**Journal of Information System, Technology and Engineering**
Volume 2, No. 2, pp. 261-268
E-ISSN: 2987-6117
http://gemapublisher.com/index.php/jiste
Received: April 2024
Accepted: May 2024
Published: June 2024

# Particle Swarm Optimization-based Linear Regression for Software Effort Estimation

**Puguh Jayadi[1], Khairul Adilah binti Ahmad[2], Rayhan Zulfitri Dwi Cahyo[3], Jofanza Denis Aldida[4]**

Universitas PGRI Madiun[1,3,4], Universitas Negeri Malang[1], Universiti Teknologi MARA Cawangan Kedah[2]

Correspondence Email: puguh.jayadi@unipma.ac.id[1]

***Abstract***

In the context of software effort estimation, this study investigates the use of Particle Swarm Optimization (PSO)-based Linear Regression to improve estimation accuracy. The main problem faced is the limitations of standard Linear Regression models in accurately estimating the effort required for software development projects. This research aims to improve the quality of estimation of software efforts to optimize resource management and project schedules. The method used was the integration of PSOs in Linear Regression, which was evaluated using three different COCOMO datasets. Experimental results show that LR+PSO models consistently outperform standard Linear Regression with lower MAE, MSE, and RMSE, as well as higher R-squared. In conclusion, integrating PSOs in Linear Regression effectively improves the estimation accuracy of software efforts, demonstrating great potential for improving estimation quality in software project management practices.

**Keywords: linear regression, particle swarm optimization, software effort estimation, accuracy.**

## INTRODUCTION

Software effort estimation is an important aspect of software project management, which directly impacts resource allocation and project scheduling. Linear Regression (LR) has long been used in software effort estimation practices, but in some cases, these models have limitations in accurately estimating the effort required (Sharma & Chaudhary, 2020). Several studies have been conducted to calculate effort in software development with algorithmic approaches such as Use Case Point, Function Point, and COCOMO (Feizpour et al., 2023; Kaushik et al., 2016; Park et al., 2016; Silhavy et al., 2023).

In addition, some studies use Machine Learning to improve accuracy in classification, estimation, and prediction using various methods, including metaheuristic techniques. For example, Cho et al. (2017) proposed the use of Particle Swarm Optimization (PSO) algorithms in Support Vector Machine (SVM) to optimize parameters in classifying Power Distribution Systems whose results showed a significant improvement in classification accuracy (Cho & Thom Hoang, 2017). In addition, Chahar et al. (2022) applied Genetic Algorithms and Neural Networks to software testing, and the results showed consistent improvements in prediction accuracy (Chahar & Bhatia, 2022). Particle Swarm Optimization (PSO) is a metaheuristic that has proven effective in solving non-linear optimization problems (Hidayat, 2023; Wu et al., 2018). Integrating PSO in Linear Regression for software effort estimation promises improved accuracy and performance of estimation models.

Based on this background, this study aims to answer several research questions about whether integrating PSO in Linear Regression can improve the accuracy of software effort

**DOI: https://doi.org/10.61487/jiste.v2i2.69**

estimation. An important part is how the performance of the LR+PSO model compares to the standard Linear Regression model. This study uses public datasets such as COCOMO81, COCOMO_Nasa_v1, and COCOMO_Nasa_v2 because these datasets are often used in several studies on the topic of Software Effort Estimation (Banimustafa, 2018; Hoc et al., 2022; Kumar & Srinivas, 2023).

This research is expected to significantly contribute to software effort estimation practice by introducing a new approach that combines Linear Regression (LR) with PSO. The results of this research can help organizations and practitioners produce more accurate and reliable estimates of software efforts, thereby minimizing the risk of estimation errors and improving the efficiency of software project management. A major contribution of this research was the development of software effort estimation models that leverage the power of PSO to improve estimation accuracy and performance. In addition, it provides a deeper understanding of metaheuristic integration in the context of software effort estimation and a comprehensive comparison between LR+PSO models and standard Linear Regression.

## METHOD

This section explains the dataset and the stages used in conducting research. The public dataset is commonly used in Software Effort Estimation research (Marco et al., 2019, 2023). Table 1 contains a description of each dataset. The stages of the research are shown in Figure 1.

**Table 1. Dataset Description**

| Dataset | Project | Feature | Min Effort | Max Effort |
|---|---|---|---|---|
| COCOMO81 | 63 | 17 | 5.9 | 11,400 |
| COCOMO Nasa v1 | 60 | 16 | 8.4 | 3,240 |
| COCOMO Nasa v2 | 93 | 17 | 8.4 | 8211 |

Broadly speaking, the research stages consist of 3 major parts, namely Preprocessing, Modeling, and Prediction, and the last is Evaluation. In the Processing section, the dataset ready to be used will be changed to Byte Literals, namely changing from String type to Object type to prevent errors in subsequent data processing and ensure it is in the appropriate format when conducting analysis. Data normalization using StandardScaler so that the range of data to be processed is not too far can cause proportional models. Data sharing enables training and test models on a single subset (Shafiee, 2023).

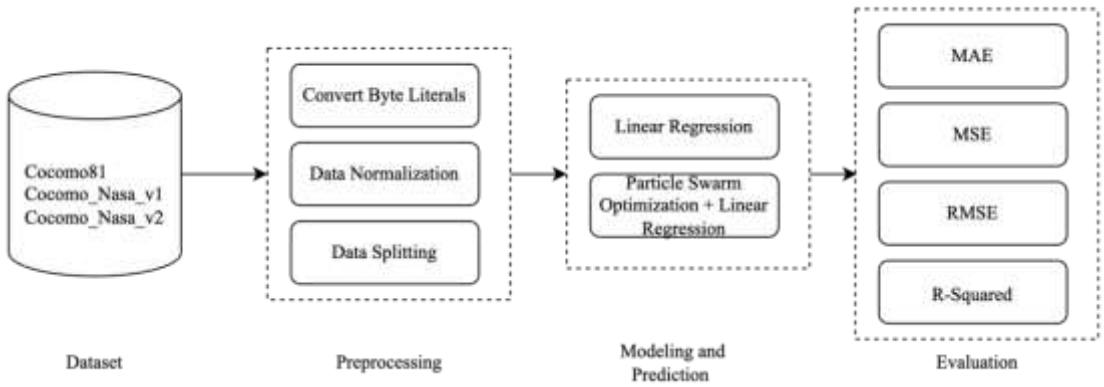


**Figure 1. Research Step**

DOI: https://doi.org/10.61487/jiste.v2i2.69

In the Modeling and Prediction section, namely making models and making predictions using Linear Regression (LR) which generally uses the formula:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon \quad (1)$$

Where $y$ is the dependent variable, $x_1$, $x_2$, ... $x_n$ is the independent variable, $\beta_0$, $\beta_1$, ..., $\beta_n$ is the regression coefficient, and $\varepsilon$ is the error. Linear Regression, as the basis of regression techniques, assumes the existence of a linear relationship between features and targets. Meanwhile, the Particle Swarm Optimization (PSO) method, is used to determine parameters for processing by Linear Regression. PSO was first recognized by Eberhart and Kennedy in 1995 which is one of the metaheuristic methods that serves to help determine the optimization of a problem (Demidova et al., 2016). PSO is inspired by the behavior of bio-natural (especially living things later referred to as particles), which cluster (become a swarm of particles) to find the same goal, with each particle determining its path or the most optimal way and will eventually be followed by other particles. PSO begins by randomly generating several solutions of particles, among which will be the optimal value as a solution for the parameter (Telikani et al., 2022). In general, the formula for PSO is divided into 2, namely Update Acceleration, and Position Update. For Acceleration Update the formula is:

$$v_i^{(t+1)} = w * v_i^{(t)} + c_1 * r_1 * (Pbest_i - x_i^{(t)}) + c_2 * r_2 * (Gbest - x_i^{(t)}) \quad (2)$$

Where $vi^{(t+1)}$ is the velocity of the particle *i at* the t+1 *iteration*. $w$ is the inertial weight that controls the impact of the previous speed on the current speed. $vi^{(t)}$ is the velocity of particle $i$ at iteration $t$. $c_1$ and $c_2$ are acceleration constants that control the influence of particle distances from $Pbest_i$ and $Gbest$ on particle movement. $r_1$ and $r_2$ is a uniform random number in a range [0, 1]. $Pbest_i$ is the best position that particle $i$ have achieved so far (Personal best). $Gbest$ is the best position that has been achieved by the entire herd so far (Global best). $x_i^{(t)}$ is the position of particle $i$ at iteration $t$. As for the Acceleration Update formula, it is:

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \quad (3)$$

Where $x_i^{(t+1)}$ is the position of particle $i$ on the iteration $t+1$. The essence of PSO is to update the velocity and position of each particle based on its best personal experience and the best experience of the herd, hoping to find an optimal global solution to a given problem. Inertial weight $w$, and acceleration constant $c_1$ and $c_2$ is a key parameter that affects PSO performance and is usually determined through experimentation or heuristic settings.

The Evaluation section uses calculations such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared ($R^2$) for each model (Ahmed et al., 2022; Shukla & Kumar, 2021). MAE is the average of the absolute values of the error between the prediction and the actual value. The formula is:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \quad (4)$$

Where $y_i$ is the actual value. $\hat{y}_i$ is the predicted value. $n$ is the number of samples. MSE is the average of the squares of the error between the prediction and the actual value. The formula is:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \quad (5)$$

**DOI: https://doi.org/10.61487/jiste.v2i2.69**

Where $y_i$ is the actual value. is the predicted value. $\hat{y}_i n$ is the number of samples. Root Mean Squared Error (RMSE) is the square root of the MSE, giving the model error measure in units equal to the target variable. The formula is:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (6)$$

Where $y_i$ is the actual value. is the predicted value. $\hat{y}_i n$ is the number of samples. R-squared ($R^2$) is the proportion of variation in the dependent variable that can be explained by the independent variable in the regression model. The formula is:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (7)$$

Where $y_i$ is the real value. $\hat{y}_i$ is the predicted value. $\bar{y}_i$ is the average of the actual values. $n$ is the number of samples. The R-squared indicates how well the data fits the regression model. A higher R-squared value indicates that the model can better explain the output variations.

**RESULT AND DISCUSSION**

This study explores the use of Particle Swarm Optimization (PSO) to improve the performance of Linear Regression (LR) models on three different datasets: COCOMO81, COCOMO_Nasa_v1, and COCOMO_Nasa_v2. The results showed significant improvements in model evaluation metrics when applying PSO to adjust regression parameters. In the COCOMO81 dataset, the PSO-optimized Linear Regression model achieved an MAE of 0.128, an MSE of 0.043, an RMSE of 0.208, and an R-squared of 0.544. Compare that to the Standard LR model, which yields an MAE of 0.766, an MSE of 1.001, an RMSE of 1.000, and a negative R-squared of -9.576. These improvements show that PSO effectively identifies a set of parameters that reduce prediction errors and improve the model's ability to explain data variability.

For COCOMO_Nasa_v1 datasets, using PSO also resulted in a marked performance improvement. The LR+PSO model achieved an MAE of 0.425, an MSE of 0.375, an RMSE of 0.612, and an R-squared of 0.831, while the Standard LR model recorded an MAE of 0.586, an MSE of 0.715, an RMSE of 0.846, and an R-squared of 0.678. This better performance suggests that PSO helps Linear Regression models more accurately predict target data.

The dataset COCOMO_Nasa_v2 shows the most notable improvement in model performance. LR+PSO achieves a very low MAE of 0.076, an MSE of 0.008, an RMSE of 0.090, and a high R-squared of 0.918, while the Standard LR model has an MAE of 0.201, MSE of 0.076, RMSE of 0.276, and an R-squared of 0.229. These results suggest that PSO can significantly improve the prediction accuracy of Linear Regression models, resulting in much lower errors and excellent model adjustment.

**Table 2. Results Comparison Results**

| Dataset | Model | MAE | MSE | RMSE | R-Squared |
|---|---|---|---|---|---|
| COCOMO81 | LR + PSO | **0.128** | **0.043** | **0.208** | **0.544** |
| COCOMO81 | LR Standar | 0.766 | 1.001 | 1.000 | -9.576 |
| COCOMO_Nasa_v1 | LR + PSO | **0.425** | **0.375** | **0.612** | **0.831** |
| COCOMO_Nasa_v1 | LR Standar | 0.586 | 0.715 | 0.846 | 0.678 |
| COCOMO_Nasa_v2 | LR + PSO | **0.076** | **0.008** | **0.090** | **0.918** |
| COCOMO_Nasa_v2 | LR Standar | 0.201 | 0.076 | 0.276 | 0.229 |

**DOI: https://doi.org/10.61487/jiste.v2i2.69**

Table 2 shows that LR+PSO outperforms Standard LR in all evaluation metrics in each dataset. These improvements signal the effectiveness of PSO in optimizing the parameters of the Linear Regression model, resulting in better model adjustment and lower prediction errors. The performance improvement of the Linear Regression model after using PSO was significant in all tested datasets. Notably, R-squared improvements suggest that PSO-optimized models can better account for variations in target data. In the COCOMO81 context, this increase is crucial given that Standard LR models have a negative R-squared, indicating very poor adjustment. PSOs improve prediction accuracy and turn inadequate models into models that account for more than half of the data variability. The most significant difference was seen in the R-squared metric, where models optimized with PSO achieved positive values, indicating a much better explanation of data variability than standard models that recorded negative values.

Significantly lower MAE and RMSE values in models with PSO indicate higher prediction accuracy and consistent and smaller errors in predictions. The fact that MSE also decreased sharply in models with PSO indicates that the model succeeded in reducing extreme prediction errors, which is an important aspect in assessing the quality of regression models. This discovery emphasizes the effectiveness of PSOs in improving the performance of Linear Regression models, especially in finding parameters that result in lower bias and variance in model predictions. This optimization results in better accuracy and reinforces confidence that the model will perform well against data not seen before, indicating the potential for strong generalizations.

Several comparison graphs show significant performance improvements in Particle Swarm Optimization (PSO)-optimized Linear Regression models compared to standard Linear Regression models across three different datasets. Each graph visualizes evaluation results from different models using MAE, MSE, RMSE, and R-squared metrics, providing a clear picture of the impact of PSO on model performance.

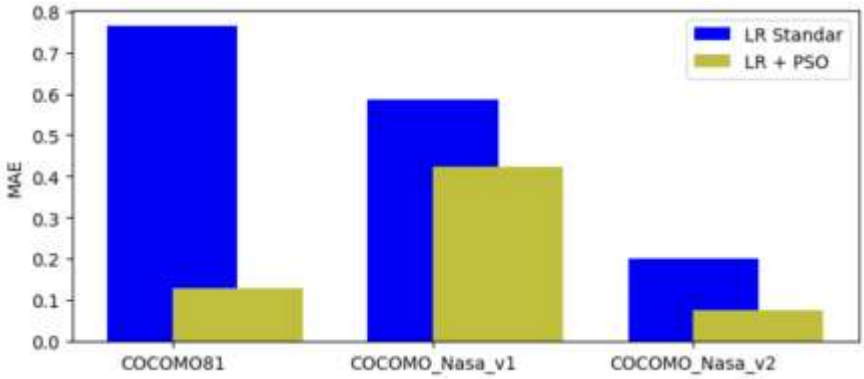


**Figure 2. Comparison of MAE**

The first graphic of Figure. 2 illustrates a lower MAE in the LR+PSO model across all datasets, indicating a more consistent and smaller mean absolute error than the model predicts. In the COCOMO81 dataset, the LR+PSO model showed a drastic reduction in MAE, showing a sharp increase in prediction accuracy.

**DOI: https://doi.org/10.61487/jiste.v2i2.69**

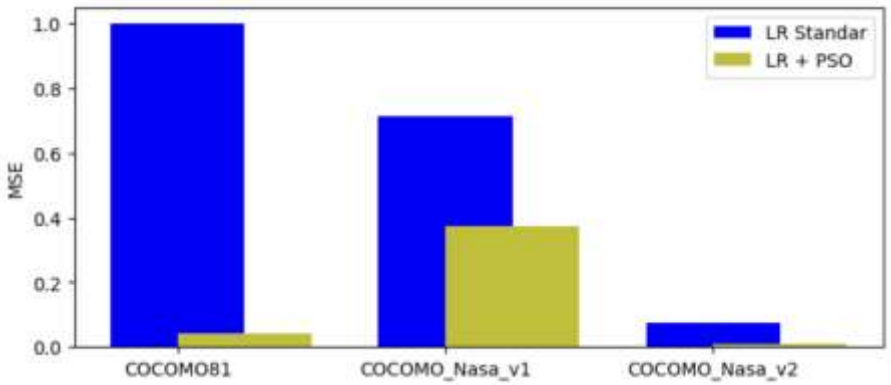


**Figure 3. Comparison of MSE**

The second graphic is Figure. 3 showed significant reductions in MSE for LR+PSO models across all datasets. This decrease underscores PSO's effectiveness in reducing greater prediction errors, with the most dramatic improvements occurring in datasets COCOMO81.

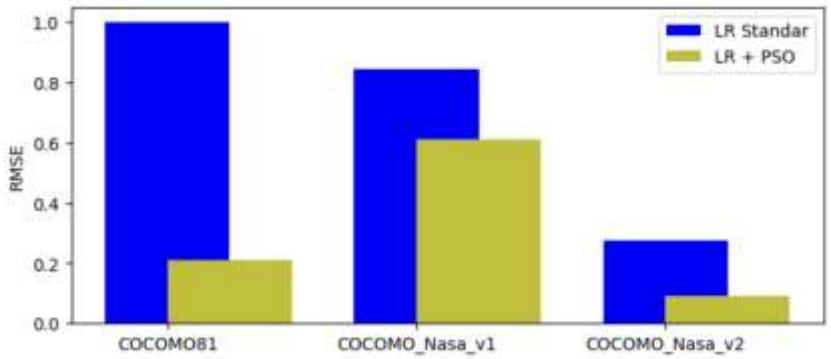


**Figure 4. Comparison of RMSE**

On the third chart in Figure. 4, The lower RMSE for the LR+PSO model confirms the consistency of smaller prediction errors, with the most noticeable differences seen in the dataset COCOMO81. This confirms that PSO provides a Linear Regression model that is more accurate in estimating actual values.

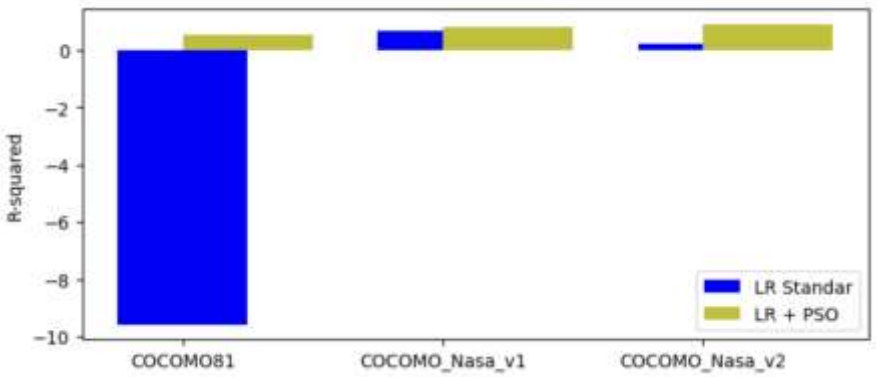


**Figure 5. Comparison of R-squared**

**DOI: https://doi.org/10.61487/jiste.v2i2.69**

The fourth graphic in Figure. 5 revealed R-squared improvements in the LR+PSO model for all datasets, with the COCOMO81 dataset displaying the most drastic increase from negative to positive. Higher R-squared values in the COCOMO_Nasa_v1 and COCOMO_Nasa_v2 datasets indicate that the PSO has successfully refined the model to explain the greater proportion of variance in the observed data. The graph visualization confirms that integrating PSO into the Linear Regression model consistently optimizes the model's parameters, decreasing prediction errors and increasing the model's ability to explain variability in the data. In conclusion, PSO stands out as a powerful optimization tool for improving the performance of Linear Regression models in various dataset conditions.

## CONCLUSION

The results confirmed that integrating Particle Swarm Optimization (PSO) in Linear Regression effectively increases the accuracy of software effort estimation. The findings show significant improvements in LR+PSO model performance compared to standard Linear Regression, with lower MAE, MSE, and RMSE and a higher coefficient of determination (R-squared). Consistently, LR+PSO models provide predictions closer to true values, demonstrating great potential in improving the quality of software effort estimation.

For future research, it is recommended that the integration of other metaheuristics in linear regression for software effort estimation be explored further. In addition to PSO, techniques such as the Genetic Algorithm or the Ant Algorithm can be tested to see if these approaches can also improve the performance of the estimation model. In addition, using more diverse and complex datasets can provide deeper insights into model performance in various software development contexts.

In addition, future research may focus on developing more adaptive and dynamic software effort estimation models. Integrating machine learning techniques, such as agency-based or kernel-based learning, can be the next step to improve the model's ability to handle variation and complexity in software effort estimation datasets. With a more sophisticated and diverse approach, it is expected to produce more accurate and reliable estimates for software project management practitioners.

## REFERENCES


Ahmed, M., Iqbal, N., Hussain, F., Khan, M. A., Helfert, M., Imran, & Kim, J. (2022). Blockchain-Based Software Effort Estimation: An Empirical Study. *IEEE Access, 10*. https://doi.org/10.1109/ACCESS.2022.3216840

Banimustafa, A. (2018). Predicting Software Effort Estimation Using Machine Learning Techniques. *2018 8th International Conference on Computer Science and Information Technology, CSIT*, 249–256. https://doi.org/10.1109/CSIT.2018.8486222

Chahar, V., & Bhatia, P. K. (2022). Performance Analysis of Software Test Effort Estimation using Genetic Algorithm and Neural Network. *International Journal of Advanced Computer Science and Applications, 13*(10), 376–383. https://doi.org/10.14569/IJACSA.2022.0131045

Cho, M.-Y., & Thom Hoang, T. (2017). Feature Selection and Parameters Optimization of SVM Using Particle Swarm Optimization for Fault Classification in Power Distribution Systems. https://doi.org/10.1155/2017/4135465

Demidova, L., Nikulchev, E., & Sokolova, Y. (2016). The SVM Classifier Based on the Modified Particle Swarm Optimization. *(IJACSA) International Journal of Advanced Computer Science and Applications, 7*(2).

Feizpour, E., Tahayori, H., & Sami, A. (2023). CoBRA without experts: New paradigm for software development effort estimation using COCOMO metrics. *Journal of Software: Evolution and Process*. https://doi.org/10.1002/smr.2569

Hidayat, W. F. (2023). Comparison of machine learning algorithm and feature selection particle swarm optimization on software effort estimation. *AIP Conference Proceedings*, 2714. https://doi.org/10.1063/5.0128433

Hoc, H. T., Silhavy, R., Prokopova, Z., & Silhavy, P. (2022). Comparing Multiple Linear Regression, Deep Learning and Multiple Perceptron for Functional Points Estimation. *IEEE Access, 10*, 112187–112198. https://doi.org/10.1109/ACCESS.2022.3215987

Kaushik, A., Soni, A. K., & Soni, R. (2016). An improved functional link artificial neural networks with intuitionistic fuzzy clustering for software cost estimation. *International Journal of System Assurance Engineering and Management, 7*, 50–61. https://doi.org/10.1007/s13198-014-0298-2

Kumar, K. H., & Srinivas, K. (2023). An accurate analogy based software effort estimation using hybrid optimization and machine learning techniques. *Multimedia Tools and Applications, 82*(20), 30463–30490. https://doi.org/10.1007/s11042-023-14522-x

Marco, R., Ahmad, S. S. S., & Ahmad, S. (2023). An Improving Long Short Term Memory-Grid Search Based Deep Learning Neural Network for Software Effort Estimation. *International Journal of Intelligent Engineering and Systems, 16*(4), 164–180. https://doi.org/10.22266/ijies2023.0831.14

Marco, R., Suryana, N., & Ahmad, S. S. S. (2019). A systematic literature review on methods for software effort estimation. *Journal of Theoretical and Applied Information Technology, 97*(2), 434–464.

Park, B. K., Moon, S. Y., & Kim, R. Y. C. (2016). Improving Use Case Point (UCP) Based on Function Point (FP) Mechanism. *2016 International Conference on Platform Technology and Service, PlatCon 2016 - Proceedings*. https://doi.org/10.1109/PlatCon.2016.7456803

Shafiee, S. (2023). An empirical evaluation of scrum training's suitability for the model-driven development of knowledge-intensive software systems. *Data and Knowledge Engineering*, 146. https://doi.org/10.1016/j.datak.2023.102195

Sharma, A., & Chaudhary, N. (2020). Linear Regression Model for Agile Software Development Effort Estimation. *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2020 - Proceeding*. https://doi.org/10.1109/ICRAIE51050.2020.9358309

Shukla, S., & Kumar, S. (2021). An Extreme Learning Machine based Approach for Software Effort Estimation. *International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE - Proceedings*, 47–57. https://doi.org/10.5220/0010397700470057

Silhavy, R., Bures, M., Alipio, M., & Silhavy, P. (2023). More Accurate Cost Estimation for Internet of Things Projects by Adaptation of Use Case Points Methodology. *IEEE Internet of Things Journal, 10*(21), 19312–19327. https://doi.org/10.1109/JIOT.2023.3281614

Telikani, A., Tahmassebi, A., Banzhaf, W., & Gandomi, A. H. (2022). Evolutionary Machine Learning: A Survey. *ACM Computing Surveys, 54*(8). https://doi.org/10.1145/3467477

Wu, D., Li, J., & Bao, C. (2018). Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation. *Soft Computing, 22*(16), 5299–5310. https://doi.org/10.1007/s00500-017-2985-9