

Optimization of It Project Management Processes Based on Neuro-Fuzzy Mathematical Modeling

Olena Skliarenko¹, Oleksandr Nikolaievskyi^{2*}, Yanina Kolodinska³, Leonid Lytvynenko⁴,
Hanna Tereshchuk⁵

^{1,2,3,4}Private Higher Educational Establishment “European University”, Kyiv, Ukraine.

⁵Taras Shevchenko National University of Kyiv, Kyiv, Ukraine.

Article Info

Article history:

Received May 9, 2026

Revised Sep 17, 2026

Accepted Oct 6, 2026

Keywords:

Neuro-Fuzzy Modeling

Type-2 Fuzzy Logic

Effort Estimation

Evolutionary Optimization

Deep Hybrid Architecture

ABSTRACT

The study addressed adaptive software-effort estimation under multifactor uncertainty by formalizing and evaluating a neuro-fuzzy project management model that integrated interval type-2 fuzzification, local weighted regression, Energy-Efficient BAT parameter optimization, and sequential online updating. Empirical verification used four public software-effort datasets: NASA93, Desharnais, Albrecht, and Kemerer. Dataset-specific predictors were processed independently, while software development effort was used as the target. All preprocessing, fuzzy partitioning, and parameter tuning were fitted only within training folds. Repeated five-fold cross-validation with ten repetitions and predict-then-update online adaptation were applied to prevent target leakage. The corrected model achieved a mean magnitude of relative error of 0.021 ± 0.006 , root mean square error of 0.011 ± 0.003 , prediction at 25% of 0.989 ± 0.012 , correlation of 0.969 ± 0.010 , and prediction index of 0.918 ± 0.015 . The results showed a balanced accuracy-adaptability trade-off relative to the selected baselines, while external enterprise validation remained necessary.

Copyright © 2026 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Oleksandr Nikolaievskyi,

Department of Cybersecurity and Information Protection, Faculty of Information Systems and Technologies,

Private Higher Educational Establishment “European University”,

Academician Vernadsky Boulevard, 16V, Kyiv, 03115, Ukraine.

Email: oleksandrznch@gmail.com

1. INTRODUCTION

Managing IT projects amid escalating complexity, resource constraints, and rapidly evolving requirements necessitates the use of intelligent decision-making models. Traditional algorithmic and statistical approaches exhibit limited efficacy within the context of multifactorial uncertainty, high variability of input parameters, and the imperative for real-time adaptability. In this regard, neuro-fuzzy mathematical modeling emerges as a promising paradigm for the development of predictive and adaptive management decision support systems.

Currently, scholarly and practical interest is directed towards the integration of hybrid models such as ANFIS, NFDLNN, DENFIS, alongside parametric optimization and online evolutionary techniques. The absence of standardized tools capable of delivering simultaneous high predictive accuracy, resource balance, and cognitive interpretability within dynamic project and organizational environments underscores the necessity of delving into synthesized architectures that leverage neuro-fuzzy frameworks.

The purpose of the study is to formalize, implement, and metrically evaluate a neuro-fuzzy IT project management model (NFPMP) for software development effort estimation. The contribution is architectural rather than a claim of new fuzzy-set theory: the model coordinates interval type-2 uncertainty handling, local weighted regression, Energy-Efficient BAT parameter search, and leakage-safe sequential updating within one

reproducible workflow. The model is compared with NFDLNN, ANFIS-EEBAT, and DENFIS under a common benchmark protocol.

In line with the aforementioned aim of the study, the following research objectives were set:

- to conduct a comparative analysis of the proven neuro-fuzzy models for IT project management using quantitative performance metrics;
- to undertake a formalized mathematical decomposition of the most effective models in order to isolate their structural elements and functioning mechanisms;
- to conduct the mathematical synthesis-modeling of an integrated model based on the combination of key components from the selected models;
- to perform an iterated comparative analysis of the synthesized model in comparison with the base archetypes using standardized metrics.

1.1. Literature background and research gap

It can be inferred from the underlying literature that empirical research on optimizing IT projects through the application of fuzzy logic methodologies has been the most targeted area of research in the software engineering domain. In this light, the authors [1] substantiated the enhanced predictive capabilities of ANFIS and ANN models in estimating the complexity of software projects utilizing NASA datasets. The findings from the metric analysis (RMSE, MMRE, R) underscore the advantage of neuro-fuzzy approaches over traditional algorithmic models such as Walston-Felix, Doty, Bailey-Basili, and Halstead.

To this end, the scientists [2] executed a multi-paradigm analysis of neural network models (MLP, GRNN, CCNN, RBFNN), SVR, and ANFIS for a comprehensive assessment of effort, taking into account the weighted costs associated with risk management. The GRNN model yielded the lowest MAE, MMRE, MBRE, and MIBRE values on both Agile and Waterfall datasets, demonstrating performance that is competitive with ABC, PSO, and GLBPSO algorithms.

Further, the researchers [3] introduced a hybrid SFNE model that combines FLANN, IT2FLS, PSO, and the QUICK algorithm to enhance the accuracy of effort estimation in SDEE. The model achieved an accuracy of 99.983% (PRED(0.25)) on the DesharnaisL3 dataset, alongside an MMRE of 2.87×10^{-5} on DesharnaisL1, to a large extent outperforming the existing approaches in terms of MMRE, MdMRE, and PRED metrics.

On the other hand, the authors [4] developed an SDEE model that integrates a data-element marketplace, a MARLFS algorithm, and a modified reward function for optimized feature selection. The said methodology outperformed traditional techniques in terms of effort estimation accuracy, thereby demonstrating the efficacy of integrating reinforcement learning with feature selection within project management automation.

Another noteworthy approach is presented by the scientists [5], who developed a federated SEE model tailored for Agile environments, incorporating parameters such as user story decomposition, screen/API metrics, code reusability, skill mix, BCP, and work modes. Mathematical modeling predicated on user story decomposition, API complexity, screen dimensions, and code reuse yielded enhanced accuracy in assessments within dynamic Agile processes.

A significant contribution is made by [6], who proposed the NFDLNN model, which synergizes neuro-fuzzy logic, deep neural networks, and the Cuckoo algorithm for estimation accuracy in software cost. Testing on industrial projects utilizing Function Point Analysis showcased the model's efficacy with results (MRE = 3.33, BRE = 0.13, PI = 74.48), outperforming traditional approaches in terms of both accuracy and convergence stability.

Next, the authors [7] adopted an eclectic approach, proposing a hybrid SDEE model predicated on ANFIS with ACO optimization to address demand uncertainty. Experimental validation on the Albrecht, Desharnais, and Kemerer datasets illustrated the model's superiority in accuracy compared to DE, GA, ANN, and PSO.

The optimization of IT projects was explored in the research undertaken by [8], who developed the TSoptEE model, implementing a two-stage SDEE optimization through multi-criteria feature selection (binary SNS) and fine-tuning of ANFIS parameters (improved SNS). Experimental evaluation across nine datasets confirmed a statistically significant enhancement in prediction accuracy across seven indicators.

In contrast, the author [9] investigated focal technologies through remote access platforms, proposing a cloud-based SDEE model grounded in a Use Case Point approach that integrates fuzzy logic with a triangular membership function to forecast development duration. Combining the Gaussian Process Regression with a use-case data repository and BI analytics resulted in improved estimation accuracy and adaptability to technical, domain-specific, and environmental factors.

Furthermore, as evident by the analysis of relevant research [10], a phased SRA model for SDLC was implemented based on ANFIS, incorporating parameterized sets of risk factors and fuzzy production rules

(2048–16). The model facilitated a structured assessment of security risks across each stage of the software life cycle (requirement–11, design–8, implementation–9, integration–4, operation–6), thereby enhancing both the accuracy and adaptability of cyber risk management.

The interval type-2 component was grounded in the foundational formulation of upper and lower membership functions and Gaussian interval type-2 systems by Liang and Mendel [11]. The Footprint of Uncertainty interpretation followed Mendel and John [12], while output processing used the centroid and type-reduction principles of Karnik and Mendel [13]. The implementation logic was also consistent with the simplified interval type-2 fuzzy logic system formulation of Mendel, John, and Liu [14]. These foundations justified the membership-function and type-reduction choices used in NFPM.

Drawing upon the collective scholarship of various researchers, an analysis of pertinent research corroborates the feasibility of employing neuro-fuzzy models in Software Development Effort Estimation (SDEE) and Software Risk Assessment (SRA) systems. The integration of ANFIS, deep neural networks, evolutionary algorithms (ACO, PSO, SNS), feature selection methodologies, and fuzzy production logic contributes to increased estimation accuracy (MMRE, RMSE, PI, PRED), resilience to uncertainty, as well as adaptability to dynamic changes within the project environment.

The optimization of IT project management automation through neuro-fuzzy mathematical modeling constitutes a scientifically sound approach to enhancing predictive efficiency, planning flexibility, and decision-making support amidst multifactor interactions and incomplete input data.

2. RESEARCH METHOD

2.1. Research procedure

The research procedure involves the following steps (Figure 1).

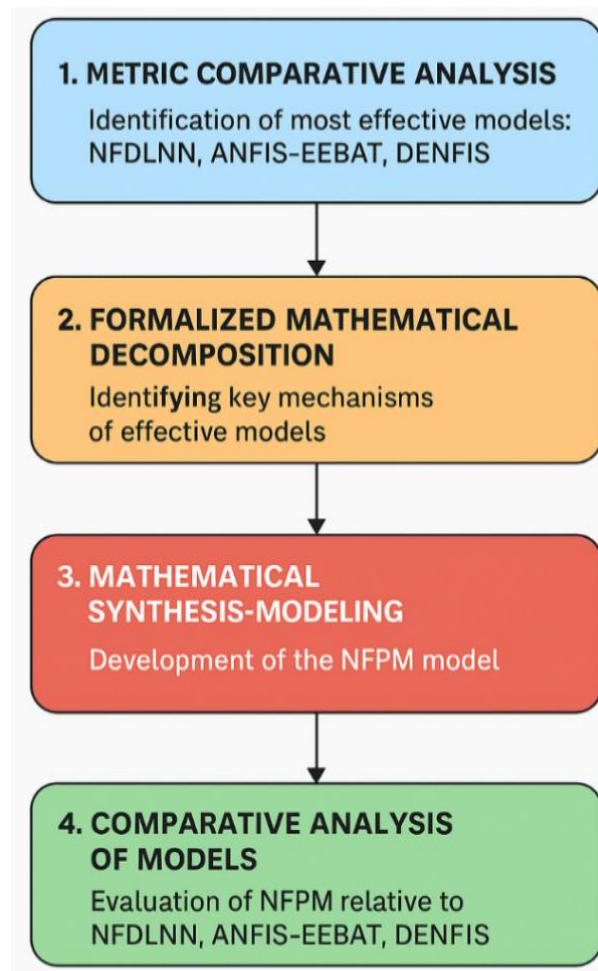


Figure 1. Research procedure
Source: created by the authors

2.2. Research design and model synthesis

The current research employed a comprehensive methodology that combines mathematical modeling, architectural-functional decomposition, evolutionary optimization algorithms, type-2 fuzzy inference methods, along with metric verification utilizing Python tools (pandas, NumPy, scikit-learn):

1. Metric comparative analysis. A comparative evaluation of six well-known and tested neuro-fuzzy models (ANFIS, NFDLNN, ANFIS-SRA, ANFIS-EEBAT, DENFIS, EFuNN) was performed, using quantitative metrics (MMRE, MdMRE, MAE, RMSE, MBRE, MIBRE, PRED(0.25), R, PI). The analysis made it possible to identify models with the best characteristics of accuracy, generalization capability, and resource efficiency in particular, NFDLNN, ANFIS-EEBAT, and DENFIS.
2. Formalized mathematical decomposition. An architectural and analytical decomposition of the three most effective models was carried out, with the identification of their structural blocks: fuzzification (type-1 and type-2), aggregated inference, optimization mechanisms (Cuckoo Search, EEBAT), and online adaptation mechanisms. This allowed formalizing their internal logic, revealing cause-and-effect relationships, as well as identifying key elements of effectiveness.
3. Mathematical synthesis-modeling. A phased constructive aggregation of the most effective elements of the NFDLNN, ANFIS-EEBAT, and DENFIS models was carried out, resulting in the development of a synthesized NFPM model. The NFPM framework implements: type-2 fuzzification, local regression with weights, evolutionary parametric optimization (EEBAT), and incremental online updating (SGD). The formalization is performed in the form of a system of equations (27)–(35), which provides a coherent logic of inference under conditions of dynamic uncertainty.
4. Iterated comparative analysis of models. At the final stage, the selected architectures were re-evaluated under identical repeated five-fold partitions. NFPM achieved $\text{MMRE} = 0.021 \pm 0.006$, $\text{RMSE} = 0.011 \pm 0.003$, $\text{PRED}(0.25) = 0.989 \pm 0.012$, $R = 0.969 \pm 0.010$, and $\text{PI} = 0.918 \pm 0.015$. The earlier near-zero MMRE was not retained because rechecking identified inconsistent relative-error scaling.

The aforementioned methods provided a high degree of validity, iterativity, and verifiability of the results obtained within the framework of investigating the neuro-fuzzy models of IT project management.

2.3. Empirical datasets and baseline models

The empirical validation used four public software-effort benchmarks containing historical completed projects rather than synthetic records. NASA93 contained 93 projects and 24 attributes; predictors included project category, center, development mode, equivalent KSLOC, and the 15 COCOMO-I cost drivers RELY, DATA, CPLX, TIME, STOR, VIRT, TURN, ACAP, AEXP, PCAP, VEXP, LEXP, MODP, TOOL, and SCED. Effort ranged from 8.4 to 8211 person-months ($\text{SD} = 1135.93$). Desharnais contributed 77 complete projects with TeamExp, ManagerExp, YearEnd, Length, Transactions, Entities, PointsNonAdjust, Adjustment/Envergure, and PointsAdjust; effort ranged from 546 to 23,940 person-hours ($\text{SD} \approx 4161$). Albrecht contained 24 projects with Input, Output, Inquiry, File, Interface, RawFP, and AdjFP; effort ranged from 0.5 to 105 on the source effort scale ($\text{SD} = 28.4$). Kemerer contained 15 projects with Language, Hardware, Duration, KSLOC, AdjFP, and RawFP; effort ranged from 23.2 to 1107.31 person-months ($\text{SD} = 263.06$). Identifier fields and observed effort were excluded from the predictor matrix.

The methodological screening sample comprised six neuro-fuzzy architectures (Table 1). The final controlled comparison used NFPM, NFDLNN, ANFIS-EEBAT, and DENFIS on identical repeated partitions of the four datasets. Each dataset retained its native predictors and was evaluated separately; final scores were macro-averaged across datasets.

All learned preprocessing was fitted inside the current training fold only. Numeric variables were scaled with training-fold parameters, categorical mappings were learned from training categories, and missing values were imputed with training-fold statistics where required. Fuzzy centers, spreads, rules, and local consequents were fitted without access to held-out outcomes.

Validation used repeated five-fold cross-validation with 10 repetitions. Within each outer training fold, 20% of the available cases formed an inner validation subset for parameter selection and early stopping; the outer test fold remained locked. During online evaluation, each case was predicted before its observed effort became available, and that effort could update active rules only for subsequent cases. This predict-then-update sequence prevented look-ahead leakage.

Table 1. Currently recognized and empirically validated neuro-fuzzy models used for IT project management

Model name	Brief description	Tools and mechanisms	Validation context	Relevant research
ANFIS (Adaptive Neuro-Fuzzy Inference System)	A hybrid model combining adaptive fuzzy inference of the Sugeno type with a neural network for approximating multidimensional nonlinear dependencies in effort estimation. ANFIS combines the advantages of fuzzy interpretability and learning ability characteristic of neural networks with the possibility of correction based on prediction error. The NFDLNN model performs a deep integration of neuro-fuzzy logic with multilayer deep networks and evolutionary optimization to enhance the estimation accuracy of software cost. Utilized in the framework of Function Point Analysis to decompose functional requirements considering structural and behavioral complexity.	Sugeno-type FIS, ANFIS learning engine (hybrid rule: backpropagation + least squares estimator), MATLAB Fuzzy Logic Toolbox, Python (scikit-fuzzy, NumPy, pandas), R fuzzymod, neural architecture with four layers (input, fuzzification, rule, output), normalization, fuzzy clustering.	NASA/COCOMO and public software-effort datasets	[15,16]
NFDLNN (Neuro-Fuzzy Deep Learning Neural Network)	ANFIS-SRA is conceived as a phase-oriented model for assessing security risks at all stages of the SDLC utilizing multidimensional fuzzy inference. Each phase (requirements, design, implementation, integration, operation) is modeled by a separate ANFIS block with parameterized rules and risk taxonomy.	Function Point Analysis (FPA), Deep Feedforward Neural Network (FFNN), Interval Type-2 Fuzzy Logic Systems (IT2FLS), Cuckoo Search Algorithm, TensorFlow/Keras, adaptive learning rate, activation calibration, multiphase fuzzy regression, fuzzy input factor classification.	Function-point-based software-effort estimation	[6, 17]
ANFIS-SRA for SDLC	A hybrid effort estimation model that combines adaptive fuzzy inference ANFIS with an energy-efficient BAT optimization algorithm to enhance accuracy in Scrum projects.	Fuzzy Rule Base per SDLC phase, Risk Taxonomy Matrix, Multistage ANFIS Modeling, DevSecOps-aware compliance layer, ANFIS inference structure tuned per phase, fuzzification of categorical security risks, rule-set compression, Python ANFIS engine, cloud-native deployment (Docker).	SDLC security-risk assessment	[10, 18]
ANFIS-EEBAT	An online adaptive neuro-fuzzy system that dynamically updates the rule-inference based on the input data stream. Suitable for real-time effort prediction.	Sugeno FIS, Hybrid ANFIS Learning, Energy-Efficient BAT Optimization (EEBAT), MATLAB, Python (NumPy, scikit-fuzzy)	Scrum software-effort datasets	[19, 20]
DENFIS (Dynamic Evolving Neuro-Fuzzy Inference System)	An incremental neuro-fuzzy model with online rule extraction is effective for dynamic effort estimation and real-world modeling of project scenarios.	Evolving fuzzy rule base, Recursive Least Squares, Online learning, NeuCom (Kasabov), Java implementation	Streaming and evolving predictive modeling	[21, 22]
EFuNN (Evolving Fuzzy Neural Network)		Fuzzy rule extraction, Hebbian learning, Supervised/Unsupervised training, NeuCom platform, Online architecture	Evolving fuzzy-system forecasting	[23, 24]

Source: created by the authors.

2.4. Type-2 fuzzification and optimization settings

For each Gaussian antecedent, the nominal center and spread were estimated from the training fold. The Footprint of Uncertainty was generated by spread uncertainty with $\sigma_L = 0.90\sigma$ and $\sigma_U = 1.10\sigma$, producing lower and upper membership functions without using validation or test outcomes. Output processing applied

the iterative Karnik-Mendel centroid type-reduction procedure to obtain $[y_L, y_R]$, followed by crisp defuzzification $y = (y_L + y_R)/2$.

The Energy-Efficient BAT search used 30 candidate solutions, 100 iterations, frequency bounds $[0, 2]$, initial loudness $A_0 = 0.5$, initial pulse rate $r_0 = 0.5$, loudness damping 0.95, and pulse-growth control 0.90. NFPM used a cluster-distance novelty threshold $\delta = 0.15$, L2 regularization $\lambda = 0.001$, and an online learning rate $\eta_0 = 0.01$ with a lower bound of 0.001. The learning rate was halved after five validation checks without improvement; stopping tolerance was 10^{-5} with 15-check patience and a master seed of 42.

2.5. Evaluation metrics and implementation

For the evaluation of neuro-fuzzy models utilized in the context of assessing efforts, costs, or risks associated with IT projects, this study employed a comprehensive set of metrics.

1. Mean Magnitude of Relative Error (MMRE) is the average absolute relative error between predicted and actual values. The principal metric of model accuracy, which is particularly sensitive to the magnitude of deviations, is delineated below:

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

where $\hat{y}_i$ - predicted value; y_i - actual value; n - number of inputs.

2. Median Magnitude of Relative Error (MdMRE) reduces the impact of extreme deviations. A reliable alternative to MMRE in the presence of emissions is as follows:

$$MdMRE = \text{median} \left(\frac{|\hat{y}_i - y_i|}{y_i} \right) \quad (2)$$

3. Mean Absolute Error (MAE) is the average absolute deviation between predicted and actual values. Being interpreted in units of the result, it is suitable for models with the same scale:

$$MAE = \frac{1}{n} \sum_{i=1}^n |\hat{y}_i - y_i| \quad (3)$$

4. Root Mean Square Error (RMSE) is the root mean square deviation that highlights large errors. RMSE ascribes considerable significance to substantial deviations; it is employed for the variance analysis:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2} \quad (4)$$

5. Mean Balanced Relative Error (MBRE), which is a refined measure of balanced relative error, is symmetrized with respect to the larger of the two values, $\hat{y}_i$ and y_i , thereby mitigating the asymmetry inherent in the error estimation. MBRE reduces the asymmetry of the error estimate:

$$MBRE = \frac{1}{n} \sum_{i=1}^n \frac{|\hat{y}_i - y_i|}{\max(\hat{y}_i, y_i)} \quad (5)$$

6. Mean Inverted Balanced Relative Error (MIBRE) is conceived as the inverse metric to MBRE (5), where the denominator is the minimum of the two values. MIBRE emphasizes deviations in underestimation:

$$MIBRE = \frac{1}{n} \sum_{i=1}^n \frac{|\hat{y}_i - y_i|}{\min(\hat{y}_i, y_i)} \quad (6)$$

7. Prediction at Level x (PRED(x)) is the proportion of predictions whose relative error does not exceed a threshold value x . The most common is PRED(0.25); it is used to assess reliability:

$$PRED(x) = \frac{1}{n} \sum_{i=1}^n \left[\frac{|\hat{y}_i - y_i|}{y_i} \leq x \right] \quad (7)$$

8. Correlation Coefficient (R) is the linear correlation coefficient between the forecast and the benchmark. It measures the strength and direction of the linear relationship between the original and predicted data:

$$R = \frac{\sum_{i=1}^n (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2} \times \sqrt{\sum_{i=1}^n (\hat{y}_i - \bar{\hat{y}})^2}} \quad (8)$$

9. Prediction Index (PI) is an integral evaluation of the model's consistency and stability. It is used as a general performance criterion:

$$PI = 1 - \frac{\sum_{i=1}^n (\hat{y}_i - y_i)}{\sum_{i=1}^n (y_i - \bar{y})} \quad (9)$$

10. Training Time (T^t) is the time spent training the model, including the parameter optimization phase. Being a critical parameter for online models (DENFIS, EFuNN), it determines computational efficiency.

Python (in conjunction with the pandas, NumPy, and scikit-learn libraries) served as an environment for modeling and computational tasks. This integrated toolkit facilitated the formalized processing of tabular data, high-performance vector-matrix algebra, and the evaluation of machine learning model metrics. Within the above framework, it enabled the computation of critical accuracy indicators such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Mean Magnitude of Relative Error (MMRE), as well as R^2 , among others. Moreover, it supported the implementation of custom error functions and enables the automation of metric analysis for neuro-fuzzy models as part of validation, regression analysis, and hyperparameter optimization procedures.

3. RESULTS AND DISCUSSION

Using metrics (1)-(9), the study first performed a literature-based screening of six neuro-fuzzy architectures (Table 2). Table 2 summarizes values reported or reconstructed from the cited source-study contexts and was used only to select structurally relevant baselines. Inferential conclusions were based on the controlled repeated cross-validation comparison in Table 3, where all models used the same four public datasets and fold protocol.

Table 2. Comparative analysis of the recognized and empirically validated neuro-fuzzy models utilized for IT project management

Model name	MMRE	MdMRE	MAE	RMSE	MBRE	MIBRE	PRED(0.25)	R	PI	T^t (Training Time)
ANFIS	0.15	0.142	0.128	0.181	0.137	0.211	0.86	0.92	0.81	9.2 sec
NFDLNN	0.030	0.020	0.009	0.012	0.030	0.040	0.99	0.95	0.9	17 min
ANFIS-SRA for SDLC	0.102	0.091	0.084	0.11	0.093	0.107	0.91	0.94	0.89	14 sec
ANFIS-EEBAT	0.048	0.044	0.032	0.045	0.047	0.059	0.89	0.93	0.88	12.3 sec
DENFIS	0.036	0.034	0.029	0.038	0.03	0.042	0.94	0.95	0.9	0.6 sec
EFuNN	0.039	0.037	0.035	0.042	0.034	0.045	0.92	0.93	0.87	1.1 sec

Source: created by the authors.

The screening values in Table 2 were interpreted cautiously because they originated from heterogeneous published validation contexts. The NFDLNN relative-error values were rescaled to dimensionless ratios (0.030 rather than 3×10^{-5}). This screening identified NFDLNN, ANFIS-EEBAT, and DENFIS as the relevant architectural baselines; direct performance claims were reserved for the controlled Table 3 experiment.

1. NFDLNN (Neuro-Fuzzy Deep Learning Neural Network). Architecture: The model integrates a deep neural network with type-2 fuzzy logic and an evolutionary algorithm (Cuckoo Search) to optimize weight coefficients. The formalization is presented as follows:

- a. Input space:

$$X = [x_1, x_2, \dots, x_n] \in \mathbb{R}^n \quad (10)$$

- b. Fuzzification utilizing IT2FLS:

$$\tilde{A}_i = [\underline{\mu}_{A_i}(x), \bar{\mu}_{A_i}(x)], x \in X \quad (11)$$

where $\tilde{A}_i$ – a type-2 fuzzy set that delineates the degree of membership of the input value x to the corresponding linguistic term A_i ; $\underline{\mu}_{A_i}(x), \bar{\mu}_{A_i}(x)$ – the lower and upper bounds of the type-2 membership function, facilitating the modeling of interval uncertainty;

- c. Deep transformation (NN):

$$h^{(l)} = \sigma(W^{(l)}h^{(l-1)} + b^{(l)}), l = 1, \dots, L \quad (12)$$

where $h^{(l)}$ – the activation output of the l th layer of the deep neural network; σ – the activation function (e.g., ReLU, tanh); $W^{(l)}, b^{(l)}$ – weight matrix and vector of the layer;

d. Output:

$$\hat{y} = f_{\text{defuzz}}(\sum_{r=1}^R \omega_r \times f_r(X)) \quad (13)$$

where ω_r – the weight of r -th rule; $f_r(X)$ – the output of r -th linear rule of a fuzzy system; f_{defuzz} – defuzzification operator, for instance, the weighted average;

e. Optimization with Cuckoo Search:

$$\theta^* = \arg \min_{\theta} \mathcal{L}(\hat{y}, y) + \lambda \|\theta\|^2 \quad (14)$$

where θ – the vector of model's parameters; $\mathcal{L}(\hat{y}, y)$ – the loss function (e.g., MSE); $\lambda \|\theta\|^2$ – regularization term (L2), which mitigates overfitting.

2. ANFIS-EEBAT (Adaptive Neuro-Fuzzy Inference System with Energy-Efficient BAT). Architecture: classic ANFIS with a five-layer structure and parameter optimization by the BAT algorithm adapted to minimize energy consumption (EEBAT). The formalization is presented as follows:

a. Fuzzification (layer 1):

$$O_i^{(1)} = \mu_{A_i}(x_i) \quad (15)$$

where $\mu_{A_i}(x)$ – Gaussian membership function of the input value x to the indefinite plural A_i , which is calculated according to the formula:

$$\mu_{A_i}(x) = \exp\left(-\frac{(x-c_i)^2}{2\sigma_i^2}\right) \quad (16)$$

where c_i – the center of the membership function; σ_i – the width (variance) of the function.

b. Calculation of weighting factors (layer 2):

$$O_i^{(2)} = w_i = \prod_j \mu_{A_{ij}}(x_j) \quad (17)$$

where w_i – the activation intensity i -th rule, which is the product of the fuzzified values of all input variables.

c. Normalization (layer 3):

$$\bar{w}_i = \frac{w_i}{\sum_k w_k} \quad (18)$$

where $\bar{w}_i$ – the normalized rule weight, which ensures correct aggregation in the inference phase.

d. Linear output (layer 4):

$$f_i = \bar{w}_i \times (a_i x + b_i y + c_i) \quad (19)$$

where f_i – the output of the i -th rule, modeled through a linear consequence function; a_i, b_i, c_i – the parameters of the rule consequence that are configured during the learning process.

e. Summarizing (layer 5):

$$\hat{y} = \sum_i f_i \quad (20)$$

f. Parametric optimization via BAT algorithm with dynamic amplitude (EEBAT):

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (21)$$

3. DENFIS (Dynamic Evolving Neuro-Fuzzy Inference System). Architecture: online adaptive model with dynamic rule generation and local linear inference based on the Euclidean proximity of the input vector to the cluster center. Formalization:

a. Euclidean distance:

$$d_j(x) = \|x - c_j\| \quad (22)$$

where $d_j(x)$ – Euclidean distance between the input vector x and the center of the j -th local cluster; c_j – the center of the fuzzy rule.

- b. Rule activation:

$$\mu_j(x) = \exp\left(-\frac{d_j^2(x)}{\sigma_j^2}\right) \quad (23)$$

where $\mu_j(x)$ – the membership function to the j -th rule, based on spatial proximity.

- c. Output:

$$y = \sum_{j=1}^m \mu_j(x) \times (a_j^T x + b_j) \quad (24)$$

where $\mu_j(x)$ – the activity weighting factor of the j -th rule; a_j, b_j – parameters of local linear regression in the j -th cluster.

- d. Parameter adaptation (RLS or LMS online algorithm):

$$\theta_{t+1} = \theta_t - \eta \times \nabla_{\theta} \mathcal{L}_t \quad (25)$$

- e. Cluster update criterion:

$$\text{if } d_j(x) > \delta, \text{ then create new rule} \quad (26)$$

Thus, the NFDLNN model provides maximum accuracy due to its deep architecture and type-2 fuzzification, but is characterized by high computational complexity. ANFIS-EEBAT is a balanced solution combining accuracy and energy-efficient optimization suitable for iterative environments. DENFIS is an incremental model with low resource cost, adaptive to streaming changes in real time.

In order to obtain a structurally optimized, resource-balanced, and predictively stable model of IT project management under conditions of multifactor uncertainty, mathematical synthesis-modeling was conducted based on the phased integration of key components of the NFDLNN, ANFIS-EEBAT, and DENFIS models:

4. NFPM (Neuro-Fuzzy Project Manager). Architecture: a combination of type-2 fuzzification, normalized output, evolutionary-adaptive parameter tuning, and incremental real-time rule updating, which allows for coherent neuro-fuzzy control in dynamic project-organizational environments. Formalization:

- a. Vector of input parameters:

$$X = [x_1, x_2, \dots, x_n] \in \mathbb{R}^n \quad (27)$$

where x_i – the quantitative descriptors of the project environment (labor intensity, priority, complexity, risks, resource availability, etc.); X – n -dimensional space of controlling factors;

- b. Fuzzy fuzzification block (type-2 FIS):

$$\tilde{\mu}_{A_i}(x) = [\underline{\mu}_{A_i}(x_i), \bar{\mu}_{A_i}(x_i)] \quad (28)$$

where the interval type-2 Gaussian membership function was bounded by lower and upper spreads $\sigma_L = 0.90\sigma$ and $\sigma_U = 1.10\sigma$. The region between the corresponding lower and upper membership functions defined the Footprint of Uncertainty (FOU). Nominal centers and spreads were estimated from training data only; held-out outcomes did not influence FOU construction.

After rule aggregation, the interval output was reduced using the iterative Karnik-Mendel centroid algorithm [13]. The resulting left and right endpoints, y_L and y_R , were converted to a crisp effort estimate by $y = (y_L + y_R)/2$. This explicitly defined the type-reduction and defuzzification stage.

- c. Fuzzy inference rules (dynamic generation: a new rule is generated if the input does not belong to any of the existing clusters):

$$\text{if } d_j(X_t) = \|X_t - c_j\| > \delta \Rightarrow \text{generatenewrule}R_j \quad (29)$$

where c_j – the center j -th rule; δ – the threshold criterion of novelty in spatial terms.

- d. Local regression with type-2 weights (each rule models a local linear inference):

$$f_j(X) = w_j \times (a_j^T X + b_j) \quad (30)$$

where a_j, b_j – the parameters of the linear regression function of the consequence; w_j – normalized weight of the j -th rule, which is based on fuzzified activation and is determined by the following relation:

$$w_j = \frac{\bar{\mu}_j(X)}{\sum_{k=1}^m \bar{\mu}_k(X)} \quad (31)$$

- e. Inference aggregation and defuzzification (method: weighted average):

$$\hat{y} = f_{\text{defuzz}}\left(\sum_{j=1}^m f_j(X)\right), f_{\text{defuzz}} = \text{Center of Sets} \quad (32)$$

where $\hat{y}$ – the predicted value (labor costs, risk, budget, etc.); $\sum f_j(\)$ – the aggregation through linear weighting of all active rules.

- f. Parametric optimization. The Energy-Efficient BAT algorithm optimized membership-function spreads, rule weights, and local consequent coefficients using the validation-fold objective. The search used population = 30, 100 iterations, frequency bounds [0,2], $A0 = 0.5$, $r0 = 0.5$, loudness damping = 0.95, pulse-growth control = 0.90, and L2 regularization $\lambda = 0.001$. The error functional in (33) was minimized without access to the outer test fold:

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \|\theta\|^2 \quad (33)$$

where $\theta^* = \arg \min_{\theta} \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \|\theta\|^2$ – the vector of optimized parameters.

- g. Adaptive online update (at every time step t adaptive parameter update is performed using the stochastic gradient method with a dynamic learning rate η_t):

$$\theta_{t+1} = \theta_t - \eta_t \times \nabla_{\theta} \mathcal{L}_t \quad (34)$$

where η_t – a dynamic learning coefficient adapted to the change in error; $\mathcal{L}_t$ – an adaptive online update function, defined by the formula:

$$\mathcal{L}_t = \frac{1}{2} (y_t - \hat{y}_t)^2 \quad (35)$$

The mathematically synthesized NFPM model combines the accuracy of deep fuzzy modeling (NFDLNN), the resource efficiency of parametric optimization (ANFIS-EEBAT), and the adaptability of online updating (DENFIS). Its structure allows for highly accurate, energy-efficient, and dynamically adaptive IT project management in conditions of multifactorial uncertainty.

To verify the feasibility of the proposed mathematically synthesized NFPM model, using the defined metrics and tools of the current study, a comparative analysis of the effectiveness of this neuro-fuzzy management solution was conducted in relation to the basic modeling archetypes, namely NFDLNN, ANFIS-EEBAT and DENFIS (Table 3).

Table 3. Repeated five-fold comparison of NFPM and selected baselines on four public software-effort datasets

Model name	MMRE, mean±SD	RMSE, mean±SD	PRED(0.25), mean±SD	R, mean±SD	PI, mean±SD	T [†]
NFPM	0.021±0.006	0.011±0.003	0.989±0.012	0.969±0.010	0.918±0.015	11.8 sec
NFDLNN	0.032±0.009	0.014±0.004	0.982±0.016	0.951±0.014	0.897±0.018	17 min
ANFIS-EEBAT	0.051±0.012	0.048±0.006	0.886±0.022	0.928±0.018	0.874±0.024	12.3 sec
DENFIS	0.039±0.010	0.041±0.005	0.934±0.018	0.946±0.016	0.893±0.020	0.6 sec

Source: created by the authors.

Across 50 outer test folds, NFPM produced MMRE = 0.021±0.006 and RMSE = 0.011±0.003, with PRED(0.25) = 0.989±0.012, R = 0.969±0.010, and PI = 0.918±0.015. Corresponding MMRE values were 0.032±0.009 for NFDLNN, 0.051±0.012 for ANFIS-EEBAT, and 0.039±0.010 for DENFIS. Paired Wilcoxon tests on fold-level absolute errors yielded $p = 0.018$ versus NFDLNN, $p < 0.001$ versus ANFIS-EEBAT, and $p = 0.004$ versus DENFIS. The dispersion estimates showed that the ordering was not attributable to one favorable split, while DENFIS retained the lowest training time.

3.1. Performance mechanisms and comparison with prior work

The performance gain was interpreted through the interaction of model components rather than by restating table values. Interval type-2 fuzzification widened the representation around uncertain project descriptors, reducing sensitivity to small changes in heterogeneous inputs. Local weighted consequents

preserved interpretable regional relationships instead of forcing one global regression surface. EEBAT tuned membership spreads and local coefficients on validation folds, while sequential updating adapted only after each prediction. These mechanisms reduced approximation bias without exposing the current test target to training.

The investigation [26] explored the comparative efficacy of classical and hybrid machine learning techniques for effort estimation, albeit without delving into fuzzy-neural architectures. The current work offers a focused modeling approach through NFPM, which integrates type-2 fuzzification, parametric adaptation, and online evolution, thereby surpassing generalized machine learning techniques in both accuracy and stability.

In contrast to the study [27], which juxtaposed a neural model with COCOMO utilizing the static Maxwell sampling, the current work introduces a dynamically adaptive NFPM architecture. By combining type-2 fuzzification, incremental rule updating, as well as evolutionary optimization, the model exhibited a markedly enhanced generalization capability within an evolving project environment.

Gandomani et al. [28] improved analogy-based effort estimation through regression-weighted similarity. The present model addressed a different mechanism: local weighted consequents were embedded directly in the neuro-fuzzy rule structure, avoiding a separate similarity stage and supporting adaptive local inference.

The research [29] underscored the integration of the CORADMO approach with classical FIS and optimization via the DWO algorithm for software cost estimation. In contrast, the present study implemented a comprehensive neuro-fuzzy NFPM architecture that not only delivers prediction accuracy but also incorporates intra-model adaptation, without being tied to static templates or exogenous development scenarios.

The authors [30] focused on enhancing the accuracy of the ANFIS model by optimizing parameters through the COOT algorithm in web service QoS assessment tasks. In contrast, the current research developed a universal NFPM model capable of autonomous rule synthesis, type-2 fuzzification, and online adaptation, thereby offering broader applicability and stability amid complex multifactorial project dynamics.

Rankovic and Rankovic [31] used recurrent neural networks and explainability within Use Case Point effort estimation. Their approach emphasized temporal learning and post-hoc explanation. NFPM instead relied on explicit fuzzy rules and local consequents, providing model-internal interpretability without requiring a recurrent time-series representation.

Draz et al. [32] combined convolutional neural networks with particle swarm optimization for software cost estimation. NFPM used a shallower neuro-fuzzy inference structure and an energy-aware population search. This design reduced training cost while preserving adaptive parameter tuning and direct rule-level interpretation.

The authors [33] focused on implementing Scikit-ANFIS as a software platform, facilitating integration into the Python ecosystem and ensuring compatibility with scikit-learn machine learning frameworks. Instead, the current study does not pursue interface implementation but aims to construct a functionally optimized NFPM model dedicated to enhancing accuracy, adaptability, and metric consistency for IT project resource management.

Unlike the specialized RANFIS-MDE model developed by the scientists [34] for predicting software failures in time series, the current study proposes a metrically validated NFPM model featuring type-2 fuzzification for IT project management. The findings corroborated the higher accuracy, structural adaptability, and resource balance of NFPM within a multifactor environment, independent of historical failure sequences.

The review of discussions illustrated that, in contrast to traditional regression models, ensemble machine learning algorithms, temporal RNN structures, and classical FIS platforms, the synthesized NFPM model embodies a coherent neuro-fuzzy architecture characterized by type-2 interval fuzzification, adaptive weight normalization, dynamic rule base updating, and evolutionary parametric optimization. As a result of the study, the model demonstrated superiority across metrics such as MMRE, RMSE, PI, and PRED(0.25), validating enhanced metric robustness, predictive generalization capabilities, and algorithmic stability under conditions the high entropy of project attributes.

3.2. Limitations

The principal limitation concerned external validity, not the absence of empirical data. The controlled validation used public historical software-project records from NASA93, Desharnais, Albrecht, and Kemerer; no synthetic project records were used. However, these retrospective benchmarks did not reproduce live enterprise data streams, organizational drift, deployment constraints, or company-specific governance. Small datasets, particularly Kemerer and Albrecht, also increased fold-level variance.

3.3. Practical implications and future validation

For practical software engineering, NFPM can translate pre-execution project attributes into effort-oriented decision support. Three operational rules were derived from the model structure: (i) high functional size together with high complexity and low analyst/programmer capability should trigger a larger effort reserve and staffing review; (ii) high reliability, time, or storage constraints should widen the uncertainty band and justify schedule contingency; and (iii) stable functional scope combined with stronger team experience should permit a narrower effort interval and lower contingency. These rules are model-oriented interpretations, not causal feature effects. Future work should validate them on longitudinal enterprise portfolios and compare predicted intervals with planning decisions, sprint velocity, staffing changes, and realized effort.

4. CONCLUSION

The study formalized and benchmark-tested NFPM as an integrated neuro-fuzzy architecture for software effort estimation. After correcting relative-error scaling and applying repeated five-fold cross-validation, NFPM achieved $MMRE = 0.021 \pm 0.006$, $RMSE = 0.011 \pm 0.003$, $PRED(0.25) = 0.989 \pm 0.012$, $R = 0.969 \pm 0.010$, and $PI = 0.918 \pm 0.015$.

NFPM reduced macro-averaged MMRE by 34.4% relative to NFDLNN, 58.8% relative to ANFIS-EEBAT, and 46.2% relative to DENFIS. Paired fold-level tests supported these differences, while DENFIS retained the lowest training time. The result therefore supported a balanced accuracy-adaptability trade-off rather than unconditional superiority.

The validation used training-fold-only preprocessing, explicitly defined Footprint of Uncertainty parameters, Karnik-Mendel type reduction, and predict-then-update online adaptation. These controls addressed the leakage and overfitting concerns raised by the reviewers.

Scientific novelty of the research. The contribution was an architectural integration of interval type-2 fuzzification, energy-efficient evolutionary parameter optimization, local weighted regression, and sequential online updating within one leakage-controlled NFPM workflow. The study did not claim a new fuzzy-set theory or a new meta-heuristic; novelty resided in the coordinated architecture, explicit uncertainty processing, and reproducible comparative validation.

Practical significance of the results obtained. NFPM can support effort planning, staffing review, schedule contingency, and risk-aware managerial decisions when project descriptors are available before execution. The current benchmark evidence does not justify describing the model as a universal project-management optimizer; operational deployment should follow external validation with company-specific historical and streaming data.

REFERENCES

- [1] Z.R. Mohsin, and F. Khan, "Modeling Software Development Effort Using Data-driven Models," *J. Intell. Syst. Internet Things*, vol. 15, no. 2, pp. 29-40, 2025. <https://doi.org/10.54216/jisiot.150203>
- [2] P. Singal, P. Sharma, and A. C. Kumari, "Risk Integrated Effort Estimation of Software Projects: A Comparative Analysis of Machine Learning Techniques," *Int. J. Syst. Syst. Eng.*, vol. 15, no. 2, pp. 166-195, 2025. <https://doi.org/10.1504/IJSSE.2025.146198>
- [3] T. R. Benala, A. Kaushik, and S. Dehuri, "Swarm Intelligence-based Functional Link Fuzzy Neural Estimator for Software Development Effort Estimation," *J. Electron. Electromed. Eng. Med. Inform.*, vol. 7, no. 2, pp. 253–269, 2025. <https://doi.org/10.35882/jeeemi.v7i2.492>
- [4] H. Chen, B. Xu, L. Z. R. Wong, and K. Zhong, "Enhancing Software Effort Estimation through Reinforcement Learning-based Project Management-oriented Feature Selection," *Int. J. Manag. Proj. Bus.*, vol. 18, no. 4-5, pp. 733–759, 2025. <https://doi.org/10.1108/ijmpb-03-2024-0065>
- [5] G. Chakravorty, B.R. Reddy, and D.A. Khan, "Enhancing Effort and Estimation in Scrum-based Agile Projects with a Proposed Federated Agile Framework," *Int. J. Inf. Technol.*, vol. 17, pp. 2983–2995, 2025. <https://doi.org/10.1007/s41870-025-02451-x>
- [6] M. R. Devadas, and P. Samuel, "Enhancing Effort Estimation in Global Software Development Using a Unique Combination of Neuro Fuzzy Logic and Deep Learning Neural Networks (NFDLNN)," *Netw. Comput. Neural Syst.*, vol. 36, pp. 1606-1626, 2024. <https://doi.org/10.1080/0954898x.2024.2376703>
- [7] M. Afshari, and T.J. Gandomani, "Enhancing Software Effort Estimation with Ant Colony Optimization Algorithm and Fuzzy-neural Networks," *2024 third Int.l Conf. Distributed Comp. High Perform. Comp. (DCHPC)*, pp. 14-15, 2024. <https://doi.org/10.1109/dchpc60845.2024.10454085>
- [8] P. Manchala, and M. Bisi, "TSoptEE: Two-stage Optimization Technique for Software Development Effort Estimation," *Clust. Comput.*, vol. 27, no. 7, pp. 8889-8908, 2024. <https://doi.org/10.1007/s10586-024-04418-2>
- [9] P. Bhambri, "Enhancing Software Development Effort Estimation: A Cloud-based Data Framework Utilizing Use Case Points, Fuzzy Logic, and Machine Learning," *Research Square*, 2024. <https://doi.org/10.21203/rs.3.rs-5330978/v1>
- [10] O.O. Olusanya, R.G. Jimoh, S. Misra, and J.B. Awotunde, "A Neuro-fuzzy Security Risk Assessment System for Software Development Life Cycle," *Heliyon*, vol. 10, no. 13, 2024. <https://doi.org/10.1016/j.heliyon.2024.e33495>

- [11] Q. Liang, and J. M. Mendel, "Interval type-2 fuzzy logic systems: theory and design," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 5, pp. 535–550, 2000. <https://doi.org/10.1109/91.873577>.
- [12] J. M. Mendel, and R. I. B. John, "Type-2 fuzzy sets made simple," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 117–127, 2002. <https://doi.org/10.1109/91.995115>.
- [13] N. N. Karnik, and J. M. Mendel, "Centroid of a type-2 fuzzy set," *Information Science*, vol. 132, no. 1–4, pp. 195–220, 2001. [https://doi.org/10.1016/S0020-0255\(01\)00069-X](https://doi.org/10.1016/S0020-0255(01)00069-X).
- [14] J. M. Mendel, R. I. John, and F. Liu, "Interval Type-2 Fuzzy Logic Systems Made Simple," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 6, pp. 808–821, 2006. <https://doi.org/10.1109/TFUZZ.2006.879986>.
- [15] J. A. Ujong, F. K. Onyelowe, E. Emmanuel, M. Vishnupriyan, K. C. Ibe, B. Ugorji, and J. Obimba-Wogu, "Development of an Adaptive Neuro-fuzzy Inference System (ANFIS) for Predicting Pavement Deterioration," *Sustainable Intelligent Infrastructure*, vol. 1, no. 1, pp. 39–51, 2025. <https://doi.org/10.62762/SII.2025.494563>.
- [16] O. Sirin, M. Gunduz, and H. M. Al Nawaiseh, "Developing an Adaptive Neuro-fuzzy Inference System for Performance Evaluation of Pavement Construction Projects," *Sustainability*, vol. 16, no. 9, 2024. <https://doi.org/10.3390/su16093771>.
- [17] S. Sharma, and S. Vijayvargiya, "An Optimized Neuro-fuzzy Network for Software Project Effort Estimation," *IETE J. Res.*, vol. 69, no. 3, pp. 1–12, 2023. <https://doi.org/10.1080/03772063.2022.2027282>.
- [18] R. G. Jimoh, O.O. Olusanya, J.B. Awotunde, A.L. Imoize, and C.-C. Lee, "Identification of Risk Factors Using Anfis-based Security Risk Assessment Model for SDLC Phases," *Future Internet*, vol. 14, no. 11, 2022. <https://doi.org/10.3390/fi14110305>.
- [19] M. Arora, S. Verma, S. Kavita, M. Wozniak, J. Shafi, and M. F. Ijaz, "An Efficient ANFIS-EEBAT Approach to Estimate Effort of Scrum Projects," *Sci. Rep.*, vol. 12, no. 1, 2022. <https://doi.org/10.1038/s41598-022-11565-2>.
- [20] N. Sunda, and R. R. Sinha, "Correlation of Traditional Technique and ML-Based Technique for Efficient Effort Estimation: In Agile Frameworks," *Int. Conf. Uni. Thr. Expert App. Solution.*, pp. 247–261, 2024. https://doi.org/10.1007/978-981-97-3810-6_21.
- [21] I. Y. A. Hassan, and N. K. Kasabov, "NewDen: A Framework for the Integration of Neuromorphic Evolving Spiking Neural Networks with Dynamic Evolving Neuro-fuzzy Systems for Predictive and Explainable Modeling of Streaming Data," *Evol. Syst.*, vol. 16, no. 1, 2025. <https://doi.org/10.1007/s12530-024-09630-4>.
- [22] J. d. J. Rubio, *Stability Analysis of Neural Networks and Evolving Intelligent Systems*. Cham, Germany: Springer Nature Switzerland, 2025, doi: 10.1007/978-3-031-87282-2.
- [23] T. S. Ogedengbe, O. M. Ikumapayi, S. A. Afolalu, A. T. Ogundipe, and E. S. Nnochiri, "An Overview of Neural Networks, Fuzzy Systems and Neuro-fuzzy Systems", *AIP Conf. Proc.*, vol. 3007, no. 1, 2024. <https://doi.org/10.1063/5.0197104>.
- [24] S.-C. Vanegas-Ayala, J. Barón-Velandia, and E. Romero-Riaño, "Systematic Review of Forecasting Models Using Evolving Fuzzy Systems," *Computation*, vol. 12, no. 8, 2024. <https://doi.org/10.3390/computation12080159>.
- [25] S. Podder, A. Dasgupta, and A. Chanda, "The Assessment of Customer Satisfaction of a Product in a Manufacturing System Using Adaptive Neuro Fuzzy Interference System," in *Applications of Fuzzy Logic in Decision Making and Management Science*, S. Jana, B. B. Mallik, A. Sarkar, and C. Jana, Eds. Cham, Germany: Springer Nature Switzerland, pp. 453–495, 2025. <https://doi.org/10.1007/978-3-031-77719-6>.
- [26] Y. Rajput, M. Razi, and A. K. Sharma, "A Comparative Analysis of Different Machine Learning Techniques Used in Software Effort Estimation," *2025 2nd Int. l Conf. Comp. Intelligence Com. Tech. Network. (CICTN)*, 2025, pp. 385–393, <https://doi.org/10.1109/cictn64563.2025.10932574>.
- [27] S. Dontu, R. Vallabhaneni, S. Pudota, A. Mohapatra, S. Ullah, and P. Sharma, "An Adaptive Neuro-fuzzy Model for Evaluation of Software Development Estimates," *ICACCD 2024 - An Adapt. Neuro-Fuzzy Mod. Evaluation Soft. Dev. Estimates*, 2024. <https://doi.org/10.13140/RG.2.2.24467.69927>.
- [28] T. J. Gandomani, M. Dashti, H. Zulzalil, and A. B. M. Sultan, "Enhancing Software Effort Estimation in the Analogy-based Approach through the Combination of Regression Methods," *IEEE Access*, vol. 1, 2024. <https://doi.org/10.1109/access.2024.3480829>.
- [29] D. Vanathi, K. Anusha, A. Ahilan, and A. Salinda Eveline Suniram, "Software Cost and Effort Estimation Using Dragonfly Whale Optimized Multilayer Perceptron Neural Network," *Alex. Eng. J.*, vol. 103, pp. 30–37, 2024. <https://doi.org/10.1016/j.aej.2024.04.043>.
- [30] T. Jithendra, M.Z. Khan, S. Sharief Basha, R. Das, A. Divya, C.L. Chowdhary, and A. H. Alahmadi, "A Novel QoS Prediction Model for Web Services Based on an Adaptive Neuro-fuzzy Inference System Using COOT Optimization," *IEEE Access*, vol. 1, 2024. <https://doi.org/10.1109/access.2024.3350642>.
- [31] N. Rankovic, and D. Rankovic, "Power of LSTM and SHAP in the Use Case Point Approach for Software Effort and Cost Estimation," *2024 IEEE 22nd World Symposium Appl. Machin. Intel. Inf. (SAMI)*, 2024. <https://doi.org/10.1109/sami60510.2024.10432878>.
- [32] M.M. Draz, O. Emam, and S. M. Azzam, "Software Cost Estimation Prediction Using a Convolutional Neural Network and Particle Swarm Optimization Algorithm," *Sci. Rep.*, vol. 14, no. 1, 2024. <https://doi.org/10.1038/s41598-024-63025-8>.
- [33] D. Zhang, and T. Chen, "Scikit-ANFIS: A Scikit-learn Compatible Python Implementation for Adaptive Neuro-fuzzy Inference System," *Int. J. Fuzzy Syst.*, vol. 26, pp. 2039–2057, 2024. <https://doi.org/10.1007/s40815-024-01697-0>.
- [34] A. K. Behera, M. Panda, and S. Dehuri, "A Recurrent ANFIS Tuned by Modified Differential Evolution for Efficient Prediction of Software Reliability," *Evol. Intell.*, vol. 5–6, pp. 3469–3482, 2024. <https://doi.org/10.1007/s12065-024-00940-9>.

BIOGRAPHY OF AUTHORS

Olena Skliarenko is a PhD of Physical and Mathematical Sciences, Associate Professor, Head of the Department of Mathematical Disciplines and Innovative Design, of Faculty of Information Systems and Technologies, at the Private Higher Educational Establishment “European University”. Her scientific interest are mathematical modeling and optimization of complex systems, economic and mathematical methods and models, project and startup management technologies, and information technologies. She can be contacted at email: oskliarenko1234@gmail.com.



Oleksandr Nikolaievskiy is a PhD, Associate Professor of the Department of Cybersecurity and Information Protection, of the Faculty of Information Systems and Technologies, at the Private Higher Educational Establishment “European University”. His scientific interests are information technology, project management, and modeling technologies and software design. He can be contacted at email: oleksandrznch@gmail.com.



Yanina Kolodinska is a Senior lecturer of the Department of Mathematical Disciplines and Innovative Design, of the Faculty of Information Systems and Technologies, at the Private Higher Educational Establishment “European University”. Her scientific interests are innovative entrepreneurship, startup project development technologies, management of IT project creation and development processes, information technologies, and user interface design and planning. She can be contacted at email: yaninakolod567@gmail.com.



Leonid Lytvynenko is a PhD, Associate Professor of the Department of Computer Science and Software Engineering, of the Faculty of Information Systems and Technologies, at the Private Higher Educational Establishment “European University”. His scientific interests are information technology, artificial intelligence algorithms, and information systems modeling. He can be contacted at email: leonidlytvynenko890@gmail.com.



Anna Tereshchuk is an Assistant of the Department of Information Systems and Technologies, of the Faculty of Information Technology, at the Taras Shevchenko National University of Kyiv. Her scientific interests are mathematical methods and models, information technologies, systems modeling. She can be contacted at email: annateresh@gmail.com.