Power of Artificial Neural Networks and Taguchi's Orthogonal Arrays in Software Effort and Cost Estimation

Nevena Rankovic [1], Dragica Rankovic [2] and Mirjana Ivanovic [3]

[1]*Tilburg University, School of Humanities and Digital Sciences, Department of Cognitive Science and Artificial Intelligence, Warandelaan 2, 5037 AB Tilburg, The Netherlands*
[2]*Union University "Nikola Tesla", Faculty of Applied Sciences, Department of mathematics, informatics and statistic, Dusana Popovica 22a, 18 000 Nis, Serbia*
[3]*University of Novi Sad, Faculty of Sciences, Department of mathematics and informatics, Trg Dositeja Obradovica 4, 21 000, Novi Sad, Serbia*

Abstract

In this paper we provide an overview of the research conducted on three novel models for estimating effort and costs in software project implementation. All three models utilize different architectures of artificial neural networks (ANNs) constructed based on Taguchi's orthogonal vector plans. The idea behind the conducted research is to optimize these novel models to avoid experiment repetition and reduce training time when estimating software projects. The structure of the proposed models could be improved by employing different encoding functions and clustering techniques for input data, aiming to mitigate the heterogeneous nature observed in various sets of real projects. Additionally, studies suggest the homogenization of input values across projects, leading to higher reliability and accuracy in the obtained results. Optimization with the Taguchi method, coupled with increased coverage of a wide range of industrial projects, results in the efficient and successful completion of various software projects, bringing benefits to the modern software industry.

Keywords

software estimation, ANN, Orthogonal Arrays, COCOMO2000, COSMIC FFP, UCP.

1. Introduction

In the contemporary software industry, there is an increasing demand for rapid, high-quality, and precise estimation of effort and costs prior to commencing software product development [1]. An essential factor for successful software project development and risk reduction involves accurately estimating the effort and costs associated with implementation. This process entails a meticulous evaluation of essential resources, such as time, personnel, and materials, necessary for project execution. The primary objective of effort and cost estimation lies in providing informed decision-making support for planning, budgeting, resource allocation, and project management [2]. The precise and reliable estimation of effort and costs facilitates efficient project management, mitigates risks, and leads to successful outcomes. Statistics reveal that only one-third of projects are successfully completed and implemented, nearly half exceed budget and implementation timelines, while around 20% experience complete failure. Therefore, precise and comprehensive estimation of effort and costs is a crucial determinant of software project success [3, 4]. It enables the determination of project initiation, conditions, and constraints necessary for successful execution and practical implementation. Consequently, software development teams must devote additional effort to ensure adequate estimation of effort and costs, thereby achieving favorable outcomes and meeting client expectations [5].

SQAMIA 2023: Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, September 10–13, 2023, Bratislava, Slovakia
EMAIL: N.Rankovic@uvt.nl (N.R); dragica.d.rankovic@gmail.com (D.R); mira@dmi.uns.ac.rs (M.I)
ORCID: 0000-0002-9910-5886 (N.R); 0000-0002-4464-0726 (D.R); 0000-0003-1946-0384 (M.I)
© 2023 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3588/p17.pdf

ANNs are artificial intelligence techniques constructed based on mathematical models and the architecture of each ANN [6] consists of an input layer, hidden layers, and an output layer. Each neuron utilizes an activation function that computes values based on the input layer and transmits them through the links to the subsequent neurons until reaching the output layer. The number of input features varies in our experiments. In the first COCOMO2000 approach, the number of input features is 3, with 1 output value and different numbers of nodes in the hidden layer [5]. In the second COSMIC FFP approach, the number of input features is 4, with 1 output value and varying numbers of nodes in the hidden layer [7]. In the third UCP approach, the number of input features is 6 for the first architecture, 4 for the second, with 1 output value, and different numbers of nodes in the hidden layer. In all experiments, the sigmoid activation function was used. All architectures were constructed based on Taguchi's orthogonal arrays. When selecting a specific architecture, considerations were made regarding the number of input features, the number of weight coefficients, and the number of levels in the orthogonal array. The aim was to construct the simplest architecture that, in the initial testing phase, achieves the minimum number of iterations, converges to the minimum error model, and satisfies the set Gradient Descent (GA) criterion, GA<0.01 [8]. The synergy between AI tools and Taguchi's optimization method proves highly potent, leading to cost reduction, enhanced quality, and accelerated development timelines for software projects. Employing a robust design strategy alongside orthogonal array plans enables the acquisition of dependable parameter information through a minimal set of experiments. Taguchi achieved optimal results through orthogonal arrays based on a unique set of Latin squares. This method significantly reduces key parameters and enables faster estimation of effort and project costs using factorial experiments with all possible combinations of parameters [9]. The robust design of Taguchi's experiment in an orthogonal array plan depends on the number of parameters, weighting factors, and the number of levels for each parameter. Each level of every parameter needs to be tested a certain number of times. For a complete factorial analysis, the number of iterations is $N=L^P$. However, by applying Taguchi's orthogonal array plan with 13 parameters at three levels, only $3^3 = 27$ experiments are needed. Taguchi's robust design method reduces the number of experiments by 99.99830649%. The orthogonal array plan selects a subset of non-repetitive combinations, properly considering all parameters. All levels of each parameter are tested at least once, and the Taguchi plan is applied for each level of a specific parameter [10] Table 1.

Table 1

Taguchi design vs. Full Factorial Design (FFD).

Taguchi design	No. of experiments	FFD	No. of experiments
$L_4(2^3)$	4	2^3	8
$L_8(2^7)$	8	2^7	128
$L_{12}(2^{11})$	12	2^{11}	2048
$L_{16}(2^{15})$	16	2^{15}	32768
$L_9(3^4)$	9	3^4	81
$L_{18}(3^7)$	18	3^7	2187

In this paper, new enhanced approaches and constructed models will be analyzed, which utilize different ANN architectures, to improve the accuracy and efficiency of effort and cost estimation during software project implementation. What is innovative compared to previous models is the construction of different ANN architectures based on Taguchi's orthogonal array plans. This enables fast, precise, and efficient estimation of effort and costs presented through three different models, using three commonly employed approaches. The main objectives of these models are: the construction and identification of the best model; the selection of an optimal ANN architecture that quickly converges to minimal magnitude relative error; reduction of the number of experiments; and shortening the time required for software effort estimation through high convergence rates. The structure of the paper is as follows: Section 2 outlines prior software estimation approaches. Section 3 introduces three new improved models' methodology. Section 4 presents achieved results. Section 5 discusses these results. Concluding remarks are in Section 6.

2. Previous approaches in software estimation

As a result of insufficiently adequate processes in the past, a large number of software projects were unsuccessful or not realized. Commonly used methods were similarity-based estimation, analysis and synthesis method, expert knowledge-based estimation, and various parametric methods [11].

The Analysis/Synthesis method is an approach to estimating software project effort and cost that involves dividing the project into smaller parts that are estimated separately. The advantage is that the method breaks down the project into smaller parts that are easier to estimate, allowing for a more detailed analysis and synthesis of effort and cost. However, this method requires more time as each part needs to be thoroughly estimated. Additionally, the overall estimation may be less reliable [6, 11].

Expert knowledge-based estimation is a rapid and simple method that relies on the experience of experts who have worked on similar software projects. The advantage of this method lies in the rapid estimation of effort and cost. However, the subjectivity of the experts can lead to variations in the estimates. Consensus among experts can be challenging, so it is important to establish clear guidelines and criteria for estimation. Validating the estimates through comparison with actual results from previous projects can improve the accuracy of the estimates [7, 11].

Parametric (algorithmic) methods rely on project metrics to construct an algorithm for determining time and cost. This method has advantages in objectivity, speed, and simplicity. Objectivity is achieved by using quantitative measures. Speed is achieved through automated calculation based on the algorithm. Ease of use is reflected in the requirement for basic knowledge of project size measurement and tracking historical data. However, drawbacks include the need for relevant and reliable historical data and lower precision in cases of high variability or novel technological concepts [9, 12].

The COCOMO2000 (Constructive Cost Model) is the most frequently employed model based on the source lines of code [13]. Parametric methods employ mathematical models that combine experimentally derived parameters to calculate the size of the system during design. The measurement of source code lines determines the size and complexity of the software project. COCOMO2000 is the prominent parametric method in this category, employing lines of source code as the measurement unit for software size. It enables estimation of the required production time. However, relying solely on lines of code for effort estimation has limitations, including variations across programming languages (e.g., C++, Java, C#) and the need for equivalence mappings with specific databases. COCOMO2000 is an algorithmic cost model that establishes the relationship between software metrics and project costs through mathematical functions. The actual effort is expressed in person-months (PM) [6, 7].

The approach based on analyzing function points is utilized to estimate the size of software functionality during development [14]. Initially, two models were distinguished within this approach: IFPUG (International Function Point Users Group) and Mark II. Subsequently, NESMA (Netherlands Software Users Metrics Association), IFPUG (version 4.1), and COSMIC FFP (Common Software Measurement International Consortium Full Function Point) became the most commonly used within the IFPUG framework [11, 15]. Function Point Analysis (FPA) addressed the limitations of the previous method that relied on measuring system size using lines of code. FPA measures the functionality of a system based on function points. Different systems may exhibit similar functionalities but utilize distinct technologies or programming languages, resulting in variations in the number of lines of source code. COSMIC FFP is one of the latest approaches, considering four reduced input values for estimating effort and costs based on functional size parameters. Fourteen parameter systems are evaluated to reliably measure functional size, encompassing aspects such as data communication, distributed data processing, performance, heavily used configuration, transaction rate, real-time data entry, user efficiency, real-time updating, complex processing, reusability, ease of installation, ease of use, multiple locations, and change facilitation [16]. In contrast to the previous COCOMO2000 method, which relies on three input parameters, COSMIC FFP incorporates four: Entry, Exit, Read, and Write.

The approach of analyzing users and use cases is also utilized to estimate software effort. COBRA (Cost Estimation, Benchmarking, and Risk Assessment) and UCP (Use Case Point Analysis) are the most commonly used models within this approach [17]. The UCP method is primarily employed to estimate the actual size of a software project. It considers the system's use cases to assess the effort required for implementation. Twenty-one parameters are used for estimation, with thirteen for the technical characteristics of the system and the remaining for environmental factors. The technical

characteristics include features such as a distributed system, system response time, efficiency, complexity of internal processes, code reuse, ease of installation and use, portability, maintenance, concurrency, security requirements, and user training. Environmental factors include compliance with development processes, application experience, knowledge of object-oriented technologies, analyst capability, team motivation, stability requirements, team availability, and programming language complexity. The UCP method combines system users and use cases, categorizing them based on their interaction complexity and transaction volume, to determine the actual size using weighted factors. The system size is defined using a four or six-dimensional vector representing user(s) and use case(s) complexity [8, 11].

3. The Methodology of three new models

In this section, three new models constructed using different ANN architectures based on Taguchi's orthogonal vector plans will be presented [11, 17].

3.1. New COCOMO in combination with ANN based on Orthogonal Arrays

Twenty-two parameters are the input variables of the COCOMO2000 model, divided into two groups: The first group consists of five parameters known as scale factors: PREC, FLEX, RESL, TEAM, PMAT. The second group consists of seventeen parameters known as effort multipliers: RELY, CPLX, DATA, RUSE, TIME, STOR, PVOL, ACAP, PCAP, PCON, APEX, PLEX, LTEX, TOOL, SCED, SITE, DOCU. Finally, the COCOMO2000 formula was obtained by formulas (1)-(7):

$$Effort = A \times [SIZE]^E \times \prod_{i=1}^{17} EM_i, \quad (1)$$

$$E = B + 0.01 \times \sum_{j=1}^{5} SFj\,, A = 2.94, B = 0.91, \quad (2)$$

$$Effort[PM] = 2.94 \times [SIZE]^E \times PEM_i, \quad (3)$$

$$PEM_i = \prod_{i=1}^{17} EM_i, \quad (4)$$

$$Time = C \times (Effort)^F, \quad (5)$$

$$F = D + 0.2 \times 0.01 \times \sum_{j=1}^{5} SFj\,, C = 3.67, D = 0.28, \quad (6)$$

$$People = \frac{Effort}{Time}, \quad (7)$$

where A and B are basic calibration constants; KSLOC (thousands of lines of source code) is the size of the software project; SF_j is five scale factor; EM_i is seventeen effort multiplier. Our experiments utilize the COCOMO2000 Post Architecture model. This model combines the effort factors and multipliers to calculate the required person-months [PM] for the implementation of a specific software project. Variables E, PEM_i, and KSLOC are used as input for four different ANN architectures constructed based on Taguchi's orthogonal vector plans: ANN-L9, ANN-L18, ANN-L27, and ANN-L36 [6, 17].

The first ANN architecture, in the COCOMO2000 approach, ANN-L9, is based on Taguchi's orthogonal vector plan (L9) with four parameters (W_i, i=($\overline{1,4}$)) and three different levels (L1, L2, L3). The experiments are conducted with 9 ANNs candidates labeled as ANN_1,..., ANN_9 Figure 1. The second constructed ANN architecture, labeled as ANN-L18, is based on Taguchi's orthogonal vector plan (L18) with eight parameters (W_i, i=($\overline{1,8}$)) and three different levels (L1, L2, L3). The experiments are conducted with 18 candidate ANNs labeled as ANN_1,..., ANN_{18} [5, 17] Figure 2. The third constructed ANN architecture, labeled as ANN-L27, is based on Taguchi's orthogonal vector plan (L27) with 13 parameters (W_i, i=($\overline{1,13}$)) and three different levels (L1, L2, and L3). The experiments are conducted with 27 candidate ANNs labeled as ANN_1, ANN_2,..., ANN_{27} Figure 3. The fourth constructed ANN architecture, labeled as ANN-L36prim, is based on Taguchi's orthogonal vector plan (L36prim) with 23 parameters (W_i, i=($\overline{1,23}$)) and three different levels (L1, L2, and L3). The experiments are conducted with 36 candidate artificial neural networks labeled as ANN_1, ANN_2,..., ANN_{36} Figure 4. Table 2 provides basic statistical data on the used datasets in all three phases of the experiment: training, testing, and validation [7, 17].

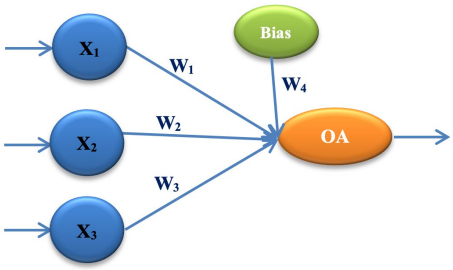

Figure 1. ANN architecture with none hidden layer (ANN-L9).

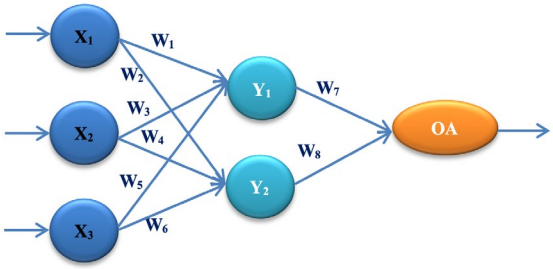

Figure 2. ANN architecture with one hidden layer (ANN-L18).

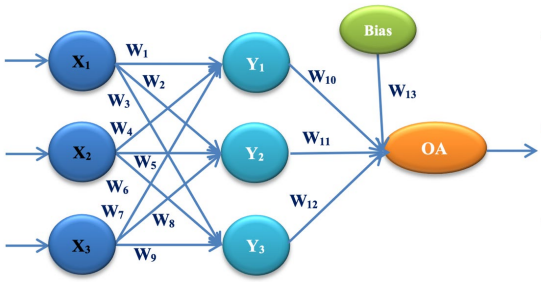

Figure 3. ANN architecture with one hidden layer (ANN-L27).

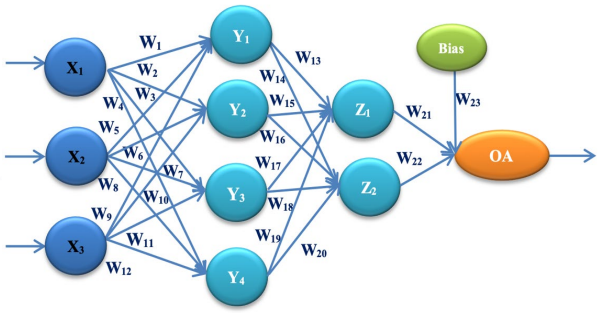

Figure 4. ANN architecture with two hidden layers (ANN-L36).

Table 2

Basic statistics about datasets (COCOMO2000).

Dataset	N	Min	Max	Mean	Std. deviation	Experiment
COCOMO2000	100	6.0	8211.0	616.0	1131.5	Training
COCOMO2000	20	28.1	606.8	277.0	206.3	Testing
COCOMO81	51	33.0	11399.9	841.8	1994.9	Validation1
NASA	60	8.4	3240.0	406.4	656.9	Validation2
Kemerer	15	23.2	1780.0	316.7	456.7	Validation3

3.2. New COSMIC FFP in combination with ANN based on Orthogonal Arrays

Unlike the previous COCOMO2000 method, which relies on three input parameters, COSMIC FFP relies on four: Entry, Exit, Read, and Write. It can be concluded that the functional size of the system represents the total number of all used messages. The system can be viewed as a four-dimensional vector space representing the total number of messages in form of: input data, reported data, written data, or data read from files. The advantage of this method is its technology independence and absence of an upper limit for the value of the functional quantity. Therefore, there is no saturation as the complexity of functionality can increase indefinitely depending on the number of messages in the system [15, 17].

Functional size is determined based on a four-dimensional vector denoted as FFP as follows (8):

$$FFP = (E, X, R, W) \tag{8}$$

where FFP represents the total number of messages in the entire observed system and is calculated as the norm of the vector, formula (9):

$$\left\|\overrightarrow{FFP}\right\| = E + X + W + R \tag{9}$$

where E = Entry, X = Exit, W = Write, and R = Read.

The first constructed ANN architecture, in the COSMIC FFP approach, labeled as ANN-L12, is based on Taguchi's orthogonal vector plan (L12) with 11 parameters (W_i, i=$\overline{1,11}$) and two different

levels (L1, L2). The experiments are conducted with 12 ANNs candidates labeled as ANN_1,..., ANN_{12} Figure 5. The second constructed ANN architecture, labeled as ANN-L36prim, is based on Taguchi's orthogonal vector plan (L36prim) with 16 parameters (W_i, i=$\overline{(1,16)}$) and combined two and three different levels (L1, L2, L3). The experiments are conducted with 36 ANNs candidates labeled as ANN_1,..., ANN_{36} Figure 6. In Table 3, basic statistics is provided for the datasets used in all three phases of the experiment: training, testing, and validation. The table includes the dataset names, the number of projects in each dataset, as well as the minimum, maximum, mean, and standard deviation values expressed in Functional Size (FS) [11,15,17].

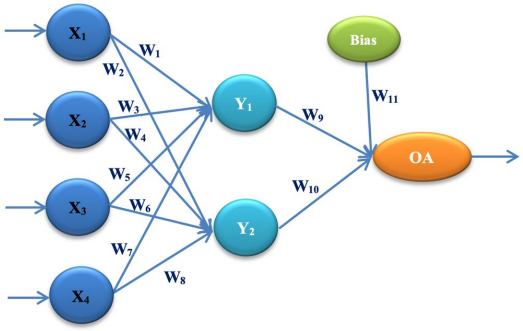

Figure 5. ANN architecture with one hidden layer (ANN-L12).

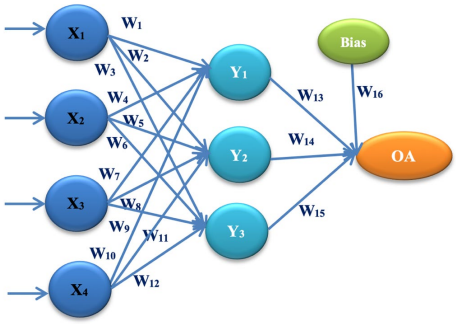

Figure 6. ANN architecture with one hidden layer (ANN-L12).

Table 3

Basic statistics about datasets (COSMIC FFP).

Datasets	N	Min	Max	Mean	Std. deviation	Experiment
ISBSG (FS<10)	52	2.0	9.0	5.404	2.4031	Training, Testing
ISBSG (10< FS<50	62	10.0	48.0	24.823	11.3203	Training, Testing
ISBSG (50< FS<100)	43	50.0	99.0	77.116	15.1441	Training, Testing
ISBSG (100< FS<500)	77	104.0	492.0	234.130	113.8159	Training, Testing
ISBSG (FS >500)	21	561.0	2090.0	1016.048	458.4442	Training, Testing
Desharnais datasets	14	140.0	3860.0	1011.429	920.4251	Validation1
Combined datasets	33	493.0	2589.0	1193.424	419.4201	Validation2

3.3. New UCP in combination with ANN based on Orthogonal Arrays

The estimated value of the UCP method utilizes twenty-one parameters for estimation, of which thirteen parameters represent the technical characteristics of the system, while the remaining eight are environmental factors. It is calculated based on G. Karner's formulas. The representation of actual effort using the UCP approach is a six-dimensional vector, where its value is calculated as the norm of the vector as follows, formulas (10), (11):

$$UCP = (UAW, UUCW, UUCP, TCF, ECF, AUCP) \quad (10)$$

$$\left\|\overrightarrow{UCP}\right\| = UAW + UUCW + UUCP + TCF + ECF + AUCP \quad (11)$$

where UAW is the unadjusted actor weight, UUCW is the unadjusted use case weight, UUCP is calculated as UUCP = UUCW + UAW, TCF is the technical factor, ECF is the environmental factor, and AUCP is calculated as AUCP = UUCP x TCF x ECF. Representation of the actual effort using the UCP approach as a four-dimensional vector is achieved calculating the value, as the vector norm, by formulas (12), (13):

$$UCP = (UAW, UUCW, TCF, ECF) \quad (12)$$

$$\left\|\overrightarrow{UCP}\right\| = UAW + UUCW + TCF + ECF \quad (13)$$

where UUCP= UAW+ UUCW, and AUCP= UUCPxTXFxECF.

In both cases, the Real Effort is obtained as the norm of the UCP vector and represents the actual functional size or the number of use case points. This method is currently widely used for effort estimation, although it is not standardized within the ISO standard like the previous two methods. Four input variables, UAW, UUCW, TCF, ECF, or six input variables, UAW, UUCW, UUCP, TCF, ECF, AUCP, are used as inputs for two different ANN architectures constructed based on Taguchi's orthogonal vector plans, namely ANN-L16 and ANN-L36prim [11, 17].

The first constructed ANN architecture, denoted as ANN-L16 in the UCP approach, is based on Taguchi's orthogonal vector plan (L16) with 15 parameters (W_i, $i=\overline{(1,15)}$) and two different levels (L1, L2). The experiments are performed with 16 ANN candidates labeled as ANN_1,..., ANN_{16} Figure 7. The second constructed ANN architecture, denoted as ANN-L36prim in the UCP approach, is based on Taguchi's orthogonal vector plan (L36prim) with 23 parameters (W_i, $i=\overline{(1,23)}$) and three different levels (L1, L2, L3). The experiments are conducted with 36 ANN candidates labeled as ANN_1, ..., ANN_{36} Figure 8. Table 4 provides basic statistical data on the datasets used in all three phases of the experiment: training, testing, and validation. The table includes the dataset names, the number of projects in each dataset, as well as the minimum, maximum, mean values, and standard deviation expressed in Real Effort (RE) [8,17].

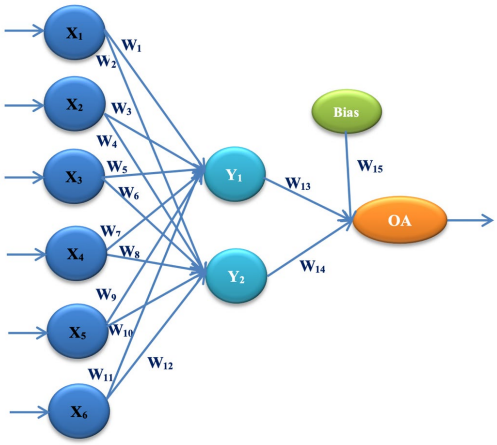

Figure 7. ANN architecture with one hidden layer (ANN-L16).

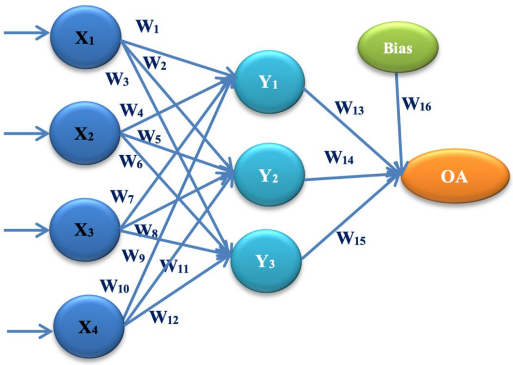

Figure 8. ANN architecture with one hidden layer (ANN-L36prim).

Table 4

Basic statistics about dataset (UCP).

Datasets	N	Min	Max	Mean	Std. deviation	Experiment
UCP Benchmark	50	5775.0	7970.0	6506.940	653.0308	Training
UCP Benchmark	21	6162.6	6525.3	6393.993	118.1858	Testing
Combined	18	2692.1	3246.6	2988.392	233.2270	Validation1
Combined Industrial	17	2176.0	3216.0	2589.400	352.0859	Validation2

3.4. Experimental setup

The algorithm of robust experimental design involves the following steps:

Step 1. Input values:

COCOMO2000 approach has three input values: X1=E, X2=PEM_i, and X3=KLOC for all four proposed architectures: ANN-L9, ANN-L18, ANN-L27, and ANN-L36;

COSMIC FFP approach has four input values: X1=Entry, X2=Exit, X3=Read and X4=Write for both proposed architectures: ANN-L12 and ANN-L36prim;

UCP approach for the first proposed architecture ANN-L16 has six input values: X1=UAW, X2=UUCW, X3=UUCP, X4=TCF, X5=ECF and X6=AUCP, while for the second proposed architecture ANN-L36prim there are four input values: X1=UAW, X2=UUCW, X3=TCF and X4=ECF.

Step 2. Fuzzification of the input values $\mu_D(X)$: $R \rightarrow [0, 1]$ formula (14):

$$\mu_D(X_i) = (X_i - X_{min})/(X_{max} - X_{min}). \quad (14)$$

Step 3. The sigmoid function, as the activation function of the hidden layer was used (15):

$$y_{i=\frac{1}{1+e^{-x_i}}}, \quad i=\overline{1,n}, \quad y_i \in [0,1]. \quad (15)$$

Example: Hidden and output layer functions for ANN-L36prim architecture (16)-(19):

$$Y1 = \frac{1}{1+e^{-(X1 \cdot W1 + X2 \cdot W4 + X3 \cdot W7 + X4 \cdot W10)}} \quad (16)$$

$$Y2 = \frac{1}{1+e^{-(X1 \cdot W2 + X2 \cdot W5 + X3 \cdot W8 + X4 \cdot W11)}} \quad (17)$$

$$Y3 = \frac{1}{1+e^{-(X1 \cdot W3 + X2 \cdot W6 + X3 \cdot W9 + X4 \cdot W12)}} \quad (18)$$

$$EstEffortANN - L36prim = \frac{1}{1+e^{-(Y1 \cdot W13 + Y2 \cdot W14 + Y3 \cdot W15 + 1 \cdot W16)}}. \quad (19)$$

In the first proposed ANN-L16 architecture, an orthogonal vector plan of two levels L1 and L2, and the initial values of the weighting factors W_i that take the values from the interval [-1, 1], were used.

The second proposed architecture has an orthogonal vector plan of three levels and the initial values of the weighting factors W_i that take the values from the interval [-1, 0, 1]. For each subsequent iteration, new weight factor values must be calculated as follows (e.g., for ANN-L16 architecture) [17], (20):

$$\begin{aligned} &W_1L1 = cost1 + cost2 + \ldots + cost8, \\ &W_1L2 = cost9 + cost10 + \ldots + cost16, \\ &\ldots. \\ &W_{15}L1 = cost1 + cost6 + \ldots + cost16, \\ &W_{15}L2 = cost2 + cost3 + \ldots + cost15, \\ &\text{where } cost(i) = \Sigma\, MRE(ANN(i)). \end{aligned} \quad (20)$$

For each subsequent iteration, the interval [-1, 1] is divided depending on the cost effect function as follows [17], (21):

$$\begin{aligned} &W_1L1new = W_1L1old \\ &W_1L2new = W_1L2old + (W_1L3old - W_1L2old)/2 \\ &W_1L3new = W_1L3old \end{aligned} \quad (21)$$

where W_1L1old, W_1L2old, and W_1L3old are values form the previous iteration.

The set of input values of each dataset converges depending on the value of the cost effect function.

Step 4. Defuzzification of the input values, formula (22):

$$X_i = (X_{min} + \mu D(X_i)) \cdot (X_{max} - X_{min}), \quad (22)$$

$OA(ANN_i) = X_i$, where $i = 16$, $i = 36$.

Step 5. Different evaluation metrics are used to validate the obtained results, formulas (23)-(28):

$$Deviation = |ActEffort - EstEffort|, \quad (23)$$

$$MAE_i = \frac{1}{n}\sum_{i=1}^{n} |ActEffort - EstEffort|, \quad (24)$$

$$MRE = Deviation/ActEffort, \quad (25)$$

$$MRE = \frac{1}{n} \cdot \sum_{i=1}^{n} MRE_i, \quad (26)$$

MMRE = *mean* (MRE), (27)

where MRE refers to Magnitude Relative Error and MAE refers to Mean Absolute Error.

For each of the experimental part in every iteration, the Gradient Descent is monitored with the condition GA<0.01, calculated as (28):

$$GA = MRE_{i1} - MRE_{i2} < 0.01, \quad where\ i = 1, \ldots, n \quad n\ is\ a\ number\ of\ ANN. \quad (28)$$

Step 6. Pearson's, Spearman's and R^2 coefficients are monitored during the experiment, formula (29):

$$Correl(X,Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \quad (29)$$

Additionally, Prediction at 25%, 30%, and 50% is the percentage of the total number of ANNs that meet the GA criterion (30).

$$PRED(x)\frac{1}{n}\cdot\sum_{i=1}^{n}\begin{cases}1, & if\ MRE \le x,\\ 0, & otherwise.\end{cases}$$

PRED(k) = count(MRE) < 25%,
PRED(k) = count(MRE) < 30%,
PRED(k)=count(MRE)<50%,
where k=25, k= 30, and k = 50 [5,8,17]. (30)

4. Overview of the Achieved Results

After conducting numerous experiments with different ANN architectures within the three approaches and utilizing various activation functions, the authors in [5, 7, 8, 17] concluded that the number of iterations in the first COCOMO2000 approach is significantly higher compared to the other two, reaching a maximum of 8 iterations for the ANN-L18 architecture, while the lowest number of iterations (6) was achieved with the ANN-L27 architecture, Figure 9. By analyzing the achieved MMRE (%), it can be observed that it is highest in the parametric COCOMO2000 model, reaching 193.1%, which is completely unacceptable and acknowledged by the authors themselves, who sought alternative solutions. The application of the new methodology on the improved COCOMO2000 model leads to a significant reduction, especially in more complex architectures. The simplest ANN-L9 architecture in this approach achieves an error of 72.0%, while the more complex ANN-L36 architecture achieves a significantly lower error of 43.3%. Using the improved models within the COSMIC FFP approach, the value of MMRE (%) decreases significantly, with the ANN-L36prim architecture achieving the lowest error on all 7 used datasets, 28.8%. For UCP approach, both architectures achieved the lowest MMRE (%) to data, 7.5%, which is a significant result considering that the lowest values in multiple research studies were around 10% Figure 10. High values of correlation coefficients (Pearson's and Spearman's rho and R^2) indicate the degree of agreement between estimated and actual effort and cost values, reaffirming the effectiveness and reliability of the proposed models. In the ANN-L36prim architecture of the UCP approach, the value of the Pearson's coefficient is high at 0.784, the Spearman's rho coefficient is very high at 0.983, and the R^2 value is again very high at 0.931, indicating an extremely strong relationship between the estimated and actual values obtained. In all models, all three coefficients have values greater than 0.6, indicating a high and very high degree of correlation [18] Table 5. In the COCOMO2000 approach, approximately one-quarter of the projects have a prediction of 25, around one-third of the total number of projects have a prediction of 30, while the prediction of 50 is higher than 50%, for more than half of the projects. In the COSMIC FFP approach, the prediction of 25 is higher than 40%, the prediction of 30 is higher than 50%, and the prediction of 50 is higher than 80%. In the UCP approach, the prediction of 25 for both models is 100%, thus the prediction of 30 and the prediction of 50 were also 100% [7, 8, 9, 17] Table 6.

Table 5
Pearson's, Spearman's rho and R^2 correlation

Correlation	ANNs	Pearson's	Spearman's rho	R^2
	ANN-L9	0.617	0.869	0.827
COCOMO2000	ANN-L18	0.615	0.864	0.861
	ANN-L27	0.714	0.862	0.862
	ANN-L36	0.866	0.904	0.869
COSMIC FFP	ANN-L12	0.794	0.787	0.803
	ANN-L36prim	0.751	0.772	0.794
UCP	ANN-L16	0.875	0.982	0.917
	ANN-L36prim	0.784	0.983	0.931

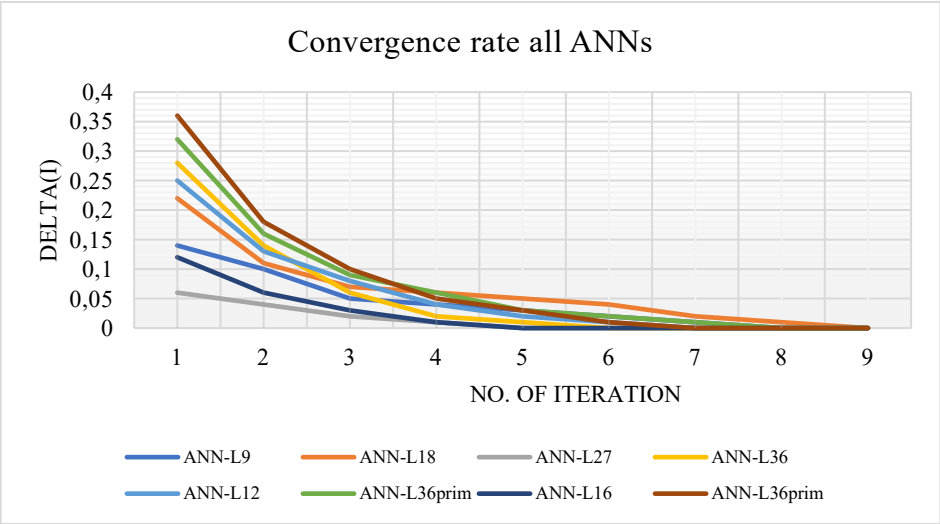

Figure 9. Convergence rate for all ANN architectures in COCOMO2000, COSMIC FFP and UCP approach.

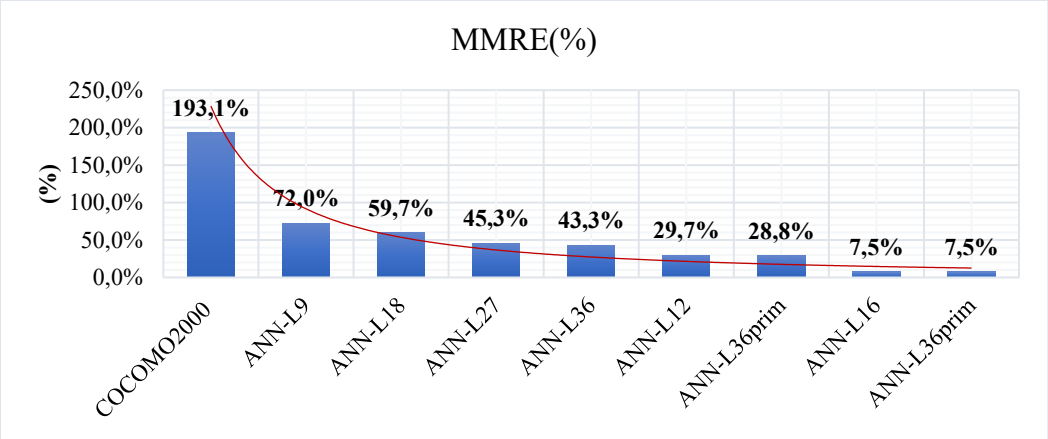

Figure 10. MMRE value for used approaches.

. **Table 6**

Prediction on 25, 30 and 50.

Prediction	ANNs	PRED(25) (%)	PRED(30) (%)	PRED(50) (%)
	ANN-L9	26.1	35.1	66.8
COCOMO2000	ANN-L18	24.3	28.3	58.2
	ANN-L27	27.2	31.2	54.8
	ANN-L36	25.7	34.2	57.4
COSMIC FFP	ANN-L12	43.4	50.5	81.2
	ANN-L36prim	44.8	56.0	87.0
UCP	ANN-L16	100.0	100.0	100.0
	ANN-L36prim	100.0	100.0	100.0

5.Discussion

In the new COCOMO2000 model, four different neural network architectures were used, along with five datasets divided into three clusters, sigmoid activation function, fuzzification method, and Taguchi method for effort and cost estimation. The experiments showed that this approach guarantees reliable and stable results, as confirmed by monitoring the MMRE values. The convergence speed of the

architecture depends on the cost function and project nature, and the number of iterations is less than 10. More complex architecture exhibits faster convergence, shorter iteration time, and minimal MMRE. Fuzzification as an encoding function along with clustering of input data proposed in [7] partially alleviate the heterogeneous structure of the projects. The ANN-L36 architecture yields the best results with the lowest MMRE of 43.3%, which is nearly twice as good as many previous studies where the lowest error was 80.9%. The advantages of the model include short estimation time, high coverage of actual effort, and minimal MMRE. The drawback is the need to find new methods to further reduce the MMRE. There are no specific limitations for applying this approach [5, 9, 17].

The new COSMIC FFP model in this study demonstrates that the application of two different ANN architectures, based on Taguchi Orthogonal Arrays, further reduces the MMRE value. This model belongs to user-functional requirement-based approaches with four input values. By using data input clustering methods from the ISBSG dataset and fuzzification as encoding method, the different project structures are successfully controlled and mitigated [16]. The results show that models constructed based on these two proposed ANN architectures (ANN-L12 and ANN-L36prim) significantly reduce the MMRE value by approximately 14.5% compared to the previous experiment on the improved COCOMO2000 model. The efficiency and stability of the proposed model are confirmed through the calculation of two correlation coefficients, while tracking the predictions on three different criteria further validates the accuracy and reliability of this model. Additional advantages of this approach include a lower number of iterations (5 to 6), the use of simple ANN architectures, optimization through Taguchi Orthogonal Arrays, wide coverage of different functional size values of software projects, and the utilization of the ISBSG repository of real project data collected from various companies. This approach is not limited and can be applied in various business and scientific domains [15, 17].

The new UCP model utilizes two different ANN architectures and four distinct datasets, a sigmoid activation function, fuzzification method, and Taguchi method for software development effort and cost estimation. This model yields significantly better results compared to the previous two, as evidenced by the MMRE value and convergence rate of each architecture. Based on the three parts of the experiment, it is concluded that the ANN-L16 architecture converges after the fourth iteration, resulting in an MMRE value of only 7.5%, which is 35.8% better than the first COCOMO2000 model. The error value of the UCP model is 21.3% lower than the second proposed COSMIC FFP model. Both architectures of this model demonstrate 100% accuracy in predictions. The advantages of this model include a lower number of iterations (4-6), simple architectures, high coverage of different effort values, and the lowest MMRE value of 7.5%. A potential drawback is the need for finding new methods to further reduce of the MMRE. This model can be used independently or in combination with the previous two, depending on the company's historical data. Although not standardized, it is increasingly used in the software industry for effort estimation for software project implementations [8, 17].

6. Conclusion

The proposed, new models in the studies [5, 7, 8, 9, 12, 16, 17, 18] can inspire the development of efficient tools for accurate and reliable estimation of effort and costs in all phases of software project development. These models target software companies, engineers, and project managers, enabling them to obtain fast and precise results for proper assessment of project requirements. Using these models can significantly reduce common problems in software engineering, improving efficiency and facilitating work for professionals and teams. The selection of models relies on historical data from software companies and their direct applicability to real-world situations. Looking ahead, our endeavors will encompass the utilization of specific types of Recurrent Neural Networks and the implementation of WHAT-IF simulations to enhance this capability.

7. References

[1] A Guide to the Project Management Body of Knowledge (PMBOK Guide). Third Edition, Project Management Institute, Inc. 2004. ISBN: 1-930699-45-X.

[2] B. W. Boehm, C. Abts, and S. Chulani. Software development cost estimation approaches-A survey. Annals of software engineering, 10 (1): 177-205, 2000.

[3] P. S. Kumar et al. Advancement from neural networks to deep learning in software effort estimation: Perspective of two decades. Computer Science Review, 28 (11):100288, 2020.
[4] P. S. Kumar and H. Behera. Estimating software effort using neural network: An experimental investigation. Computational Intelligence in Pattern Recognition, Proceedings of CIPR, pages 165–180, Springer, 2020.
[5] D. Rankovic, N. Rankovic, M. Ivanovic, & L. Lazic. Convergence rate of Artificial Neural Networks for estimation in software development projects. Information and Software Technology, 138, 106627, Elsevir, 2021.
[6] S. Devnani-Chulani et al. Calibration Approach and Results of the COCOMO II Post-Architecture Model. In Proceedings of the 20th Annual Conference of the International Society of Parametric Analysts (ISPA) and the 8th Annual Conference of the Society of Cost Estimating and Analysis (SCEA), 1998.
[7] N. Rankovic, D. Rankovic, M. Ivanovic, & L. Lazic. A new approach to software effort estimation using different artificial neural network architectures and Taguchi orthogonal arrays. Ieee access, 9, 26926-26936, 2021.
[8] N. Rankovic, D. Rankovic, M. Ivanovic, & L. Lazic. A novel UCP model based on artificial neural networks and orthogonal arrays. *Applied Sciences*, *11*(19), 8799, 2021.
[9] N. Rankovic, D. Rankovic, M. Ivanovic, & L. Lazic. Influence of input values on the prediction model error using artificial neural network based on Taguchi's orthogonal array. Concurrency and Computation: Practice and Experience, 34(20), e6831, 2022.
[10] N. Rankovic, D. Rankovic, M. Ivanovic, & L. Lazic. (2021). Improved effort and cost estimation model using artificial neural networks and taguchi method with different activation functions. Entropy, 23(7), 854, 2021.
[11] J. Popović. Enhancing methods for effort estimation in software projects. Doctoral dissertation, University of Belgrade, School of Electrical Engineering, Belgrade, Serbia, 2016.
[12] N. Rankovic, D. Rankovic, M. Ivanovic, & L. Lazic. Artificial Neural Network Architecture and Orthogonal Arrays in Estimation of Software Projects Efforts. In 2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA) (pp. 1-6). IEEE, 2021.
[13] B. W. Boehm. Safe and simple software cost analysis. IEEE software, 17 (5): 14-17, 2000.
[14] A. J. Albrecht and J. E. Gaffney. Software function, source lines of code, and development effort prediction: a software science validation. IEEE transactions on software engineering, (6): 639-648, 1983.
[15] R. Meli et al. On the applicability of COSMIC-FFP for measuring software throughout its life cycle. In Proceedings of the 11th European Software Control and Metrics Conference, pages 18-20, Springer, 2000.
[16] N. Rankovic, D. Rankovic, M. Ivanovic, & L. Lazic. COSMIC FP method in software development estimation using artificial neural networks based on orthogonal arrays. Connection Science, 34(1), 185-204, 2022.
[17] N. Ranković, Estimation of Effort and Costs in the Development of Software Projects Using Artificial Neural Networks Based On Taguchi's Orthogonal Vector Plans (Doctoral dissertation, University of Novi Sad (Serbia)), 2022.
[18] D. Rankovic, N. Rankovic, M. Ivanovic, & L. Lazic. The Generalization of Selection of an Appropriate Artificial Neural Network to Assess the Effort and Costs of Software Projects. In Artificial Intelligence Applications and Innovations: 18th IFIP WG 12.5 International Conference, AIAI 2022, Hersonissos, Crete, Greece, June 17–20, 2022, Proceedings, Part I (pp. 420-431). Cham: Springer International Publishing, June 2022.