236/jsea.2015.83014.

[23] V. Resmi, S. Vijayalakshmi, and R. S. Chandrabose, "An effective software project effort estimation system using optimal firefly algorithm," *Cluster Comput.*, vol. 22, no. S5, pp. 11329–11338, Sep. 2019, doi: 10.1007/s10586-017-1388-0.

[24] K. Langsari and R. Sarno, "Optimizing effort and time parameters of COCOMO II estimation using fuzzy multi-objective PSO," in *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2017, vol. 2017-Decem, no. September, pp. 1–6, doi: 10.1109/EECSI.2017.8239157.

[25] N. A. Zakaria, A. R. Ismail, N. Z. Abidin, N. H. M. Khalid, and A. Y. Ali, "Optimized COCOMO parameters using hybrid particle swarm optimization," *Int. J. Adv. Intell. Informatics*, vol. 7, no. 2, pp. 177–187, Apr. 2021, doi: 10.26555/ijain.v7i2.583.

[26] M. D. Alanis-Tamez, C. López-Martín, and Y. Villuendas-Rey, "Particle Swarm Optimization for Predicting the Development Effort of Software Projects," *Mathematics*, vol. 8, no. 10, pp. 1–21, Oct. 2020, doi: 10.3390/math8101819.

[27] S. Chhabra and H. Singh, "Optimizing Design of Fuzzy Model for Software Cost Estimation Using Particle Swarm Optimization Algorithm," *Int. J. Comput. Intell. Appl.*, vol. 19, no. 01, pp. 1–16, Mar. 2020, doi: 10.1142/S1469026820500054.

[28] M. Khazaiepoor, A. Khatibi Bardsiri, and F. Keynia, "A Hybrid Approach for Software Development Effort Estimation using Neural networks, Genetic Algorithm, Multiple Linear Regression and Imperialist Competitive Algorithm," *Int. J. Nonlinear Anal. Appl*, vol. 11, no. 1, pp. 207–224, 2020, doi: 10.22075/ijnaa.2020.4259.

[29] K. E. Rao and G. A. Rao, "Ensemble learning with recursive feature elimination integrated software effort estimation: a novel approach," *Evol. Intell.*, vol. 14, no. 1, pp. 151–162, Mar. 2021, doi: 10.1007/s12065-020-00360-5.

[30] S. P. Singh, G. Dhiman, P. Tiwari, and R. H. Jhaveri, "A soft computing based multi-objective optimization approach for automatic prediction of software cost models," *Appl. Soft Comput.*, vol. 113, p. 107981, Dec. 2021, doi: 10.1016/j.asoc.2021.107981.

[31] A. Kaushik, S. Verma, H. J. Singh, and G. Chhabra, "Software cost optimization integrating fuzzy system and COA-Cuckoo optimization algorithm," *Int. J. Syst. Assur. Eng. Manag.*, vol. 8, no. S2, pp. 1461–1471, Nov. 2017, doi: 10.1007/s13198-017-0615-7.

[32] S. Kumari and S. Pushkar, "Cuckoo search based hybrid models for improving the accuracy of software effort estimation," *Microsyst. Technol.*, vol. 24, no. 12, pp. 4767–4774, Dec. 2018, doi: 10.1007/s00542-018-3871-9.

[33] M. R. Braz and S. R. Vergilio, "Using fuzzy theory for effort estimation of object-oriented software," in *16th IEEE International Conference on Tools with Artificial Intelligence*, 2004, no. Ictai, pp. 196–201, doi: 10.1109/ICTAI.2004.119.

[34] A. Ardiansyah, R. Ferdiana, and A. E. Permanasari, "MUCPSO: A Modified Chaotic Particle Swarm Optimization with Uniform Initialization for Optimizing Software Effort Estimation," *Appl. Sci.*, vol. 12, no. 3, p. 1081, Jan. 2022, doi: 10.3390/app12031081.

[35] G. Robiolo and R. Orosco, "Employing use cases to early estimate effort with simpler metrics," *Innov. Syst. Softw. Eng.*, vol. 4, no. 1, pp. 31–43, Apr. 2008, doi: 10.1007/s11334-007-0043-y.

[36] 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, 2005. ICSE 2005.*, 2005, pp. 303–311, doi: 10.1109/ICSE.2005.1553573.

[37] B. Anda, H. Dreiem, D. I. K. Sjøberg, and M. Jørgensen, "Estimating Software Development Effort Based on Use Cases — Experiences from Industry," in *International Conference on the Unified Modeling Language*, 2001, pp. 487–502. doi: 10.1007/3-540-45441-1_35

[38] B. Anda, E. Angelvik, and K. Ribu, "Improving Estimation Practices by Applying Use Case Models," in *Product Focused Software Process Improvement*, vol. 2559, no. 1325, 2002, pp. 383–397. doi: 10.1007/3-540-36209-6_32

[39] M. Ochodek, J. Nawrocki, and K. Kwarciak, "Simplifying effort estimation based on Use Case Points," *Inf. Softw. Technol.*, vol. 53, no. 3, pp. 200–213, Mar. 2011, doi: 10.1016/j.infsof.2010.10.005.

[40] M. Ochodek, B. Alchimowicz, J. Jurkiewicz, and J. Nawrocki, "Improving the reliability of transaction identification in use cases," *Inf. Softw. Technol.*, vol. 53, no. 8, pp. 885–897, Aug. 2011, doi: 10.1016/j.infsof.2011.02.004.

[41] H. L. T. K. Nhung, V. Van 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, 2022, doi: 10.1109/ACCESS.2021.3139183.

[42] A. B. Nassif, "Towards an Early Software Estimation Using Log-Linear Regression and a Multilayer Perceptron Model," *J. Syst. Softw.*, vol. 86, no. 1, pp. 144–160, 2013. doi: 10.1016/j.jss.2012.07.050

[43] M. Azzeh and A. B. Nassif, "A hybrid model for estimating software project effort from Use Case Points," *Appl. Soft Comput.*, vol. 49, pp. 981–989, Dec. 2016, doi: 10.1016/j.asoc.2016.05.008.

[44] A. B. Nassif, L. F. Capretz, D. Ho, and M. Azzeh, "A Treeboost Model for Software Effort Estimation Based on Use Case Points," in *2012 11th International Conference on Machine Learning and Applications*, 2012, no. December 2012, pp. 314–319, doi: 10.1109/ICMLA.2012.155.

[45] K. Qi, A. Hira, E. Venson, and B. W. Boehm, "Calibrating use case points using bayesian analysis," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2018, pp. 1–10, doi: 10.1145/3239235.3239236.

[46] R. Silhavy, P. Silhavy, and Z. Prokopova, "Using Actors and Use Cases for Software Size Estimation," *Electronics*, vol. 10, no. 5, pp. 1–20, Mar. 2021, doi: 10.3390/electronics10050592.

[47] H. Le Thi Kim Nhung, 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*, vol. 1294, 2020, pp. 757–768. doi: 10.1007/978-3-030-63322-6_64

[48] K. Qi and B. W. Boehm, "Detailed use case points (DUCPs)," in *Proceedings of the 10th International Workshop on Modelling in Software Engineering - MiSE '18*, 2018, pp. 17–24, doi: 10.1145/3193954.3193955.

[49] D. D. Galorath and M. W. Evans, *Software Sizing, Estimation, and Risk Management*. Boca Raton: Auerbach Publications, 2006. doi: 10.1201/9781420013122

[50] 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, pp. 1–5, doi: 10.1109/CISE.2009.5364515.

[51] M. Manzoor and A. Wahid, "Revised Use Case Point (Re-UCP) Model for Software Effort Estimation," *Int. J. Adv. Comput. Sci. Appl.*, vol. 6, no. 3, pp. 65–71, 2015, doi: 10.14569/IJACSA.2015.060310.

[52] A. Minkiewicz, "Use Case Sizing," in *International Forum on COCOMO and Software Cost Modelin*, 2004. Available at: Google Scholar

[53] A. B. Nassif, "Software Size and Effort Estimation from Use Case Diagrams Using Regression and Soft Computing Models," The University of Western Ontario, 2012. Available at: Google Scholar

[54] H. T. Hoc, V. Van Hai, and H. Le Thi Kim Nhung, "AdamOptimizer for the Optimisation of Use Case Points Estimation," in *Advances in Intelligent Systems and Computing*, vol. 1294, 2020, pp. 747–756. doi: 10.1007/978-3-030-63322-6_63

[55] Sholiq, A. P. Subriadi, F. A. Muqtadiroh, and R. S. Dewi, "A model of owner estimate cost for software development project in Indonesia," *J. Softw. Evol. Process*, vol. 31, no. 10, p. e2175, Oct. 2019, doi: 10.1002/smr.2175.

[56] Subriadi, A. Pribadi, and P. A. Ningrum, "Critical Review of the Effort Rate Value in Use Case Point Method for Estimating Software Development Effort," vol. 59, no. 3, pp. 735–744, 2014. Available at: Google Scholar

[57] M. Azzeh and A. B. Nassif, "Project productivity evaluation in early software effort estimation," *J. Softw. Evol. Process*, vol. 30, no. 12, p. e2110, Dec. 2018, doi: 10.1002/smr.2110.

[58] R. Silhavy, P. Silhavy, and Z. Prokopova, "Analysis and selection of a regression model for the Use Case Points method using a stepwise approach," *J. Syst. Softw.*, vol. 125, pp. 1–14, Mar. 2017, doi: 10.1016/j.jss.2016.11.029.

[59] A. Ali and C. Gravino, "An empirical comparison of validation methods for software prediction models," *J. Softw. Evol. Process*, vol. 33, no. 8, pp. 1–38, Aug. 2021, doi: 10.1002/smr.2367.

[60] E. Alpaydin, *Introduction to machine learning*. Massachusetts: MIT press, 2014. Available at: Google Books

[61] E. Kocaguneli and T. Menzies, "Software effort models should be assessed via leave-one-out validation," *J. Syst. Softw.*, vol. 86, no. 7, pp. 1879–1890, Jul. 2013, doi: 10.1016/j.jss.2013.02.053.

[62] Q. Li, Q. Wang, Y. Yang, and M. Li, "Reducing biases in individual software effort estimations," in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '08*, 2008, pp. 223–232, doi: 10.1145/1414004.1414041.

[63] Ardiansyah, R. Ferdiana, and A. E. Permanasari, "Use Case Points based software effort prediction using regression analysis," in *2019 International Conference on Advanced Computer Science and information Systems (ICACSIS)*, 2019, pp. 15–20, doi: 10.1109/ICACSIS47736.2019.8979851.

[64] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, Aug. 2012, doi: 10.1016/j.infsof.2011.12.008.

[65] P. D. Ellis, *The Essential Guide to Effect Sizes*. Cambridge: Cambridge University Press, 2010. doi: 10.1017/CBO9780511761676

[66] J. Cohen, "A power primer.," *Psychol. Bull.*, vol. 112, no. 1, pp. 155–159, 1992, doi: 10.1037/0033-2909.112.1.155.

[67] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Population size in Particle Swarm Optimization," *Swarm Evol. Comput.*, vol. 58, no. May, p. 100718, Nov. 2020, doi: 10.1016/j.swevo.2020.100718.

[68] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 1999, vol. 3, pp. 1945–1950, doi: 10.1109/CEC.1999.785511.

[69] I. D. Kenestie and Sholiq, "Determining Effort Rate (ER) Value for Use Case Points based Educational Software Development Effort Estimation," *J. Tek. POMITS*, pp. 1–11, 2011. Available at: Google Scholar

[70] Sholiq, T. Sutanto, A. P. Widodo, and W. Kurniawan, "Effort Rate on Use Case Point Method for Effort Estimation of Website Development," *J. Theor. Appl. Inf. Technol.*, vol. 63, no. 1, pp. 209–218, 2014. Available at: Google Scholar

[71] R. Silhavy, P. Silhavy, and Z. Prokopova, "Evaluating subset selection methods for use case points estimation," *Inf. Softw. Technol.*, vol. 97, no. June 2017, pp. 1–9, May 2018, doi: 10.1016/j.infsof.2017.12.009.

[72] A. B. Nassif, D. Ho, and L. F. Capretz, "Towards an early software estimation using log-linear regression and a multilayer perceptron model," *J. Syst. Softw.*, vol. 86, no. 1, pp. 144–160, Jan. 2013, doi: 10.1016/j.jss.2012.07.050.

[73] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, vol. 4, no. 2, pp. 1942–1948, doi: 10.1109/ICNN.1995.488968.

[74] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2009. Available at: Google Scholar

[75] M. Harman, "The Current State and Future of Search Based Software Engineering," in *Future of Software Engineering (FOSE '07)*, 2007, pp. 342–357, doi: 10.1109/FOSE.2007.29.