UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCES
DEPARTMENT OF
MATHEMATICS AND INFORMATICS

Estimation of effort and costs in the development of software projects using artificial neural networks based on Taguchi's orthogonal vector plans

- Doctoral Dissertation -

Procena napora i troškova za razvoj softverskih projekata pomoću veštačkih neuronskih mreža zasnovanih na Tagučijevim ortogonalnim vektorskim planovima

- Doktorska disertacija -

Mentor:
prof. dr Mirjana Ivanović

Kandidat:
Nevena Ranković

Novi Sad, 2021

УНИВЕРЗИТЕТ У НОВОМ САДУ
ПРИРОДНО-МАТЕМАТИЧКИ ФАКУЛТЕТ

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА[1]

Врста рада:	Докторска дисертација
Име и презиме аутора:	Невена Ранковић
Ментор (титула, име, презиме, звање, институција)	др Мирјана Ивановић, редовни професор, Природно-математички факултет, Универзитет у Новом Саду
Наслов рада:	Процена напора и трошкова за развој софтверских пројеката помоћу вештачких неуронских мрежа заснованих на Тагучијевим ортогоналним векторским плановима
Језик публикације (писмо):	Енглески језик и Српски језик (латиница)
Физички опис рада:	Унети број: Страница: 195 Поглавља: 7 Референци: 133 Табела: 80 Слика: 64 Графикона: 0 Прилога: 0
Научна област:	Рачунарске науке
Ужа научна област (научна дисциплина):	Софтверско инжењерство Вештачка интелигенција
Кључне речи / предметна одредница:	процена напора и трошкова, вештачке неуронске мреже, Тагучијеви ортогонални векторски планови, робусни дизајн експеримента
Резиме на језику рада:	Savremena softverska industrija zahteva brzo, kvalitetno i precizno predviđanje napora i troškova, pre nego što se stvarni napor uloži u realizaciju softverskog proizvoda. Ovako postavljeni zahtevi predstavljaju izazov za bilo koju softversku kompaniju, koja mora biti spremna da ispuni postavljena očekivanja naručioca softvera. Glavni faktor uspešnog razvoja softverskih projekata i smanjenja rizika od grešaka je adekvatna procena uloženog napora i troškova tokom njegove realizacije. U ovoj doktorskoj disertaciji biće analizirani dosadašnji različitih pristupi i modeli koji nisu u najvećoj meri bili dovoljno precizni i efikasni, što je za posledicu imalo samo oko 30% uspešno realizovanih softverskih rešenja. Glavni cilj biće predstavljanje tri nova poboljšana modela zasnovana na efikasnim alatima veštačke inteligencije, veštačkim neuronskim mrežama

[1] Аутор докторске дисертације потписао је и приложио следеће Обрасце:
5б – Изјава о ауторству;
5в – Изјава о истоветности штампане и електронске верзије и о личним подацима;
5г – Изјава о коришћењу.
Ове Изјаве се чувају на факултету у штампаном и електронском облику и не кориче се са тезом.

	(engl. Artificial neural networks/ANN). Sva tri poboljšana modela koriste različite arhitekture veštačkih neuronskih mreža, konstruisanih na osnovu Tagučijevih ortogonalnih vektorskih planova. Cilj je optimizacija poboljšanih modela kako bi se izbeglo ponavljanje broja eksperimenata i dugotrajno vreme za njihovo obučavanje, odnosno treniranje. Primenom metode klasterizacije na više različitih skupova realnih projekata dodatno se ublažava njihova heterogena struktura. Dodatno, ulazne vrednosti projekata se homogenizuju metodom fazifikacije čime se postiže još veća pouzdanost i tačnost dobijenih rezultata. Optimizacija Tagučijevom metodom uz povećanje pokrivenosti širokog spektra različitih projekata, dovodi do efikasnog i uspešnog dovršavanja što više različitih softverskih projekta. Glavni doprinosi ove disertacije su: konstruisanje i identifikovanje najboljeg modela za procenu napora i troškova, odabir najbolje ANN arhitekture čije vrednosti najbrže konvergiraju minimalnoj magnitudnoj relativnoj greški, postizanje malog broja izvedenih eksperimenata, smanjeno vreme procene softverskog napora zbog stope konvergencije. Uvode se i dodatni kriterijumi i ograničenja za nadgledanje i izvršavanje eksperimenata pomoću preciznog algoritma za izvršavanje nad sva tri nova predložena modela. Pored praćenja brzine konvergencije svake arhitekture veštačke neuronske mreže, prati se i uticaj ulaznih veličina svakog modela na promenu vrednosti magnitudne relativne greške modela. Na ovakav način konstruisani modeli eksperimentalno su više puta proveravani i potvrđeni na različitim skupovima realnih projekata i mogu se praktično primenjivati, a dobijeni rezultati ukazuju da su postignute vrednosti grešaka niže od dosadašnjih predstavljenih. Samim tim se predloženi modeli u ovoj disertaciji mogu pouzdano primenjivati i koristiti ne samo za procenu napora i troškova za razvoj softverskih već i za razvoj projekata u drugim oblastima industrije i nauke.
Датум прихватања теме од стране надлежног већа:	
Датум одбране: (Попуњава одговарајућа служба)	
Чланови комисије: (титула, име, презиме, звање, институција)	Председник: др Владимир Курбалија, редовни професор, Природно-математички факултет, Универзитет у Новом Саду Члан: др Милош Рацковић, редовни професор, Природно-математички факултет, Универзитет у Новом Саду Члан: др Тихана Галинац Грбац, редовни професор, Технички факултет у Пули, Јурај Добрила Универзитет у Пули. Члан: др Александра Клашња-Милићевић, ванредни професор, Природно-математички факултет, Универзитет у Новом Саду. Члан: др Љубомир Лазић, редовни професор, Рачунарски факултет, Универзитет Унион Београд.
Напомена:	

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCES

KEY WORD DOCUMENTATION[2]

Document type:	Doctoral dissertation
Author:	Nevena Ranković
Supervisor (title, first name, last name, position, institution)	Ph.D Mirjana Ivanović, , full professor, Faculty of Sciences, University of Novi Sad
Thesis title:	„Estimation of effort and costs in the development of software projects using artificial neural networks based on Taguchi's orthogonal vector plans“
Language of text (script):	English language and Serbian language (latin script)
Physical description:	Number of: Pages: 195 Chapters: 7 References: 133 Tables: 80 Illustrations: 64 Graphs: 0 Appendices: 0
Scientific field:	Computer Science
Scientific subfield (scientific discipline):	Software Engineering Artificial Intelligence
Subject, Key words:	estimation of effort and costs, artificial neural networks, Taguchi's orthogonal vector plans, robust design of experiment
Abstract in English language:	The modern software industry requires fast, highquality, and accurate forecasting of efforts and costs before the actual effort is invested in realizing the software product. Such requirements are a challenge for any software company, which must be ready to meet the expectations of the software customer. The main factor in the successful development of software projects and reducing the risk of errors is an adequate of the effort and costs invested during its implementation. In this doctoral dissertation, different approaches and models that have not been sufficiently precise and efficient so far will be analyzed, which resulted in only about 30% of successfully implemented software solutions. The main goal is to present three new, improved models based on efficient artificial intelligence tools, artificial neural networks. All three improved models use different architectures of artificial neural networks (ANN), constructed based on Taguchi's orthogonal vector plans. The goal is to optimize the improved models to avoid repeating the number of experiments and long time for their training. Applying the clustering method to several

[2] The author of doctoral dissertation has signed the following Statements:
5б – Statement on the authority,
5в – Statement that the printed and e-version of doctoral dissertation are identical and about personal data,
5г – Statement on copyright licenses.
The paper and e-versions of Statements are held at he faculty and are not included into the printed thesis.

	different sets of real projects further mitigates their heterogeneous structure. In addition, the input values of the projects are homogenized by the method of fuzzification, which achieves even greater reliability and accuracy of the obtained results. Optimization by the Taguchi method and increasing the coverage of a wide range of different projects leads to the efficient and successful completion of as many different software projects as possible. The main contributions of this paper are: constructing and identifying the best model for estimating effort and cost, selecting the best ANN architecture whose values converge the fastest to the minimum magnitude relative error, achieving a small number of experiments, reduced software effort estimation time due to convergence rate. Additional criteria and constraints are introduced to monitor and execute experiments using a precise algorithm to execute all three new proposed models. In addition to monitoring the convergence rate of each artificial neural network architecture, the influence of the input values of each model on the change in the value of the magnitude relative error of the model is also monitored. The models constructed in this way have been experimentally checked and confirmed several times on different sets of real projects and can be practically applied, and the obtained results indicate that the achieved error values are lower than those presented so far. Therefore, the proposed models in this dissertation can be reliably applied and used to assess the efforts and costs for software development and projects in other areas of industry and science.
Accepted on Scientific Board on:	
Defended: (Filled by the faculty service)	
Thesis Defend Board: (title, first name, last name, position, institution)	President: Ph. D Vladimir Kurbalija, full professor, Faculty of Sciences, University of Novi Sad Member: Ph.D Miloš Racković, Ph, full professor, Faculty of Sciences, University of Novi Sad Member: Ph. D Tihana Galinac Grbac, full professor, Faculty of Engineering, Juraj Dobrila University of Pula Member: Ph. D Aleksandra Klašnja-Milićević, associate professor, Faculty of Sciences, University of Novi Sad Member: Ph. D Ljubomir Lazić, full professor, School of Computing, Union University Belgrade
Note:	

Contents

List of Figures

List of Tables

Shorthands and Abbreviations Used

ANN	Artificial Neural Network
GA	Gradient Descent
MMRE	Mean Magnitude Relative Error
COCOMO2000	Constructive Cost Model 2000
FPA	Function Point Analysis
ISBSG	International Software Benchmarking Standards repository
UCP	Use Case Point
MAE	Mean Absolute Error
MRE	Magnitude Relative Error
PRED	Prediction
FFP	Full Factorial Plan
IFPUG	International Function Point Users Group
NESMA	Netherlands Software Users Metrics Association
COSMIC FFP	COmmon Software Measurement International Consortium Full Function Point
COBRA	Cost Estimation, Benchmarking, and Risk Assessment
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
SLIM	Software Life Cycle Management
SEER	System Evaluation and Estimation of Resource
SEM	Software Evaluation Model
COCOMO81	Constructive Cost model 81
FP	Function Point
PM	Person-months
NFFPCM	Neuro-Fuzzy Function Point Calibration Model
HCI	Human-centric
MFFN	Multilayer Feed Forward Neural Network
ANN-L9	Artificial Neural Networks based on the orthogonal plan of Taguchi (L9)
ANN-L18	Artificial Neural Networks based on the orthogonal plan of Taguchi (L18)
ANN-L27	Artificial Neural Networks based on the orthogonal plan of Taguchi (L27)
ANN-L36	Artificial Neural Networks based on the orthogonal plan of Taguchi (L36)
ANN-L12	Artificial Neural Networks based on the orthogonal plan of Taguchi (L12)
ANN-L16	Artificial Neural Networks based on the orthogonal plan of Taguchi (L16)
ANN-L36prim	Artificial Neural Networks based on the orthogonal plan of Taguchi (L36prim)
SVM	Support Vector Regression
ML	Machine Learning

RBF	Radial Basis Function
OATM	Orthogonal Array Tunning Method

Acknowledgments

First and foremost, I would like to thank my advisor, prof. Dr. Mirjana Ivanovic, on many years of patience, understanding, and advice that she has always unreservedly given to me. Thank you for recognizing me as a future lecturer during my student days and for always encouraging me to learn and progress more and more. Even as a mentor on my master's thesis, she helped me with her experience and knowledge to research and write in an adequate way. With her I gained the confidence to do this doctoral dissertation and I owe her infinite gratitude and respect.

I would also like to thank the members of the committee, prof. Dr. Vladimir Kurbalija, prof. Dr. Milos Rackovic, prof. Dr. Tihana Galinac Grbac, prof. Dr. Aleksandra Klasnja-Milicevic and prof. Dr. Ljubomir Lazic for their time spent on reading and commenting on this thesis.

I would like to express special gratitude to prof. Dr. Ljubomir Lazic for introducing me to the topic of this thesis and for many hours spent in meetings regarding it. He was always there for teaching me important research lessons and appreciating my work and commitment, believing in me and having patience for all my questions. Also, he taught me how to be a researcher and encouraged me to determine appropriate solution methodologies to organize research findings.

My greatest gratitude goes to my family.

Abstract

The modern software industry requires fast, high-quality, and accurate forecasting of efforts and costs before the actual effort is invested in realizing the software product. Such requirements are a challenge for any software company, which must be ready to meet the expectations of the software customer. The main factor in the successful development of software projects and reducing the risk of errors is an adequate estimation of the effort and costs invested during its implementation.

In this doctoral dissertation as a starting point, different approaches and models that have not been sufficiently precise and efficient so far will be analyzed, which resulted in only about 30% of successfully implemented software solutions.

The main goal of the dissertation is to present three new, improved models based on efficient artificial intelligence tools, artificial neural networks. All three improved models use different architectures of artificial neural networks (ANN), constructed based on Taguchi's orthogonal vector plans. The idea is to optimize the improved models to avoid repeating the number of experiments and the long time needed for their training. Applying the clustering method to several different sets of real projects further mitigates their heterogeneous structure. In addition, the input values of the projects are homogenized by the method of fuzzification, which achieves even greater reliability and accuracy of the obtained results. Optimization by the Taguchi method and increasing the coverage of a wide range of different projects leads to the efficient and successful completion of as many different software projects as possible.

The main contributions of this dissertation are:

- constructing and identifying the best model for estimating effort and cost,
- selecting the best ANN architecture whose values converge the fastest to the minimum magnitude relative error,
- achieving a small number of experiments, and
- reduced software effort estimation time due to convergence rate.

Additional criteria and constraints are introduced to monitor and execute experiments using a precise algorithm to execute all three new proposed models. In addition to monitoring the convergence rate of each artificial neural network architecture, the influence of the input values of each model on the change in the value of the magnitude relative error of the model is also monitored. The models constructed in this way have been experimentally checked and confirmed several times on different sets of real projects and can be practically applied.

The obtained results indicate that the achieved error values are lower than those presented so far. Therefore, the proposed models in this dissertation can be reliably

applied and used to assess the effort and costs for software development and projects in other areas of industry and science.

Izvod

Savremena softverska industrija zahteva brzo, kvalitetno i precizno predviđanje napora i troškova, pre nego što se stvarni napor uloži u realizaciju softverskog proizvoda. Ovako postavljeni zahtevi predstavljaju izazov za bilo koju softversku kompaniju, koja mora biti spremna da ispuni postavljena očekivanja naručioca softvera. Glavni faktor uspešnog razvoja softverskih projekata i smanjenja rizika od grešaka je adekvatna procena uloženog napora i troškova tokom njegove realizacije. U ovoj doktorskoj disertaciji, kao početna tačka, biće analizirani dosadašnji različitih pristupi i modeli koji nisu u najvećoj meri bili dovoljno precizni i efikasni, što je za posledicu imalo samo oko 30% uspešno realizovanih softverskih rešenja.

Glavni cilj ove disertacije biće predstavljanje tri nova poboljšana modela zasnovana na efikasnim alatima veštačke inteligencije, veštačkim neuronskim mrežama (*engl.* Artificial neural networks/ANN). Sva tri poboljšana modela koriste različite arhitekture veštačkih neuronskih mreža, konstruisanih na osnovu Tagučijevih ortogonalnih vektorskih planova. Ideja ove disertacije je i optimizacija poboljšanih modela kako bi se izbeglo ponavljanje broja eksperimenata i dugotrajno vreme za njihovo obučavanje, odnosno treniranje. Primenom metode klasterizacije na više različitih skupova realnih projekata dodatno se ublažava njihova heterogena struktura.

Dodatno, ulazne vrednosti projekata se homogenizuju metodom fazifikacije čime se postiže još veća pouzdanost i tačnost dobijenih rezultata. Optimizacija Tagučijevom metodom uz povećanje pokrivenosti širokog spektra različitih projekata, dovodi do efikasnog i uspešnog dovršavanja što više različitih softverskih projekta.

Glavni doprinosi ove disertacije su:

- konstruisanje i identifikovanje najboljeg modela za procenu napora i troškova,
- odabir najbolje ANN arhitekture čije vrednosti najbrže konvergiraju minimalnoj magnitudnoj relativnoj greški,
- postizanje malog broja izvedenih eksperimenata,
- smanjeno vreme procene softverskog napora zbog stope konvergencije.

Uvode se i dodatni kriterijumi i ograničenja za nadgledanje i izvršavanje eksperimenata pomoću preciznog algoritma za izvršavanje nad sva tri nova predložena modela. Pored praćenja brzine konvergencije svake arhitekture veštačke neuronske mreže, prati se i uticaj ulaznih veličina svakog modela na promenu vrednosti magnitudne relativne greške modela. Na ovakav način konstruisani modeli eksperimentalno su više puta proveravani i potvrđeni na različitim skupovima realnih projekata i mogu se praktično primenjivati, a dobijeni rezultati ukazuju da su postignute vrednosti grešaka niže od dosadašnjih. Samim tim se predloženi modeli u ovoj disertaciji mogu pouzdano

primenjivati i koristiti ne samo za procenu napora i troškova za razvoj softverskih već i za razvoj projekata u drugim oblastima industrije i nauke.

Preface

Estimation of effort and costs for realizing software projects is one of the most critical problems and tasks in software companies. In practice, software teams usually make estimation based on the knowledge of experts in the field or based on similarities with previous projects. Estimation of effort and costs is a crucial factor, which will determine the beginning of the project, the conditions, and limitations under which it will be realized and successfully completed, and applied in practice. The main reason for the insufficiently successful completion of projects still lies in insufficiently reasonable and inaccurate estimates. As a consequence of this inadequate, various world statistics show that only a third of projects are successfully completed and practically implemented. At the same time, almost half remain unresolved, i.e., break through the budget and implementation time, and about 20% of them fail entirely.

This dissertation aims to analyze the existing estimation methods and to present and experimentally confirm new, improved models based on them, giving better results than the existing ones. The idea is to significantly increase the accuracy, efficiency, and reliability of the proposed models with the currently most powerful artificial intelligence tools, such as artificial neural networks, and appropriate optimization methods, such as the Taguchi method, with additional criteria and methods. Then it is necessary to repeatedly check and confirm the correctness and accuracy of the new models of effort and cost estimation on real different datasets. The introduction of additional methods and techniques will improve the efficiency of the proposed models, which will have no limitations and can still be tested on new sets of real projects from any area for which the software project is implemented.

Additionally, the influence of different parameters, both input values and weight coefficients for different architectures of artificial neural networks, will be monitored. The aim is to identify the best model and the best architecture of an artificial neural network that will experimentally give the best results in the proposed models, i.e., the lowest error value. The proposed models aim to better, more accurate, and faster of efforts and costs and thus increase project implementation success. This would significantly improve the field of software engineering.

The dissertation is structured in the following way:

- The first chapter will explain in more detail the problems of effort and cost estimation.
- The second chapter will present various parametric and non-parametric approaches and models that have been most commonly used in practice to solve problems.

- The third chapter will describe in detail and present three new, improved models from three different approaches and the methodology used in the proposed models.
- The fourth chapter will present experimentally obtained and repeatedly tested results.
- The fifth chapter explains the general conclusions and reaffirms the reliability of the proposed approaches.
- The sixth chapter will present the application of the proposed models and an explanation for their scientific contribution and general social significance.
- The last chapter brings concluding remarks.

Chapter 1: Introduction

The essential activity in the software development process is effort, which involves estimating the time and money to complete a software project successfully. Project development is of exceptional importance, both for the project customers and for the project implementers. The amount of money needed to invest in a project affects whether the project will start or not or whether it will be completed successfully. In practice, the price of a project is usually compared to the cost of similar projects, which have been successfully completed. Time and necessary money are not the only factors that define the beginning of the project, but other parameters must be taken into accounts, such as quality, the complexity of the project, overtime work of the team, and others.

Preliminary estimates of effort can usually lead to delays in project implementation, requests for additional funds, overtime work of experts, and the like. Also, inaccurate estimates can directly affect the quality of the software. Due to not considering all the necessary parameters for implementing the software, certain activities, such as additional testing, completion of documentation, and additional definition of user requirements, are reduced to a minimum effort. All this can lead to a vast number of unexplained projects, which means that they have either exceeded the deadline or need more money for their implementation. The consequences of undefined and unexplained projects are currently a mirror of the software industry.

Based on many years of research by the Project Management Institute, their analysis [1], and research by Professor Barry Boehm [2], it was concluded that the effort and cost converge to accurate values depending on the stages of project development. This characteristic can be presented as a “cone of uncertainty” of the effort and convergence estimate towards the final error value, see Figure 1.

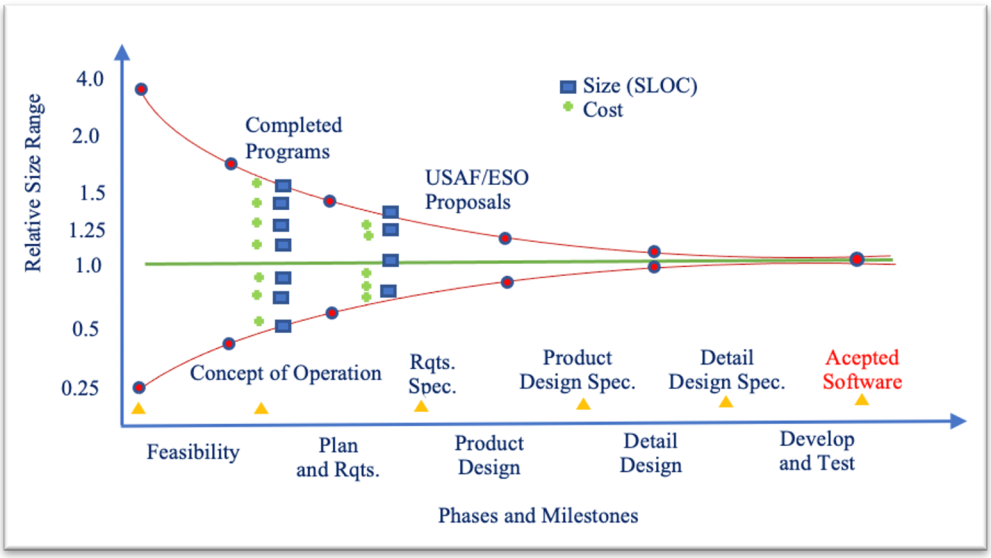


Figure 1. "Cone of uncertainty" according to the different phases of the project.
Slika 1. "Kupa nesigurnosti" u zavisnosti od različitih faza projekta.

From Table 1. [3], [4], [5], it can be concluded that in the period from 2010 to 2020, the number of successfully completed projects decreased by 6%. The percentage of successfully completed projects for 2020 is only 31%. The number of failed projects also decreased by 2%, and in 2020 is 19%. The number of unresolved projects has increased by 8% and now stands at 50%, see Figure 2 [4], [5].

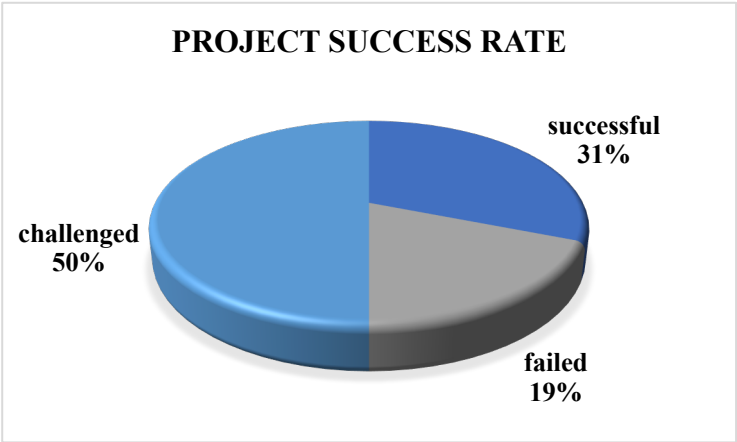


Figure 2. Graphic representation of successful, unsuccessful, and unresolved projects for 2020.
Slika 2. Grafička reprezentacija uspešnih, neuspešnih i nerešenih projekata za 2020.god.

Table 1. Percentage of successful, unsuccessful, and unresolved projects for 10 years.
Tabela 1. Procenat uspešnih, neuspešnih i nerešenih projekata u proteklih 10 god.

Project (%)	2010	2015	2020
successful	37	36	31
failed	21	17	19
challenged	42	47	50

Based on the report of the Standish Group [6], which maintains a repository on global statistics on successful, unsuccessful, and unresolved projects, can be concluded that: in unsuccessful and unresolved projects, most of them exceed deadlines and budgets; deadlines are exceeded by about 80% of projects; the budget exceeds about 50% of the projects. It can be also stated that time and budget are the two most important causes of unsuccessful project completion. Therefore, the most complex and essential task facing software teams is a reliable, accurate and fast of the effort required to implement the project. The conclusions of the research conducted by the Standish Group also show that the probability of failure of more extensive projects is higher [4], [5], [6]. That is, more complex projects have a higher chance of failure [7], [8].

Software companies use a variety of software tools and services to meet customer requirements. In order to construct reliable software of high standards, many researchers [9], [10], [11], [12] have proposed different combinations of parametric and non-parametric models of effort and cost estimation. It is necessary to analyze and experimentally check the most successful methods and models so far to be adequately improved [13], [14], [15].

1.1 Software evaluation problems

Previous methods of assessing software development have been based on unreliable and inaccurate methods, techniques, and models. The result of such inadequate processes is a huge number of unsuccessful and unrealized projects. The most commonly used methods are similarity method, method of analysis and synthesis, based on the knowledge of experts in these fields, and various parametric (algorithmic) methods [13].

1. Similarity method: effort and cost estimation is done based on similar projects. In practice, there are no completely similar projects, and this method can be used to assess certain functionalities of a similar project that can be used. It is necessary to identify differences that must again be assessed by similarity or some other method.

- The advantage of the similarity method is that the of effort and costs is performed on real projects from practice.

- The disadvantages are reflected in the fact that in practice, there are no similar projects, so the method is reduced to the of similar projects in which there are differences that are re-assessed, which increases the time [13], [14], [15].

2. Analysis/Synthesis: estimating effort and cost using this method is done by dividing it into smaller parts of the project, which are evaluated individually. Then, based on them, an overall of the project's success is made.

- Advantages of evaluation of such approach are: parts of the project are easier and simpler to assess.
- Disadvantages are: This method requires more time, but the accuracy of the evaluation in the total sum of smaller parts of the project is less reliable [13], [16].

3. Expert knowledge-based : this method involves estimating the effort and cost based on the experience of one/more experts who have worked on similar projects.

- Advantages of this approach are: is fast and straightforward.
- Disadvantages are: the knowledge of experts is subjective, and if the is made based on the opinion of several experts, it is not easy to reach an objective solution [17], [18].

4. Parametric (algorithmic) methods: estimation of effort and costs using this method is performed based on the size (measure) of the project, and then an algorithm is constructed to determine the time and costs for its implementation.

- Advantages of this approach are: is objective, fast, and easy to use.
- Disadvantages are: this method requires sound knowledge and monitoring of historical data, and thus the algorithm is based on historical, collected data [13], [19], [20].

1.2 Subject and goal of the research

Assessing the effort and costs required to implement projects is a significant factor in the software development process. It largely depends on a vast number of software functional and non-functional requirements. Functional requirements such as business rules, transaction corrections, adjustments, and cancellations, administrative functions, authentication, authorization levels, audit tracking, external interfaces, certification requirements, reporting requirements and historical data. Non-functional requirements such as speed or how fast an application responds to commands, security of sensitive data,

portability, compatibility, capacity, reliability, environment, localization, etc. The influence of various factors on the of effort and costs during the development of a software project is shown in Figure 3 [13].

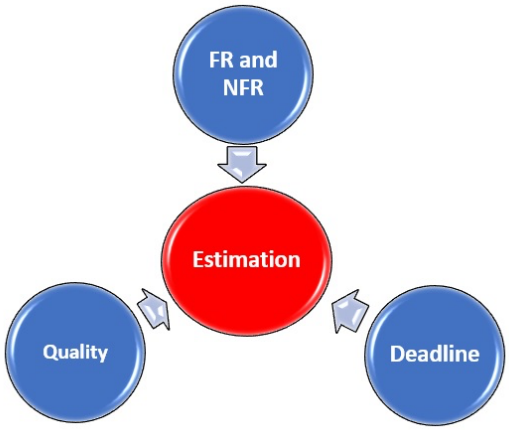


Figure 3. The influence of various factors on the effort and costs for the implementation of software projects.
Slika 3. Uticaj različitih faktora na napor i troškove potrebne za implementaciju softverskih projekata.

In addition to the total volume of necessary work, which takes into account the functional and non-functional requirements of users, factors of great importance are:

1. Software quality: meeting requirements in accordance with the highest standards defined by the organization. These are most often: processing a large amount of data, increased speed and sending requests and responses, etc., which requires additional time to meet the requirements.
2. Time limit: this factor is the most influential in project failure. Therefore, software development must not depend on the number of team members, additional time to improve the quality of software, implementation of the software itself, time of delivering the first version, and waiting for the completion of overall project tasks for a given software product [13], [21], [22], [23].

This dissertation's subject is the analysis of existing methods, techniques, and models for estimating efforts and costs to realize software projects and present new, improved models through three different approaches.

The performed experiments aim to construct three new, improved models based on different architectures of artificial neural networks. Improvement of existing methods and models would be realized by applying artificial intelligence, which would serve as a

powerful tool for better results when evaluating software. Using different architectures of ANN constructed based on Taguchi's orthogonal vector plans [7], [8], [24], [25], [26], [27], it is possible to optimize the proposed models. A realistic and correct estimation is of utmost importance in managing software projects to avoid cancellation, deterioration, or exceeding the time or budget provided for the project. Improving existing methods and models would reduce the risks of project cancellation or failure. This would be of great importance both for software companies and customers who expect the product within the set budget and time frame [28], [29].

Additional value of this dissertation is to experimentally test and determine the best model for reliable, efficient, and accurate of efforts and costs for the implementation of software projects.

This model should meet the following set criteria:

1. Choosing the simplest ANN architecture

In the experimental part, which involves training each ANN, the most straightforward architecture is chosen depending on the input values. The selected ANN architecture is constructed based on the corresponding Taguchi orthogonal vector plan. Then, by adding nodes in the hidden layer, a more complex architecture is chosen. If the magnitude relative error (MRE) value is less than or equal to 1%, the addition of new nodes and new hidden layers is suspended.

2. Minimum number of iterations (less than 10)

The Gradient Descent (GA) value for each ANN architecture, i.e., the "stop criterion," is monitored in the training process. If the GA value is less than 0.01 of 1% of the MRE, the training procedure is suspended. The number of iterations performed in each proposed model is expected to be less than 10.

3. Selection of the architecture that converges the fastest to the mean magnitude relative error (MMRE) value

By training different architectures, it can be established that some of them have a smaller number of iterations (up to 5) and that they meet the "stop criterion" much faster. In the proposed models, it is possible to monitor the convergence rate of each selected ANN architecture.

4. Appropriate selection of the activation function of the hidden layer of the ANN architecture

In the experimental part of the training of different ANN architectures, different activation functions of the hidden and output layers were used. Sigmoid function, tangent hyperbolic, and Gaussian function were experimented with.

5. Appropriate division of the used data sets into a certain number of clusters

The nature of each observed data set is heterogeneous. In order to obtain an optimal estimate, it is necessary to divide each data set into an appropriate number of clusters.

In the first Constructive Cost Model (COCOMO2000) approach selected, a division into three clusters was used: small, medium, and large, for each of the data sets used.

The second selected Function Point Analysis (FPA) approach experimented with the International Software Benchmarking Standards Group (ISBSG) repository divided into five clusters. Each cluster is divided at a scale of 70:30, where 70 projects are used for the training process and 30 for the testing process.

In the third selected Use Case Point (UCP) approach, a 70:30 scale data set split was used, where 70% projects were used for the training process and 30% for the testing process.

1.3 Expected scientific contribution

The scientific contribution of this dissertation is based on a set of original and improved methods and models for estimating the effort and cost of developing software projects. Through the three approaches presented, improved methods based on existing models will be using: different data sets, clustering methods [30], and fuzzification methods [31], [32] for different ANN architectures constructed based on Taguchi's orthogonal vector plans and different activation functions [33] [34]. In addition, various metrics were calculated, and criteria set such as Mean Absolute Error (MAE), Magnitude Relative Error (MRE), Mean Magnitude Relative Error (MMRE) [35], Prediction on three criteria (PRED) [36], [37], Correlation (Pearson's, Spearman's and R^2) [38].

With different data sets, the correctness of new, improved models for each proposed approach is checked. The experimental part of the research is performed on different data sets and includes three parts:

1. **Training** of the proposed ANN architecture who compete to become a "Winner" network on a specific data set; E.g., it is necessary to train several ANN candidates at the same time in depending on the selected Taguchi's orthogonal vector plan, the most accurate ANN the network becomes a "Winner" network;
2. **Testing** on the same data set, but on other projects using the "Winner" network that gave the best results in the first part, i.e., the lowest value of MMRE;
3. **Validation** on other data sets using the "Winner" network.

Due to the different nature of projects and their heterogeneous structure, the clustering method is used. Different data sets are divided into smaller parts (clusters) according to the corresponding project sizes. In order to further homogenize the input

values of the used data sets, the fuzzification method is used. This method involves mapping the real values of input values into values from the interval [0, 1].

The dissertation will use different ANN architectures in each of the models and approaches that depend on the number of the input values and weight factors (coefficients) used in all three parts of the experiment. Each of the ANN architectures used was constructed based on the corresponding Taguchi orthogonal vector plan. In this way, faster, more accurate, and more precise convergence towards the final value of MMRE is achieved. Also, the number of iterations is significantly reduced concerning the Full Factorial Plan (FFP) [39], [40], [41], which reduces the time required for.

Based on several experiments and the use of different activation functions in the hidden and output layer of the proposed architectures, the lowest value of MMRE was achieved using the sigmoid function [42], [43], [44].

The following metrics are additionally calculated based on the estimated value: deviation [45], MAE, MRE, and MMRE [35]. The actual and estimated values are compared, the correlation is monitored, and the prediction on several criteria is performed.

The results of extensive experiments showed that the values of errors in evaluating new, proposed models based on existing and improved methods are significantly less than the values of errors caused by the evaluation of previous parametric and non-parametric methods and models. It is possible to experimentally identify the best model that can be efficiently and reliably applied in practice.

Chapter 2: Typical existing methods for effort and cost estimation of software projects

In this chapter, the existing typical and most frequently used estimation methods, such as parametric, nonparametric and combined methods, will be presented.

2.1 Parametric methods

Many methods measure a software's size, complexity, or the time it takes to build software. All of them can be divided into two main groups: parametric and non-parametric. They can also differ according to different approaches of estimation, three are the most commonly used: **Code analysis and effort**, **Function Point Analysis**, and **Analysis of actors (users) and use cases**, see Figure 4.

1. An approach is based on analysis of the number of source code lines and assessing the effort required to design and implement a project. The most commonly used model of this approach is COCOMO2000 (Constructive Cost Model) [46].

2. An approach is based on the functional points for estimating the size of software functionality being developed [47]. Within this approach, two models were initially distinguished: IFPUG (International Function Point Users Group) [48] and Mark II [49], and later within the IFPUG model, the most commonly used were: NESMA (Netherlands Software Users Metrics Association) [50], IFPUG (version 4.1) [51] and COSMIC FFP (Common Software Measurement International Consortium Full Function Point) [52].

3. An approach is based on the analysis of users and use cases for effort estimation. Within this approach, the most commonly used models are COBRA (Cost Estimation, Benchmarking, and Risk Assessment) [53] and UCP (Use Case Point Analysis) [54].

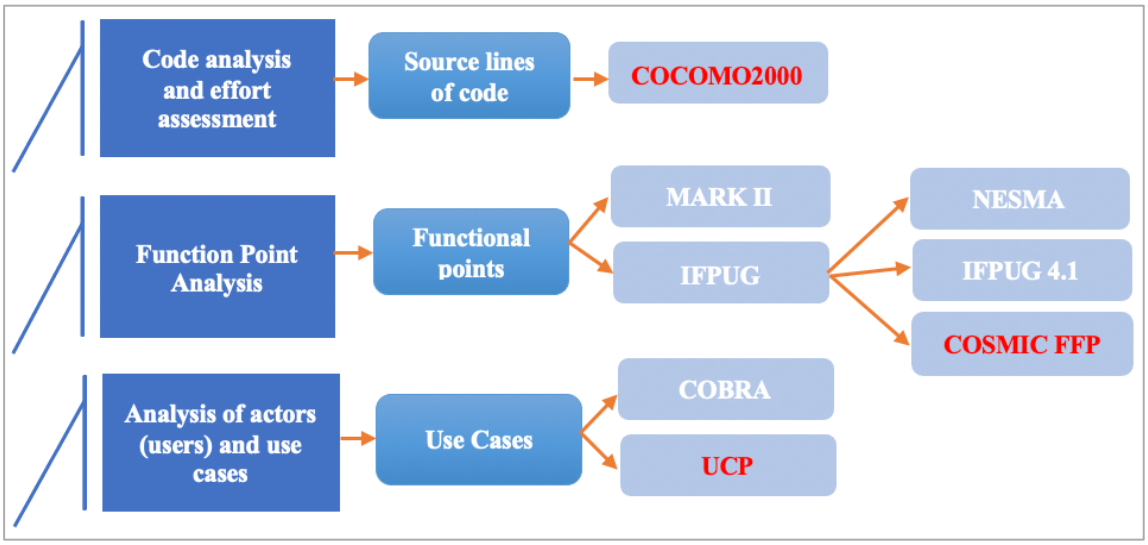


Figure 4. Parametric methods.
Slika 4. Parametarske metode.

2.2 Linear regression and statistical estimates

In many studies, various authors have used the regression method to estimate the effort and cost of developing software projects. Linear regression is a method that represents the relationship between dependent and independent variables on a given data set [55]. Based on this method, the estimated value with the slightest error can be determined. The relationship between the dependent variables Y_i, $where\ i = \overline{1,n}$; and the independent X_i, $where\ i = \overline{1,n}$; can be represented by the following formula (1):

$$Y_i = f(X_i) + r_i, where\ i = \overline{1,n}; \quad (1)$$

The real value of Y_i is the sum of the estimated value of $f(X_i)$, $where\ i = \overline{1,n}$; and the error in estimating r_i $where\ i = \overline{1,n}$;

The main goal of regression models is to choose the appropriate function f_i to minimize the error r_i. The estimated value represents the time required to realize the software project, and $f(X_i)$ represents the various functions that estimate the impact of input values. Input values are model parameters that affect the output value, representing the actual effort, functional size, and real effort depending on the experimental approach.

The most commonly used examples of regression models use linear, polynomial, logarithmic, and exponential functions. Combinations of these functions are also possible to find the best model to give the lowest error value. An example of a combination of a linear and an exponential function is the COCOMO parametric method [56], [57], see formulas (4)-(10).

In addition to calculating the error in regression models, it is necessary to use specific statistical metrics that primarily describe the essential characteristics of data sets. The most common features of the data set based on which further estimation is performed are minimum and maximum value, mean value, and standard deviation. The mean value represents the arithmetic mean of the data set and is calculated according to the following formula (2):

$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$, where N is the total number of the observed data. (2)

The standard deviation represents the expected deviations of all values of a given set from the mean value and is calculated according to the following formula (3):

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \quad (3)$$

In addition to basic information about the data set, it is necessary to compare the dependence of the data using the correlation coefficient. In this dissertation, Pearson's, Spearman's, and R^2 correlation coefficients were used to determine the relationship between real and estimated values in all three proposed models.

Additionally, the prediction was introduced as another parameter to confirm the quality of the proposed approach. The prediction was monitored on three criteria: PRED(25), PRED(30), and PRED(50). This metric represents the percentage of projects that have a relative error value less than a given criterion.

Some authors [58], [59], [60] use other various statistical tests (ANOVA test, Wilcoxon's test, Mann-Whitney test) to compare the real and estimated value and to estimate the value of relative error depending on the proposed approach.

2.3 Code analysis and effort estimation

This chapter will explain in detail the approach based on the analysis of the number of source lines of code. The most commonly used model of this approach is COCOMO2000.

2.3.1 Essential characteristics of the COCOMO2000 model

In parametric methods, the size of the system is calculated as a combination of mathematical models whose primary data are parameters obtained experimentally - by

measuring real values during the design. Measurement-based on the number of lines of source code is used to determine the size and complexity of a software project. It is most often used to show the effort and time needed to realize a project. Indeed, the most crucial method for estimating the effort from this group is the COCOMO2000 parametric method, which uses the number of source lines of code as a unit for measuring software size. In this way, with COCOMO2000, it is possible to estimate the required production time. The number of lines of code is an easy way to estimate the effort and cost, but there are also many disadvantages, such as differences in the programming language used (C ++, Java, C #, etc.) and establishing the equivalence of specific databases.

The COCOMO2000 is an algorithmic cost model where, with the help of mathematical functions, a context is created between software metrics and project costs. The actual effort represents the actual value of the project, based on the number of lines of code expressed in person-months [PM]. The twenty-two parameters represent input values, divided into two groups [61]:

- The first group cosists of five parameters, denoted as scale factors: PREC, FLEX, RESL, TEAM, PMAT;
- The second group consists of seventeen parameters, denoted as effort multipliers: RELY, CPLX, DATA, RUSE, TIME, STOR, PVOL, ACAP, PCAP, PCON, APEX, PLEX, LTEX, TOOL, SCED, SITE, DOCU.

Each of the scale factors is described as follows:

1. **PREC** (*Precedentedness*) - represents the objectives of the project and the required technology.
2. **FLEX** (*Development Flexibility*) - represents the adaptation of development concerning the requirements, standards, and restrictions.
3. **RESL** (*Architecture and Risk Resolution*) - is a measure of software components, quality, and potential risks.
4. **TEAM** (*Team Cohesion*) - represents the complexity and adaptation of team members to teamwork.
5. **PMAT** (*Process Maturity*) - represents the maturity of the process, i.e., monitors the development of software-based and Capability Maturity Model-Capability Maturity Model Integration (CMM-CMMI).

Each of the effort multipliers is described as follows:

1. **RELY** (*Required Software Reliability*) - represents the reliability of performing certain software functions up to a specific period.

2. **DATA** (*Data Base Size*) - represents the size of the database.
3. **CPLX** (*Product Complexity*) - represents the operations performed on the product, and they are divided into five areas: control operations, computational operations, device-dependent operations, data management operations, and user interface management operations.
4. **RUS**E (*Required Reusability*) - represents the reusability of the software.
5. **DOCU** (*Documentation*) - represents the documentation required to accompany each part software life cycle.
6. **TIME** (*Time Constraint*) - represents the time significance of software execution.
7. **STORE** (*Storage Constraint*) - represents data storage and its limitations.
8. **PVOL** (*Platform Volatility*) represents the impact of software and hardware architecture changes on the system.
9. **ACAP** (*Analyst Capability*) - is a measure of the ability of analysts.
10. **PCAP** (*Programmer Capability*) - is a measure of the capability of the programmer.
11. **AEXP** (*Applications Experience*) - is a measure of programmer experience working on a given application.
12. **PEXP** (*Platform Experience*) - is a measure of experience working on a particular platform.
13. **LTEX** (*Language and Tool Experience*) - is a measure of experience working with different programming languages and tools.
14. **PCON** (*Personnel Continuity*) - represents the complexity and efficiency of the team.
15. **TOOL** (*Use of Software Tools*) - is a specific measure of used software tools.
16. **SITE** (*Multisite development*) - represents the geographical distance of team members.
17. **SCED** (*Required Development Schedule*) - represents the required schedule of tasks on the project.

Finally, the COCOMO2000 formula was obtained as follows [61], see formulas (4)-(10):

$$Effort = A \times [SIZE]^E \times \prod_{i=1}^{17} EM_i \quad (4)$$

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

$$Effort[PM] = 2.94 \times [SIZE]^E \times PEM_i \quad (6)$$

$$PEM_i = \prod_{i=1}^{17} EM_i \quad (7)$$

$$Time = C \times (Effort)^F \quad (8)$$

$$F = D + 0.2 \times 0.01 \times \sum_{j=1}^{5} SFj\,, C = 3.67, D = 0.28. \quad (9)$$

$$People = Effort/Time \quad (10)$$

A and B are the fundamental constants for calibration; KSLOC (thousands of source lines of code) represents the size of the software project; SF_j denotes five scale factors; EM_i denotes seventeen effort multipliers. As previously mentioned, performed experiments are using the COCOMO2000 Post Architecture model. This model combines, as mentioned above, factors and effort multipliers to calculate needed person-months [PM] for the implementation of a particular software project.

All factors are measured in values ranging from "*very low*" to "*extremely high.*" An example of several parameters assigned in the COCOMO2000 model is given in Table 2.

Table 2. Example of weighting coefficients assigned to COCOMO2000 parameters.
Tabela 2. Primer težinskih koeficijenata dodeljenih COCOMO2000 parametrima.

Parameter	Symbol	Very Low	Low	Neutral	High	Very High	Extreme High
PREC	SF_1	0.05	0.04	0.03	0.02	0.01	0
FLEX	SF_2	0.05	0.04	0.03	0.02	0.01	0
RESL	SF_3	0.05	0.04	0.03	0.02	0.01	0
DOCU	EM_5	0.85	0.93	1	1.08	1.17	
PCAP	EM_{10}	1.37	1.16	1	0.87	0.74	
PCON	EM_{11}	1.26	1.11	1	0.91	0.83	
AEXP	EM_{12}	1.23	1.1	1	0.88	0.8	

2.3.2 Previous research in the COCOMO2000 approach

It is a constant struggle and challenge for many large software companies engaged in delivering various industrial tools, services, and products to meet the rigorous demands and needs of software clients/customers. A number of researches [9], [10], [11], [12] apply various software models and methods of effort and cost estimation in order to meet high standards and construct quality software. Some of the previous effort models used in various software development projects are SLIM (Software Life Cycle Management) [62], SEER-SEM (System Evaluation and Estimation of Resource Software Evaluation

Model) [63], COCOMO81 (Constructive Cost model) [64] and COCOMO2000 [65], [66].

The COCOMO2000 model uses cost factors, scale factors, and software size to estimate effort and cost, expressed in the number of source lines of code. There are two versions of the COCOMO2000 model: Early Design and Post Architecture, which are tested to achieve the best results.

The Post Architecture model is a more detailed version and it was used in the first approach of this dissertation (3.1) for the training and testing during the performed experiments. A data set of the COCOMO2000 model consisting of 161 projects was also used. The authors in the study [66], in their approaches to software evaluation, used the technique of neural networks for backpropagation through the data set COCOMO2000, which consists of 60 and 93 projects, respectively, for estimating software costs and Dolphin algorithm. The results showed a lower value of MRE. However, in this dissertation, within this approach, it is shown that it is possible to reduce the value of MRE, i.e., achieve better results with a simple ANN architecture to perform a minimum number of iterations, which means reducing estimation time.

Recent research, such as [67], presents the use of a combined COCOMO2000 parametric approach technique and a neural network (non-parametric approach) into a single structure (model) for estimating effort and cost. The results showed that the estimated cost of COCOMO2000 is closer to the actual cost, i.e., that a lower MRE value is achieved.

In another study [68], the authors experimentally showed that the application of Machine Learning improves the COCOMO2000 model using ANN. The obtained results again showed a high value of relative error. The authors in [69] present different algorithms and neural networks and compare them for a more accurate and reliable estimation of software costs. Also, this study used different hidden neural network activation functions that gave worse results than the sigmoid function.

In a study [70], a two-layer network was used as a model to reduce MRE values further. Various models for estimating efforts and costs in the development of software projects [71], [72] based on non-parametric approaches such as multilayer neural networks have achieved slightly better results, i.e., a lower value of MRE.

The paper [73] presents the results of research on over 33,000 different experiments, where different ANNs were used, which use real project values for data processing. Data obtained from these experiments were collected from different sources and evaluated using different metrics. In studies [74], [75], [76], [77], the authors use different architectures of artificial neural networks to process the data set. Compared with previous research, it has been shown that different results are obtained depending on the heterogeneous nature of the data sets. It can be concluded that there is no unified methodology to ensure adequate and reliable .

Contrary to previous research, different data sources will be used in this dissertation, and the obtained results will be compared using the same methodology. This achieves the stability, efficiency, and reliability of the proposed approach. In the experimental part, different data sets were used to confirm the use of the proposed methodology on different project values. COCOMO81, COCOMO2000, NASA60, NASA93, Kemerer15, and Desharnais datasets were used.

In addition to different datasets, four different architectures of artificial neural networks constructed based on Taguchi's orthogonal vector plans were used. To further improve the proposed model within the COCOMO2000 approach, the clustering method divided the datasets into three clusters to mitigate the heterogeneous structure of different projects. For additional homogenization of input values, the fuzzification method was used. Unlike other research, this dissertation uses a combination of parametric and non-parametric methods and models that have so far experimentally yielded the best results in effort and cost estimation. By introducing a machine learning rate as a criterion for stopping iterations, i.e., a "stop criterion," the convergence rate of each proposed architecture can be monitored, and the experiment stopped after the set conditions are met. The introduced machine learning rate criterion is Gradient Descent and it is used to monitor the progress and speed of the experiment. In this way, the best model is obtained with the lowest number of iterations, thus reduced execution time and the lowest MMRE value. Based on this criterion, it is possible to determine the most reliable, stable, and most accurate model of the proposed ANN architectures.

The critical decisions that define the new, improved model within the COCOMO2000 approach in this dissertation are as follows:

- Examining the convergence rate of four different proposed ANN architectures;
- Analysis of the obtained MMRE values;
- Division of all used datasets into clusters;
- Examination of different activation functions of the hidden ANN layer;
- Finding the most efficient methods of encoding and decoding input values, such as the fuzzification method;
- Required minimum number of performed experiments;
- Testing and validation on other different datasets.

2.4 Function Point Analysis

This chapter will explain in detail the approach based on the analysis of the functional points. The most commonly used model of this approach is COSMIC FFP.

2.4.1 Essential characteristics of the COSMIC FFP model

Function Point Analysis (FPA) is an approach that has emerged in order to overcome the shortcomings of a previous approache that measuerd system size only based on the number of lines of code (source lines of code - SLOC). In the analysis of functional points, the system's functionality is measured based on the values expressed in functional points. Different systems may have similar functionalities but may use, for example, different technologies or programming languages, and therefore differ in the number of source lines of code. This approach has developed many models based on the proposed method to most effectively and accurately estimate the functional size.

This dissertation will present the recent method from the family of functional points - COSMIC FFP (COmmon Software Measurement International Consortium Full Function Point). This method is used to estimate the effort and cost of the functional size of software projects based on fourteen parameters that are reduced to four input values: 1. Entry, 2. Exit, 3. Read and 4. Write (see Figure 19). The fourteen system parameters are evaluated to measure the functional size reliably:

- **Data communication** - data on the transfer and exchange of information between the users and system;
- **Distributed data processing** - testing whether the data processing is distributed;
- **Performance** - testing the required performance of the system;
- **Heavily used configuration** - hardware and software platform;
- **Transaction rate** - the frequency of execution of transactions in the system;
- **On-Line data entry** - the percentage of data that is entered directly;
- **End-user efficiency** - efficient operation of software users;
- **On-Line update** - the amount of data that is updated;
- **Complex processing** - the complexity of data processing in the system;
- **Reusability** - code reuse;
- **Installation ease**;
- **Operational ease** - ease of use;
- **Multiple sites** - division of the team into several locations;

- **Facilitate change** [13].

Unlike previous methods that were based on five input variables, COSMIC FFP is based on four. The four input variables presented in the model we proposed based on the COSMIC FFP method represent the input values for two different ANN architectures. Each of the proposed architectures is constructed based on Taguchi's orthogonal vector plan. There are four input variables are:

1. Entry - messages that users send to the system or messages that one system sends to another to transmit the necessary data. These messages do not have necessary to be system entries.

2. Exit - messages that the system or module returns in response to the user in the form of data that can be read from files. These messages can also be in the form of arithmetic and logic operations.

3. Read - messages that update data in the system. They can be different files, tables, and other data.

4. Write - messages that send data from the system. They can be in the form of tables, files, and do not represent formulas or operations.

It can be concluded that the functional size of the system represents the total number of all messages used. The system can be viewed as a four-dimensional vector space that represents the total number of messages. Messages are data entered, data that is logged out, data that is written in a file, or data read from files. This method enables the detection and determination of the influence and the slightest change on the functional size. The advantage of this method is that it is independent of technology and has no upper limit on the value of the functional quantity. Therefore, there is no saturation because the complexity of the functionality can grow indefinitely depending on the number of messages in the system [78], [79].

Functional size is determined based on a four-dimensional vector denoted as FFP as follows (11):

$$FFP = (E, X, R, W) \tag{11}$$

FFP represents the total number of messages in the entire observed system and is calculated as the norm of the vector in the formula (12):

$$\left\|\overrightarrow{FFP}\right\| = E + X + W + R \tag{12}$$

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

The COSMIC FFP method is important because it can be applied to different systems as even the slightest change can affect the change of the system, and there is no limit to the size of the functionality in the chosen dataset.

2.4.2 Previous research in the FPA approach

The first research related to the FPA approach appears in the works of Albrecht and Gafni (1983) [47] and Kemerer (1987) [80]. The research presented by Albrecht refers to the measurement of the functionality on IBM projects, where the relationship between the size of the project and the time required to create it is observed. Measuring functionality in this approach, Albrecht calls the **functional point**. This approach was later further developed and adopted by the International Function Point User Group (1994) [81] and is considered the basis for the further development of the FPA approach.

Similar models also have been later developed, such as the NESMA by the Netherlands Software Metrics Association in the Netherlands [50] and the MARK II [82] model in the United Kingdom. With the further improvement and development of these models, IFPUG version 4.1 and the COSMIC FFP method developed by the COmmon Software Measurement International Consortium [83] are emerging. COSMIC FFP is the latest and most commonly used method based on size functionality. This method measures the system's functionality based on the functional points that are exchanged in the system. Functional points in the COSMIC FFP approach are communication messages. The essential functional points that are analyzed in this method are Entry, Exit, Write and Read.

Many authors, such as [84], [85], [86], have shown that despite attempts to introduce new metrics, such as user stories in agile methodologies, the COSMIC FFP method has proven to be more efficient and accurate in several experiments.

The paper [87] compares the effects of improving effort prediction using COSMIC FFP and IFPUG methods. COSMIC FFP again gave better results for predicting efforts to develop software projects.

The paper [88] presented an approximation model with a convolutional neural network and achieved a reliable prediction accuracy using a word embedding model trained on Wikipedia + Gigaworld and functional point measures.

Another study [89] proposed different machine learning algorithms to measure software development also using COSMIC functional size.

The authors in [90], [91] proposed a calibration of the functional point complexity (FP) weights and an FP calibration model called the Neuro-Fuzzy Function Point Calibration Model (NFFPCM). They used the ISBSG repository [15] and demonstrated

improved accuracy of the mean relative error (MMRE) value in estimating software effort after calibration. They used only parts of the ISBSG datasets, while in this dissertation, the entire dataset is used without any calibration or adjustment. Also, a method of clustering input values for the whole data set is presented, in addition to fuzzification and a smaller number of required iterations on different combinations of data sets from the ISBSG repository. Next, two more sets of data are introduced to confirm the obtained results.

The critical decisions that define the new, improved model within the COSMIC FFP approach in this dissertation are as follows:

- Examination of the influence of four input values on the change of MMRE value;
- Comparative analysis of two different architectures of artificial neural networks and the obtained results;
- Division of the used dataset into clusters depending on their functional size and different nature of each project within the ISBSG repository;
- Finding the most efficient methods of encoding and decoding input values, such as the fuzzification method;
- Required minimum number of performed experiments;
- Testing and validation on other different datasets.

2.5 Analysis of actors (users) and Use Cases

This chapter will explain in detail the approach based on the analysis of the use case points. The most commonly used model of this approach is UCP model.

2.5.1 Essential characteristics of the UCP model

The UCP (Use Case Point Analysis) method is most often used in estimating the real size of a software project. This method for estimating the effort to implement a particular system considers use cases of the system. Also, analyzes system users and different scenarios to adequately assess the effort required. It uses twenty-one parameters for, of which thirteen parameters are technical characteristics of the system, and the remaining eight are environmental factors.

The technical characteristics of the system being evaluated are:

- **Is system distributed**;
- **System response time**;
- **Efficiency of the system**;
- **Complexity of internal processes**;
- **Posibility of code reuse**;
- **Ease of installation of the final product**;
- **Ease of use of the final product**;
- **Transfer to other required platforms**;
- **System maintenance**;
- **Competitiveness**, **parallel processing**;
- **Security requirements of the system**;
- **Access to external systems**;
- **End-user training**.

The environmental factors being assessed are:

- **Compliance with the used development process**;
- **Experience with applications that will be used**;
- **Experience in object-oriented technologies that will be used**;
- **Ability of chief analyst**;
- **Team motivation**;
- **Stability requirements of the system**;
- **Working hours of team members**;
- **Complexity of the programming language that will be used**.

System users and use cases were used together to determine the real size of the UCP method.

Users of the system are divided into three groups: simple (depending on the interaction with the system, they are assigned a weight factor of 1), medium (depending on internal/external communication, they are assigned a weight factor of 2), and complex (depending on the complexity of interactions weighted by factor 3).

There are also three categories of use cases that are defined based on the number of executed transactions (number of users and system messaging): simple (for less than 3 transactions and a weighting factor of 5 is assigned), medium (from 4 to 7 transactions and a weighting factor of 10 is assigned), and complex (more than 8 transactions and a weighting factor of 15 is assigned).

The size of the system is defined based on a six-dimensional vector whose elements represent the complexity of the previously mentioned users and the cases of users in the system.

The estimated value is calculated based on G. Karner formulas [92]:

UAW (*Unadjusted Actor Weight*) - this input value is a functional point that can determine the level of complexity of system users. Users can be simple system operators or other external systems. Each user is ranked according to their level of complexity and can be: *Simple*, *Average*, and *Complex*, see formulas (13)-(16).

$SimpleA = \sum(SimpleActor) * SimpleWeight$, where $SimpleWeight = 1$; (13)
$AverageA = \sum(AverageActor) * AverageWeight$, where $AverageWeight = 2$; (14)
$ComplexA = \sum(ComplexActor) * ComplexWeight$, where $ComplexWeight = 3$; (15)

UAW = *SimpleA* + *AverageA* + *ComplexA* (16)

UUCW (*Unadjusted Use Case Weight*) - this input value is a functional point that can determine the level of complexity of use cases. Each use case is ranked according to its level of complexity and can be: *Simple*, *Average* and *Complex*, see formulas (17)-(20).

$SimpleUUCW = \sum(SimpleUCW) * SimpleWeight$,
where $SimpleWeight = 5$, (transactions<=3, analysis classes<5) (17)

$AverageUUCW = \sum(AverageUCW) * AverageWeight$, where $AverageWeight = 10$,
(4<=transactions<=7, 5<=analysis classes<=10) (18)

$ComplexUUCW = \sum(ComplexUCW) * ComplexWeight$, where $ComplexWeight = 15$;
(transactions>7, analysis classes>=10) (19)

UUCW = *SimpleUUCW* + *AverageUUCW* + *ComplexUUCW* (20)

By using the two sizes listed above, the size of the system to be developed is obtained.

UUCP (*Unadjusted Use Case Points*) is determined by following equation (21):

UUCP=UUCW+UAW (21)

TCF (*Technical Complexity Factor*) is an estimate of the technical complexity of the system and can be described by the following formulas (22), (23):

TCF = 0.6 + (0.01*FactorT) (22)

FactorT = $\sum Weight * AssignedValue$, where *AssignedValue* is from 0 to 5 and represents a technical factor of the estimated process, see Table 3. (23)

Table 3. Thirteen technical complexity factors.
Tabela 3. Trinaest tehničkih faktora kompleksnosti.

Factor	Description	Weight	Assigned Value	Weight x Assigned Value
T1	Distributed system	2.0	5	10
T2	Response time/performance objectives	1.0	5	5
T3	End-user efficiency	1.0	3	3
T4	Internal processing complexity	1.0	2	2
T5	Code reusability	1.0	3	3
T6	Easy to install	0.5	1	0.5
T7	Easy to use	0.5	5	2.5
T8	Portability to other platforms	2.0	2	4
T9	System maintenance	1.0	2	2
T10	Concurrent/parallel processing	1.0	3	3
T11	Security features	1.0	5	5
T12	Access for third parties	1.0	1	1
T13	End-user training	1.0	1	1
			Total (TF):	42

ECF (*Environmental Complexity Factor*) is one of the factors affecting the size of the project expressed in Use Case points. It is calculated according to the following formulas (24), (25):

ECF = 1.4+(-0.03xFactorE) (24)

FactorE = $\sum Weight * AssignedValue$, where *AssignedValue* from 0 to 5 and represents a environmental factor of the estimated process, see Table 4. (25)

Table 4. Eight environmental factors.
Tabela 4. Osam faktora okruženja.

Factor	Description	Weight	Assigned Value	Weight x Assigned Value
E1	Familiarity with development process used	1.5	3	4.5
E2	Application experience	0.5	3	1.5
E3	Object-oriented experience of team	1.0	3	2
E4	Lead analyst capability	0.5	5	2.5
E5	Motivation of the team	1.0	2	2
E6	Stability of requirements	2.0	1	2
E7	Part-time staff	-1.0	0	0
E8	Difficult programming language	-1.0	4	-4
			Total (EF):	10.5

AUCP (*Adjusted Use Case Point*) is the final size of the system expressed in Use Case points and is calculated as follows (26):

AUCP = UUCPxTCFxECF (26)

Representation of real effort by UCP approach as a six-dimensional vector, where its value is calculated as the norm of the vector as follows, see formulas (27), (28):

UCP = (UAW, UUCW, UUCP, TCF, ECF, AUCP) (27)

$$\left\|\overrightarrow{UCP}\right\| = UAW + UUCW + UUCP + TCF + ECF + AUCP \quad (28)$$

Representation of real effort by UCP approach as a four-dimensional vector, where its value is calculated as the norm of the vector as follows, see formulas (29) (30):

UCP = (UAW, UUCW, TCF, ECF) (29)

$$\left\|\overrightarrow{UCP}\right\| = UAW + UUCW + TCF + ECF \quad (30)$$

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

In both cases, Real Effort is obtained as the norm of the UCP vector and represents the real functional size or number of points of use cases. This method is currently most commonly used to assess effort [56], although it is not standardized within ISO standards as the previous two are.

2.5.2 Previous research in the UCP approach

The UCP method is the latest and the most widespread method for estimating the effort and costs for realizing software products. The most significant advantage of this method is that the lowest values of relative error in estimation are obtained, between 20% and 35% [57]. The best result achieved by this method is an error value of about 10% [95]. Many researchers [96], [97], [98], [99] have combined this method with other parametric models and models of artificial intelligence.

In the study [100], the UCP method is used for estimating size and effort for mobile applications. Android mobile applications are considered as a case study, and modified UCP has also been proposed.

Authors in [101] proposed a framework for UCP-based techniques to promote reusability in developing software applications. The results showed that the framework

had met five quality attributes, and it can be used in the early stages of software development.

In [102], a systematic review of studies with the best practices of use case point (UCP) and expert judgment–based effort estimation techniques was given.

The study [103] presented the results of four different models that include the UCP method and Neuro-Fuzzy logic. It is concluded that the Neuro-fuzzy logic model using revised use case points and modified environmental gives the best fitting accuracy at an early stage than other models.

In the study [104], the authors compare the benefits of a statistical analysis of the effort estimation for seven real-world software development projects. Also, they contrast a conventional Use Case points method with iUCP, an HCI (Human-centric) enhanced model. Furthermore, they propose an enhancement of the iUCP original effort estimation formula.

The critical decisions that define the new, improved model within the UCP approach in this dissertation are as follows:

- Examination of the influence of two linearly dependent input values (UUCP and AUCP) on the change of MMRE value;
- Comparative analysis of two different architectures of artificial neural networks and the obtained results;
- Dividing the used dataset to a scale of 70:30, i.e., 70% projects of the selected dataset are used for the training process, while 30% are used for the testing process;
- Finding the most efficient methods of encoding and decoding input values, such as the fuzzification method;
- Required minimum number of performed experiments;
- Testing and validation on other different datasets.

2.6 Application of ANN in software estimation

Artificial neural networks (ANNs) are gaining importance in the 90s of the last century, with the accelerated computer technologies. Each ANN is a system that consists of interconnected elements that we call nodes or artificial neurons. Neurons are connected by certain connections (synapses) through which data is transmitted. The architecture of each network represents the connection of neurons into one that differs depending on the number of layers. The first layer is always called the input layer, and the last layer the output layer. All layers in between are called hidden layers. The first layer, i.e., the input

layer, can have several different input sizes. The corresponding data to be processed is entered via the input variables. The data is transferred to a hidden layer in which it is processed according to defined rules and further passed through synapses to the output layer. The strength of the connections between neurons is called the weighting factor (coefficient) [105], [106].

Generally, this artificial intelligence tool can be justified when any specific rules cannot determine the final output value. First, it is necessary to train the artificial neural network for its further use. Training implies setting input values, defining the rules or functions according to which one trains and according to which the output value is obtained. Each ANN has a particular architecture, i.e., the scheme according to which the nodes are connected. It contains an input layer, usually one to three hidden layers, and an output layer. An essential role of each neural network is played by the activation function of neurons, which in combination with weight coefficients represents the essence of training. In the initial phase of training, the errors are much larger, and then in each subsequent iteration, by changing the weighting coefficients, the errors are reduced and converged towards precise, estimated values [107], [108].

2.6.1 Classification of artificial neural networks

There are different divisions and architectures of artificial neural networks. They can be distinguished according to the connection method: layered, wholly connected, and cellular artificial neural networks. According to the direction of signal transmission and data processing, we distinguish feedforward - networks in which the signal transmission takes place in one direction and feedback - recurrent or return networks.

Depending on the way of training, artificial neural networks can be supervised or unsupervised [109]. Supervised neural networks use labeled input and output data, while an unsupervised do not.

2.6.2 Artificial neural networks in other fields

It is expected that in the future, this artificial intelligence tool will be used more and more because it is suitable for modeling complex systems that have a large number of conditioned, indeterminate, or unknown factors. One of these examples is estimating time and costs for the development of software projects, where the previous analytical methods can be replaced by models that use artificial neural networks.

With the apperance development of artificial intelligence, the application of ANN in solving various problems, which previously could not be solved by other classical computer technology methods, spread rapidly. Today, ANN can be used in image

processing, speech recognition, recognition of various objects, and sensory signals. They are also frequently used in areas like application in natural languages processing, recognition of printed texts, handwritten texts, and so on. They also play a notable role in medical diagnostics, military and police purposes, and telecommunications [110]. They certainly have the most significant contribution to computer science and information and communication technolgy.

The subject of the dissertation is convenient to use artificial neural networks because their advantage over other nonlinear models is based on estimating any function with optional precision. In the dissertation various Taguchi Orthogonal Arrays are used to simplify optimization problems, representing the MFFN (Multilayer Feed Forward Neural Network) class, which has a crucial role in solving various problems in science, engineering, medicine, pattern recognition, nuclear sciences, and other fields [7]. In order to construct a high-performance MFFN, no approved theory allows the calculation of ideal parameter settings. This leads to the conclusion that even small changes in parameters can cause significant differences in the behavior of almost all networks. In [8], an analysis of NN design factors and object functions is given, in which an architecture with one or two hidden layers is recommended. Based on the Kolmogorov-Smirnov test, a recommendation is given that the number of neurons in the hidden layer should be doubled the number of input neurons increased by one, i.e., $2 \times N_{input} + 1$. The results for each of the 240 experiments performed in [8] were collected. It was shown that a specific neural network configuration is required to achieve convergence, along with the accuracy of the trained network, when a set of test data is obtained. Also, the number of hidden layers (one or two) has a minimal effect on the network accuracy but is rather significant at the convergence speed. Considering these results, a trial and error strategy is adopted because most existing theoretical works for generalization fail to explain the performance of neural networks in practice.

The idea adopted in the dissertation is experimenting with the **simplest ANN topology** - with none hidden layer, and then **more complex** - with one hidden layer, and finally with two hidden layers architecture, keeping in mind that the size of the set of observations during ANN training will not exceed the number of projects in the dataset.

2.7 Robust design - Taguchi Orthogonal Arrays

Dr. Genichi Taguchi proposed a robust design method in 1986. The main idea of this method is to find the interaction between the control and noise variables to provide appropriate settings of the control factors to reduce the changes caused by the noise variables. It is one of the powerful methods available to reduce costs, improve quality and at reducing the time necessary to perform the experiments. The robust design strategy also

involves using orthogonal vector plans to gather reliable information on project parameters with a small number of experiments [111], [112].

A robust method design implies meeting the prescribed criteria when planning and implementing software. This can be achieved with Taguchi's method regardless of the different influences of external, additional requirements. The best effect in planning a robust product design, Taguchi achieved through Orthogonal Arrays. Taguchi's orthogonal vector plans are based on a unique set of Latin Squares. The discovery of orthogonal vector plans and their application minimizes the number of crucial parameters for the project's successful development. The impact parameters are not duplicated, which achieves a much faster estimation of the efforts and costs of a particular project. This design method is based on a "factorial experiment" that is realized only with all possible experimental combinations of parameter values. Taguchi's orthogonal vector plans play the most crucial role in experimenting.

Taguchi's robust design of the experiment in each orthogonal vector plan depends on the number of parameters, the weighting factors (coefficients), and the number of levels of each parameter. That is, how many times it is necessary to test each level for each parameter. The number of experiments required for a complete factorial analysis is $N = L^P$ (for example: when three levels with 13 parameters are used according to the full factorial plan (FFP) it is necessary to execute $N = 3^{13} = 1\ 594\ 323$ experiments). Using the Taguchi orthogonal vector plan with 13 parameters (weight coefficients) on three levels, only $3^3 = 27$ experiments is necessary. The Taguchi method of robust design reduces the number of experiments for 99.99830649% (0.9999830649 = 1 - (27/1 594 323)).

Taguchi's orthogonal vector plan takes the selected cluster of combinations without repetition but so that all parameters are equally taken into account. They can also be assessed independently of each other. Taguchi's orthogonal vector plan is observed for each level of a particular parameter [113]. All *L* levels of each of the *(P-1)* other parameters are tested at least once, see Figure 5.

Taguchi design	Experiment number	FFD	Experiment number
$L_4(2^3)$	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

Figure 5. Taguchi design vs. Full Factorial Design (FFD).

Slika 5. Taguči dizajn u poređenju sa potpunim faktorijalnim dizajnom.

Chapter 3: New, proposed models for three software approaches

In software development, both in practice and in a large number of researches for estimating efforts and costs, one of the most frequently used tools of artificial intelligence are artificial neural networks (ANN). ANN is a good technique for information processing and can significantly contribute to constructing new models for software evaluation [114], [115]. Because of ANNs ability to learn from different data sets, it is possible to generate accurate and reliable results and assess the risk of possible errors.

The general structure of ANN consists of three layers: input, hidden, and output. In the experiments in this dissertation, different architectures with three, four, and six input values will be presented in combination with the different number of weight factors (coefficients). In the new, proposed models, the input values and weight parameters are constructed according to Taguchi's orthogonal vector plans [116], [117]. The goal of each proposed model is to find the simplest network that meets all the additional criteria. In the hidden layer of each proposed architecture, a different number of nodes is selected depending on the appropriate orthogonal vector plan. The output layer has one obtained value (estimated value) where it can be used to calculate: Deviation, MRE (Magnitude Relative Error), and MAE (Mean Absolute Error) for each ANN.

The MRE value is the error value for each ANN within the proposed architecture. Using the MRE value, the difference between the actual and the estimated effort concerning the actual effort is measured for a given project. This value considers the numerical value of each observation in the data distribution and is sensitive to individual predictions.

The MAE value determines accuracy and represents the absolute difference between the actual and estimated value.

In addition to the values of MRE and MAE, MMRE (Mean Magnitude Relative Error) is measured. MMRE is the mean value of MRE. Within the ANN-L27 architecture, there are 27 different combinations (“ANN candidates”), where the goal is to determine the ANN that gives the lowest MMRE value after the first part of the experiment (after ANN training).

Why measuring the value of selected error criteria?

Each of the new, proposed ANN models uses different Taguchi orthogonal vector plans to efficiently, reliably, and accurately estimate the effort and cost of developing software projects. Taguchi's orthogonal vector plan allows taking the selected cluster of

combinations without repetition. In this way, all factors are taken into account equally and can be assessed independently. By reducing the Full Factorial plan [118], [40] instead of testing all possible combinations of ANN, this method considers only pairs of combinations. This allows data to be collected to obtain information on which factors most influence the quality of the product being developed. It takes a minimum number of experiments, saves time and resources. The choice of the appropriate Taguchi orthogonal vector plan depends on the number of weighting factors and the number of input values. The general goal of Taguchi's method is to create a high-quality product with a possible reduction in development costs.

Different ANN architectures constructed on Taguchi's orthogonal vector plans were used in different approaches to constructing the three improved models:

- New, improved COCOMO2000 model,
- New, improved COSMIC FFP model, and
- New, improved UCP model.

3.1 New, improved COCOMO2000 model

As part of the COCOMO approach, an improved COCOMO2000 model is presented in this subsection. For the improved COCOMO2000 model, the following architectures and corresponding orthogonal vector plans L9, L18, L27 and L36 are used:

1. COCOMO2000 and ANN-L9

The first proposed ANN architecture, denoted as ANN-L9, is the simplest because there are no hidden layers. It is based on the orthogonal vector plan of Taguchi (L9) with four parameters (Wi , i = $\overline{1,4}$) and three different levels [7], [8], [119]: L1, L2 and L3, see Figure 6, Table 5. It is experimented with nine ANN candidates denoted as ANN1, ANN2,...,ANN9 who compete to become a "Winner" network. Bias represents an additional weighting factor to complete the selected orthogonal plan and has a value of 1.

Table 5. Taguchi orthogonal vector plan (L9=3^4).
Tabela 5. Taguči ortogonalni vektorski plan (L9=3^4).

ANN-L9	W_1	W_2	W_3	W_4
ANN1	L1	L1	L1	L1
ANN2	L1	L2	L2	L2
ANN3	L1	L3	L3	L3
ANN4	L2	L1	L2	L3
ANN5	L2	L2	L3	L1
ANN6	L2	L3	L1	L2
ANN7	L3	L1	L3	L2
ANN8	L3	L2	L1	L3
ANN9	L3	L3	L2	L1

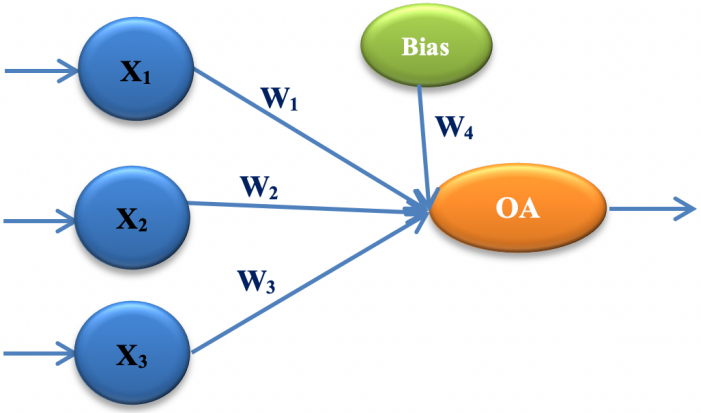


Figure 6. ANN architecture with none hidden layer (ANN-L9).
Slika 6. ANN arhitektura bez skrivenog sloja (ANN-L9).

2. COCOMO2000 and ANN-L18

The second proposed ANN architecture, denoted as ANN-L18, is designed with a single hidden layer. It is based on the Taguchi orthogonal vector plan (L18) with eight parameters (Wi , i = $\overline{1,8}$) and three different levels [7], [8], [119]: L1, L2 and L3, see Figure 7, Table 6. Compared to the previous architecture, this is a combined version, because the first parameter has only two levels, and the remaining seven have all three levels. It is experimented with eighteen ANN candidates denoted as ANN1, ANN2,...,ANN18 who compete to become a "Winner" network.

Table 6. Taguchi orthogonal vector plan ($L18=2^1 3^7$).
Tabela 6. Taguči ortogonalni vektorski plan ($L18=2^1 3^7$).

ANN-L18	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8
ANN1	L1	L1	L1	L1	L1	L1	L1	L1
ANN2	L1	L1	L2	L2	L2	L2	L2	L2
ANN3	L1	L1	L3	L3	L3	L3	L3	L3
ANN4	L1	L2	L1	L1	L2	L2	L3	L3
ANN5	L1	L2	L2	L2	L3	L3	L1	L1
ANN6	L1	L2	L3	L3	L1	L1	L2	L2
ANN7	L1	L3	L1	L2	L1	L3	L2	L3
ANN8	L1	L3	L2	L3	L2	L1	L3	L1
ANN9	L1	L3	L3	L1	L3	L2	L1	L2
ANN10	L2	L1	L1	L3	L3	L2	L2	L1
ANN11	L2	L1	L2	L1	L1	L3	L3	L2
ANN12	L2	L1	L3	L2	L2	L1	L1	L3
ANN13	L2	L2	L1	L2	L3	L1	L3	L2
ANN14	L2	L2	L2	L3	L1	L2	L1	L3
ANN15	L2	L2	L3	L1	L2	L3	L2	L1
ANN16	L2	L3	L1	L3	L2	L3	L1	L2
ANN17	L2	L3	L2	L1	L3	L1	L2	L3
ANN18	L2	L3	L3	L2	L1	L2	L3	L1

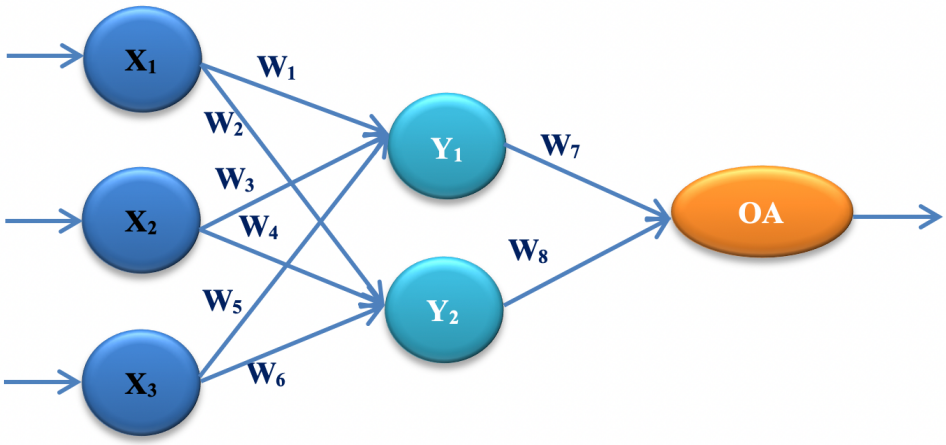


Figure 7. ANN architecture with one hidden layer (ANN-L18).
Slika 7. ANN arhitektura sa jednim skrivenim slojem (ANN-L18).

3. COCOMO2000 and ANN-L27

The third proposed ANN architecture, denoted as ANN-L27, is also designed with one hidden layer. It is based on the Taguchi orthogonal vector plan (L27) with thirteen parameters (Wi , i = $\overline{1,13}$) and three different levels [7], [8], [119]: L1, L2 and L3, see Figure 8, Table 7. It has experimented with twenty-seven ANN candidates denoted as

ANN1, ANN2,...,ANN27 who compete to become a "Winner" network. Bias represents an additional weighting factor to complete the selected orthogonal plan and has a value of 1.

Table 7. Taguchi orthogonal vector plan (L27=3^{13}).
Tabela 7. Taguči ortogonalni vektorski plan (L27=3^{13}).

ANN-L27	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
ANN1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
ANN2	L1	L1	L1	L1	L2	L2	L2	L2	L2	L2	L2	L2	L2
ANN3	L1	L1	L1	L1	L3	L3	L3	L3	L3	L3	L3	L3	L3
ANN4	L1	L2	L2	L2	L1	L1	L1	L2	L2	L2	L3	L3	L3
ANN5	L1	L2	L2	L2	L2	L2	L2	L3	L3	L3	L1	L1	L1
ANN6	L1	L2	L2	L2	L1	L1	L1	L3	L3	L3	L2	L2	L2
ANN7	L1	L3	L3	L3	L1	L1	L1	L3	L3	L3	L2	L2	L2
ANN8	L1	L3	L3	L3	L2	L2	L2	L1	L1	L1	L3	L3	L3
ANN9	L1	L3	L3	L3	L3	L3	L3	L2	L2	L2	L1	L1	L1
ANN10	L2	L1	L2	L3	L1	L2	L3	L1	L2	L3	L1	L2	L3
ANN11	L2	L1	L2	L3	L2	L3	L1	L2	L3	L1	L2	L3	L1
ANN12	L2	L1	L2	L3	L3	L1	L2	L3	L1	L2	L3	L1	L2
ANN13	L2	L2	L3	L1	L1	L2	L3	L2	L3	L1	L3	L1	L2
ANN14	L2	L2	L3	L1	L2	L3	L1	L3	L1	L2	L1	L2	L3
ANN15	L2	L2	L3	L1	L3	L1	L2	L1	L2	L3	L2	L3	L1
ANN16	L2	L3	L1	L2	L1	L2	L3	L3	L1	L2	L2	L3	L1
ANN17	L2	L3	L1	L2	L2	L3	L1	L1	L2	L3	L3	L1	L2
ANN18	L2	L3	L1	L2	L3	L1	L2	L2	L3	L1	L1	L2	L3
ANN19	L3	L1	L3	L2	L1	L3	L2	L1	L3	L2	L1	L3	L2
ANN20	L3	L1	L3	L2	L2	L1	L3	L2	L1	L3	L2	L1	L3
ANN21	L3	L1	L3	L2	L3	L2	L1	L3	L2	L1	L3	L2	L1
ANN22	L3	L2	L1	L3	L1	L3	L2	L2	L1	L3	L3	L2	L1
ANN23	L3	L2	L1	L3	L2	L1	L3	L3	L2	L1	L1	L3	L2
ANN24	L3	L2	L1	L3	L3	L2	L1	L1	L3	L2	L2	L1	L3
ANN25	L3	L3	L2	L1	L1	L3	L2	L3	L2	L1	L2	L1	L3
ANN26	L3	L3	L2	L1	L2	L1	L3	L1	L3	L2	L3	L2	L1
ANN27	L3	L3	L2	L1	L3	L2	L1	L2	L1	L3	L1	L3	L2

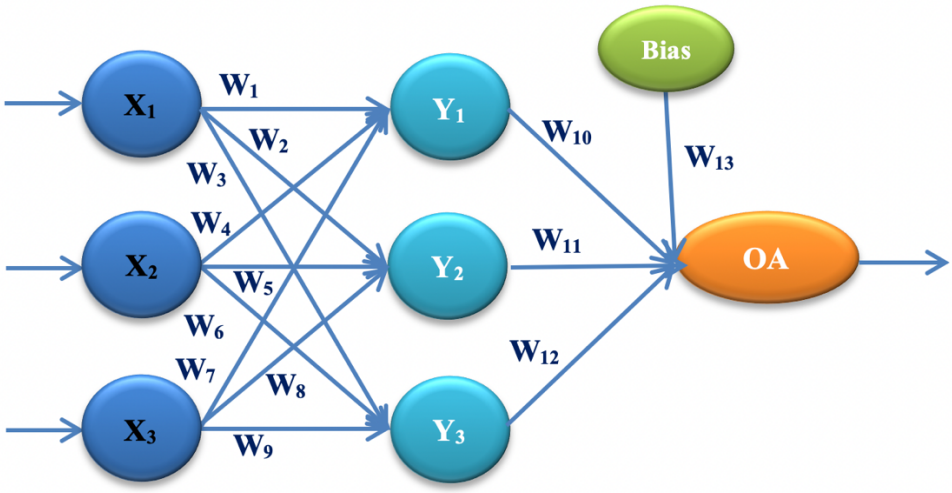


Figure 8. ANN architecture with one hidden layer (ANN-L27).
Slika 8. ANN arhitektura sa jednim skrivenim slojem (ANN-L27).

4. COCOMO2000 and ANN-L36

The fourth proposed ANN architecture, denoted as ANN-L36, consists of two hidden layers. It is based on the Taguchi orthogonal vector plan (L36) with twenty-three parameters (Wi, i = $\overline{1,23}$) and three different levels [7], [8], [119]: L1, L2 and L3, see Figure 9, Table 8. This architecture is a combined network, because the first eleven parameters have only two levels, and the remaining twelve have all three levels. It is experimented with thirty-six ANN candidates denoted as ANN1, ANN2,...,ANN36 who compete to become a "Winner" network. Bias represents an additional weighting factor to complete the selected orthogonal plan and has a value of 1.

Table 8. Taguchi orthogonal vector plan (L36=$2^{11}3^{12}$).
Tabela 8. Taguči ortogonalni vektorski plan (L36=$2^{11}3^{12}$).

ANN-L36	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}	W_{16}	W_{17}	W_{18}	W_{19}	W_{20}	W_{21}	W_{22}	W_{23}
ANN1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
ANN2	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2
ANN3	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3
ANN4	L1	L1	L1	L1	L1	L2	L2	L2	L2	L2	L2	L1	L1	L1	L1	L2	L2	L2	L2	L3	L3	L3	L3
ANN5	L1	L1	L1	L1	L1	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L3	L3	L3	L3	L1	L1	L1	L1
ANN6	L1	L1	L1	L1	L1	L2	L2	L2	L2	L2	L2	L3	L3	L3	L3	L1	L1	L1	L1	L2	L2	L2	L2
ANN7	L1	L1	L2	L2	L2	L1	L1	L1	L2	L2	L2	L1	L1	L2	L3	L1	L2	L3	L3	L1	L2	L2	L3
ANN8	L1	L1	L2	L2	L2	L1	L1	L1	L2	L2	L2	L2	L2	L3	L1	L2	L3	L1	L1	L2	L3	L3	L1
ANN9	L1	L1	L2	L2	L2	L1	L1	L1	L2	L2	L2	L3	L3	L1	L2	L3	L1	L2	L2	L3	L1	L1	L2
ANN10	L1	L2	L1	L2	L2	L1	L2	L2	L1	L1	L2	L1	L1	L3	L2	L1	L3	L2	L3	L2	L1	L3	L2
ANN11	L1	L2	L1	L2	L2	L1	L2	L2	L1	L1	L2	L2	L2	L1	L3	L2	L1	L3	L1	L3	L2	L1	L3
ANN12	L1	L2	L1	L2	L2	L1	L2	L2	L1	L1	L2	L3	L3	L2	L1	L3	L2	L1	L2	L1	L3	L2	L1
ANN13	L1	L2	L2	L1	L2	L2	L1	L2	L1	L2	L1	L1	L2	L3	L1	L3	L2	L1	L3	L3	L2	L1	L2
ANN14	L1	L2	L2	L1	L2	L2	L1	L2	L1	L2	L1	L2	L3	L1	L2	L1	L3	L2	L1	L1	L3	L2	L3
ANN15	L1	L2	L2	L1	L2	L2	L1	L2	L1	L2	L1	L3	L1	L2	L3	L2	L1	L3	L2	L2	L1	L3	L1
ANN16	L1	L2	L2	L2	L1	L2	L2	L1	L2	L1	L1	L1	L2	L3	L2	L1	L1	L3	L2	L3	L3	L2	L1
ANN17	L1	L2	L2	L2	L1	L2	L2	L1	L2	L1	L1	L2	L3	L1	L3	L2	L2	L1	L3	L1	L1	L3	L2
ANN18	L1	L2	L2	L2	L1	L2	L2	L1	L2	L1	L1	L3	L1	L2	L1	L3	L3	L2	L1	L2	L2	L1	L3
ANN19	L2	L1	L2	L2	L1	L1	L2	L2	L1	L2	L1	L1	L2	L1	L3	L3	L3	L1	L2	L2	L1	L2	L3
ANN20	L2	L1	L2	L2	L1	L1	L2	L2	L1	L2	L1	L2	L3	L2	L1	L1	L1	L2	L3	L3	L2	L3	L1
ANN21	L2	L1	L2	L2	L1	L1	L2	L2	L1	L2	L1	L3	L1	L3	L2	L2	L2	L3	L1	L1	L3	L1	L2
ANN22	L2	L1	L2	L1	L2	L2	L2	L1	L1	L1	L2	L1	L2	L2	L3	L3	L1	L2	L1	L1	L3	L3	L2
ANN23	L2	L1	L2	L1	L2	L2	L2	L1	L1	L1	L2	L2	L3	L3	L1	L1	L2	L3	L2	L2	L1	L1	L3
ANN24	L2	L1	L2	L1	L2	L2	L2	L1	L1	L1	L2	L3	L1	L1	L2	L2	L3	L1	L3	L3	L2	L2	L1
ANN25	L2	L1	L1	L2	L2	L2	L1	L2	L2	L1	L1	L1	L3	L2	L1	L2	L3	L3	L1	L3	L1	L2	L2
ANN26	L2	L1	L1	L2	L2	L2	L1	L2	L2	L1	L1	L2	L1	L3	L2	L3	L1	L1	L2	L1	L2	L3	L3
ANN27	L2	L1	L1	L2	L2	L2	L1	L2	L2	L1	L1	L3	L2	L1	L3	L1	L2	L2	L3	L2	L3	L1	L1
ANN28	L2	L2	L2	L1	L1	L1	L1	L2	L2	L1	L2	L1	L3	L2	L2	L2	L1	L1	L3	L2	L3	L1	L3
ANN29	L2	L2	L2	L1	L1	L1	L1	L2	L2	L1	L2	L2	L1	L3	L3	L3	L2	L2	L1	L3	L1	L2	L1
ANN30	L2	L2	L2	L1	L1	L1	L1	L2	L2	L1	L2	L3	L2	L1	L1	L1	L3	L3	L2	L1	L2	L3	L2
ANN31	L2	L2	L1	L2	L1	L2	L1	L1	L1	L2	L2	L1	L3	L3	L3	L2	L3	L2	L2	L1	L2	L1	L1
ANN32	L2	L2	L1	L2	L1	L2	L1	L1	L1	L2	L2	L2	L1	L1	L1	L3	L1	L3	L3	L2	L3	L2	L2
ANN33	L2	L2	L1	L2	L1	L2	L1	L1	L1	L2	L2	L3	L2	L2	L2	L1	L2	L1	L1	L3	L1	L3	L3
ANN34	L2	L2	L1	L1	L2	L1	L2	L1	L2	L2	L1	L1	L3	L1	L2	L3	L2	L3	L1	L2	L2	L3	L1
ANN35	L2	L2	L1	L1	L2	L1	L2	L1	L2	L2	L1	L2	L1	L2	L3	L1	L3	L1	L2	L3	L3	L1	L2
ANN36	L2	L2	L1	L1	L2	L1	L2	L1	L2	L2	L1	L3	L2	L3	L1	L2	L1	L2	L3	L1	L1	L2	L3

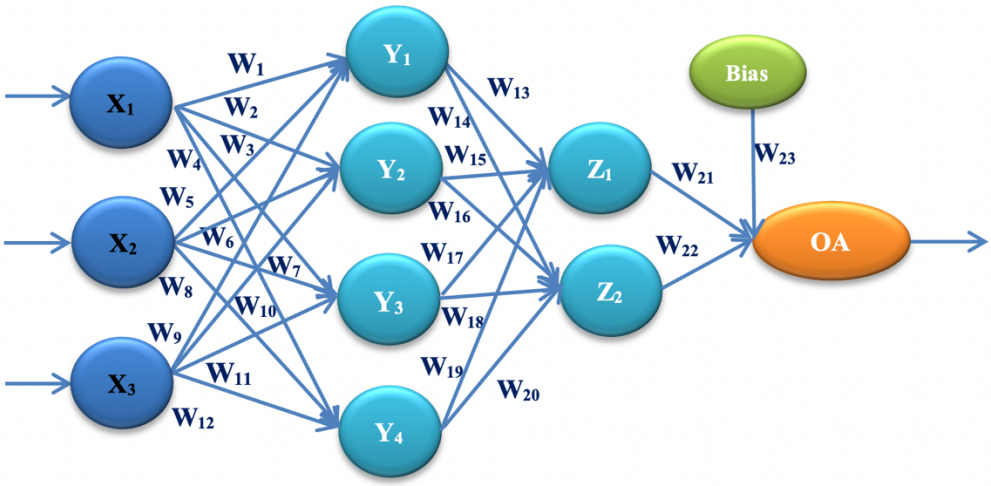


Figure 9. ANN architecture with two hidden layers (ANN-L36).
Slika 9. ANN arhitektura sa dva skrivena sloja (ANN-L36).

Each experiment, depending on the chosen approach and used datasets, consists of three parts:

1. Training of the proposed ANN architecture who compete to become a "Winner" network.
2. Testing on ANN "Winner" that gave the lowest value of MMRE in the first part of the experiment on the same dataset;
3. Validation on the ANN that gave the lowest value of MMRE in the first part of the experiment, but on other datasets.

3.1.1 Data sets used in the COCOMO2000 approach

The data on which the experiments were performed has heterogeneous nature; therefore it is necessary to check the data on several different data sources. Since the experiment uses artificial neural network architectures with three input values, the estimated effort for the software development project is calculated according to the COCOMO2000 formula (4-10) and is presented as the number of person-months [PM].

For the first experiment devoted to the training of models, a COCOMO2000 dataset of 100 projects for all four proposed ANN architectures was used. The COCOMO2000 dataset of 20 projects for the second stage of experiment, i.e. for testing developed models for all four proposed ANN architectures, was used.

To increase the reliability of our proposed approach, several datasets such as COCOMO81, where 51 random projects were selected, a NASA dataset with 60 projects, and a Kemerer dataset with 15 projects were used for the last stage of the experiment - i.e. for the validation of developed models, see Table 9.

From the datasets used [120], it can be observed that the actual effort ranges from a minimum of 6.0 person-months (PM) in Dataset_1 to a maximum of 11399.9 PM in Dataset_3. This leads to the conclusion that the range of data is with very large standard deviation, see Table 10.

Table 9. Information on used datasets (COCOMO2000).
Tabela 9. Informacije o korišćenim skupovima podataka (COCOMO2000).

	Dataset	Number of projects	Experiment
Dataset_1	COCOMO2000 dataset	100	Training
Dataset_2	COCOMO2000 dataset	20	Testing
Dataset_3	COCOMO81 dataset	Random 51	Validation1
Dataset_4	NASA dataset	60	Validation2
Dataset_5	Kemerer dataset	15	Validation3

Table 10. Basic statistics about datasets (COCOMO2000).
Tabela 10. Osnovni statistički podaci o korišćenim skupovima podataka (COCOMO2000).

Dataset	No. of projects	Min [PM]	Max [PM]	Mean [PM]	Std. deviation [PM]
Dataset_1	100	6.0	8211.0	616.0	1131.5
Dataset_2	20	28.1	606.8	277.0	206.3
Dataset_3	51	33.0	11399.9	841.8	1994.9
Dataset_4	60	8.4	3240.0	406.4	656.9
Dataset_5	15	23.2	1780.0	316.7	456.7

3.1.2 The methodology used within the improved COCOMO2000 model

The appropriate methodology was selected for the experimental part in the COCOMO2000 approach based on several trial experiments. The order of the steps in the experiments was constructed based on a robust design algorithm and shown in the Figure 10.

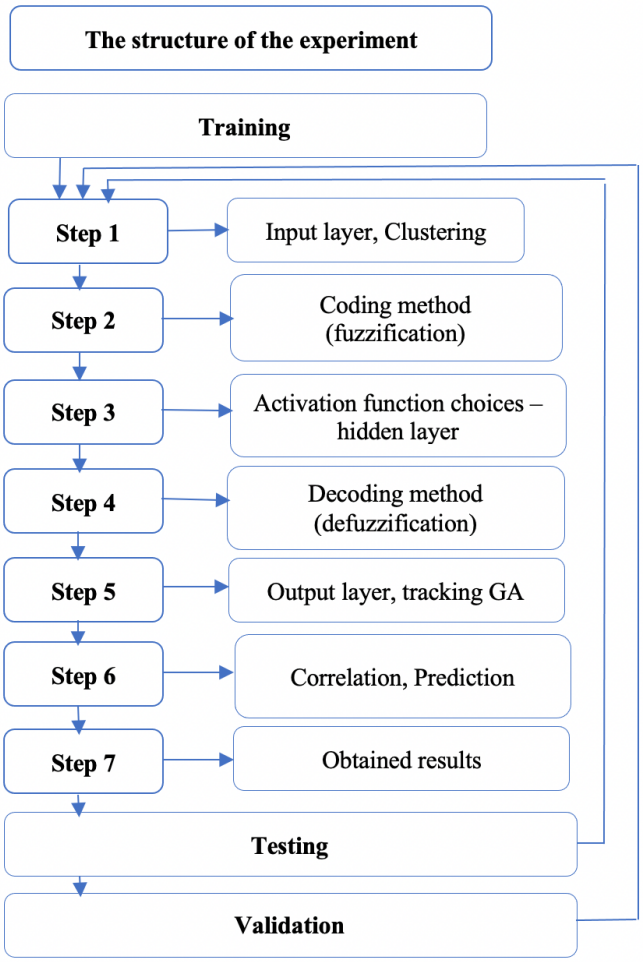


Figure 10. Robust design algorithm for performing the experiment (COCOMO2000).
Slika 10. Algoritam robusnog dizajna za izvođenje eksperimenta (COCOMO2000).

Step 1: Input layer, Clustering

The input layer consists of data from the first 100 projects of the COCOMO2000 dataset, divided into three clusters according to the value of actual effort expressed in person-months [PM]: less than 90PM (small projects), between 90PM and 500PM (medium projects), and more than 500PM (large projects). The three input values for the proposed neural network architectures ANN-L9, ANN-L18, ANN-L27, and ANN-L36 are: X1 = E, X2 = PEM_i, and X3 = KLOC (see formulas 5-7).

Step 2: Coding method (fuzzification)

In all trial experiments, the method that gave the best results was fuzzification which includes: mapping of all input values X1, X2 and X3 into real values given in the

interval [0, 1]: The function $\mu_D(X)$: $R \rightarrow [0, 1]$, translates the real values of input signals into coded values from the interval [0, 1] , in the following way: $\mu_D(X_i) = (X_i - X_{min})/(X_{max} - X_{min})$ (min-max normalization) [121], where D is the set of data on which the experiment is performed.

Step 3: Activation function choices - hidden layer

The functions of the hidden layer and the output layer use two different activation functions for proposed ANN architectures. *EstEffANN* represents the output value of proposed model (31):

1. $y_i = \frac{1}{1+e^{-x_i}}, i = \overline{1,n}$ (31)

a) None hidden and one output layer function for ANN-L9 architecture, see Figure 6, Table 5 (32):

$$EstEffANN - L9 = 1/\left(1 + e^{-(X_1 \cdot W_1 + X_2 \cdot W_2 + X_3 \cdot W_3 + 1 \cdot W_4)}\right) \quad (32)$$

where *EstEffANN-L9* is the output value.

b) One hidden and one output layer functions for ANN-L18 architecture, see Figure 7, Table 6 (33)-(35):

$$Y_1 = 1/\left(1 + e^{-(X_1 \cdot W_1 + X_2 \cdot W_3 + X_3 \cdot W_5)}\right) \quad (33)$$
$$Y_2 = 1/\left(1 + e^{-(X_1 \cdot W_2 + X_2 \cdot W_4 + X_3 \cdot W_6)}\right) \quad (34)$$
$$EstEffANN - L18 = 1/\left(1 + e^{-(Y_1 \cdot W_7 + Y_2 \cdot W_8)}\right) \quad (35)$$

where Y_1 and Y_2 are calculated values from hidden layer and *EstEffANN-L18* is the output value.

c) One hidden and one output layer functions for ANN-L27 architecture, see Figure 8, Table 7 (36)-(39):

$$Y_1 = \frac{1}{1+e^{-(X_1 \cdot W_1 + X_2 \cdot W_4 + X_3 \cdot W_7)}} \quad (36)$$
$$Y_2 = \frac{1}{1+e^{-(X_1 \cdot W_2 + X_2 \cdot W_5 + X_3 \cdot W_8)}} \quad (37)$$
$$Y_3 = \frac{1}{1+e^{-(X_1 \cdot W_3 + X_2 \cdot W_6 + X_3 \cdot W_9)}} \quad (38)$$
$$EstEffANN - L27 = \frac{1}{1+e^{-(Y_1 \cdot W_{10} + Y_2 \cdot W_{11} + Y_3 \cdot W_{12} + 1 \cdot W_{13})}} \quad (39)$$

where Y_1, Y_2, and Y_3 are calculated values from hidden layer and *EstEffANN-L27* is the output value.

d) Two hidden and one output layer functions for ANN-L36 architecture, see Figure 9, Table 7 (40)-(46):

$$Y_1 = \frac{1}{1+e^{-(X_1 \cdot W_1 + X_2 \cdot W_5 + X_3 \cdot W_9)}} \quad (40)$$

$$Y_2 = \frac{1}{1+e^{-(X_1 \cdot W_2 + X_2 \cdot W_6 + X_3 \cdot W_{10})}} \quad (41)$$

$$Y_3 = \frac{1}{1+e^{-(X_1 \cdot W_3 + X_2 \cdot W_7 + X_3 \cdot W_{11})}} \quad (42)$$

$$Y_4 = \frac{1}{1+e^{-(X_1 \cdot W_4 + X_2 \cdot W_8 + X_3 \cdot W_{12})}} \quad (43)$$

$$Z_1 = \frac{1}{1+e^{-(Y_1 \cdot W_{13} + Y_2 \cdot W_{15} + Y_3 \cdot W_{17} + Y_4 \cdot W_{19})}} \quad (44)$$

$$Z_2 = \frac{1}{1+e^{-(Y_1 \cdot W_{14} + Y_2 \cdot W_{16} + Y_3 \cdot W_{18} + Y_4 \cdot W_{20})}} \quad (45)$$

$$EstEffANN - L36 = \frac{1}{1+e^{-(Z_1 \cdot W_{21} + Z_2 \cdot W_{22} + 1 \cdot W_{23})}} \quad (46)$$

where Y_1, Y_2, Y_3, and Y_4 are calculated values from the first hidden layer, Z_1 and Z_2 are calculated values from the second hidden layer and *EstEffANN-L36* is the output value.

Example of hyperbolic tangent for ANN-L27 and ANN-L36 according to formula (47):

$$2.\ y_i = \frac{e^{x_i} - e^{-x_i}}{e^{x_i} + e^{-x_i}},\ i = \overline{1,n} \quad (47)$$

a) Hidden and output layer functions for ANN-L27 architecture, according to formulas (48)-(51):

$$Y_1 = \frac{e^{(X_1 \cdot W_1 + X_2 \cdot W_4 + X_3 \cdot W_7)} - e^{-(X_1 \cdot W_1 + X_2 \cdot W_4 + X_3 \cdot W_7)}}{e^{(X_1 \cdot W_1 + X_2 \cdot W_4 + X_3 \cdot W_7)} + e^{-(X_1 \cdot W_1 + X_2 \cdot W_4 + X_3 \cdot W_7)}} \quad (48)$$

$$Y_2 = \frac{e^{(X_1 \cdot W_2 + X_2 \cdot W_5 + X_3 \cdot W_8)} - e^{-(X_1 \cdot W_2 + X_2 \cdot W_5 + X_3 \cdot W_8)}}{e^{(X_1 \cdot W_2 + X_2 \cdot W_5 + X_3 \cdot W_8)} + e^{-(X_1 \cdot W_2 + X_2 \cdot W_5 + X_3 \cdot W_8)}} \quad (49)$$

$$Y_3 = \frac{e^{(X_1 \cdot W_3 + X_2 \cdot W_5 + X_3 \cdot W_9)} - e^{-(X_1 \cdot W_3 + X_2 \cdot W_5 + X_3 \cdot W_9)}}{e^{(X_1 \cdot W_3 + X_2 \cdot W_5 + X_3 \cdot W_9)} + e^{-(X_1 \cdot W_3 + X_2 \cdot W_5 + X_3 \cdot W_9)}} \quad (50)$$

$$EstEffANN - L27 = \frac{e^{(Y_1 \cdot W_{10} + Y_2 \cdot W_{11} + Y_3 \cdot W_{12} + 1 \cdot W_{13})} - e^{-(Y_1 \cdot W_{10} + Y_2 \cdot W_{11} + Y_3 \cdot W_{12} + 1 \cdot W_{13})}}{e^{(Y_1 \cdot W_{10} + Y_2 \cdot W_{11} + Y_3 \cdot W_{12} + 1 \cdot W_{13})} + e^{-(Y_1 \cdot W_{10} + Y_2 \cdot W_{11} + Y_3 \cdot W_{12} + 1 \cdot W_{13})}} \quad (51)$$

where Y_1, Y_2, Y_3 are calculated values from hidden layer and *EstEffANN-L27* is the output value.

b) Hidden and output layer functions for ANN-L36 architecture, according to formulas (52)-(58):

$$Y_1 = \frac{e^{(X_1 \cdot W_1 + X_2 \cdot W_5 + X_3 \cdot W_9)} - e^{-(X_1 \cdot W_1 + X_2 \cdot W_5 + X_3 \cdot W_9)}}{e^{(X_1 \cdot W_1 + X_2 \cdot W_5 + X_3 \cdot W_9)} + e^{-(X_1 \cdot W_1 + X_2 \cdot W_5 + X_3 \cdot W_9)}} \quad (52)$$

$$Y_2 = \frac{e^{(X_1 \cdot W_2 + X_2 \cdot W_6 + X_3 \cdot W_{10})} - e^{-(X_1 \cdot W_2 + X_2 \cdot W_6 + X_3 \cdot W_{10})}}{e^{(X_1 \cdot W_2 + X_2 \cdot W_6 + X_3 \cdot W_{10})} + e^{-(X_1 \cdot W_2 + X_2 \cdot W_6 + X_3 \cdot W_{10})}} \quad (53)$$

$$Y_3 = \frac{e^{(X_1 \cdot W_3 + X_2 \cdot W_7 + X_3 \cdot W_{11})} - e^{-(X_1 \cdot W_3 + X_2 \cdot W_7 + X_3 \cdot W_{11})}}{e^{(X_1 \cdot W_3 + X_2 \cdot W_7 + X_3 \cdot W_{11})} + e^{-(X_1 \cdot W_3 + X_2 \cdot W_7 + X_3 \cdot W_{11})}} \quad (54)$$

$$Y_4 = \frac{e^{(X_1 \cdot W_4 + X_2 \cdot W_8 + X_3 \cdot W_{12})} - e^{-(X_1 \cdot W_4 + X_2 \cdot W_8 + X_3 \cdot W_{12})}}{e^{(X_1 \cdot W_4 + X_2 \cdot W_8 + X_3 \cdot W_{12})} + e^{-(X_1 \cdot W_4 + X_2 \cdot W_8 + X_3 \cdot W_{12})}} \quad (55)$$

$$Z_1 = \frac{e^{(Y_1 \cdot W_{13} + Y_2 \cdot W_{15} + Y_3 \cdot W_{17} + Y_4 \cdot W_{19})} - e^{-(Y_1 \cdot W_{13} + Y_2 \cdot W_{15} + Y_3 \cdot W_{17} + Y_4 \cdot W_{19})}}{e^{(Y_1 \cdot W_{13} + Y_2 \cdot W_{15} + Y_3 \cdot W_{17} + Y_4 \cdot W_{19})} + e^{-(Y_1 \cdot W_{13} + Y_2 \cdot W_{15} + Y_3 \cdot W_{17} + Y_4 \cdot W_{19})}} \quad (56)$$

$$Z_2 = \frac{e^{(Y_1 \cdot W_{14} + Y_2 \cdot W_{16} + Y_3 \cdot W_{18} + Y_4 \cdot W_{20})} - e^{-(Y_1 \cdot W_{14} + Y_2 \cdot W_{16} + Y_3 \cdot W_{18} + Y_4 \cdot W_{20})}}{e^{(Y_1 \cdot W_{14} + Y_2 \cdot W_{16} + Y_3 \cdot W_{18} + Y_4 \cdot W_{20})} + e^{-(Y_1 \cdot W_{14} + Y_2 \cdot W_{16} + Y_3 \cdot W_{18} + Y_4 \cdot W_{20})}} \quad (57)$$

$$EstEffANN - L36 = \frac{e^{(Z_1 \cdot W_{21} + Z_2 \cdot W_{22} + Y_3 \cdot W_{18+1} \cdot W_{23})} - e^{-(Z_1 \cdot W_{21} + Z_2 \cdot W_{22} + Y_3 \cdot W_{18+1} \cdot W_{23})}}{e^{(Z_1 \cdot W_{21} + Z_2 \cdot W_{22} + Y_3 \cdot W_{18+1} \cdot W_{23})} + e^{-(Z_1 \cdot W_{21} + Z_2 \cdot W_{22} + Y_3 \cdot W_{18+1} \cdot W_{23})}} \quad (58)$$

where Y_1, Y_2, Y_3, and Y_4 are calculated values from the first hidden layer, Z_1 and Z_2 are calculated values from the second hidden layer and *EstEffANN-L36* is the output value.

Initial values in all proposed ANN architectures for weighting factors are taken from set [-1, 0, 1]. After the first iteration, the value of the cost effect function is calculated and arranged based on Taguchi Orthogonal Array and according to the following formula - for example, ANN-L27 and ANN-L36 are arranged as follows [7], [119] (59):

$$\begin{aligned}
&\mathrm{L1W}_1 = \mathrm{cost1} + \mathrm{cost2} + \ldots + \mathrm{cost9} \\
&\mathrm{L2W}_1 = \mathrm{cost10} + \mathrm{cost11} + \ldots + \mathrm{cost18} \\
&\mathrm{L3W}_1 = \mathrm{cost19} + \mathrm{cost20} + \ldots + \mathrm{cost27} \\
&\ldots. \\
&\mathrm{L1W}_{13} = \mathrm{cost1} + \mathrm{cost5} + \ldots + \mathrm{cost26} \\
&\mathrm{L1W}_{13} = \mathrm{cost2} + \mathrm{cost6} + \ldots + \mathrm{cost27} \\
&\mathrm{L1W}_{13} = \mathrm{cost3} + \mathrm{cost4} + \ldots + \mathrm{cost25}
\end{aligned}$$

$$\text{where } \mathrm{cost}(i) = \Sigma\, \mathrm{MRE}(\text{ANN-L27}(i)) \quad (59)$$

Calculating the levels for ANN-L36 architecture [7], [119] according to formula (60):

$$\begin{aligned}
&\mathrm{L1W}_1 = \mathrm{cost1} + \mathrm{cost2} + \ldots + \mathrm{cost18} \\
&\mathrm{L2W}_1 = \mathrm{cost19} + \mathrm{cost20} + \ldots + \mathrm{cost36} \\
&\ldots. \\
&\mathrm{L1W}_{23} = \mathrm{cost1} + \mathrm{cost5} + \ldots + \mathrm{cost34} \\
&\mathrm{L2W}_{23} = \mathrm{cost2} + \mathrm{cost6} + \ldots + \mathrm{cost35} \\
&\mathrm{L3W}_{23} = \mathrm{cost3} + \mathrm{cost4} + \ldots + \mathrm{cost36}
\end{aligned}$$

$$\text{where } \mathrm{cost}(i) = \Sigma\, \mathrm{MRE}(\text{ANN-L36}(i)) \quad (60)$$

For each subsequent iteration, the interval is divided as follows [7], [119] according to formula (61):

$$L1W_1new = L2W_1old$$
$$L2W_1new = L2W_1old + (L3W1old - L2W1old)/2$$
$$L3W_1new = L3W_1old \quad (61)$$

where suffix "*old*" means values from the interval of the previous iteration, and "*new*" means the value calculated based on the division of the previous intervals.

The set of input values of each dataset converges depending on the value of the cost effect function. An example of dividing the interval and calculating the value of the cost effect function through 6 iterations are given in Figures 11-16, and Tables 11-20.

Table 11. Cost effect function values - 1st iteration.
Tabela 11. Vrednosti funkcije troškova u prvoj iteraciji.

Cost effect function values for the 1st iteration													
W_i	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
L1	70.64	70.04	70.68	70.72	70.48	70.53	70.59	70.66	70.54	70.91	70.89	70.84	71.05
L2	70.59	70.86	70.59	70.57	70.71	70.74	70.52	70.63	70.63	70.54	70.57	70.55	70.77
L3	70.58	70.91	70.55	70.52	70.62	70.54	70.70	70.64	70.64	70.36	70.34	70.42	69.99

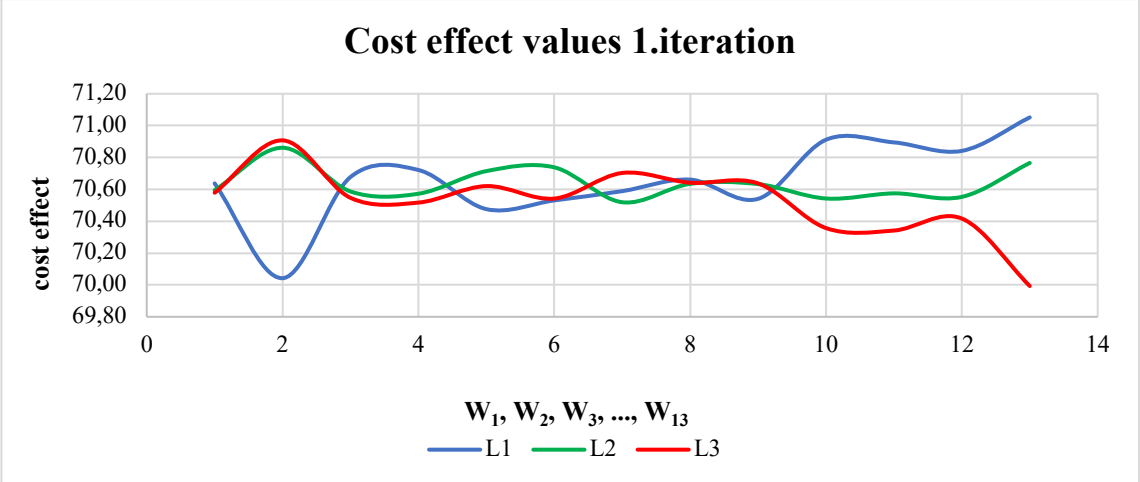


Figure 11. Graphical representation of the cost effect function values - 1st iteration.
Slika 11. Grafička reprezentacija vrednosti funkcije troškova tokom prve iteracije.

From Figure 11 and Table 11 it can be concluded that the weighting factors W_2, W_{10} and W_{13} have the greatest influence on the change of MRE values in the next iteration. Weighting factor W_8 has the smallest impact in this iteration.

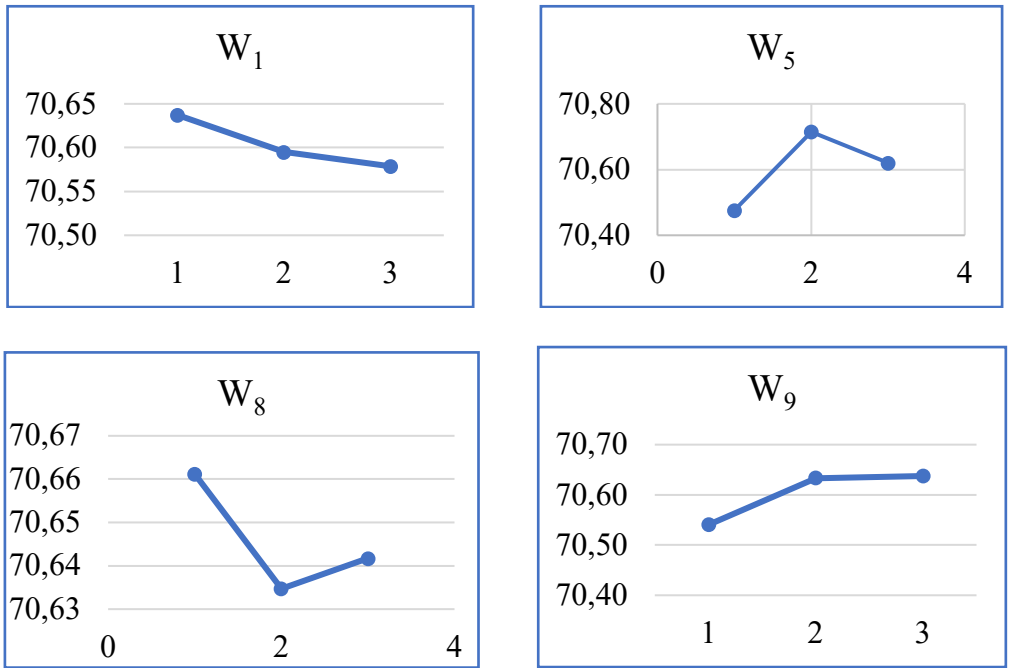


Figure 12. Division of the interval according to the value of the cost effect function.
Slika 12. Podela intervala prema vrednostima funkcije troškova.

Table 12. Intervals for the 2nd iteration.
Tabela 12. Vrednosti intervala u drugoj iteraciji.

Intervals for the 2nd iteration													
W_i	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
L1	0.00	-1.00	0.00	0.00	-1.00	-1.00	-1.00	0.00	-1.00	0.00	0.00	0.00	0.00
L2	0.50	-0.50	0.50	0.50	-0.50	-0.50	-0.50	0.50	-0.50	0.50	0.50	0.50	0.50
L3	1.00	0.00	1.00	1.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00	1.00	1.00

Table 13. Cost effect function values - 2nd iteration.
Tabela 13. Vrednosti funkcije troškova u drugoj iteraciji.

Cost effect function values for the 2nd iteration													
W_i	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
L1	69.88	69.68	69.85	69.87	69.74	69.83	69.86	69.91	69.84	69.88	69.99	69.87	70.06
L2	69.78	69.90	69.80	69.79	69.85	69.83	69.81	69.78	69.84	69.77	69.79	69.77	69.83
L3	69.76	69.84	69.78	69.76	69.83	69.76	69.75	69.78	69.75	69.77	69.65	69.78	69.53

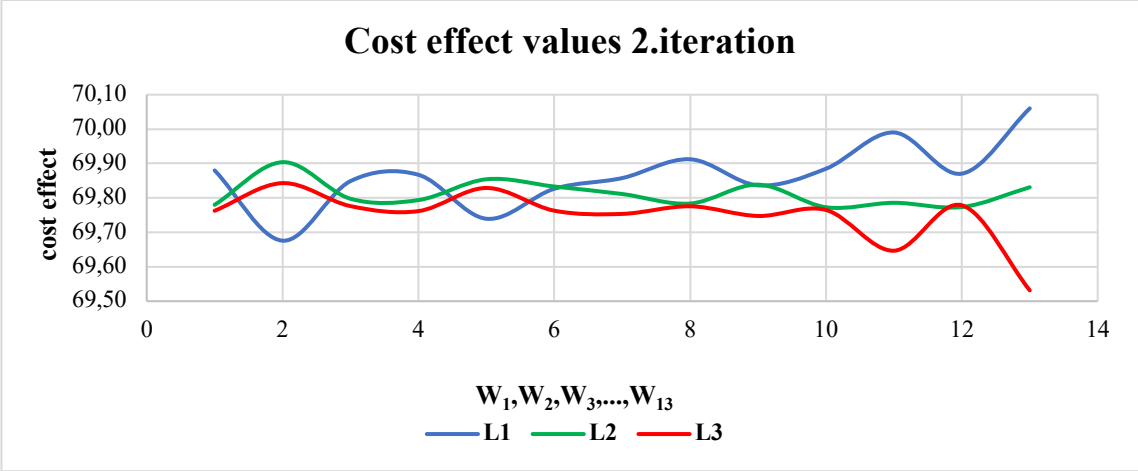


Figure 13. Graphical representation of the cost effect function values - 2nd iteration.
Slika 13. Grafička reprezentacija vrednosti funkcije troškova tokom druge iteracije.

From Figure 13 and Table 13 it can be concluded that the weighting factors W_{11} and W_{13} have the greatest influence on the change of MRE values in the next iteration. Weighting factors W_3 and W_5 have the smallest impact in this iteration.

Table 14. Intervals for the 3rd iteration.
Tabela 14. Vrednosti intervala u drugoj iteraciji.

Intervals for the 3rd iteration													
W_i	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
L1	0.50	-1.00	0.50	0.50	-1.00	-0.50	-0.50	0.50	-0.50	0.50	0.50	0.50	0.50
L2	0.75	-0.75	0.75	0.75	-0.75	-0.25	-0.25	0.75	-0.25	0.75	0.75	0.75	0.75
L3	1.00	-0.50	1.00	1.00	-0.50	0.00	0.00	1.00	0.00	1.00	1.00	1.00	1.00

Table 15. Cost effect function values - 3rd iteration.
Tabela 15. Vrednosti funkcije troškova u trećoj iteraciji.

Cost effect function values for the 3rd iteration													
W_i	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
L1	69.41	69.38	69.41	69.42	69.39	69.40	69.44	69.42	69.43	69.43	69.46	69.41	69.46
L2	69.37	69.39	69.37	69.37	69.38	69.39	69.38	69.38	69.39	69.37	69.37	69.37	69.39
L3	69.37	69.38	69.36	69.36	69.38	69.36	69.33	69.37	69.33	69.35	69.32	69.36	69.31

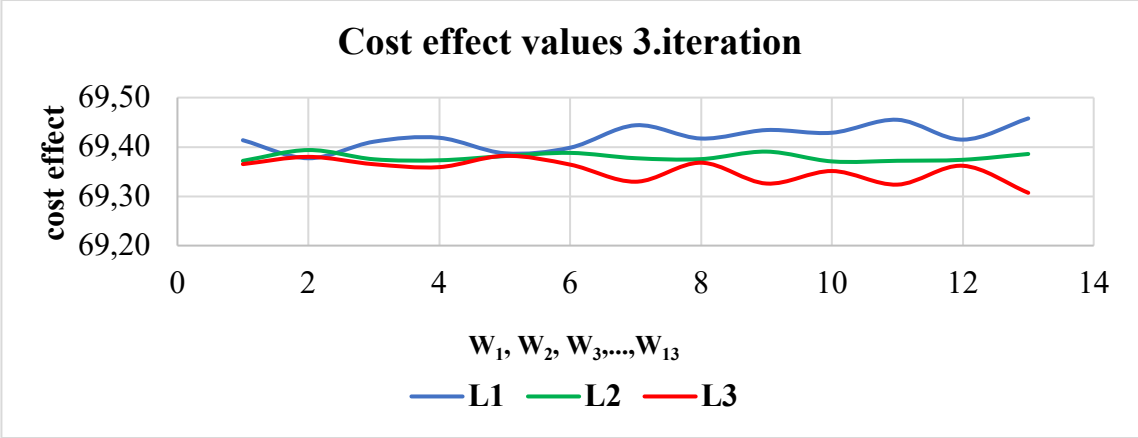


Figure 14. Graphical representation of the cost effect function values - 3rd iteration.
Slika 14. Grafička reprezentacija vrednosti funkcije troškova tokom treće iteracije.

From Figure 14 and Table 15 it can be concluded that the weighting factors W_{11} and W_{13} have the greatest influence on the change of MRE values in the next iteration. Weighting factors W_2 and W_5 have the smallest impact in this iteration.

Table 16. Intervals for the 4th iteration.
Tabela 16. Vrednosti intervala u četvrtoj iteraciji.

Intervals for the 4th iteration													
W_i	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
L1	0.75	-1.00	0.75	0.75	-0.75	-0.25	-0.25	0.75	-0.25	0.75	0.75	0.75	0.75
L2	0.88	-0.88	0.88	0.88	-0.63	-0.13	-0.13	0.88	-0.13	0.88	0.88	0.88	0.88
L3	1.00	-0.75	1.00	1.00	-0.50	0.00	0.00	1.00	0.00	1.00	1.00	1.00	1.00

Table 17. Cost effect function values - 4th iteration.
Tabela 17. Vrednosti funkcije troškova u četvrtoj iteraciji.

Cost effect function values for the 4th iteration													
W_i	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
L1	69.22	69.22	69.22	69.23	69.22	69.22	69.24	69.22	69.24	69.23	69.24	69.23	69.24
L2	69.21	69.22	69.21	69.21	69.21	69.21	69.21	69.21	69.22	69.21	69.21	69.21	69.21
L3	69.21	69.21	69.21	69.21	69.21	69.21	69.19	69.21	69.19	69.20	69.20	69.20	69.19

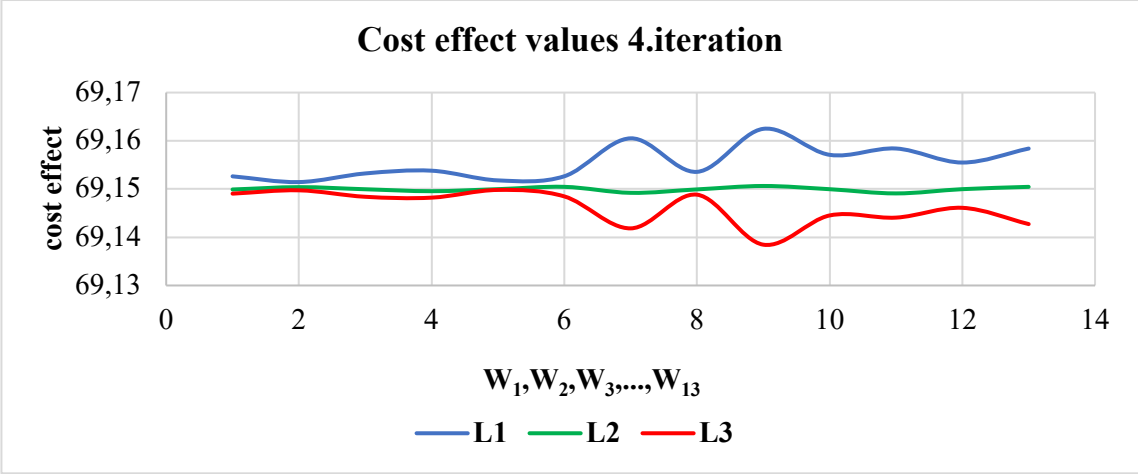


Figure 15. Graphical representation of the cost effect function values - 4th iteration.
Slika 15. Grafička reprezentacija vrednosti funkcije troškova tokom četvrte iteracije.

From Figure 15 and Table 17 it can be concluded that the weighting factors W_7 and W_9 have the greatest influence on the change of MRE values in the next iteration. Weighting factors W_2 and W_5 have the smallest impact in this iteration.

Table 18. Intervals for the 5th iteration.
Tabela 18. Vrednosti intervala u petoj iteraciji.

Intervals for the 5th iteration													
W_i	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
L1	0.88	-0.88	0.88	0.88	-0.63	-0.13	-0.13	0.88	-0.13	0.88	0.88	0.88	0.88
L2	0.94	-0.81	0.94	0.94	-0.56	-0.06	-0.06	0.94	-0.06	0.94	0.94	0.94	0.94
L3	1.00	-0.75	1.00	1.00	-0.50	0.00	0.00	1.00	0.00	1.00	1.00	1.00	1.00

Table 19. Cost effect function values - 5th iteration.
Tabela 19. Vrednosti funkcije troškova u petoj iteraciji.

Cost effect function values for the 5th iteration													
W_i	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
L1	69.15	69.15	69.15	69.15	69.15	69.15	69.16	69.15	69.16	69.16	69.16	69.16	69.16
L2	69.15	69.15	69.15	69.15	69.15	69.15	69.15	69.15	69.15	69.15	69.15	69.15	69.15
L3	69.15	69.15	69.15	69.15	69.15	69.15	69.14	69.15	69.14	69.14	69.14	69.15	69.14

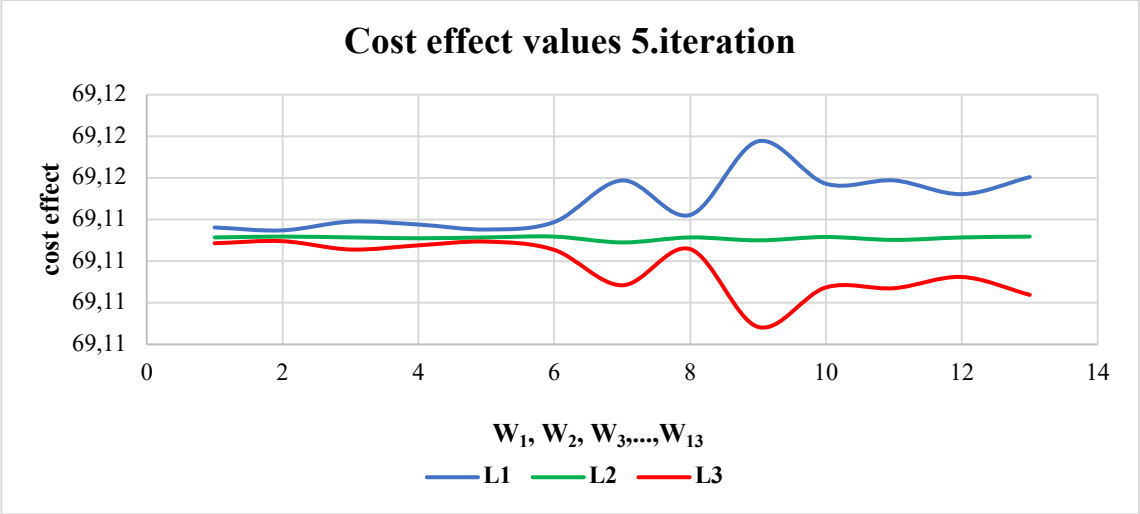


Figure 16. Graphical representation of the cost effect function values - 5th iteration.
Slika 16. Grafička reprezentacija vrednosti funkcije troškova tokom pete iteracije.

From Figure 16 and Table 19 it can be concluded that the weighting factors W_9 and W_{10} have the greatest influence on the change of MRE values in the next iteration. Weighting factors W_2 and W_5 have the smallest impact in this iteration.

Table 20. Intervals for the 6th iteration.
Tabela 20. Vrednosti intervala u šestoj iteraciji.

Intervals for the 6th iteration													
W_i	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}
L1	0.88	-0.88	0.88	0.88	-0.63	-0.13	-0.13	0.88	-0.13	0.88	0.88	0.88	0.88
L2	0.94	-0.81	0.94	0.94	-0.56	-0.06	-0.06	0.94	-0.06	0.94	0.94	0.94	0.94
L3	1.00	-0.75	1.00	1.00	-0.50	0.00	0.00	1.00	0.00	1.00	1.00	1.00	1.00

By the interval division procedure, the MRE value decreases significantly in each subsequent iteration. The number of iterations required to reach the minimum relative error depends on the cost effect function. The rate at which each of the proposed ANNs advances toward the minimal magnitude relative error is actually its rate of convergence. e.g., the cost effect values of the ANN-L27 network function are calculated for each level L1, L2, and L3 after each iteration is performed. The cost effect function needs to be calculated for each of the architectures ANN-L9, ANN-L18, ANN-L27, and ANN-L36 according to the Taguchi Orthogonal Arrays. This procedure should be repeated until the end of iterations, i.e. until the conditions are reached when GA <0.01. The figures above show the value of the cost effect function at all three levels, for all 5 iterations - for example, on the selected ANN-L27 architecture. From the figures it can also be concluded

how much is the influence of each weighting factor on the rate of convergence of the proposed ANN architecture. There are 13 weighting factors in the ANN-L27 architecture: W_1, W_2,...,W_{13}. The closer the curves are, the contribution of weighting factors is minimal, for example from W_1 to W_5 on Figure 16. In contrast, if the curves are more distant, the contribution of their weighting factors to the convergence rate is higher, for example, from W_8 to W_{10} on Figure 16. From this it can be concluded that not all weight factors have a uniform influence on the rate of convergence. The same procedure is repeated for all proposed architectures for each iteration. The rate of convergence of each ANN architecture depends on the appropriate division of the interval, which is performed based on the values of the cost effect function. As the values of input signals (projects) are heterogeneous, then the values of the cost effect function for all three levels are different. In the presented Figures 11-16 it can be seen that the values of the cost effect function for all three levels are uneven, i.e. that the network initially converges poorly. Dividing the interval properly, in each iteration performed, the symmetry of the first and third levels is achieved. When the second level has equal values, the ANN architecture converges to a specific value (MRE).

Step 4: Decoding method (defuzzification)

In all parts of the performed experiment, the appropriate method of defuzzification (decoding) was used according to the following formulas [122] (62)-(64):

$$X_i = (X_{min} + \mu_D(X_i)) \cdot (X_{max} - X_{min}) \quad (62)$$

$$OA(ANN - L27 = X_i), i = \overline{(1, 27)} \quad (63)$$

$$OA(ANN - L36 = X_i), i = \overline{(1, 36)} \quad (64)$$

where i=9, i=18, i=27 and i=36.

Step 5: Output layer, tracking GA

The performance of estimation strategies should be evaluated. There are a range of quantitative relation measurements of accuracy metrics (etc.) that could be used for assessing the quality of a particular estimation strategy. For each iteration in this experiment, the output values are obtained according to the following formulas/measures (65)-(69) [123]:

$$Deviation = |ActEffort - EstEffort| \quad (65)$$

$$MAE_i = \frac{1}{n}\sum_{i=1}^{n} |ActEffort - EstEffort| \quad (66)$$

$$MRE = Deviation/ActEffort \quad (67)$$

$$MRE = \frac{1}{n} \cdot \sum_{i=1}^{n} MRE_i \quad (68)$$

$$\text{MMRE} = mean\,(\text{MRE}) \quad (69)$$

For each of the experimental parts in every iteration, the Gradient Descent is monitored with the condition GA<0.01, calculated as (70):

$$GA = MRE_{i1} - MRE_{i2} < 0.01,$$
$$where\ i = 1, \dots, n \quad n\ is\ a\ number\ of\ ANN\ a\ Winner\ candidates. \quad (70)$$

The difference of the minimum values for each iteration in each ANN architecture is denoted by delta, where delta(i) = δ_i, and is calculated as follows (71), (72):

$$\delta_i = MMRE_{ik} - MMRE_{(i+1)k} \quad i - number\ of\ ANN,\ k - number\ of\ iteration. \quad (71)$$

$$\text{if } \delta_i > \delta_{i+1},\ then\ ANN_i\ converges\ to\ MMRE_i\ for\ each\ of\ the\ proposed\ ANN\ architectures. \quad (72)$$

In this way, it can be determined which of the ANN architectures meets the given criterion of rapidity, i.e. which one converges at the fastest rate. By experimenting with different ANN architectures, a GA criterion was set to meet the required number of iterations or stopping convergence at GA <0.01.

In the training part of the experiment of the each "ANN candidate" of the selected ANN architecture according to Taguchi's Orthogonal Array, in each subsequent iteration, a reduction of MRE of less than 1% is achieved, which in this experiment represents a "stop criterion" [114], [115].

Step 6: Correlation, Prediction

Correlation represents the interrelationship between real and estimated value. The higher the correlation coefficient, the more stable the correlation. In this experiment, Pearson's [124], Spearman's [125] and R^2 [126] test were followed. Correlation ranges from +1 to -1 and a correlation of +1 means that there is a perfect positive relationship between variables. The general formula for calculating the correlation coefficients between two variable is as follows (73):

$$Correl(X,Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \quad (73)$$

Prediction at three criteria: PRED(25), PRED(30), and PRED(50) to calculate the percentage of the total number of ANNs that meet the GA criterion, see formula (74), [127].

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

PRED(k) = count(MRE) < 25%
PRED(k) = count(MRE) < 30%
PRED(k) = count(MRE) < 50% , where k = 25, k = 30, and k = 50. (74)

Step 7: Obtained results

Testing of trained models is performed following the same methodology and algorithm related to other projects (these are not used in the training part of the experiment) in the same dataset. Validation as another quality check of proposed and trained models is performed also following the same methodology, but on different sources/other datasets. Also, testing and validation, are performed on the ANN network that gives the best results for each of the proposed ANN architectures.

3.2 New, improved COSMIC FFP model

As part of the COSMIC FFP approach, an improved COSMIC FFP model is presented in this subsection. For the improved COSMIC FFP model, the following architectures and corresponding orthogonal vector plans L12 and L36prim are used:

1. COSMIC FFP and ANN-L12

The first proposed architecture is ANN-L12. It consists of four input values, one hidden layer with two nodes, one output, and the total number of eleven weighting factors (Wi, i = $\overline{1,11}$) and whose initial values are from the interval [-1, 1]. The Taguchi Orthogonal Array used in the construction of this proposed architecture contains two levels L1 and L2, see Figure 17, Table 21 [7], [8], [114]. Bias represents an additional weighting factor to complete the selected orthogonal plan and has a value of 1.

Table 21. Taguchi orthogonal vector plan (L12=2^{11}).
Tabela 21. Taguči ortogonalni vektorski plan (L12=2^{11}).

ANN-L12	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}
ANN1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
ANN2	L1	L1	L1	L1	L1	L2	L2	L2	L2	L2	L2
ANN3	L1	L1	L2	L2	L2	L1	L1	L1	L2	L2	L2
ANN4	L1	L2	L1	L2	L2	L1	L2	L2	L1	L1	L2
ANN5	L1	L2	L2	L1	L2	L2	L1	L2	L1	L2	L1
ANN6	L1	L2	L2	L2	L1	L2	L2	L1	L2	L1	L1
ANN7	L2	L1	L2	L2	L1	L1	L2	L2	L1	L2	L1
ANN8	L2	L1	L2	L1	L2	L2	L2	L1	L1	L1	L2
ANN9	L2	L1	L1	L2	L2	L2	L1	L2	L2	L1	L1
ANN10	L2	L2	L2	L1	L1	L1	L1	L2	L2	L1	L2
ANN11	L2	L2	L1	L2	L1	L2	L1	L1	L1	L2	L2
ANN12	L2	L2	L1	L1	L2	L1	L2	L1	L2	L2	L1

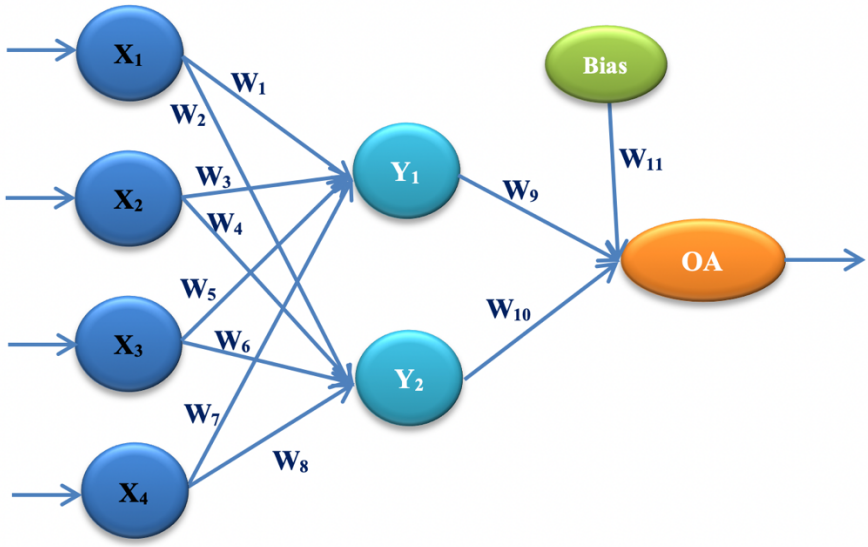


Figure 17. ANN architecture with one hidden layer (ANN-L12).
Slika 17. ANN arhitektura sa jednim skrivenim slojem (ANN-L12).

2. COSMIC FFP and ANN-L36prim

The second proposed architecture is ANN-L36prim. It consists of four input values, one hidden layer with three nodes, one output, and the total number of sixteen weighting factors (Wi, i = $\overline{1,16}$) and whose initial values are from the interval [-1, 1]. The Taguchi Orthogonal Array used in the construction of this proposed architecture

contains two levels L1 and L2, see Figure 18, Table 22 [7], [8], [114]. Bias represents an additional weighting factor to complete the selected orthogonal plan and has a value of 1.

Table 22. Taguchi orthogonal vector plan (L36prim=$3^{11}2^{4}3^{1}$).
Tabela 22. Taguči ortogonalni vektorski plan (L36prim=$3^{11}2^{4}3^{1}$).

ANN-L36 prim	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}	W_{16}
ANN1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
ANN2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L1	L1	L1	L1
ANN3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L1	L1	L1	L1
ANN4	L1	L1	L1	L1	L2	L2	L2	L2	L3	L3	L3	L3	L1	L2	L2	L1
ANN5	L1	L1	L1	L1	L3	L3	L3	L3	L2	L2	L2	L2	L1	L2	L2	L1
ANN6	L3	L3	L3	L3	L1	L1	L1	L1	L2	L2	L2	L2	L1	L2	L2	L1
ANN7	L1	L1	L2	L3	L1	L2	L3	L3	L1	L1	L1	L3	L2	L1	L2	L1
ANN8	L2	L2	L3	L1	L2	L3	L1	L1	L2	L3	L3	L1	L2	L1	L2	L1
ANN9	L3	L3	L1	L2	L3	L1	L2	L2	L3	L1	L1	L2	L2	L1	L2	L1
ANN10	L1	L1	L3	L2	L1	L3	L2	L3	L2	L1	L3	L2	L2	L2	L1	L1
ANN11	L2	L2	L1	L3	L2	L1	L3	L1	L3	L2	L1	L3	L2	L2	L1	L1
ANN12	L3	L3	L2	L1	L3	L2	L1	L2	L1	L3	L2	L1	L2	L2	L1	L1
ANN13	L1	L2	L3	L1	L3	L2	L1	L3	L3	L2	L1	L2	L1	L1	L1	L2
ANN14	L2	L3	L1	L2	L1	L3	L2	L1	L1	L3	L2	L3	L1	L1	L1	L2
ANN15	L3	L1	L2	L3	L2	L1	L3	L2	L2	L1	L3	L1	L1	L1	L1	L2
ANN16	L1	L2	L3	L2	L1	L1	L3	L2	L3	L3	L2	L1	L1	L2	L2	L2
ANN17	L2	L3	L1	L3	L2	L2	L1	L3	L1	L1	L3	L2	L1	L2	L2	L2
ANN18	L3	L1	L2	L1	L3	L3	L2	L1	L2	L2	L1	L3	L1	L2	L2	L2
ANN19	L1	L2	L1	L3	L3	L3	L1	L2	L2	L1	L2	L3	L2	L1	L2	L2
ANN20	L2	L3	L2	L1	L1	L1	L2	L3	L3	L2	L3	L1	L2	L1	L2	L2
ANN21	L3	L1	L3	L2	L2	L2	L3	L1	L1	L3	L1	L2	L2	L1	L2	L2
ANN22	L1	L2	L2	L3	L3	L1	L2	L1	L1	L3	L3	L2	L2	L2	L1	L2
ANN23	L2	L3	L3	L1	L1	L2	L3	L2	L2	L1	L1	L3	L2	L2	L1	L2
ANN24	L3	L1	L1	L2	L2	L3	L1	L3	L3	L2	L2	L1	L2	L2	L1	L2
ANN25	L1	L3	L2	L1	L2	L3	L3	L1	L3	L1	L2	L2	L1	L1	L1	L3
ANN26	L2	L1	L3	L2	L3	L1	L1	L2	L1	L2	L3	L3	L1	L1	L1	L3
ANN27	L3	L2	L1	L3	L1	L2	L2	L3	L2	L3	L1	L1	L1	L1	L1	L3
ANN28	L1	L3	L2	L2	L2	L1	L1	L3	L2	L3	L1	L3	L1	L2	L2	L3
ANN29	L2	L1	L3	L3	L3	L2	L2	L1	L3	L1	L2	L1	L1	L2	L2	L3
ANN30	L3	L2	L1	L1	L1	L3	L3	L2	L1	L2	L3	L2	L1	L2	L2	L3
ANN31	L1	L3	L3	L3	L2	L3	L2	L2	L1	L2	L1	L1	L2	L1	L2	L3

ANN32	L2	L1	L1	L1	L3	L1	L3	L3	L3	L3	L2	L2	L2	L1	L2	L3
ANN33	L3	L2	L2	L2	L1	L2	L1	L1	L3	L1	L3	L3	L2	L1	L2	L3
ANN34	L1	L3	L1	L2	L3	L2	L3	L1	L2	L2	L3	L1	L2	L2	L1	L3
ANN35	L2	L1	L2	L3	L1	L3	L1	L2	L3	L3	L1	L2	L2	L2	L1	L3
ANN36	L3	L2	L3	L1	L2	L1	L2	L3	L1	L1	L2	L3	L2	L2	L1	L3

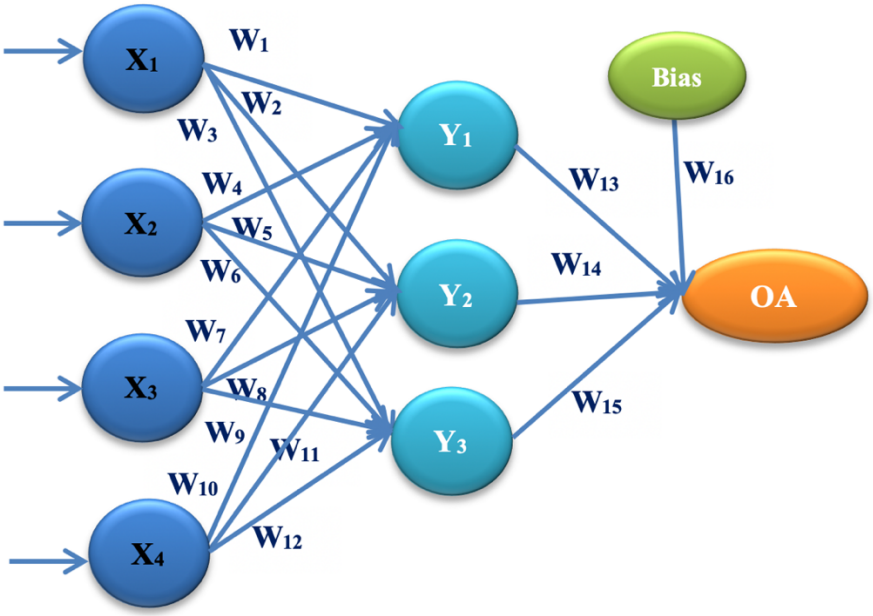


Figure 18. ANN architecture with one hidden layer (ANN-L36prim).
Slika 18. ANN arhitektura sa jednim skrivenim slojem (ANN-L36prim).

The experiment presented in this approach consists of three parts:

1. Training of two different ANN architectures constructed according to the corresponding Taguchi orthogonal vector plans (ANN-L12 and ANN36prim);
2. Testing on ANN "Winner", which gave the best results (the lowest MMRE value) in the first part of the experiment, for two proposed architectures on the same dataset;
3. Validation on ANN "Winner" that gave the best results (the lowest MMRE value) in the first part of the experiment, for each selected architecture, but using some other i.e. different datasets.

3.2.1 Data sets used in the COSMIC FFP approach

For the first and second part of the experiment, the ISBSG [15] repository was used. In the 3rd part, different data sets are used: Desharnais dataset (avaiable at *http://promise.site.uottawa.ca*) and a combined dataset composed of projects of different companies. ISBSG offers a variety number of information regarding practices from

various organizations, applications, and development types, which represent its main potential. The ISBSG suggests that the most important criteria for estimation purposes are the functional values; the development type (new development, enhancement, or re-development); the primary programming language or the language type/generation (e.g., 3GL, 4GL); and the development platform (mainframe, midrange or PC).

The results in Table 23. indicate the heterogeneous nature of the designs of each dataset used and within all three parts of the experiment. It can be seen that data sets in this approach [128], [129] are very different in terms of the programming languages used, the duration of application development, and an extensive range of functional values, with a large standard deviation, see Table 24.

Table 23. Information on used datasets (COSMIC FFP).
Tabela 23. Informacije o korišćenim skupovima podataka (COSMIC FFP).

	Datasets	Number of project	Experiment
Dataset_1	ISBSG (Functional Size<10)	37	Training
	ISBSG (Functional Size<10)	15	Testing
Dataset_2	ISBSG (10<Functional Size<50)	45	Training
	ISBSG (10<Functional Size<50)	17	Testing
Dataset_3	ISBSG (50<Functional Size<100)	30	Training
	ISBSG (10<Functional Size<100)	13	Testing
Dataset_4	ISBSG (100<Functional Size<500)	60	Training
	ISBSG (10<Functional Size<500)	17	Testing
Datset_5	ISBSG (Functional Size>500)	14	Training
	ISBSG (Functional Size>500)	7	Testing
Dataset_6	Desharnais	14	Validation1
Dataset_7	Combined	33	Validation2

Table 24. Basic statistics about datasets (COSMIC FFP).
Tabela 24. Osnovni statistički podaci o korišćenim skupovima podataka (COSMIC FFP).

Datasets	N	Min [PM]	Max [PM]	Mean [PM]	Std. deviation [PM]
Dataset_1	52	2.0	9.0	5.404	2.4031
Dataset_2	62	10.0	48.0	24.823	11.3203
Dataset_3	43	50.0	99.0	77.116	15.1441
Dataset_4	77	104.0	492.0	234.130	113.8159
Dataset_5	21	561.0	2090.0	1016.048	458.4442
Dataset_6	14	140.0	3860.0	1011.429	920.4251
Dataset_7	33	493.0	2589.0	1193.424	419.4201

3.2.2 The methodology used within the improved COSMIC FFP model

The appropriate methodology was selected for the experimental part in the COSMIC FFP approach, i.e. it is based on several trial experiments. The order of the steps in the experiment was constructed based on a robust design algorithm and shown in the Figure 19.

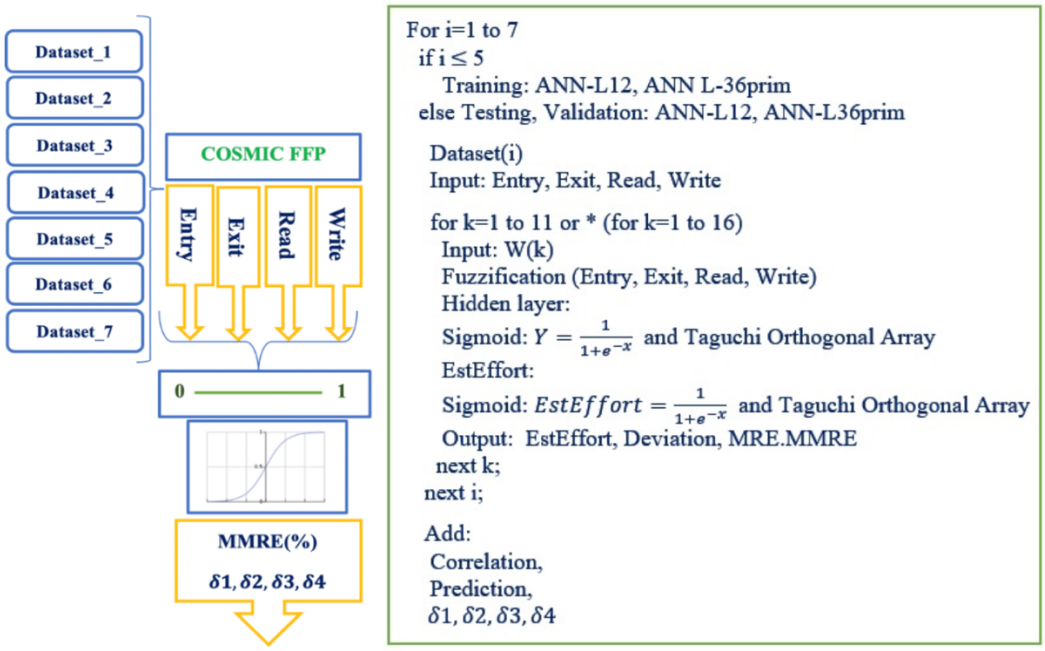


Figure 19. Robust design algorithm for performing the experiment (COSMIC FFP).
Slika 19. Algoritam robusnog dizajna za izvođenje eksperimenta (COSMIC FFP).

Step 1: Input values are the data about particular project from the selected ISBSG dataset and are represented by four parameters (Entry, Exit, Read, Write), which describe the functional size.

Step 2: All input values are transformed according to the following formula:

The function $\mu_D(X)$: $R \rightarrow [0, 1]$, translates the real values of input signals into coded values from the interval [0, 1] , in the following way: $\mu_D(X_i) = (X_i - X_{min})/(X_{max} - X_{min})$ (min-max normalization) [121], where D is the set of data on which the experiment is performed, Xi is the input value, X_{min} is the smallest input value, and X_{max} the greatest input value on the observed dataset.

Step 3: The sigmoid function, as the activation function of the hidden layer was used (31):

$$y_i = \frac{1}{1+e^{-x_i}}, i = \overline{1,n} \tag{31}$$

The construction of the activation function is based on a combination of input values and corresponding weight coefficients W_i for each of the proposed ANN architectures.

a) Hidden and output layer functions for ANN-L12 architecture, see Figure 17, Table 21 (75)-(77):

$$Y1 = \frac{1}{1+e^{-(X1\cdot W1+X2\cdot W3+X3\cdot W5+X4\cdot W7)}} \tag{75}$$

$$Y2 = \frac{1}{1+e^{-(X1\cdot W2+X2\cdot W4+X3\cdot W6+X4\cdot W8)}} \tag{76}$$

$$EstEffortANN-L12 = \frac{1}{1+e^{-(Y1\cdot W9+Y2\cdot W10+1\cdot W11)}} \tag{77}$$

where Y_1 and Y_2 are the hidden layer functions and *EstEffortANN-L12* represents output function.

b) Hidden and output layer functions for ANN-L36prim architecture, see Figure 18, Table 22 (78)-(81):

$$Y1 = \frac{1}{1+e^{-(X1\cdot W1+X2\cdot W4+X3\cdot W7+X4\cdot W10)}} \tag{78}$$

$$Y2 = \frac{1}{1+e^{-(X1\cdot W2+X2\cdot W5+X3\cdot W8+X4\cdot W11)}} \tag{79}$$

$$Y3 = \frac{1}{1+e^{-(X1\cdot W3+X2\cdot W6+X3\cdot W9+X4\cdot W12)}} \tag{80}$$

$$EstEffortANN-L36prim = \frac{1}{1+e^{-(Y1\cdot W13+Y2\cdot W14+Y3\cdot W15+1\cdot W16)}} \tag{81}$$

where Y_1, Y_2, and Y_3 are the hidden layer functions and *EstEffortANN-L36prim* represents output function.

In the first proposed ANN-L12 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 L1, L2, and L3, and the initial values of the weighting factors W_i that takes 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-L12 architecture) [7], [119] (82):

$$W_1{\cdot}L1 = cost1 + cost2 + \ldots + cost6$$
$$W_1{\cdot}L2 = cost7 + cost8 + \ldots + cost12$$
$$\ldots.$$
$$W_{12}{\cdot}L1 = cost1 + cost5 + \ldots + cost12$$
$$W_{12}{\cdot}L2 = cost2 + cost3 + \ldots + cost11$$

$$\text{where } cost(i) = \Sigma\, MRE(ANN(i)) \tag{82}$$

For each subsequent iteration, the interval [-1, 1] is divided depending on the cost effect function as follows (83) [7], [119]:

$$W_1{\cdot}L1new = W_1{\cdot}L1old$$
$$W_1{\cdot}L2new = W_1{\cdot}L2old + (W_1{\cdot}L2old - W_1{\cdot}L1old)/2 \tag{83}$$

where $W_1{\cdot}L1old$ and $W_1{\cdot}L2old$ 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: Method of defuzzification was used according to the following formula (84), (85) [122]:

$$Xi = (X_{min} + \mu D(X_i)) \cdot (X_{max} - X_{min}) \tag{84}$$
$$\text{OA(ANN}\boldsymbol{i}\text{)} = X\boldsymbol{i}\text{ , where } \boldsymbol{i} = 12, \boldsymbol{i} = 36. \tag{85}$$

where OA represents actual effort of the particular project, that is calculated based on ANN-L12 and ANN-L36prim.

Step 5: For each iteration in our experiment, the output values are obtained according to the following formulas/measures [123] (65)-(70):

$$Deviation = |ActEffort - EstEffort| \tag{65}$$
$$MAE_i = \frac{1}{n}\sum_{i=1}^{n} |ActEffort - EstEffort| \tag{66}$$
$$MRE = Deviation/ActEffort \tag{67}$$
$$MRE = \frac{1}{n} \cdot \sum_{i=1}^{n} MRE_i \tag{68}$$
$$\text{MMRE} = mean\,(\text{MRE}) \tag{69}$$

For each experimental part in every iteration, the Gradient Descent with the condition GA<0.01 is monitored and calculated as [114], [115] (70):

$$GA = MRE_{i1} - MRE_{i2} < 0.01,$$
$$where\ i = 1, \dots, n \quad n\ is\ a\ number\ of\ ANN\ a\ Winner\ candidates. \tag{70}$$

Step 6: Examination of the influence of input values on the change of MMRE value (86)-(89):

1. The influence of the first input parameter (Entry) and its value on the change of MMRE value is calculated as:

$$\delta 1 = mean(\text{MMRE}) - mean(\text{MMRE}_1)$$

where MMRE_1 is *mean(*MMRE) when X1=0; (86)

2. The influence of the second input parameter (Exit) and its value on the change of MMRE value is calculated as:

$$\delta 2 = mean(\text{MMRE}) - mean(\text{MMRE}_2)$$

where MMRE_2 is mean(MMRE) when X2=0; (87)

3. The influence of the third input parameter (Read) and its value on the change of MMRE value is calculated as:

$$\delta 3 = mean(\text{MMRE}) - mean(\text{MMRE}_3)$$

where MMRE_3 is *mean(*MMRE) when X3=0; (88)

4. The influence of the fourth input parameter (Write) and its value on the change of MMRE value is calculated as:

$$\delta 4 = mean(\text{MMRE}) - mean(\text{MMRE}_4)$$

where MMRE_4 is *mean*(MMRE) when X4=0; (89)

Step 7: Correlation, Prediction

Pearson's [124], Spearman's [125] and R^2 [126] coefficients are monitored during the experiment (73).

$$Correl(X,Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \quad (73)$$

Additionally, Prediction at 25%, 30%, and 50% is the percentage of the total number of ANNs that meet the GA criterion (74) [127].

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

PRED(k) = count(MRE) < 25%
PRED(k) = count(MRE) < 30%
PRED(k) = count(MRE) < 50% , where k = 25, k = 30, and k = 50. (74)

The second and third parts of the experiment are executed in the same way as the first part, with different projects and datasets being used. The second part uses the ISBSG dataset, but with projects that were not used in the first part. In the third part, the Desharnais dataset and combined dataset are used.

3.3 New, improved UCP model

As part of the UCP approach, an improved UCP modelproposed in the dissertation. For the improved UCP model, the following architectures and corresponding orthogonal vector plans L16 and L36prim are used:

1. UCP and ANN-L16

The first proposed architecture is ANN-L16. It consists of six input values, one hidden layer with two nodes, one output, and the total number of fifteen weighting factors (Wi, i = $\overline{1, 15}$) and their initial values are from the interval [-1, 1]. The Taguchi Orthogonal Array used in the construction of this proposed architecture contains two levels L1 and L2, see Figure 20, Table 25 [7], [8], [114]. Bias represents an additional weighting factor to complete the selected orthogonal plan and has a value of 1.

Table 25. Taguchi orthogonal vector plan (L16=2^{15}).
Tabela 25. Taguči ortogonalni vektorski plan (L16=2^{15}).

ANN-L16	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}
ANN1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
ANN2	L1	L1	L1	L1	L1	L1	L1	L2	L2	L2	L2	L2	L2	L2	L2
ANN3	L1	L1	L1	L2	L2	L2	L2	L1	L1	L1	L1	L2	L2	L2	L2
ANN4	L1	L1	L1	L1	L2	L2	L2	L2	L2	L2	L2	L1	L1	L1	L1
ANN5	L1	L2	L2	L1	L1	L2	L2	L1	L1	L2	L2	L1	L1	L2	L2
ANN6	L1	L2	L2	L1	L1	L2	L2	L2	L2	L1	L1	L2	L2	L1	L1
ANN7	L1	L2	L2	L2	L2	L1	L1	L1	L1	L2	L2	L2	L2	L1	L1
ANN8	L1	L2	L2	L2	L2	L1	L1	L2	L2	L1	L1	L1	L1	L2	L2
ANN9	L2	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2	L1	L2
ANN10	L2	L1	L2	L1	L2	L1	L2	L2	L1	L2	L1	L2	L1	L2	L1
ANN11	L2	L1	L2	L2	L1	L2	L1	L1	L2	L1	L2	L2	L1	L2	L1
ANN12	L2	L1	L2	L2	L1	L2	L1	L2	L1	L2	L1	L1	L2	L1	L2
ANN13	L2	L2	L1	L1	L2	L2	L1	L1	L2	L2	L1	L1	L2	L2	L1
ANN14	L2	L2	L1	L1	L2	L2	L1	L2	L1	L1	L2	L2	L1	L1	L2
ANN15	L2	L2	L1	L2	L1	L1	L2	L1	L2	L2	L1	L2	L1	L1	L2
ANN16	L2	L2	L1	L2	L1	L1	L2	L2	L1	L1	L2	L1	L2	L2	L1

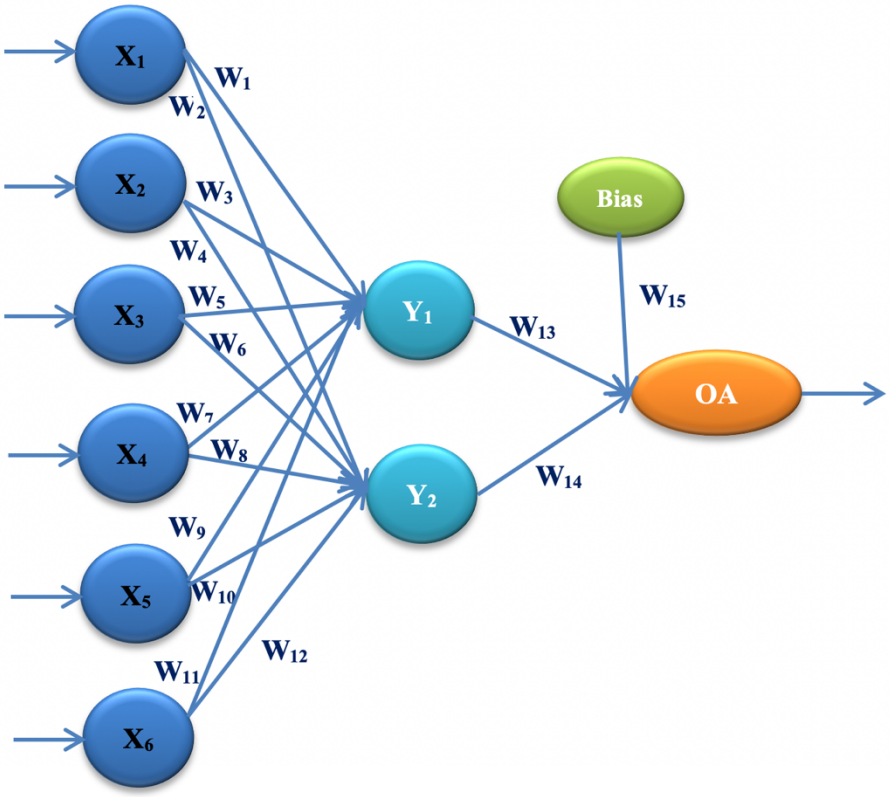


Figure 20. ANN architecture with one hidden layer (ANN-L16).
Slika 20. ANN arhitektura sa jednim skrivenim slojem (ANN-L16).

2. UCP and ANN-L36prim

The second proposed architecture is ANN-L36prim. It consists of four input values, one hidden layer with three nodes, one output, and the total number of sixteen weighting factors (Wi, i = $\overline{1,16}$), their initial values are from the interval [-1, 0, 1]. The Taguchi Orthogonal Array used in the construction of this proposed architecture is combined, where the first eleven parameters and the last sixteenth parameter are with three levels L1, L2, and L3, while the remaining four parameters are with two levels L1 and L2, see Figure 21, Table 26 [7], [8], [114]. Bias represents an additional weighting factor to complete the selected orthogonal plan and has a value of 1.

Table 26. Taguchi orthogonal vector plan (L36prim=$3^{11}2^{4}3^{1}$).
Tabela 26. Taguči ortogonalni vektorski plan (L36prim=$3^{11}2^{4}3^{1}$).

ANN-L36 prim	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	W_9	W_{10}	W_{11}	W_{12}	W_{13}	W_{14}	W_{15}	W_{16}
ANN1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1	L1
ANN2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L2	L1	L1	L1	L1
ANN3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L3	L1	L1	L1	L1
ANN4	L1	L1	L1	L1	L2	L2	L2	L2	L3	L3	L3	L3	L1	L2	L2	L1
ANN5	L1	L1	L1	L1	L3	L3	L3	L3	L2	L2	L2	L2	L1	L2	L2	L1
ANN6	L3	L3	L3	L3	L1	L1	L1	L1	L2	L2	L2	L2	L1	L2	L2	L1
ANN7	L1	L1	L2	L3	L1	L2	L3	L3	L1	L1	L1	L3	L2	L1	L2	L1
ANN8	L2	L2	L3	L1	L2	L3	L1	L1	L2	L3	L3	L1	L2	L1	L2	L1
ANN9	L3	L3	L1	L2	L3	L1	L2	L2	L3	L1	L1	L2	L2	L1	L2	L1
ANN10	L1	L1	L3	L2	L1	L3	L2	L3	L2	L1	L3	L2	L2	L2	L1	L1
ANN11	L2	L2	L1	L3	L2	L1	L3	L1	L3	L2	L1	L3	L2	L2	L1	L1
ANN12	L3	L3	L2	L1	L3	L2	L1	L2	L1	L3	L2	L1	L2	L2	L1	L1
ANN13	L1	L2	L3	L1	L3	L2	L1	L3	L3	L2	L1	L2	L1	L1	L1	L2
ANN14	L2	L3	L1	L2	L1	L3	L2	L1	L1	L3	L2	L3	L1	L1	L1	L2
ANN15	L3	L1	L2	L3	L2	L1	L3	L2	L2	L1	L3	L1	L1	L1	L1	L2
ANN16	L1	L2	L3	L2	L1	L1	L3	L2	L3	L3	L2	L1	L1	L2	L2	L2
ANN17	L2	L3	L1	L3	L2	L2	L1	L3	L1	L1	L3	L2	L1	L2	L2	L2
ANN18	L3	L1	L2	L1	L3	L3	L2	L1	L2	L2	L1	L3	L1	L2	L2	L2
ANN19	L1	L2	L1	L3	L3	L3	L1	L2	L2	L1	L2	L3	L2	L1	L2	L2
ANN20	L2	L3	L2	L1	L1	L1	L2	L3	L3	L2	L3	L1	L2	L1	L2	L2
ANN21	L3	L1	L3	L2	L2	L2	L3	L1	L1	L3	L1	L2	L2	L1	L2	L2
ANN22	L1	L2	L2	L3	L3	L1	L2	L1	L1	L3	L3	L2	L2	L2	L1	L2
ANN23	L2	L3	L3	L1	L1	L2	L3	L2	L2	L1	L1	L3	L2	L2	L1	L2
ANN24	L3	L1	L1	L2	L2	L3	L1	L3	L3	L2	L2	L1	L2	L2	L1	L2
ANN25	L1	L3	L2	L1	L2	L3	L3	L1	L3	L1	L2	L2	L1	L1	L1	L3

ANN26	L2	L1	L3	L2	L3	L1	L1	L2	L1	L2	L3	L3	L1	L1	L1	L3
ANN27	L3	L2	L1	L3	L1	L2	L2	L3	L2	L3	L1	L1	L1	L1	L1	L3
ANN28	L1	L3	L2	L2	L2	L1	L1	L3	L2	L3	L1	L3	L1	L2	L2	L3
ANN29	L2	L1	L3	L3	L3	L2	L2	L1	L3	L1	L2	L1	L1	L2	L2	L3
ANN30	L3	L2	L1	L1	L1	L3	L3	L2	L1	L2	L3	L2	L1	L2	L2	L3
ANN31	L1	L3	L3	L3	L2	L3	L2	L2	L1	L2	L1	L1	L2	L1	L2	L3
ANN32	L2	L1	L1	L1	L3	L1	L3	L3	L3	L3	L2	L2	L2	L1	L2	L3
ANN33	L3	L2	L2	L2	L1	L2	L1	L1	L3	L1	L3	L3	L2	L1	L2	L3
ANN34	L1	L3	L1	L2	L3	L2	L3	L1	L2	L2	L3	L1	L2	L2	L1	L3
ANN35	L2	L1	L2	L3	L1	L3	L1	L2	L3	L3	L1	L2	L2	L2	L1	L3
ANN36	L3	L2	L3	L1	L2	L1	L2	L3	L1	L1	L2	L3	L2	L2	L1	L3

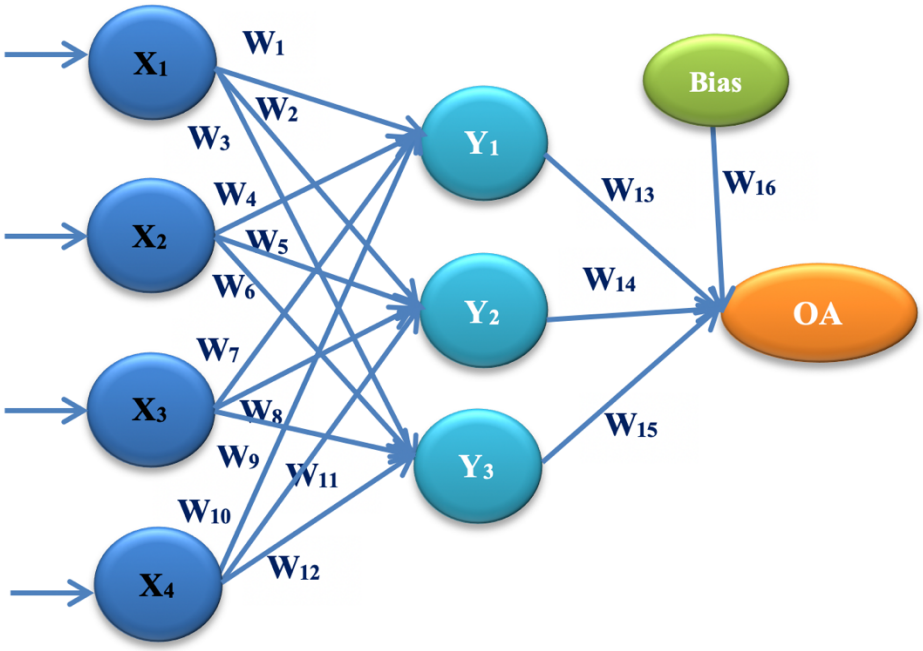


Figure 21. ANN architecture with one hidden layer (ANN-L36prim).
Slika 21. ANN arhitektura sa jednim skrivenim slojem (ANN-L36prim).

The experiment presented in this paper consists of three parts:

1. Training of two different ANN architectures constructed according to the corresponding Taguchi orthogonal vector plans (ANN-L16 and ANN36prim);
2. Testing on ANN "Winner", which gave the best results (the lowest MMRE value) in the first part of the experiment, for two proposed architectures on the same dataset;
3. Validation on ANN "Winner" that gave the best results (the lowest MMRE value) in the first part of the experiment, for each selected architecture, but using some other i.e. different datasets.

3.3.1 Data sets used in the UCP approach

For the first and second part of the experiment, the Use Case Point Benchmark Dataset by Radek Silhavy (UCP Benchmark Dataset) [130] was used. In the third part, different data sets were used, i.e. combined datasets composed of projects of different industrial companies were used. The results in Table 27. indicate a more homogeneous structure of the projects used in all three parts of the experiment. It is concluded based on the standard deviation results presented in Table 28.

Table 27. Information on used datasets (UCP).
Tabela 27. Informacije o korišćenim skupovima podataka (UCP).

	Dataset	Number of projects	Experiment
Dataset_1	UCP Benchmark Dataset	50	Training
Dataset_2	UCP Benchmark Dataset	21	Testing
Dataset_3	Combined	18	Validation1
Dataset_4	Combined Industrial projects	17	Validation2

Table 28. Basic statistics about dataset (UCP).
Tabela 28. Osnovni statistički podaci o korišćenim skupovima podataka (UCP).

Datasets	N	Min [PM]	Max [PM]	Mean [PM]	Std. deviation [PM]
Dataset_1	50	5775.0	7970.0	6506.940	653.0308
Dataset_2	21	6162.6	6525.3	6393.993	118.1858
Dataset_3	18	2692.1	3246.6	2988.392	233.2270
Dataset_4	17	2176.0	3216.0	2589.400	352.0859

3.3.2 The methodology used within the improved UCP model

The appropriate methodology was selected for the experimental part in the UCP approach based on several trial experiments. The order of the steps in the experiment was constructed based on a robust design algorithm and it is shown in Figure 22.

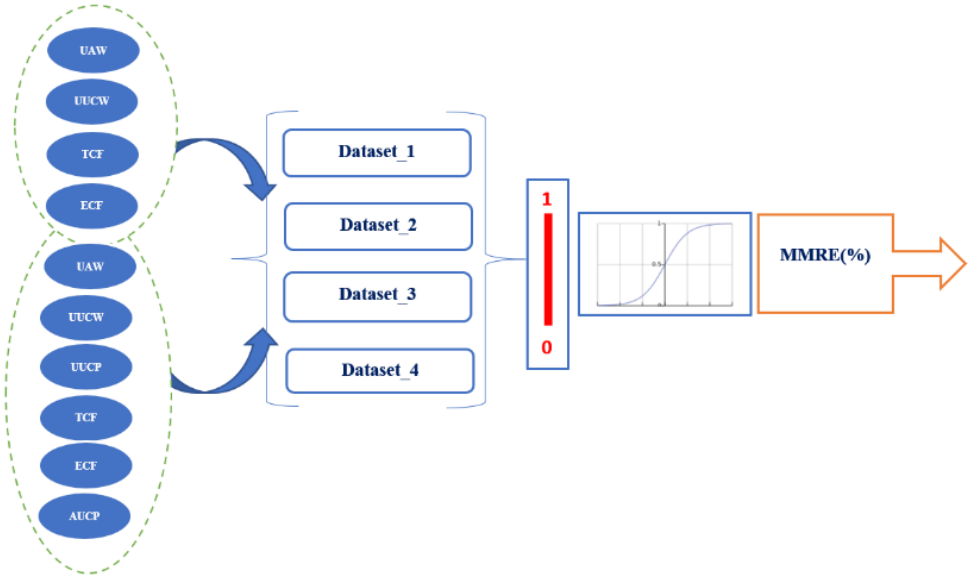


Figure 22. Robust design algorithm for performing the experiment (UCP).
Slika 22. Algoritam robusnog dizajna za izvođenje eksperimenta (UCP).

Step 1: Input layer

The input values of the first proposed architecture ANN-L16 are six input values, four of which are independent: UAW, UUCW, TCF, and ECF and two dependent: UUCP i AUCP.

The input values of the second proposed architecture ANN-L36prim are four independent input values: UAW, UUCW, TCF, and ECF.

Step 2: All input values are transformed according to the following formula:

The function $\mu_D(X)$: $R \rightarrow [0, 1]$, translates the real values of input signals into coded values from the interval [0, 1] , in the following way: $\mu_D(X_i) = (X_i - X_{min})/(X_{max} - X_{min})$ (min-max normalization) [121], where D is the set of data on which the experiment is performed, Xi is the input value, X_{min} is the smallest input value, and X_{max} the greatest input value on the observed dataset.

Step 3: The sigmoid function, as the activation function of the hidden layer was used (31):

$$y_i = \frac{1}{1+e^{-x_i}}, i = \overline{1,n} \tag{31}$$

The construction of the activation function is based on a combination of input values and corresponding weight coefficients W_i for each of the proposed ANN architectures.

a) Hidden and output layer functions for ANN-L16 architecture, see Figure 20, Table 25 (90)-(92):

$$Y_1 = 1/\left(1 + e^{-(X_1 \cdot W_1 + X_2 \cdot W_3 + X_3 \cdot W_5 + X_4 \cdot W_7 + X_5 \cdot W_9 + X_6 \cdot W_{11})}\right) \quad (90)$$

$$Y_2 = 1/\left(1 + e^{-(X_1 \cdot W_2 + X_2 \cdot W_4 + X_3 \cdot W_6 + X_4 \cdot W_8 + X_5 \cdot W_{10} + X_6 \cdot W_{12})}\right) \quad (91)$$

$$EstEffANN - L16 = 1/\left(1 + e^{-(Y_1 \cdot W_{13} + Y_2 \cdot W_{14} + 1 \cdot W_{15})}\right) \quad (92)$$

where Y_1, Y_2, and Y_3 are the hidden layer functions and *EstEffortANN-L16* represents output function.

b) Hidden and output layer functions for ANN-L36prim architecture, see Figure 21, Table 26 (78)-(81):

$$Y1 = \frac{1}{1+e^{-(X1 \cdot W1 + X2 \cdot W4 + X3 \cdot W7 + X4 \cdot W10)}} \quad (78)$$

$$Y2 = \frac{1}{1+e^{-(X1 \cdot W2 + X2 \cdot W5 + X3 \cdot W8 + X4 \cdot W11)}} \quad (79)$$

$$Y3 = \frac{1}{1+e^{-(X1 \cdot W3 + X2 \cdot W6 + X3 \cdot W9 + X4 \cdot W12)}} \quad (80)$$

$$EstEffortANN - L36prim = \frac{1}{1+e^{-(Y1 \cdot W13 + Y2 \cdot W14 + Y3 \cdot W15 + 1 \cdot W16)}} \quad (81)$$

where Y_1, Y_2, and Y_3 are the hidden layer functions and *EstEffortANN-L36prim* represents output function.

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 L1, L2, and L3, 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) [7], [119] (93):

$W_1L1 = cost1 + cost2 + \ldots + cost8$

$W_1L2 = cost9 + cost10 + \ldots + cost16$

....

$W_{15}L1 = cost1 + cost6 + \ldots + cost16$

$W_{15}L2 = cost2 + cost3 + \ldots + cost15$

where $cost(i) = \Sigma\, MRE(ANN(i))$ (93)

For each subsequent iteration, the interval [-1, 1] is divided depending on the cost effect function as follows [7], [119] (94):

$$W_1L1new = W_1L1old$$
$$W_1L2new = W_1L2old + (W_1L3old - W_1L2old)/2$$
$$W_1L3new = W_1L3old \quad (94)$$

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: Method of defuzzification was used according to the following formula (84), (85) [122]:

$$Xi = (X_{min} + \mu D(X_i)) \cdot (X_{max} - X_{min}) \quad (84)$$
$$OA(ANN\boldsymbol{i}) = X\boldsymbol{i}\text{ , where }\boldsymbol{i} = 16, \boldsymbol{i} = 36. \quad (85)$$

where OA represents actual effort of the particular project, that is calculated based on ANN-L12 and ANN-L36prim.

Step 5: For each iteration in our experiment, the output values are obtained according to the following formulas/measures [123] (65)-(70):

$$Deviation = |ActEffort - EstEffort| \quad (65)$$
$$MAE_i = \frac{1}{n}\sum_{i=1}^{n} |ActEffort - EstEffort| \quad (66)$$
$$MRE = Deviation/ActEffort \quad (67)$$
$$MRE = \frac{1}{n} \cdot \sum_{i=1}^{n} MRE_i \quad (68)$$
$$MMRE = mean\,(MRE) \quad (69)$$

For each of the experimental part in every iteration, the Gradient Descent is monitored with the condition GA<0.01, calculated as [114], [115] (70):

$$GA = MRE_{i1} - MRE_{i2} < 0.01,\ \ where\ i = 1, \dots, n \quad n\ is\ a\ number\ of\ ANN. \quad (70)$$

Step 6: Influence of dependent variables UUCP and AUCP on the change of MMRE value.

1. The influence of the input parameter UUCP and its value is calculated as (95):

$$\delta 1 = mean(MMRE) - mean(MMRE_1) \quad (95)$$

where $MMRE_1$ is *mean*(MMRE) when UUCP=0;

2. The influence of the input parameter AUCP and its value is calculated as (96):

$\delta 2 = mean(\text{MMRE}) - mean(\text{MMRE}_2)$ when AUCP=0; (96)

Step 7: Correlation, Prediction

Pearson's [124], Spearman's [125] and R^2 [126] coefficients are monitored during the experiment (73).

$$Correl(X,Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \quad (73)$$

Additionally, Prediction at 25%, 30%, and 50% is the percentage of the total number of ANNs that meet the GA criterion (74) [127].

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

PRED(k) = count(MRE) < 25%
PRED(k) = count(MRE) < 30%
PRED(k) = count(MRE) < 50% , where k = 25, k = 30, and k = 50. (74)

The second and third part are executed in the same way as the first part, with different projects and datasets being used. The second part uses the also UCP Benchmark (Mendeley) dataset, but with projects that were not used in the first part. In the third part, the combined industrial datasets were used.

Chapter 4: Analysis of the obtained results by applying three new, improved models

4.1 Obtained results using COCOMO2000 model and ANN

Within the first proposed, improved COCOMO2000 approach, four different ANN architectures based on Taguchi's orthogonal vector plans were used, designated as ANN-L9, ANN-L18, ANN-L27, and ANN-L36. In the first part of the experiment - training these ANN architectures, the used COCOMO2000 data set was divided into three clusters, according to the value of actual effort: small, medium, and large.

- Small cluster (Actual Effort <50PM);
- Medium cluster (50PM<Actual Effort<500PM);
- Large cluster (Actual Effort>500PM).

Table 29. shows the results obtained by training the first proposed ANN-L9 architecture on a small cluster. In addition to examining the MRE values for each of the nine ANN architecture candidates, the GA criterion was also monitored. Based on all MRE values in each performed iteration, the "Winner" network was determined, i.e., the ANN network with the lowest MRE value (the ANN with the best performances). Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN9) is 80.7%, and the value of MMRE is 81.1%. The required GA criterion was met after seven iterations.

Table 29. ANN-L9 results of training part on small cluster.
Tabela 29. Rezultati treniranja ANN-L9 nad malim klasterom.

No.of Iter.	**1.**		**2.**		**3.**		**4.**		**5.**		**6.**		**7.**	
ANN-L9	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	0.903	0.903	0.853	0.049	0.827	0.026	0.816	0.011	0.812	0.005	0.809	0.002	0.808	0.001
ANN2	1.038	1.038	0.932	0.107	0.871	0.061	0.839	0.032	0.823	0.016	0.815	0.008	0.811	0.004
ANN3	1.278	1.278	1.038	0.240	0.918	0.121	0.862	0.055	0.835	0.028	0.821	0.014	0.814	0.007
ANN4	1.258	1.258	1.038	0.219	0.919	0.120	0.863	0.056	0.835	0.028	0.821	0.014	0.814	0.007
ANN5	0.857	0.857	0.829	0.028	0.817	0.012	0.812	0.005	0.810	0.002	0.808	0.001	0.808	0.001
ANN6	1.038	1.038	0.917	0.121	0.862	0.055	0.834	0.028	0.821	0.014	0.814	0.007	0.811	0.003
ANN7	1.038	1.038	0.922	0.117	0.864	0.057	0.836	0.029	0.821	0.014	0.814	0.007	0.811	0.004
ANN8	1.316	1.316	1.038	0.277	0.913	0.126	0.859	0.053	0.833	0.026	0.820	0.013	0.814	0.006
ANN9	0.809	0.809	0.808	0.001	0.808	0.001	0.807	0.000	0.807	0.000	0.807	0.000	0.807	0.000
GA		**9**		**8**		**8**		**7**		**6**		**3**		**0**
Winner	**80.9%**		**80.8%**		**80.8%**		**80.7%**		**80.7%**		**80.7%**		**80.7%**	
MMRE	**106%**		**93.1%**		**86.6%**		**83.7%**		**82.2%**		**81.4%**		**81.1%**	

Table 30. shows the results obtained by training the first proposed ANN-L9 architecture on the medium cluster. In addition to examining the MRE values for each of the nine ANN architecture candidates, the GA criterion was also monitored. Based on all MRE values in each performed iteration, the "Winner" network was determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN9) is 49.4%, and the value of MMRE is 49.8%. The required GA criterion was met after six iterations.

Table 30. ANN-L9 results of training part on medium cluster.
Tabela 30. Rezultati treniranja ANN-L9 nad srednjim klasterom.

No. of Iter.	**1.**		**2.**		**3.**		**4.**		**5.**		**6.**	
ANN-L9	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**
ANN1	0.544	0.544	0.515	0.029	0.504	0.011	0.499	0.005	0.497	0.002	0.495	0.001
ANN2	0.612	0.612	0.558	0.054	0.524	0.034	0.508	0.016	0.501	0.007	0.498	0.003
ANN3	0.729	0.729	0.612	0.117	0.550	0.062	0.519	0.031	0.506	0.013	0.505	0.001
ANN4	0.718	0.718	0.612	0.106	0.551	0.061	0.519	0.031	0.506	0.013	0.500	0.006
ANN5	0.514	0.514	0.503	0.010	0.499	0.005	0.496	0.002	0.495	0.001	0.495	0.001
ANN6	0.612	0.612	0.551	0.061	0.519	0.032	0.506	0.013	0.500	0.006	0.497	0.003
ANN7	0.612	0.612	0.551	0.061	0.520	0.031	0.507	0.013	0.500	0.006	0.497	0.003
ANN8	0.751	0.751	0.612	0.139	0.548	0.064	0.517	0.030	0.505	0.012	0.500	0.006
ANN9	0.495	0.495	0.495	0.000	0.495	0.000	0.494	0.000	0.494	0.000	0.494	0.000
GA		**9**		**8**		**7**		**6**		**3**		**0**
Winner	**49.5%**		**49.5%**		**49.5%**		**49.4%**		**49.4%**		**49.4%**	
MMRE	**62.1%**		**55.7%**		**52.3%**		**50.7%**		**50.1%**		**49.8%**	

Table 31. shows the results obtained by training the first proposed ANN-L9 architecture on a large cluster. In addition to examining the MRE values for each of the nine ANN architecture candidates, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN9) is 206.9%, and the value of MMRE is 207.6%. The required GA criterion was met after eight iterations.

Table 31. ANN-L9 results of training part on large cluster.
Tabela 31. Rezultati treniranja ANN-L9 nad velikim klasterom.

No. of Iter.	1.		2.		3.		4.		5.		6.		7.		8.	
ANN-L9	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	2.683	2.683	2.204	0.163	2.132	0.072	2.100	0.033	2.084	0.016	2.076	0.008	2.073	0.004	2.071	0.002
ANN2	3.224	3.224	2.426	0.328	2.248	0.178	2.158	0.090	2.113	0.045	2.091	0.022	2.080	0.011	2.074	0.006
ANN3	4.031	4.031	2.670	0.554	2.371	0.299	2.219	0.152	2.143	0.076	2.106	0.037	2.104	0.002	2.098	0.006
ANN4	4.028	4.028	2.670	0.554	2.371	0.299	2.219	0.152	2.143	0.076	2.106	0.037	2.087	0.018	2.078	0.009
ANN5	2.311	2.311	2.107	0.072	2.094	0.013	2.081	0.013	2.075	0.006	2.072	0.003	2.070	0.002	2.070	0.001
ANN6	3.224	3.224	2.390	0.294	2.230	0.161	2.149	0.081	2.109	0.040	2.089	0.020	2.079	0.010	2.074	0.005
ANN7	3.224	3.224	2.393	0.263	2.213	0.180	2.140	0.073	2.104	0.036	2.087	0.018	2.078	0.009	2.073	0.004
ANN8	4.248	4.248	2.646	0.578	2.355	0.292	2.210	0.145	2.138	0.072	2.104	0.035	2.086	0.017	2.078	0.009
ANN9	2.116	2.116	2.080	0.011	2.074	0.006	2.072	0.003	2.070	0.001	2.070	0.001	2.069	0.000	2.069	0.000
GA		**9**		**9**		**8**		**8**		**7**		**6**		**3**		**0**
Winner	**211.6%**		**208.0%**		**207.4%**		**207.2%**		**207.0%**		**207.0%**		**206.9%**		**206.9%**	
MMRE	**323.2%**		**239.9%**		**223.2%**		**215.0%**		**210.9%**		**208.9%**		**208.1%**		**207.6%**	

Figure 23. presents the MMRE results for all three clusters on which the ANN-L9 training procedure was performed using the COCOMO2000 dataset. It can be concluded that the value of MMRE is the lowest on a small cluster (49.8%) and the highest on a large cluster (207.6%). Figure 24. presents the MMRE results for the obtained "Winner" network on all three clusters. The ANN-L9 training procedure was performed using the COCOMO2000 dataset. It can be concluded that the value of MMRE "Winner" is the lowest on a small cluster after seven iterations (49.4%) and the highest on a large cluster after the eight iterations (206.9%).

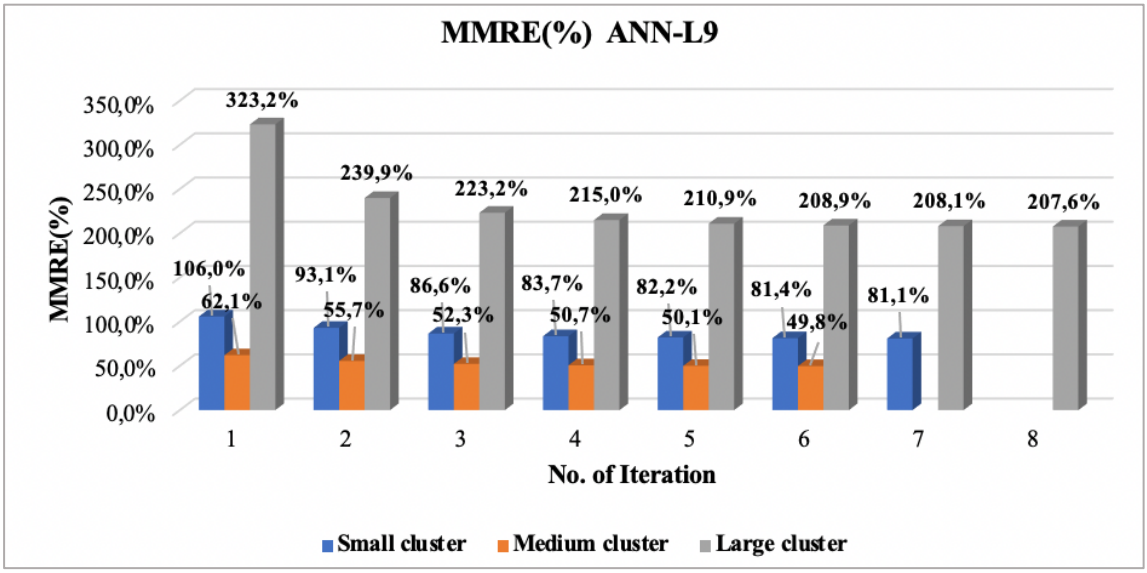


Figure 23. MMRE results in three parts of the experiment for all clusters (ANN-L9).
Slika 23. Rezultati vrednosti MMRE u sva tri dela eksperimenta nad svim klasterima (ANN-L9).

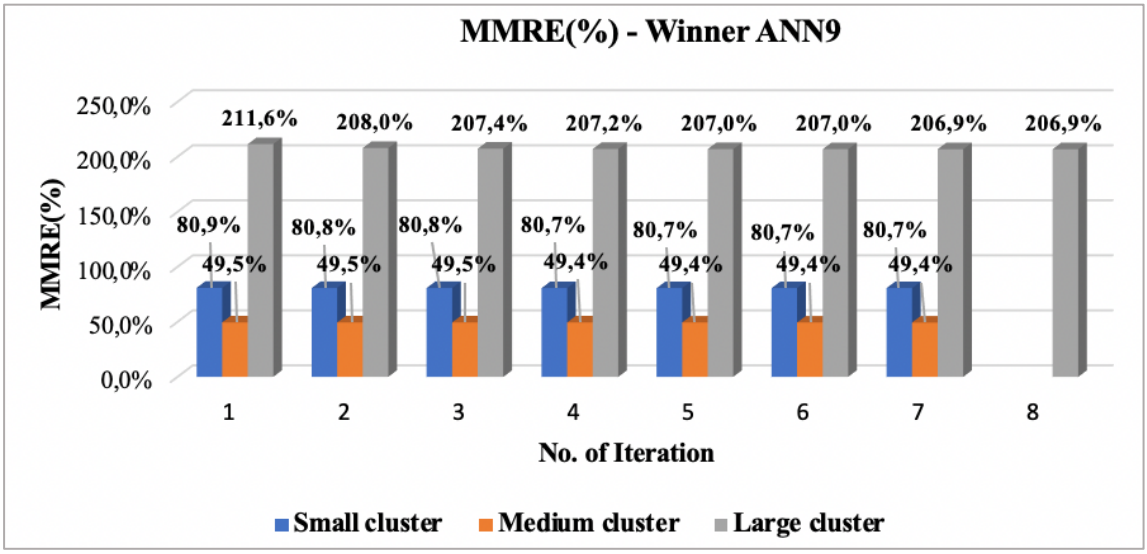


Figure 24. MMRE for "Winner" ANN9 (ANN-L9).
Slika 24. MMRE vrednosti za "Winner" ANN9 (ANN-L9).

Table 32. shows the results obtained by training the second proposed ANN-L18 architecture on a small cluster. In addition to examining the MRE values for each of the 18 ANNs, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The

obtained value of the "Winner" network (ANN5) is 63.3%, and the value of MMRE is 63.7%. The required GA criterion was met after six iterations.

Table 32. ANN-L18 results of training part on small cluster.
Tabela 32. Rezultati treniranja ANN-L18 nad malim klasterom.

No. of Iter.	1.		2.		3.		4.		5.		6.	
ANN-L18	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	0.693	0.693	0.660	0.033	0.643	0.017	0.642	0.001	0.635	0.008	0.634	0.001
ANN2	0.906	0.906	0.742	0.164	0.677	0.065	0.652	0.025	0.642	0.011	0.637	0.004
ANN3	1.400	1.400	0.906	0.494	0.730	0.176	0.672	0.058	0.649	0.023	0.641	0.008
ANN4	1.101	1.101	0.906	0.195	0.737	0.169	0.679	0.058	0.650	0.028	0.641	0.009
ANN5	0.670	0.670	0.648	0.022	0.639	0.009	0.635	0.004	0.634	0.001	0.633	0.001
ANN6	0.906	0.906	0.730	0.176	0.670	0.060	0.648	0.022	0.640	0.008	0.636	0.004
ANN7	1.146	1.146	0.823	0.324	0.703	0.119	0.665	0.038	0.642	0.024	0.639	0.003
ANN8	0.857	0.857	0.722	0.135	0.670	0.052	0.648	0.021	0.640	0.008	0.636	0.004
ANN9	0.744	0.744	0.681	0.063	0.654	0.027	0.644	0.010	0.637	0.006	0.635	0.002
ANN10	0.742	0.742	0.680	0.063	0.653	0.027	0.645	0.008	0.641	0.004	0.635	0.006
ANN11	1.119	1.119	0.819	0.300	0.701	0.118	0.665	0.036	0.645	0.020	0.639	0.006
ANN12	0.867	0.867	0.732	0.135	0.674	0.058	0.651	0.023	0.641	0.010	0.637	0.004
ANN13	1.106	1.106	0.809	0.297	0.698	0.110	0.664	0.035	0.645	0.018	0.639	0.006
ANN14	0.934	0.934	0.739	0.195	0.674	0.065	0.650	0.025	0.641	0.009	0.637	0.004
ANN15	0.768	0.768	0.682	0.085	0.652	0.030	0.643	0.009	0.637	0.006	0.635	0.003
ANN16	0.766	0.766	0.685	0.081	0.655	0.030	0.645	0.009	0.638	0.008	0.635	0.002
ANN17	1.113	1.113	0.814	0.299	0.700	0.114	0.663	0.036	0.645	0.018	0.639	0.006
ANN18	0.913	0.913	0.729	0.184	0.670	0.059	0.647	0.022	0.640	0.007	0.636	0.004
GA		**18**		**18**		**17**		**12**		**7**		**0**
Winner	**67.0%**		**64.8%**		**63.9%**		**63.5%**		**63.4%**		**63.3%**	
MMRE	**93.0%**		**75.0%**		**67.8%**		**65.3%**		**64.1%**		**63.7%**	

Table 33. shows the results obtained by training the second proposed ANN-L18 architecture on the medium cluster. In addition to examining the MRE values for each of the 18 ANNs, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN5) is 44.6%, and the value of MMRE is 44.8%. The required GA criterion was met after six iterations.

Table 33. ANN-L18 results of training part on medium cluster.
Tabela 33. Rezultati treniranja ANN-L18 nad srednjim klasterom.

No. of Iter.	1.		2.		3.		4.		5.		6.	
ANN-L18	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	0.487	0.487	0.458	0.029	0.450	0.008	0.449	0.000	0.447	0.003	0.446	0.000
ANN2	0.612	0.612	0.513	0.099	0.471	0.042	0.457	0.014	0.450	0.006	0.448	0.002
ANN3	0.854	0.854	0.612	0.242	0.504	0.109	0.468	0.036	0.454	0.013	0.450	0.004
ANN4	0.718	0.718	0.612	0.106	0.507	0.105	0.471	0.036	0.455	0.016	0.450	0.005
ANN5	0.469	0.469	0.455	0.014	0.449	0.005	0.447	0.002	0.446	0.001	0.446	0.000
ANN6	0.612	0.612	0.505	0.107	0.503	0.003	0.454	0.049	0.450	0.004	0.448	0.002
ANN7	0.767	0.767	0.566	0.201	0.488	0.078	0.463	0.025	0.450	0.012	0.449	0.001
ANN8	0.568	0.568	0.496	0.073	0.465	0.031	0.453	0.011	0.450	0.004	0.448	0.002
ANN9	0.518	0.518	0.474	0.044	0.457	0.016	0.452	0.006	0.448	0.003	0.447	0.001
ANN10	0.519	0.519	0.473	0.046	0.457	0.016	0.452	0.005	0.450	0.002	0.447	0.003
ANN11	0.725	0.725	0.562	0.163	0.487	0.075	0.463	0.024	0.452	0.011	0.449	0.003
ANN12	0.578	0.578	0.505	0.074	0.469	0.036	0.456	0.013	0.450	0.005	0.448	0.002
ANN13	0.726	0.726	0.556	0.171	0.485	0.071	0.462	0.023	0.452	0.010	0.449	0.003
ANN14	0.628	0.628	0.510	0.119	0.469	0.041	0.455	0.014	0.450	0.004	0.448	0.002
ANN15	0.525	0.525	0.473	0.052	0.456	0.017	0.451	0.005	0.448	0.003	0.447	0.001
ANN16	0.526	0.526	0.473	0.053	0.457	0.016	0.452	0.005	0.448	0.004	0.447	0.001
ANN17	0.737	0.737	0.557	0.180	0.485	0.072	0.462	0.023	0.452	0.010	0.449	0.003
ANN18	0.604	0.604	0.500	0.105	0.465	0.035	0.453	0.012	0.450	0.003	0.448	0.002
GA		**18**		**18**		**15**		**12**		**4**		**0**
Winer	**46.9%**		**45.5%**		**44.9%**		**44.7%**		**44.6%**		**44.6%**	
MMRE	**62.1%**		**51.7%**		**47.3%**		**45.7%**		**45.0%**		**44.8%**	

Table 34. shows the results obtained by training the second proposed ANN-L18 architecture on a large cluster. In addition to examining the MRE values for each of the 18 ANNs, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN5) is 137.0%, and the value of MMRE is 137.3%. The required GA criterion was met after nine iterations.

Table 34. ANN-L18 results of training part on large cluster.
Tabela 34. Rezultati treniranja ANN-L18 nad velikim klasterom.

No. of Iter.	1.		2.		3.		4.		5.		6.		7.		8.		9.	
ANN-L18	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	2.174	2.174	1.752	0.422	1.542	0.210	1.522	0.021	1.404	0.118	1.382	0.022	1.375	0.007	1.372	0.004	1.371	0.001
ANN2	3.220	3.220	2.379	0.841	1.867	0.512	1.644	0.223	1.482	0.162	1.421	0.061	1.384	0.037	1.377	0.007	1.373	0.004
ANN3	4.752	4.752	3.220	1.532	2.255	0.965	1.819	0.436	1.565	0.254	1.461	0.104	1.392	0.069	1.381	0.010	1.375	0.006
ANN4	4.024	4.024	3.220	0.804	2.278	0.942	1.824	0.454	1.573	0.252	1.465	0.108	1.395	0.070	1.382	0.013	1.376	0.006
ANN5	1.704	1.704	1.520	0.184	1.441	0.080	1.404	0.037	1.382	0.022	1.371	0.011	1.371	0.000	1.370	0.000	1.370	0.001
ANN6	3.220	3.220	2.322	0.898	2.273	0.049	1.603	0.671	1.474	0.128	1.417	0.057	1.383	0.034	1.376	0.007	1.373	0.003
ANN7	4.361	4.361	2.837	1.523	2.051	0.787	1.714	0.337	1.471	0.243	1.439	0.032	1.388	0.051	1.379	0.009	1.374	0.005
ANN8	2.789	2.789	2.162	0.626	1.770	0.392	1.567	0.204	1.463	0.104	1.411	0.051	1.382	0.030	1.375	0.006	1.372	0.003
ANN9	2.443	2.443	1.875	0.569	1.608	0.266	1.494	0.114	1.424	0.070	1.392	0.032	1.377	0.015	1.373	0.004	1.371	0.002
ANN10	2.498	2.498	1.908	0.590	1.625	0.283	1.539	0.086	1.468	0.071	1.416	0.053	1.378	0.038	1.373	0.005	1.371	0.002
ANN11	4.037	4.037	2.783	1.254	2.045	0.738	1.748	0.297	1.520	0.228	1.439	0.081	1.389	0.050	1.379	0.011	1.374	0.005
ANN12	2.874	2.874	2.265	0.610	1.828	0.437	1.623	0.204	1.476	0.147	1.418	0.058	1.384	0.034	1.376	0.008	1.373	0.003
ANN13	4.087	4.087	2.740	1.347	2.015	0.725	1.688	0.327	1.513	0.175	1.459	0.054	1.389	0.071	1.378	0.010	1.374	0.005
ANN14	3.367	3.367	2.327	1.040	1.821	0.505	1.596	0.225	1.470	0.126	1.415	0.055	1.382	0.033	1.376	0.006	1.373	0.003
ANN15	2.379	2.379	1.836	0.544	1.588	0.247	1.486	0.102	1.417	0.069	1.389	0.029	1.376	0.013	1.373	0.003	1.371	0.002
ANN16	2.435	2.435	1.833	0.602	1.581	0.251	1.497	0.084	1.415	0.083	1.387	0.027	1.375	0.012	1.372	0.003	1.371	0.002
ANN17	4.173	4.173	2.735	1.438	2.002	0.733	1.677	0.325	1.509	0.168	1.434	0.075	1.387	0.047	1.378	0.009	1.374	0.004
ANN18	3.020	3.020	2.163	0.857	1.748	0.416	1.559	0.189	1.454	0.105	1.407	0.047	1.380	0.027	1.375	0.006	1.372	0.003
GA		18		18		18		18		18		18		16		4		0
Winner	170.4%		152.0%		144.1%		140.4%		138.2%		137.1%		137.1%		137.0%		137.0%	
MMRE	319.8%		232.6%		185.2%		161.1%		147.1%		141.8%		138.3%		137.6%		137.3%	

Figure 25. presents the MMRE results for all three clusters on which the ANN-L18 training procedure was performed using the COCOMO2000 dataset. It can be concluded that the value of MMRE is the lowest on a small cluster (44.6%) and the highest on a large cluster (137.0%). Figure 26. presents the MMRE results for the obtained "Winner" network on all three clusters on which the ANN-L18 training procedure was performed using the COCOMO2000 dataset. It can be concluded that the value of MMRE "Winner" is the lowest on a small cluster after the sixth iteration (44.8%) and the highest on a large cluster after the ninth iteration (137.3%).

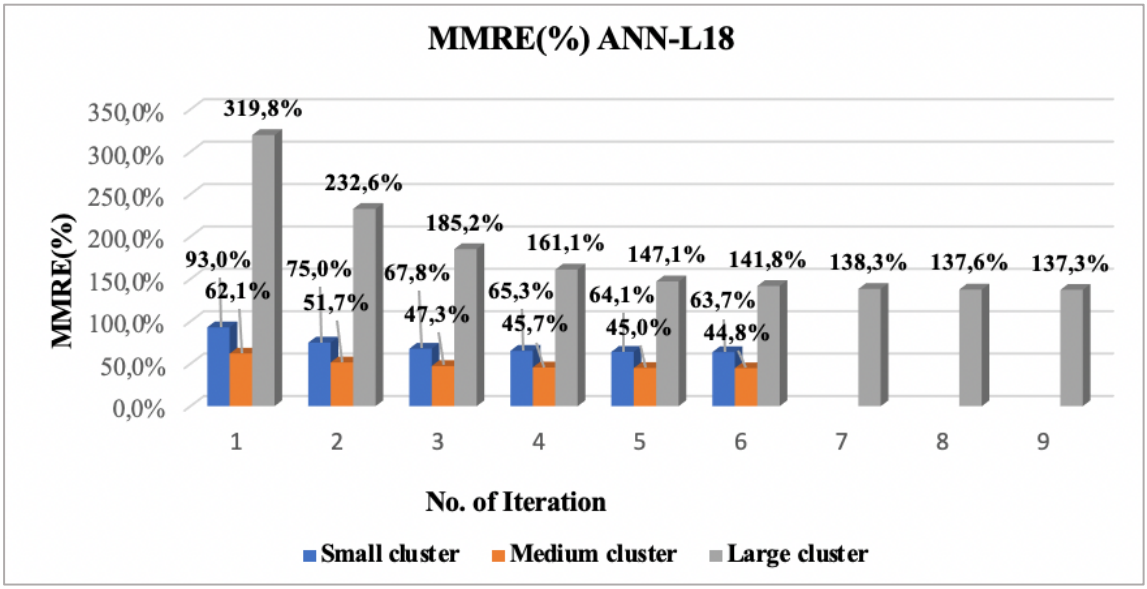


Figure 25. MMRE results in three parts of the experiment for all clusters (ANN-L18).
Slika 25. Rezultati vrednosti MMRE u sva tri dela eksperimenta nad svim klasterima (ANN-L18).

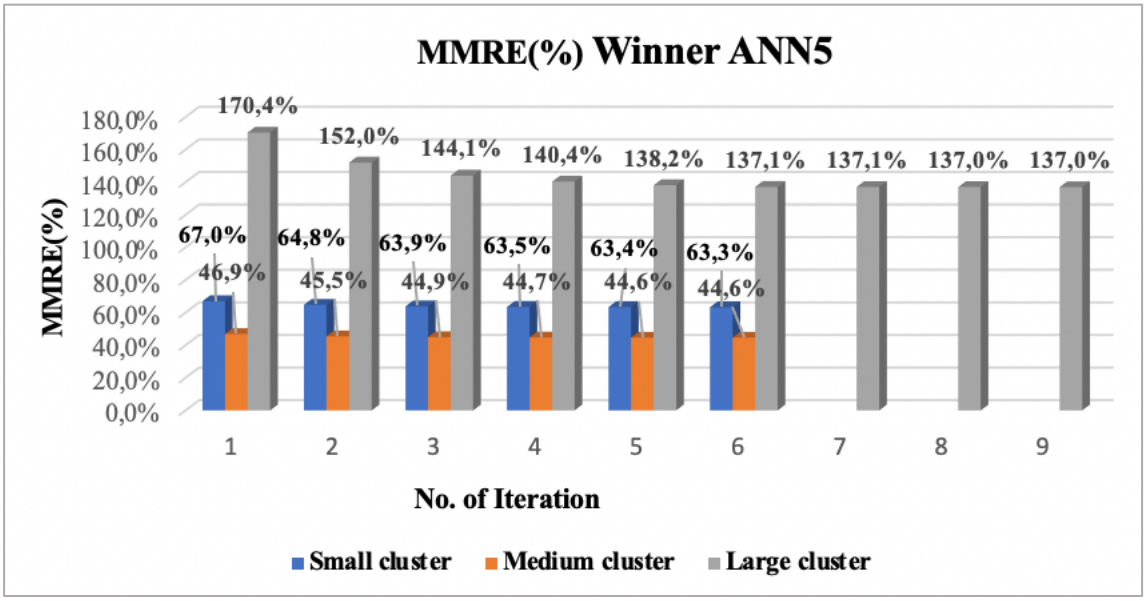


Figure 26. MMRE for "Winner" ANN5 (ANN-L18).
Slika 26. MMRE vrednosti za "Winner" ANN5 (ANN-L18).

Table 35. shows the results obtained by training the third proposed ANN-L27 architecture on a small cluster. In addition to examining the MRE values for each of the 27 ANNs, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The

obtained value of the "Winner" network (ANN5) is 59.8%, and the value of MMRE is 60.4%. The required GA criterion was met after six iterations.

Table 35. ANN-L27 results of training part on small cluster.
Tabela 35. Rezultati treniranja ANN-L27 nad malim klasterom.

No. of Iter.	1.		2.		3.		4.		5.		6.	
ANN-L27	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	0.631	0.631	0.621	0.010	0.614	0.007	0.611	0.003	0.605	0.006	0.604	0.001
ANN2	0.938	0.938	0.699	0.239	0.639	0.060	0.618	0.021	0.601	0.007	0.601	0.000
ANN3	1.929	1.929	1.628	0.301	0.750	0.878	0.654	0.096	0.620	0.004	0.607	0.000
ANN4	1.777	1.777	1.043	0.734	0.725	0.318	0.649	0.076	0.624	0.025	0.609	0.002
ANN5	0.612	0.612	0.610	0.002	0.610	0.000	0.609	0.001	0.607	0.002	0.598	0.009
ANN6	0.775	0.775	0.657	0.119	0.623	0.034	0.613	0.010	0.601	0.012	0.602	0.001
ANN7	1.156	1.156	0.766	0.390	0.657	0.109	0.627	0.030	0.611	0.015	0.605	0.006
ANN8	1.649	1.649	0.938	0.711	0.697	0.240	0.640	0.057	0.621	0.019	0.606	0.001
ANN9	0.621	0.621	0.617	0.004	0.614	0.003	0.611	0.003	0.602	0.009	0.605	0.002
ANN10	1.509	1.509	0.858	0.651	0.678	0.180	0.634	0.044	0.621	0.013	0.605	0.001
ANN11	0.673	0.673	0.634	0.039	0.617	0.017	0.612	0.004	0.603	0.009	0.603	0.000
ANN12	0.981	0.981	0.722	0.259	0.647	0.075	0.622	0.025	0.612	0.009	0.605	0.007
ANN13	0.747	0.747	0.661	0.086	0.626	0.035	0.614	0.012	0.606	0.009	0.602	0.004
ANN14	1.151	1.151	0.736	0.414	0.646	0.091	0.621	0.025	0.599	0.022	0.601	0.002
ANN15	0.908	0.908	0.690	0.218	0.639	0.051	0.619	0.020	0.601	0.018	0.602	0.001
ANN16	0.751	0.751	0.661	0.090	0.630	0.032	0.616	0.013	0.606	0.010	0.603	0.004
ANN17	1.158	1.158	0.762	0.396	0.657	0.105	0.626	0.030	0.610	0.017	0.604	0.005
ANN18	0.900	0.900	0.691	0.208	0.636	0.005	0.617	0.019	0.604	0.013	0.603	0.001
ANN19	1.026	1.026	0.710	0.316	0.641	0.069	0.618	0.022	0.601	0.017	0.602	0.001
ANN20	1.465	1.465	0.854	0.611	0.767	0.086	0.634	0.133	0.617	0.017	0.606	0.001
ANN21	0.668	0.668	0.635	0.033	0.617	0.018	0.612	0.004	0.606	0.006	0.603	0.004
ANN22	0.952	0.952	0.720	0.232	0.650	0.069	0.624	0.027	0.611	0.003	0.605	0.006
ANN23	0.711	0.711	0.654	0.057	0.627	0.027	0.616	0.001	0.614	0.002	0.604	0.001
ANN24	1.169	1.169	0.767	0.402	0.657	0.110	0.626	0.031	0.605	0.001	0.606	0.001
ANN25	0.919	0.919	0.694	0.225	0.635	0.059	0.617	0.001	0.601	0.001	0.601	0.000
ANN26	0.775	0.775	0.669	0.106	0.631	0.038	0.617	0.014	0.604	0.000	0.603	0.000
ANN27	1.102	1.102	0.733	0.369	0.650	0.083	0.623	0.026	0.606	0.000	0.602	0.000
GA		**27**		**24**		**23**		**19**		**12**		**0**
Winner	**61.2%**		**61.0%**		**61.0%**		**60.9%**		**59.9%**		**59.8%**	
MMRE	**102.4%**		**75.7%**		**67.4%**		**64.5%**		**63.0%**		**60.4%**	

Table 36. shows the results obtained by training the third proposed ANN-L27 architecture on the medium cluster. In addition to examining the MRE values for each of the 27 ANNs, the GA criterion was also monitored. Based on all MRE values in each

executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN5) is 43.2%, and the value of MMRE is 43.3%. The required GA criterion was met after five iterations.

Table 36. ANN-L27 results of training part on medium cluster.
Tabela 36. Rezultati treniranja ANN-L27 nad srednjim klasterom.

No. of Iter.	1.		2.		3.		4.		5.	
ANN-L27	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	0.442	0.442	0.439	0.003	0.436	0.003	0.435	0.001	0.434	0.001
ANN2	0.612	0.612	0.473	0.139	0.441	0.032	0.434	0.007	0.433	0.001
ANN3	1.121	1.121	0.701	0.420	0.489	0.213	0.444	0.044	0.436	0.009
ANN4	1.057	1.057	0.654	0.403	0.480	0.174	0.443	0.037	0.435	0.008
ANN5	0.444	0.444	0.438	0.006	0.436	0.001	0.435	0.001	0.434	0.001
ANN6	0.529	0.529	0.451	0.077	0.435	0.017	0.433	0.002	0.433	0.000
ANN7	0.714	0.714	0.497	0.217	0.448	0.049	0.435	0.013	0.435	0.001
ANN8	1.020	1.020	0.612	0.407	0.468	0.144	0.440	0.028	0.434	0.006
ANN9	0.449	0.449	0.439	0.010	0.436	0.003	0.435	0.000	0.434	0.001
ANN10	0.930	0.930	0.557	0.374	0.455	0.102	0.436	0.019	0.433	0.003
ANN11	0.455	0.455	0.442	0.013	0.436	0.006	0.434	0.002	0.433	0.001
ANN12	0.627	0.627	0.482	0.145	0.444	0.038	0.435	0.009	0.433	0.002
ANN13	0.502	0.502	0.456	0.045	0.440	0.017	0.435	0.005	0.434	0.001
ANN14	0.725	0.725	0.492	0.232	0.445	0.047	0.435	0.011	0.433	0.002
ANN15	0.605	0.605	0.464	0.141	0.438	0.026	0.436	0.002	0.434	0.003
ANN16	0.502	0.502	0.452	0.051	0.435	0.017	0.433	0.002	0.433	0.000
ANN17	0.754	0.754	0.504	0.250	0.448	0.056	0.435	0.012	0.433	0.002
ANN18	0.583	0.583	0.470	0.113	0.441	0.029	0.435	0.007	0.433	0.002
ANN19	0.697	0.697	0.479	0.217	0.440	0.039	0.433	0.007	0.432	0.001
ANN20	0.894	0.894	0.551	0.343	0.455	0.096	0.436	0.019	0.433	0.003
ANN21	0.452	0.452	0.444	0.007	0.437	0.007	0.435	0.003	0.434	0.001
ANN22	0.629	0.629	0.481	0.148	0.442	0.039	0.434	0.009	0.433	0.001
ANN23	0.479	0.479	0.452	0.027	0.437	0.015	0.433	0.003	0.433	0.001
ANN24	0.751	0.751	0.511	0.240	0.449	0.062	0.435	0.015	0.433	0.001
ANN25	0.594	0.594	0.473	0.121	0.443	0.030	0.436	0.007	0.434	0.002
ANN26	0.529	0.529	0.456	0.073	0.437	0.019	0.434	0.004	0.433	0.001
ANN27	0.690	0.690	0.487	0.203	0.444	0.044	0.434	0.010	0.433	0.000
GA		**27**		**24**		**22**		**9**		**0**
Winner	**44.2%**		**43.8%**		**43.5%**		**43.3%**		**43.2%**	
MMRE	**65.9%**		**49.5%**		**44.6%**		**43.5%**		**43.3%**	

Table 37. shows the results obtained by training the third proposed ANN-L27 architecture on a large cluster. In addition to examining the MRE values for each of the 27 ANN architecture candidates, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN5) is 48.8%, and the value of MMRE is 49.0%. The required GA criterion was met after eight iterations.

Table 37. ANN-L27 results of training part on large cluster.
Tabela 37. Rezultati treniranja ANN-L27 nad velikim klasterom.

No. of Iter.	1.		2.		3.		4.		5.		6.		7.		8.	
ANN-L27	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	0.812	0.812	0.582	0.230	0.525	0.057	0.502	0.022	0.494	0.009	0.490	0.003	0.489	0.002	0.488	0.001
ANN2	3.343	3.343	1.323	2.020	0.688	0.635	0.572	0.117	0.524	0.048	0.503	0.021	0.494	0.008	0.491	0.004
ANN3	6.378	6.378	3.343	3.035	1.118	2.225	0.687	0.432	0.560	0.127	0.519	0.041	0.501	0.018	0.494	0.007
ANN4	6.115	6.115	2.871	3.245	1.043	1.827	0.654	0.390	0.550	0.104	0.514	0.035	0.499	0.015	0.493	0.006
ANN5	0.973	0.973	0.677	0.296	0.567	0.109	0.522	0.045	0.502	0.020	0.494	0.008	0.490	0.004	0.489	0.002
ANN6	2.610	2.610	0.996	1.614	0.642	0.354	0.547	0.095	0.513	0.034	0.499	0.015	0.493	0.006	0.491	0.002
ANN7	4.052	4.052	1.485	2.567	0.757	0.728	0.577	0.180	0.526	0.051	0.504	0.022	0.495	0.009	0.491	0.004
ANN8	5.958	5.958	2.353	3.605	0.909	1.443	0.616	0.293	0.538	0.078	0.509	0.029	0.500	0.009	0.492	0.008
ANN9	0.650	0.650	0.559	0.091	0.521	0.039	0.501	0.019	0.494	0.007	0.490	0.004	0.489	0.002	0.488	0.001
ANN10	5.445	5.445	1.973	3.472	0.836	1.136	0.597	0.239	0.533	0.065	0.507	0.026	0.496	0.011	0.491	0.005
ANN11	1.776	1.776	0.852	0.924	0.607	0.245	0.538	0.069	0.509	0.029	0.497	0.012	0.492	0.005	0.489	0.002
ANN12	3.414	3.414	1.236	2.177	0.704	0.533	0.564	0.140	0.521	0.044	0.501	0.019	0.494	0.008	0.490	0.004
ANN13	2.150	2.150	0.921	1.229	0.623	0.298	0.541	0.081	0.511	0.031	0.498	0.013	0.492	0.005	0.490	0.003
ANN14	4.191	4.191	1.397	2.794	0.734	0.663	0.569	0.165	0.522	0.046	0.502	0.020	0.494	0.008	0.490	0.004
ANN15	3.410	3.410	1.312	2.098	0.721	0.590	0.568	0.153	0.522	0.046	0.502	0.020	0.494	0.008	0.490	0.004
ANN16	2.201	2.201	1.009	1.192	0.653	0.356	0.552	0.101	0.516	0.036	0.499	0.016	0.493	0.006	0.490	0.003
ANN17	4.473	4.473	1.601	2.872	0.778	0.824	0.582	0.196	0.516	0.066	0.505	0.011	0.495	0.009	0.491	0.004
ANN18	3.055	3.055	1.089	1.966	0.582	0.507	0.553	0.029	0.516	0.037	0.500	0.016	0.493	0.006	0.490	0.003
ANN19	4.050	4.050	1.285	2.765	0.698	0.587	0.560	0.138	0.518	0.041	0.500	0.018	0.493	0.007	0.490	0.003
ANN20	5.232	5.232	1.837	3.395	0.814	1.023	0.591	0.224	0.530	0.061	0.506	0.024	0.496	0.010	0.491	0.004
ANN21	1.503	1.503	0.797	0.706	0.592	0.205	0.532	0.060	0.507	0.025	0.496	0.011	0.491	0.005	0.489	0.002
ANN22	3.555	3.555	1.346	2.210	0.728	0.617	0.570	0.158	0.523	0.047	0.503	0.021	0.494	0.008	0.491	0.004
ANN23	1.860	1.860	0.902	0.958	0.624	0.278	0.543	0.081	0.512	0.031	0.498	0.014	0.492	0.006	0.490	0.003
ANN24	4.406	4.406	1.444	2.962	0.744	0.700	0.571	0.173	0.523	0.048	0.503	0.021	0.494	0.008	0.491	0.004
ANN25	3.115	3.115	1.055	2.060	0.654	0.401	0.548	0.106	0.513	0.034	0.499	0.015	0.493	0.006	0.490	0.003
ANN26	2.610	2.610	1.018	1.593	0.644	0.373	0.548	0.097	0.513	0.034	0.498	0.015	0.493	0.006	0.490	0.003
ANN27	3.987	3.987	1.459	2.528	0.754	0.705	0.575	0.179	0.525	0.050	0.503	0.021	0.495	0.009	0.491	0.004
GA		**27**		**27**		**27**		**27**		**25**		**24**		**4**		**0**
Winner	**65.0%**		**55.9%**		**52.1%**		**50.1%**		**49.4%**		**49.0%**		**48.9%**		**48.8%**	
MMRE	**338.2%**		**136.0%**		**71.3%**		**56.6%**		**52.0%**		**50.1%**		**49.3%**		**49.0%**	

Figure 27. presents the MMRE results for all three clusters on which the ANN-L27 training procedure was performed using the COCOMO2000 dataset. It can be concluded that the value of MMRE is the lowest in the medium cluster (43.3%) and the highest in the small cluster (60.4%). Figure 28. presents the MMRE results for the obtained "Winner" network on all three clusters on which the ANN-L27 training procedure was performed using the COCOMO2000 dataset. It can be concluded that the value of MMRE "Winner" is the lowest in the medium cluster, after the fifth iteration (43.2%), and the highest in the small cluster after the sixth iteration (59.8%).

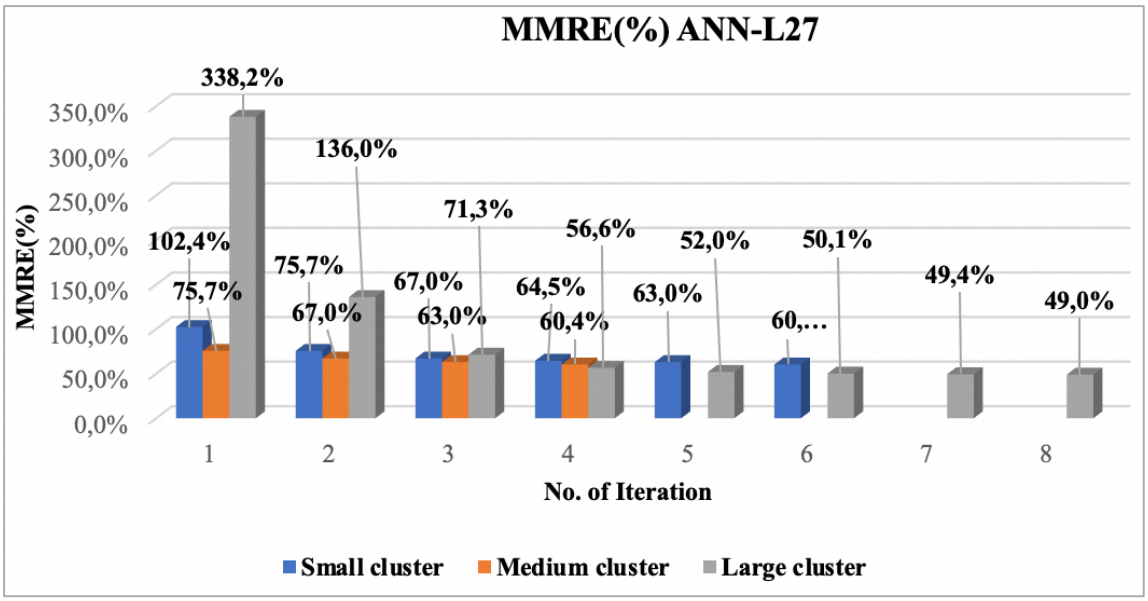


Figure 27. MMRE results in three parts of the experiment for all clusters (ANN-L27).
Slika 27. Rezultati vrednosti MMRE u sva tri dela eksperimenta nad svim klasterima (ANN-L27).

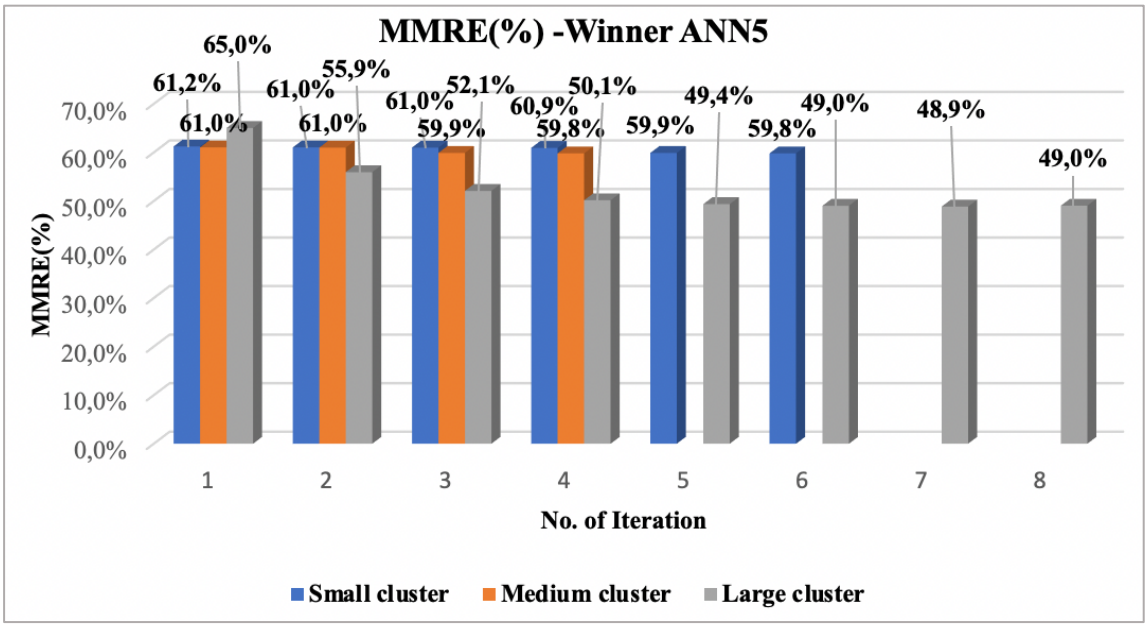


Figure 28. MMRE for "Winner" ANN5 (ANN-L27).
Slika 28. MMRE vrednosti za "Winner" ANN5 (ANN-L27).

Table 38. shows the results obtained by training the fourth proposed ANN-L36 architecture on a small cluster. In addition to examining the MRE values for each of the 36 ANNs, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN23) is 52.4%, and the value of MMRE is 52.6%. The required GA criterion was met after five iterations.

Table 38. ANN-L36 results of training part on small cluster.
Tabela 38. Rezultati treniranja ANN-L36 nad malim klasterom.

No. of Iter.	1.		2.		3.		4.		5.	
ANN-L36	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	1.075	1.075	0.660	0.415	0.553	0.106	0.529	0.024	0.526	0.003
ANN2	0.831	0.831	0.593	0.237	0.537	0.056	0.530	0.007	0.526	0.003
ANN3	1.078	1.078	0.592	0.486	0.538	0.054	0.530	0.008	0.526	0.003
ANN4	0.779	0.779	0.592	0.187	0.543	0.049	0.532	0.011	0.527	0.005
ANN5	0.729	0.729	0.581	0.148	0.537	0.044	0.529	0.008	0.526	0.003
ANN6	0.831	0.831	0.595	0.235	0.539	0.056	0.529	0.010	0.526	0.003
ANN7	0.592	0.592	0.545	0.047	0.534	0.012	0.529	0.005	0.526	0.003
ANN8	1.566	1.566	0.831	0.736	0.593	0.237	0.536	0.058	0.529	0.007
ANN9	0.582	0.582	0.544	0.039	0.532	0.012	0.527	0.004	0.525	0.002
ANN10	0.778	0.778	0.587	0.191	0.542	0.045	0.530	0.012	0.527	0.004
ANN11	0.538	0.538	0.537	0.000	0.532	0.005	0.527	0.005	0.525	0.002
ANN12	1.438	1.438	0.775	0.663	0.581	0.193	0.533	0.048	0.528	0.006
ANN13	0.635	0.635	0.550	0.085	0.534	0.016	0.528	0.005	0.526	0.003
ANN14	0.719	0.719	0.574	0.144	0.537	0.038	0.530	0.007	0.526	0.003
ANN15	1.328	1.328	0.693	0.635	0.568	0.125	0.533	0.035	0.528	0.005
ANN16	1.384	1.384	0.751	0.633	0.577	0.174	0.534	0.043	0.528	0.006
ANN17	0.651	0.651	0.568	0.083	0.541	0.027	0.531	0.010	0.527	0.004
ANN18	0.537	0.537	0.535	0.002	0.532	0.003	0.528	0.004	0.526	0.002
ANN19	0.537	0.537	0.534	0.004	0.531	0.002	0.527	0.004	0.525	0.002
ANN20	1.460	1.460	0.761	0.698	0.579	0.183	0.533	0.045	0.527	0.006
ANN21	0.710	0.710	0.591	0.119	0.538	0.053	0.530	0.008	0.527	0.004
ANN22	0.781	0.781	0.684	0.097	0.558	0.126	0.533	0.025	0.528	0.005
ANN23	0.534	0.534	0.526	0.008	0.526	0.000	0.524	0.002	0.524	0.000
ANN24	1.243	1.243	0.712	0.531	0.565	0.148	0.533	0.032	0.528	0.005
ANN25	0.692	0.692	0.561	0.131	0.533	0.028	0.528	0.006	0.526	0.002
ANN26	0.691	0.691	0.570	0.121	0.536	0.034	0.529	0.007	0.526	0.003
ANN27	1.441	1.441	0.734	0.707	0.565	0.170	0.532	0.032	0.527	0.005
ANN28	0.631	0.631	0.547	0.083	0.532	0.016	0.528	0.004	0.525	0.002
ANN29	1.077	1.077	0.649	0.428	0.547	0.102	0.531	0.016	0.527	0.004
ANN30	0.937	0.937	0.638	0.299	0.552	0.087	0.532	0.020	0.527	0.004
ANN31	1.028	1.028	0.631	0.396	0.536	0.095	0.529	0.007	0.526	0.003
ANN32	0.937	0.937	0.636	0.300	0.550	0.087	0.532	0.018	0.528	0.005
ANN33	0.596	0.596	0.547	0.049	0.533	0.014	0.528	0.005	0.525	0.002
ANN34	1.472	1.472	0.756	0.715	0.575	0.182	0.533	0.041	0.528	0.005

ANN35	0.926	0.926	0.607	0.320	0.537	0.069	0.530	0.008	0.526	0.003
ANN36	0.556	0.556	0.534	0.021	0.528	0.006	0.526	0.003	0.524	0.001
GA		**36**		**32**		**31**		**16**		**0**
Winner	**53.4%**		**52.6%**		**52.6%**		**52.4%**		**52.4%**	
MMRE	**89.8%**		**62.0%**		**54.6%**		**53.0%**		**52.6%**	

Table 39. shows the results obtained by training the fourth proposed ANN-L36 architecture on the medium cluster. In addition to examining the MRE values for each of the 36 ANNs, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN18) is 42.9%, and the value of MMRE is 43.0%. The required GA criterion was met after five iterations.

Table 39. ANN-L36 results of training part on medium cluster.
Tabela 39. Rezultati treniranja ANN-L36 nad srednjim klasterom.

No. of Iter	1.		2.		3.		4.		5.	
ANN-L36	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	0.748	0.748	0.517	0.231	0.450	0.067	0.433	0.017	0.430	0.003
ANN2	0.612	0.612	0.463	0.149	0.437	0.026	0.432	0.006	0.430	0.002
ANN3	0.753	0.753	0.457	0.296	0.435	0.022	0.431	0.004	0.430	0.002
ANN4	0.576	0.576	0.457	0.119	0.439	0.018	0.433	0.006	0.431	0.002
ANN5	0.543	0.543	0.460	0.082	0.437	0.023	0.431	0.006	0.430	0.002
ANN6	0.612	0.612	0.466	0.146	0.438	0.029	0.431	0.007	0.430	0.002
ANN7	0.457	0.457	0.438	0.019	0.433	0.005	0.431	0.002	0.430	0.001
ANN8	1.055	1.055	0.612	0.442	0.467	0.145	0.438	0.029	0.431	0.007
ANN9	0.451	0.451	0.436	0.015	0.431	0.005	0.430	0.002	0.429	0.001
ANN10	0.577	0.577	0.462	0.115	0.438	0.024	0.430	0.008	0.430	0.000
ANN11	0.438	0.438	0.435	0.003	0.432	0.003	0.430	0.002	0.429	0.001
ANN12	0.975	0.975	0.578	0.397	0.460	0.118	0.436	0.024	0.431	0.005
ANN13	0.481	0.481	0.442	0.039	0.433	0.009	0.431	0.003	0.429	0.001
ANN14	0.536	0.536	0.448	0.088	0.435	0.013	0.432	0.004	0.430	0.002
ANN15	0.911	0.911	0.543	0.368	0.453	0.090	0.434	0.019	0.431	0.003
ANN16	0.948	0.948	0.564	0.385	0.459	0.105	0.436	0.023	0.431	0.005
ANN17	0.495	0.495	0.452	0.043	0.438	0.014	0.432	0.006	0.430	0.002
ANN18	0.434	0.434	0.434	0.000	0.433	0.002	0.431	0.002	0.430	0.001
ANN19	0.436	0.436	0.435	0.001	0.432	0.003	0.430	0.002	0.429	0.001
ANN20	0.995	0.995	0.571	0.424	0.459	0.112	0.436	0.023	0.431	0.005
ANN21	0.536	0.536	0.459	0.077	0.437	0.022	0.431	0.006	0.430	0.002
ANN22	0.833	0.833	0.515	0.318	0.437	0.077	0.433	0.004	0.431	0.003
ANN23	0.449	0.449	0.436	0.013	0.431	0.005	0.430	0.001	0.429	0.001
ANN24	0.864	0.864	0.536	0.328	0.453	0.084	0.434	0.019	0.431	0.003
ANN25	0.519	0.519	0.446	0.073	0.434	0.012	0.431	0.003	0.430	0.002
ANN26	0.520	0.520	0.447	0.074	0.435	0.011	0.432	0.003	0.430	0.002
ANN27	0.981	0.981	0.555	0.425	0.454	0.102	0.434	0.020	0.431	0.004
ANN28	0.481	0.481	0.439	0.041	0.433	0.006	0.431	0.002	0.430	0.001
ANN29	0.766	0.766	0.495	0.270	0.438	0.057	0.432	0.006	0.430	0.002
ANN30	0.673	0.673	0.488	0.186	0.444	0.044	0.433	0.011	0.431	0.002
ANN31	0.733	0.733	0.486	0.247	0.441	0.045	0.431	0.010	0.430	0.002
ANN32	0.672	0.672	0.485	0.188	0.443	0.042	0.433	0.010	0.430	0.002
ANN33	0.457	0.457	0.438	0.019	0.432	0.006	0.430	0.002	0.429	0.001
ANN34	0.997	0.997	0.566	0.431	0.457	0.110	0.435	0.022	0.431	0.004
ANN35	0.666	0.666	0.469	0.196	0.439	0.031	0.432	0.006	0.430	0.002
ANN36	0.442	0.442	0.434	0.007	0.431	0.003	0.430	0.002	0.429	0.001
GA		**36**		**32**		**26**		**11**		**0**
Winner	**43.4%**		**43.4%**		**43.1%**		**43.0%**		**42.9%**	
MMRE	**65.6%**		**48.2%**		**44.1%**		**43.2%**		**43.0%**	

Table 40. shows the results obtained by training the fourth proposed ANN-L36 architecture on a large cluster. In addition to examining the MRE values for each of the 36 ANNs, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN23) is 49.5%, and the value of MMRE is 49.9%. The required GA criterion was met after eight iterations.

Table 40. ANN-L36 results of training part on large cluster.
Tabela 40. Rezultati treniranja ANN-L36 nad velikim klasterom.

No. of Iter.	1.		2.		3.		4.		5.		6.		7.		8.	
ANN-L36	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	4.164	4.164	1.624	2.540	0.746	0.878	0.574	0.171	0.530	0.045	0.511	0.018	0.503	0.008	0.499	0.004
ANN2	3.224	3.224	1.356	1.868	0.741	0.615	0.581	0.160	0.534	0.047	0.514	0.021	0.504	0.010	0.499	0.005
ANN3	4.200	4.200	1.624	2.577	0.799	0.825	0.596	0.203	0.540	0.056	0.516	0.024	0.505	0.011	0.500	0.005
ANN4	2.889	2.889	1.624	1.266	0.840	0.783	0.610	0.230	0.545	0.065	0.519	0.026	0.506	0.012	0.501	0.006
ANN5	2.663	2.663	0.996	1.668	0.639	0.357	0.553	0.086	0.522	0.031	0.508	0.014	0.501	0.007	0.498	0.003
ANN6	3.224	3.224	1.371	1.853	0.742	0.629	0.582	0.160	0.534	0.048	0.508	0.026	0.504	0.004	0.499	0.005
ANN7	1.624	1.624	0.888	0.736	0.624	0.264	0.549	0.075	0.520	0.029	0.507	0.013	0.501	0.006	0.498	0.003
ANN8	5.873	5.873	3.224	2.649	1.219	2.005	0.703	0.516	0.572	0.131	0.530	0.042	0.512	0.018	0.503	0.009
ANN9	1.507	1.507	0.725	0.782	0.570	0.155	0.528	0.042	0.510	0.018	0.502	0.008	0.499	0.004	0.497	0.002
ANN10	2.863	2.863	0.711	2.152	0.610	0.101	0.575	0.035	0.532	0.044	0.513	0.019	0.504	0.009	0.499	0.005
ANN11	0.945	0.945	0.894	0.050	0.564	0.330	0.526	0.038	0.510	0.017	0.502	0.008	0.501	0.001	0.499	0.002
ANN12	5.471	5.471	2.638	2.833	1.035	1.604	0.657	0.378	0.559	0.098	0.525	0.034	0.509	0.015	0.503	0.006
ANN13	2.033	2.033	0.898	1.135	0.625	0.273	0.549	0.076	0.520	0.029	0.507	0.013	0.501	0.006	0.498	0.003
ANN14	2.576	2.576	1.281	1.295	0.728	0.553	0.578	0.151	0.533	0.045	0.513	0.020	0.504	0.009	0.499	0.004
ANN15	5.148	5.148	2.060	3.088	0.876	1.183	0.616	0.261	0.546	0.069	0.519	0.027	0.507	0.012	0.501	0.006
ANN16	5.344	5.344	2.483	2.860	1.004	1.480	0.650	0.354	0.542	0.108	0.524	0.018	0.509	0.015	0.502	0.007
ANN17	2.114	2.114	1.189	0.925	0.720	0.469	0.577	0.143	0.533	0.044	0.513	0.020	0.504	0.009	0.499	0.005
ANN18	0.957	0.957	0.680	0.277	0.564	0.115	0.527	0.038	0.510	0.017	0.502	0.008	0.498	0.004	0.497	0.002
ANN19	0.894	0.894	0.661	0.233	0.561	0.101	0.525	0.036	0.509	0.016	0.502	0.007	0.498	0.004	0.497	0.002
ANN20	5.578	5.578	2.546	3.032	1.011	1.535	0.651	0.360	0.557	0.094	0.524	0.033	0.509	0.015	0.502	0.007
ANN21	2.545	2.545	2.072	0.473	0.728	1.344	0.578	0.150	0.533	0.045	0.513	0.020	0.504	0.009	0.499	0.005
ANN22	4.700	4.700	2.367	2.333	1.007	1.360	0.653	0.354	0.557	0.096	0.524	0.032	0.509	0.015	0.502	0.007
ANN23	0.625	0.625	0.537	0.088	0.512	0.025	0.503	0.010	0.499	0.004	0.497	0.002	0.496	0.001	0.495	0.000
ANN24	4.890	4.890	1.994	2.896	0.870	1.124	0.605	0.265	0.546	0.059	0.519	0.027	0.507	0.012	0.501	0.006
ANN25	2.372	2.372	0.914	1.459	0.626	0.287	0.552	0.075	0.520	0.032	0.507	0.013	0.501	0.006	0.498	0.003
ANN26	2.411	2.411	1.255	1.155	0.728	0.527	0.579	0.149	0.533	0.046	0.513	0.020	0.504	0.009	0.499	0.005
ANN27	5.505	5.505	2.136	3.369	0.882	1.254	0.616	0.266	0.521	0.095	0.519	0.002	0.507	0.012	0.501	0.006
ANN28	1.967	1.967	0.927	1.040	0.626	0.301	0.549	0.077	0.520	0.029	0.507	0.013	0.501	0.006	0.498	0.003
ANN29	4.320	4.320	1.461	2.859	0.740	0.721	0.579	0.161	0.533	0.046	0.513	0.020	0.504	0.009	0.499	0.004
ANN30	3.704	3.704	1.774	1.930	0.851	0.923	0.611	0.240	0.545	0.066	0.518	0.026	0.506	0.012	0.501	0.006
ANN31	4.106	4.106	1.360	2.745	0.727	0.634	0.575	0.152	0.532	0.043	0.523	0.009	0.503	0.020	0.499	0.004
ANN32	3.693	3.693	1.748	1.945	0.848	0.900	0.610	0.238	0.545	0.065	0.518	0.026	0.506	0.012	0.501	0.006
ANN33	1.676	1.676	0.925	0.751	0.631	0.294	0.551	0.080	0.521	0.030	0.507	0.014	0.501	0.006	0.498	0.003
ANN34	5.594	5.594	2.558	3.036	1.015	1.543	0.652	0.363	0.558	0.094	0.524	0.034	0.509	0.015	0.502	0.007
ANN35	3.643	3.643	1.373	2.270	0.733	0.640	0.578	0.155	0.533	0.045	0.513	0.020	0.504	0.009	0.499	0.004
ANN36	1.224	1.224	0.701	0.523	0.567	0.134	0.527	0.040	0.510	0.017	0.502	0.008	0.499	0.004	0.497	0.002
GA		**36**		**36**		**36**		**35**		**35**		**28**		**14**		**0**
Winner	**62.5%**		**53.7%**		**51.2%**		**50.3%**		**49.9%**		**49.7%**		**49.6%**		**49.5%**	
MMRE	**323.0%**		**148.8%**		**75.9%**		**58.4%**		**53.2%**		**51.3%**		**50.4%**		**49.9%**	

Figure 29. presents the MMRE results for all three clusters on which the ANN-L36 training procedure was performed using the COCOMO2000 dataset. It can be concluded that the value of MMRE is the lowest on the medium cluster (43.0%) and the highest on the small cluster (52.6%). Figure 30. presents the MMRE results for the obtained "Winner" network on all three clusters on which the ANN-L36 training procedure was performed using the COCOMO2000 dataset. It can be concluded that the value of MMRE "Winner" is the lowest in the medium cluster, after the fifth iteration (42.9%), and the highest in the small cluster after the five iterations (52.4%).

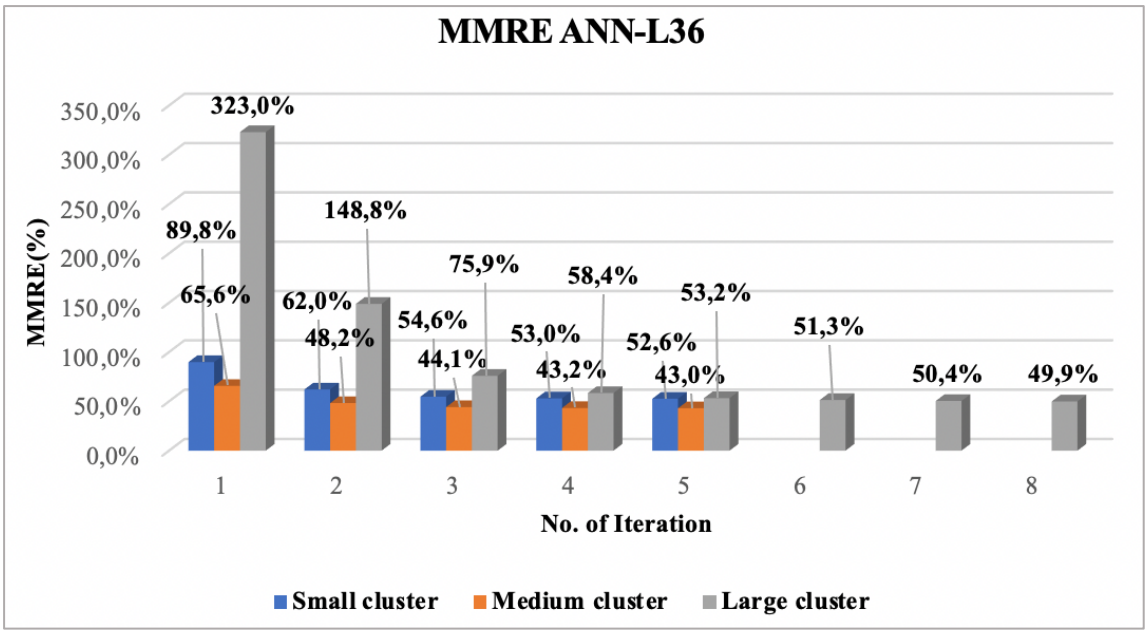


Figure 29. MMRE results in three parts of the experiment for all clusters (ANN-L36).
Slika 29. Rezultati vrednosti MMRE u sva tri dela eksperimenta nad svim klasterima (ANN-L36).

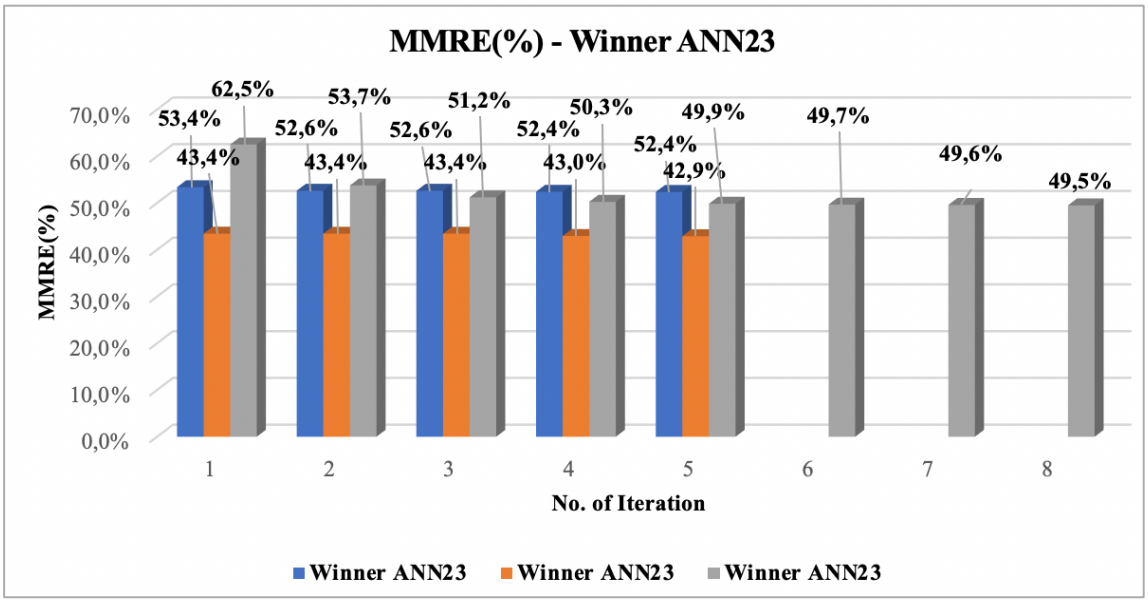


Figure 30. MMRE for "Winner" ANN23 (ANN-L36).
Slika 30. MMRE vrednosti za "Winner" ANN23 (ANN-L36).

The convergence rate describes the rate of the proposed algorithm, that is, the number of iterations required by the algorithm to achieve the smallest value, i.e. the minimum value of the MRE in experiment. From the point of view of a project, it is calculated as the minimum relative error difference between every two output iterations. The rate of convergence of each cluster depends on the particular ANN architecture. Achieving the minimum MRE and fulfilling the criteria, GA<0.01 is achieved for the small and medium cluster after 5 iterations in architectures ANN-L27 and ANN-L36, while for the large cluster this condition is achieved after 8 iterations for architectures ANN-L9, ANN-L27, and ANN- L36. The rate of convergence in this approach implies a significant reduction in the number of iterations, and therefore a reduction in the time of effort estimation during the development of software projects.

From Figure 31, it can be concluded that:
$\delta i\,(ANN-L36) > \delta i\,(ANN-L18) > \delta i\,(ANN-L9) > \delta i\,(ANN-L27)$, that is, the ANN-L36 architecture converges the fastest to the minimum MMRE on a small cluster.

From Figure 32, it can be concluded that:
$\delta i\,(ANN-L36) > \delta i\,(ANN-L27) > \delta i\,(ANN-L18) > \delta i\,(ANN-L9)$, that is, the ANN-L36 architecture again converges the fastest for medium cluster.

From Figure 33, it can be concluded that:

$\delta i\ (ANN - L36) > \delta i\ (ANN - L27) > \delta i\ (ANN - L18) > \delta i\ (ANN - L9)$, that is, again, the ANN-L36 architecture converges the fastest to the minimum MMRE observed for a large cluster.

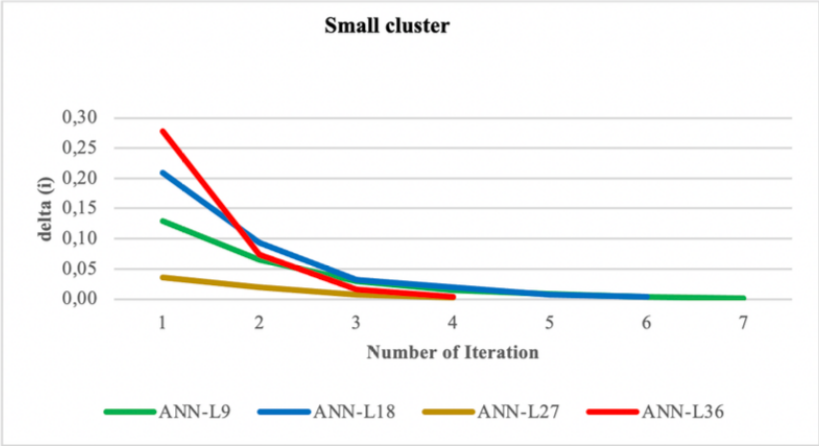


Figure 31. Convergence rate of the four proposed ANNs on small cluster.
Slika 31. Brzina konvergencije četiri predložene ANN arhitekture nad malim klasterom.

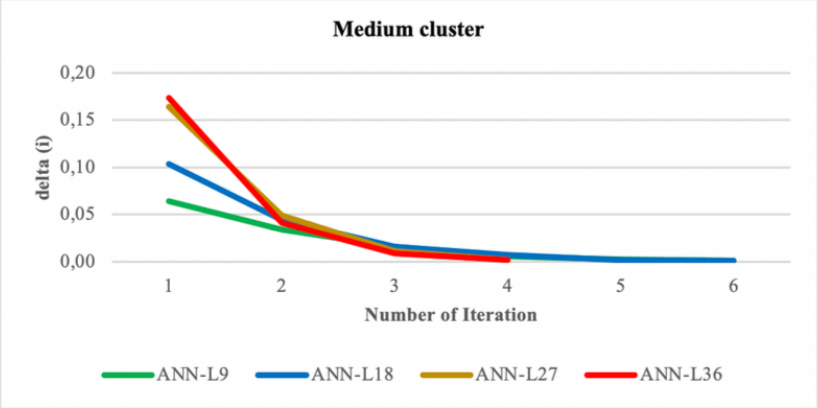


Figure 32. Convergence rate of the four proposed ANNs on medium cluster.
Slika 32. Brzina konvergencije četiri predložene ANN arhitekture nad srednjim klasterom.

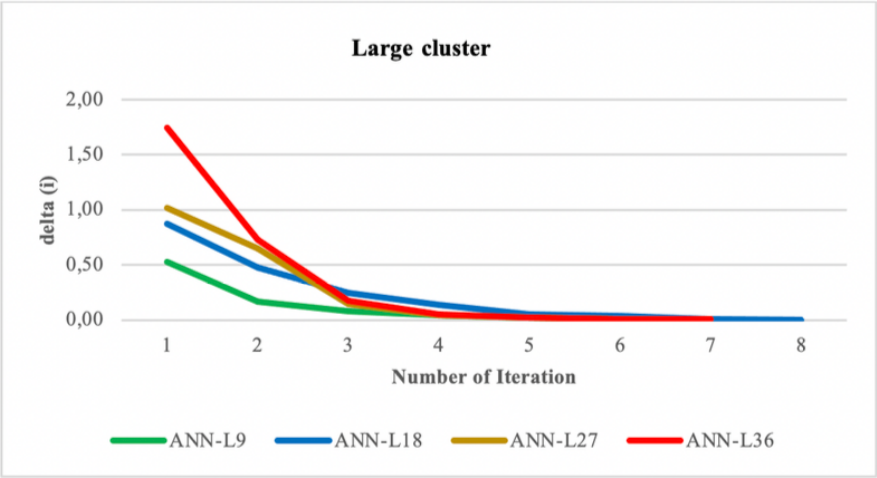


Figure 33. Convergence rate of the four proposed ANNs on large cluster.
Slika 33. Brzina konvergencije četiri predložene ANN arhitekture nad velikim klasterom.

In the first part of this experiment, devoted to training proposed architectures: ANN-L9, ANN-L18, ANN-L27 and ANN-L36, a COCOMO2000 dataset of 100 projects was used, divided into clusters according to actual effort expressed in person-months (PM) as follows: for PM<90 (small cluster of 32 projects), for 90<PM<500 (medium cluster of 40 projects), and PM>500 (large cluster of 28 projects).

The average value of MMRE during this part of the experiment is the lowest in the case of ANN-L36 network, which is the best result of all the obtained and is 43.3%.

The ANN-L27 network has an MMRE value of 45.3%, which is 2% less than the best result.

The networks ANN-L18 and ANN-L9, with 59.7% and 72.0% respectively, have a worse result, i.e. a higher value of MMRE.

Based on the results in Table 8, it can be concluded that the ANN-L36 network gives the best result and converges the fastest to the minimum value of MMRE.

Additionally, to improve the efficiency of the experiments, the Mean absolute error (MAE) was calculated. This value represents the average of differences in the absolute value between the actual effort and each estimated effort following the total number of projects, see Table 41.

From the results it can be concluded that the ANN-L36 network achieves the best behavior with an average MMRE value of 43.3% in all three parts of the experiment. A slightly higher value of MMRE, about 2%, is achieved by the ANN-L27 architecture (45.3%). A weaker result is achieved with ANN-L18 (59.7%), and the worst result is achieved with ANN-L9 (72%), see Table 42, Figure 34.

It can be concluded that by increasing the number of hidden layers, the estimation of MMRE value is more reliable.

Table 41. MAE for each proposed ANN architecture on clusters (COCOMO2000).
Tabela 41. Vrednosti MAE za svaku predloženu arhitekturu u svim klasterima (COCOMO2000).

Experiment	ANN9			ANN18			ANN27			ANN36		
	Small cluster	Medium cluster	Large cluster	Small cluster	Medium cluster	Large cluster	Small cluster	Medium cluster	Large cluster	Small cluster	Medium cluster	Large cluster
Training	36.1	230.6	3114.6	26.0	185.7	2392.9	16.9	157.8	924.7	30.5	168.5	991.3
Testing	31.5	218.6	1487.7	23.5	174.6	1163.2	30.7	157.5	651.1	19.7	157.1	706.5
Validation1	54.8	223.4	4013.6	25.2	186.5	2903.3	32.2	147.2	941.7	31.0	150.1	1106.3
Validation2	34.2	220.8	1420.6	26.4	165.8	1128.5	12.5	109.9	703.0	18.4	145.1	711.1
Validation3	36.8	246.7	2034.5	36.1	192.3	1804.2	28.3	137.2	875.5	29.1	140.2	868.3
MAE(%)	38.68	228.02	2414.2	27.44	180.98	1878.42	24.12	141.92	819.2	25.74	152.2	876.7
AVERAGE((MAE(%))		**893.6**			**695.6**			**328.4**			**351.5**	

Table 42. MMRE for each proposed ANN architecture on clusters (COCOMO2000).
Tabela 42. Vrednosti MMRE za svaku predloženu arhitekturu u svim klasterima (COCOMO2000).

Experiment	ANN-L9			ANN-L18			ANN-L27			ANN-L36		
	Small cluster	Medium cluster	Large cluster	Small cluster	Medium cluster	Large cluster	Small cluster	Medium cluster	Large cluster	Small cluster	Medium cluster	Large cluster
Training	80.7	49.6	206.9	69.6	44.8	141.1	65.4	43.3	47.3	65.6	43.1	49.7
Testing	63.3	41.6	13.6	59.9	32.3	13.6	54.9	30.8	20.3	57.4	31.3	16.1
Validation1	30.8	38.4	293.7	52.8	35.3	195.4	37.5	64.5	52.3	42.4	35.3	55.9
Validation2	53.2	48	53.8	56.4	42.7	41	68.4	45.7	45.1	59.5	41.2	44.1
Validation3	50.8	32.8	23.2	56.1	31	24	50.3	33.7	19.6	55.4	32.8	19.5
MMRE(%)	55.8	42.1	118.2	59.0	37.2	83.0	55.3	43.6	36.9	56.1	36.7	37.1
AVERAGE((MMRE(%))		72.0			59.7			45.3			43.3	

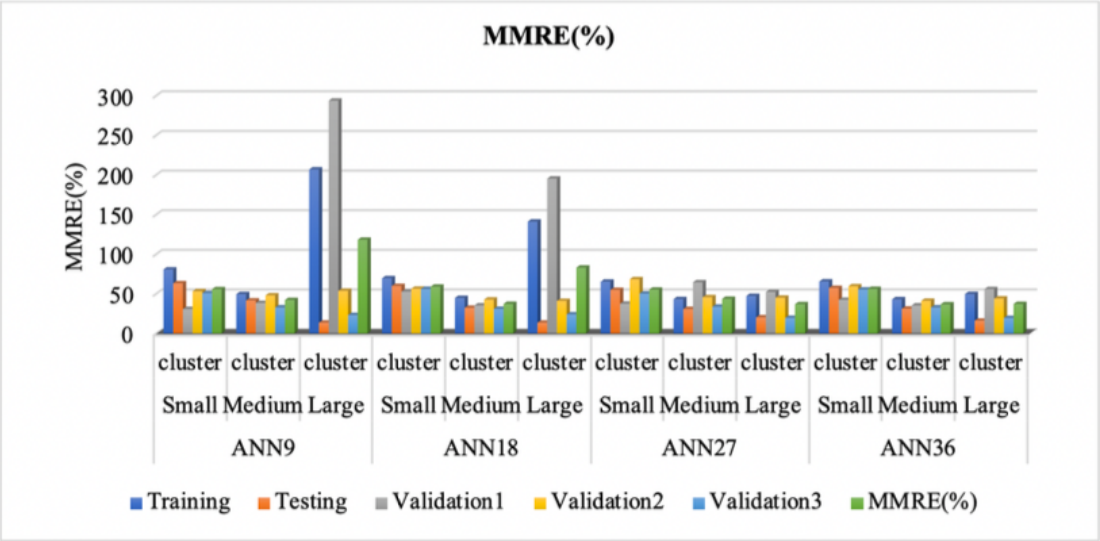


Figure 34. MMRE values of proposed ANN architectures for each part of the experiment (COCOMO2000).
Slika 34. Vrednosti MMRE za svaku predloženu arhitekturu u svakom delu eksperimenta (COCOMO2000).

In the training part of the experiment, it can be concluded that the medium cluster gives the best results for the ANN-L36 architecture (43.1%).

In the testing part of the experiment, it can be concluded that the best result is achieved by a large cluster for the architectures ANN-L9 and ANN-L18 (13.6% for both architectures).

In the validation part of the experiment, the best results were achieved on the medium cluster for all proposed architectures.

During all three parts of the experiments, the prediction PRED (%) was monitored, i.e. the number of projects that meet the set GA criterion. It can be concluded that the value of PRED is approximate in all proposed architectures.

During the training experiment, the best prediction is achieved by the ANN-L9 network, while in part of testing and validation experiments, the best PRED value is achieved with the ANN-L27 and ANN-L36 networks, depending on the dataset. During the first and second validations, the best prediction result is achieved with the ANN-L36 network, while during the third validation, the best results are obtained with the ANN-L27 architecture, see Table 43.

The results in Table 10. indicate the heterogeneous nature of the designs of each datasets used and directly affect the prediction results within all three parts of the experiment. It can be seen that data sets from 1 to 5 are very heterogeneous in terms of the programming languages used, the duration of application development, and a very large range of actual effort values, with a large standard deviation.

Table 43. Monitoring the prediction of each part of the experiment (COCOMO2000).
Tabela 43. Praćenje predikcije u svakom delu eksperimenta (COCOMO2000).

		Training		
PRED(%)	ANN-L9(%)	ANN-L18(%)	ANN-L27(%)	ANN-L36(%)
PRED(25)	20.0	21.0	16.0	16.0
PRED(30)	22.0	23.0	22.0	21.0
PRED(50)	51.0	46.0	42.0	43.0
		Testing		
PRED(25)	35.1	45.5	60.4	45.2
PRED(30)	50.4	50.2	65.3	50.3
PRED(50)	80.7	75.2	80.0	70.3
		Validation1		
PRED(25)	25.5	17.6	17.6	25.5
PRED(30)	31.4	21.6	23.5	33.3
PRED(50)	62.7	43.1	41.2	56.9
		Validation2		
PRED(25)	25.0	18.3	16.7	16.7
PRED(30)	33.3	21.7	20.0	25.0
PRED(50)	68.3	56.7	35.0	43.3
		Validation3		
PRED(25)	25.1	19.0	25.1	25.1
PRED(30)	38.2	25.1	25.1	41.6
PRED(50)	71.4	69.9	75.9	73.4

The minimum value of MRE also depends on the nature of the projects, so in addition to the obtained training results on three clusters, it is necessary to test and validate these models on different data sources, i.e. on several different datasets. The correlation coefficient in this method represents the agreement of the actual effort and the estimated value. Statistical verification of the values of the correlation coefficients for all four proposed architectures proves the reliability of the method used to obtain the estimated value of each specified architecture. The observed correlations were obtained on a complete dataset and include all three defined clusters.

A lower correlation value is observed in the simpler architectures ANN-L9 and ANN-L18, while the correlation has a higher value in the more complex architectures ANN- L27 and ANN-L36. It can be concluded that the best correlation results are obtained according to Pearson's and Spearman's for the architecture ANN-L36 and according to R^2 for the architecture ANN-L27, see Table 44. All three coefficients were used to represent the correlation between the estimated and actual values of the actual effort for each of the selected architectures ANN-L9, ANN-L18, ANN-L27, and ANN-L36.

Table 44. Correlation for each ANN architecture (COCOMO2000).
Tabela 44. Korelacija svake predložene ANN arhitekture (COCOMO2000).

Correlation	ANN-L9	ANN-L18	ANN-L27	ANN-L36
Pearson's	0.617	0.615	0.714	0.866
Spearman's rho	0.869	0.864	0.862	0.904
R^2	0.827	0.861	0.862	0.869

Based on all parts of the experiment, it can be concluded that the ANN-L36 architecture has the best results for the MMRE value (43.3%), observed on all clusters Table 42, the number of iterations is the lowest compared to other architectures, and for the small and medium cluster it is 5. The ANN-L36 architecture converges fastest compared to all proposed architectures, see Figure 31, Figure 32, and Figure 33. The three correlation coefficients attempt to best match the actual effort values for all datasets used in all parts of the experiment for ANN-L36.

Prediction monitoring shows the best result with ANN-L36 in all parts of the experiment. Finally, among all the proposed architectures, regardless of the nature of the project, ANN-L36 gives the best and most reliable results.

The results shown in the previous tables and figures of this approach (COCOMO2000) were processed in the R programming language (avaiable at: https://www.r-project.org/) and checked in the Python programming language (avaiable at: https://www.python.org/) within the RStudio environment (avaiable at: https://www.rstudio.com/products/rstudio/). The data required for statistical analysis were processed in the IBM SPSS Statistical 25 software tool (avaiable at: https://www.ibm.com/support/pages/downloading-ibm-spss-statistics-25).

4.2 Obtained results using COSMIC FFP model and ANN

Within the second proposed, improved COSMIC FFP approach, two different ANN architectures based on Taguchi's orthogonal vector plans, designated as ANN-L12 and ANN-L36prim, were used. In the first part of the experiment - training these ANN architectures, the used ISBSG data set was divided into five clusters depending on the functional size. After several experiments, it was found that the most reliable results in training and testing parts of the experiments are given by the proposed division of 70:30. In each cluster, 70% of the projects were used for training and the remaining 30% for testing.

Table 45. shows the results obtained by training the first proposed ANN-L12 architecture on the first cluster. In addition to examining the MRE values for each of the 12 ANN architecture candidates, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value (the ANN with the best performances). Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN1) is 36.1%, and the value of MMRE is 36.4%. The required GA criterion was met after five iterations.

Table 45. ANN-L12 results of training part on Dataset_1 (COSMIC FFP).
Tabela 45. Rezultati treniranja ANN-L12 u Dataset_1 (COSMIC FFP).

Dataset_1										
No. of Iter.	**1.**		**2.**		**3.**		**4.**		**5.**	
ANN-L12	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**
ANN1	0.361	0.361	0.361	0.000	0.361	0.000	0.361	0.000	0.361	0.000
ANN2	1.044	1.044	0.552	0.492	0.392	0.160	0.368	0.024	0.364	0.004
ANN3	1.053	1.053	0.552	0.502	0.397	0.155	0.368	0.029	0.364	0.004
ANN4	0.552	0.552	0.390	0.162	0.366	0.024	0.362	0.004	0.361	0.001
ANN5	0.408	0.408	0.385	0.023	0.375	0.009	0.369	0.007	0.365	0.003
ANN6	0.425	0.425	0.398	0.027	0.379	0.018	0.370	0.009	0.365	0.004
ANN7	0.436	0.436	0.401	0.035	0.380	0.020	0.370	0.010	0.366	0.005
ANN8	0.520	0.520	0.392	0.128	0.365	0.027	0.362	0.003	0.361	0.000
ANN9	0.412	0.412	0.388	0.023	0.376	0.012	0.369	0.007	0.365	0.004
ANN10	0.854	0.854	0.447	0.407	0.377	0.070	0.368	0.010	0.364	0.004
ANN11	0.906	0.906	0.448	0.458	0.376	0.072	0.367	0.009	0.364	0.003
ANN12	0.575	0.575	0.407	0.168	0.386	0.021	0.374	0.012	0.367	0.006
GA		**12**		**11**		**10**		**3**		**0**
Winner	**36.1%**		**36.1%**		**36.1%**		**36.1%**		**36.1%**	
MMRE	**62.9%**		**42.7%**		**37.8%**		**36.7%**		**36.4%**	

Table 46. shows the results obtained by training the first proposed ANN-L12 architecture on the second cluster. In addition to examining the MRE values for each of the 12 ANN architecture candidates, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN1) is 35.4% and the value of MMRE is 35.7%. The required GA criterion was met after six iterations.

Table 46. ANN-L12 results of training part on Dataset_2 (COSMIC FFP).
Tabela 46. Rezultati treniranja ANN-L12 u Dataset_2 (COSMIC FFP).

Dataset_2												
No. of Iter.	**1.**		**2.**		**3.**		**4.**		**5.**		**6.**	
ANN-L12	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**
ANN1	0.356	0.356	0.356	0.000	0.356	0.000	0.356	0.000	0.354	0.002	0.354	0.000
ANN2	1.344	1.344	0.710	0.635	0.449	0.261	0.378	0.071	0.361	0.017	0.358	0.004
ANN3	1.351	1.351	0.710	0.642	0.450	0.260	0.379	0.071	0.362	0.017	0.358	0.004
ANN4	0.704	0.704	0.450	0.254	0.376	0.073	0.359	0.017	0.356	0.003	0.356	0.000
ANN5	0.452	0.452	0.391	0.061	0.373	0.018	0.365	0.009	0.361	0.004	0.358	0.002
ANN6	0.450	0.450	0.398	0.053	0.375	0.022	0.365	0.010	0.361	0.004	0.359	0.002
ANN7	0.462	0.462	0.400	0.061	0.376	0.025	0.365	0.010	0.361	0.005	0.359	0.002
ANN8	0.683	0.683	0.444	0.239	0.374	0.069	0.359	0.015	0.356	0.003	0.356	0.000
ANN9	0.462	0.462	0.396	0.066	0.374	0.021	0.365	0.010	0.361	0.004	0.358	0.002
ANN10	1.074	1.074	0.568	0.506	0.408	0.160	0.369	0.039	0.359	0.009	0.357	0.002
ANN11	1.130	1.130	0.569	0.561	0.406	0.163	0.368	0.038	0.359	0.009	0.357	0.002
ANN12	0.712	0.712	0.454	0.259	0.389	0.065	0.369	0.020	0.362	0.007	0.359	0.003
GA		**12**		**11**		**11**		**9**		**2**		**0**
Winner	**35.6%**		**35.6%**		**35.6%**		**35.6%**		**35.4%**		**35.4%**	
MMRE	**76.5%**		**48.7%**		**39.2%**		**36.6%**		**35.9%**		**35.7%**	

Table 47. shows the results obtained by training the first proposed ANN-L12 architecture on the third cluster. In addition to examining the MRE values for each of the 12 ANN architecture candidates, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN12) is 16.6%, and the value of MMRE is 16.8%. The required GA criterion was met after five iterations.

Table 47. ANN-L12 results of training part on Dataset_3 (COSMIC FFP).
Tabela 47. Rezultati treniranja ANN-L12 u Dataset_3 (COSMIC FFP).

Dataset_3										
No. of Iter.	**1.**		**2.**		**3.**		**4.**		**5.**	
ANN-L12	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**
ANN1	0.214	0.214	0.204	0.010	0.182	0.022	0.174	0.007	0.170	0.004
ANN2	0.274	0.274	0.230	0.044	0.188	0.043	0.173	0.014	0.171	0.002
ANN3	0.272	0.272	0.213	0.059	0.177	0.036	0.173	0.004	0.171	0.002
ANN4	0.175	0.175	0.171	0.004	0.171	0.000	0.169	0.002	0.168	0.000
ANN5	0.187	0.187	0.170	0.017	0.168	0.001	0.168	0.000	0.166	0.002
ANN6	0.212	0.212	0.184	0.028	0.171	0.012	0.169	0.002	0.168	0.001
ANN7	0.215	0.215	0.175	0.040	0.169	0.006	0.169	0.001	0.166	0.002
ANN8	0.175	0.175	0.171	0.004	0.171	0.000	0.169	0.002	0.168	0.000
ANN9	0.213	0.213	0.184	0.029	0.171	0.013	0.169	0.002	0.168	0.001
ANN10	0.216	0.216	0.186	0.030	0.177	0.009	0.170	0.007	0.169	0.001
ANN11	0.236	0.236	0.191	0.045	0.178	0.013	0.170	0.008	0.170	0.000
ANN12	0.173	0.173	0.169	0.004	0.168	0.001	0.167	0.001	0.167	0.001
GA		**12**		**9**		**6**		**1**		**0**
Winner	**17.3%**		**16.9%**		**16.8%**		**16.7%**		**16.6%**	
MMRE	**21.3%**		**18.7%**		**17.4%**		**17.0%**		**16.8%**	

Table 48. shows the results obtained by training the first proposed ANN-L12 architecture on the fourth cluster. In addition to examining the MRE values for each of the 12 ANN architecture candidates, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN1) is 29.4%, and the value of MMRE is 29.6%. The required GA criterion was met after six iterations.

Table 48. ANN-L12 results of training part on Dataset_4 (COSMIC FFP).
Tabela 48. Rezultati treniranja ANN-L12 u Dataset_4 (COSMIC FFP).

Dataset_4												
No. of Iter.	**1.**		**2.**		**3.**		**4.**		**5.**		**6.**	
ANN-L12	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**
ANN1	0.297	0.297	0.297	0.000	0.297	0.000	0.297	0.000	0.297	0.000	0.297	0.000
ANN2	1.303	1.303	0.665	0.639	0.381	0.284	0.307	0.073	0.296	0.011	0.296	0.000
ANN3	1.285	1.285	0.665	0.621	0.383	0.282	0.308	0.075	0.296	0.012	0.296	0.001
ANN4	0.638	0.638	0.378	0.261	0.305	0.072	0.295	0.010	0.295	0.001	0.294	0.000
ANN5	0.387	0.387	0.316	0.071	0.307	0.009	0.302	0.005	0.300	0.003	0.298	0.001
ANN6	0.365	0.365	0.320	0.045	0.308	0.012	0.302	0.006	0.300	0.003	0.298	0.001
ANN7	0.364	0.364	0.319	0.044	0.308	0.012	0.302	0.006	0.300	0.003	0.298	0.001
ANN8	0.640	0.640	0.380	0.260	0.306	0.074	0.297	0.010	0.296	0.001	0.296	0.000
ANN9	0.373	0.373	0.320	0.053	0.309	0.012	0.303	0.006	0.300	0.003	0.298	0.001
ANN10	1.034	1.034	0.504	0.530	0.336	0.168	0.298	0.038	0.296	0.002	0.296	0.000
ANN11	1.094	1.094	0.515	0.579	0.337	0.178	0.298	0.039	0.296	0.002	0.296	0.000
ANN12	0.685	0.685	0.369	0.316	0.312	0.057	0.303	0.009	0.299	0.003	0.298	0.001
GA		**12**		**11**		**10**		**5**		**2**		**0**
Winner	**29.7%**		**29.7%**		**29.7%**		**29.5%**		**29.5%**		**29.4%**	
MMRE	**70.5%**		**42.0%**		**32.4%**		**30.1%**		**29.7%**		**29.6%**	

Table 49. shows the results obtained by training the first proposed ANN-L12 architecture on the fifth cluster. In addition to examining the MRE values for each of the 12 ANN architecture candidates, the GA criterion was also monitored. Based on all MRE values in each executed iteration, the "Winner" network is determined (the ANN with the best performances), i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN1) is 24.1%, and the value of MMRE is 24.2%. The required GA criterion was met after six iterations.

Table 49. ANN-L12 results of training part on Dataset_5 (COSMIC FFP).
Tabela 49. Rezultati treniranja ANN-L12 u Dataset_5 (COSMIC FFP).

Dataset_5												
No. of Iter.	**1.**		**2.**		**3.**		**4.**		**5.**		**6.**	
ANN-L12	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**	**MRE**	**GA**
ANN1	0.248	0.248	0.241	0.007	0.241	0.000	0.241	0.000	0.241	0.000	0.241	0.000
ANN2	1.084	1.084	0.571	0.513	0.329	0.242	0.266	0.063	0.246	0.020	0.243	0.003
ANN3	1.108	1.108	0.571	0.537	0.331	0.240	0.267	0.064	0.247	0.020	0.243	0.004
ANN4	0.550	0.550	0.316	0.235	0.263	0.053	0.246	0.017	0.243	0.003	0.241	0.001
ANN5	0.345	0.345	0.264	0.081	0.250	0.014	0.246	0.004	0.243	0.002	0.242	0.001
ANN6	0.322	0.322	0.267	0.055	0.255	0.012	0.248	0.007	0.244	0.003	0.243	0.002
ANN7	0.331	0.331	0.270	0.061	0.254	0.016	0.247	0.007	0.244	0.004	0.242	0.001
ANN8	0.514	0.514	0.311	0.203	0.261	0.050	0.245	0.016	0.242	0.003	0.241	0.001
ANN9	0.320	0.320	0.262	0.059	0.253	0.009	0.247	0.006	0.244	0.003	0.242	0.002
ANN10	0.897	0.897	0.429	0.468	0.296	0.134	0.256	0.040	0.246	0.010	0.243	0.003
ANN11	0.970	0.970	0.436	0.534	0.295	0.141	0.255	0.040	0.245	0.010	0.242	0.003
ANN12	0.557	0.557	0.334	0.223	0.267	0.067	0.250	0.018	0.244	0.005	0.242	0.002
GA		**12**		**11**		**10**		**7**		**3**		**0**
Winner	**24.8%**		**24.1%**		**24.1%**		**24.1%**		**24.1%**		**24.1%**	
MMRE	**60.3%**		**35.6%**		**27.4%**		**25.1%**		**24.4%**		**24.2%**	

The behavior of the introduced GA criterion on all five used ISBSG dataset clusters is shown in Figure 35. It can be concluded that in each cluster, five or six executed iterations are enough to achieve the minimum MMRE value.

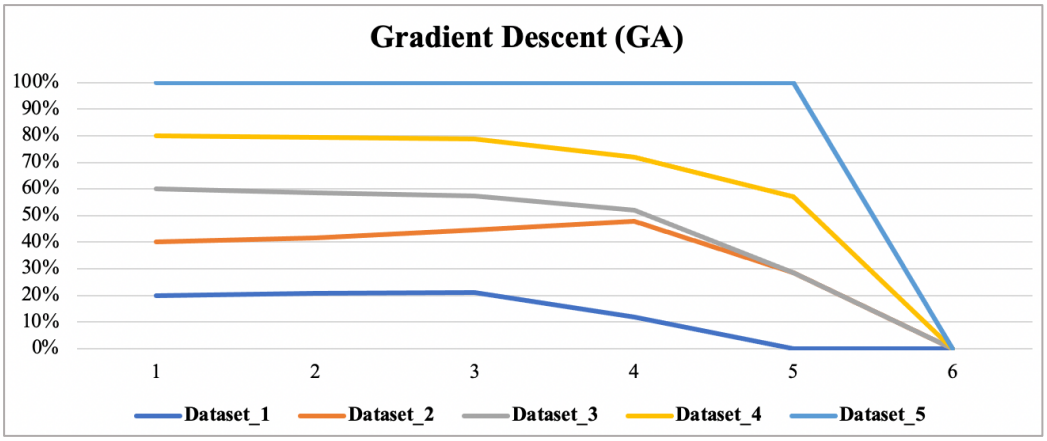


Figure 35. GA for all five clusters (ANN-L12).
Slika 35. GA vrednost za svih pet korišćenih klastera (ANN-L12).

The MMRE value for the "Winner" network in each cluster used is graphically shown in Figure 36. It can be concluded that the difference between the MMRE value in the first and last iteration is minimal and weighs a constant value.

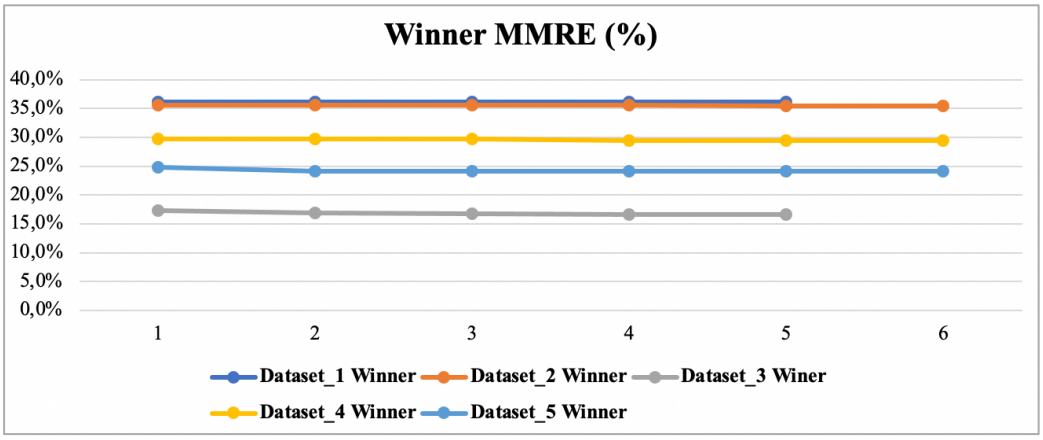


Figure 36. "Winner" MMRE for all five clusters (ANN-L12).
Slika 36. Vrednost MMRE za "Winner" mrežu za svih pet korišćenih klastera (ANN-L12).

The value of MMRE in each cluster used is graphically shown in Figure 37. Unlike the value of MMRE "Winner" network, the functions are exponentially shaped and tend towards the minimum MMRE.

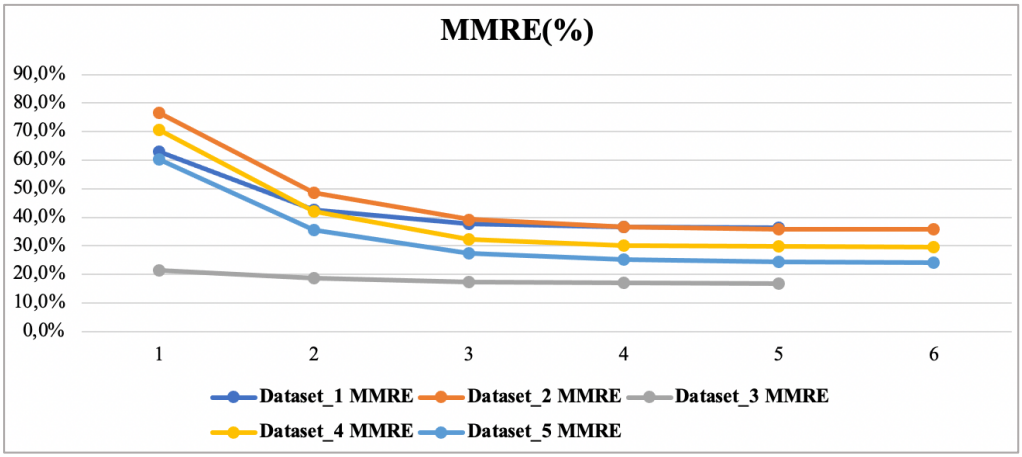


Figure 37. MMRE value for all five clusters (ANN-L12).
Slika 37. Vrednost MMRE za svih pet korišćenih klastera (ANN-L12).

Table 50. shows the results obtained by training the second proposed ANN-L36prim architecture on all five used ISBSG dataset clusters. The number of iterations in each cluster was monitored. The set GA criterion was met after six iterations for all five

clusters. A graphical representation of the GA criteria for all five clusters used is shown in Figure 38.

Table 50. GA for all five clusters (ANN-L36prim).
Tabela 50. Vrednost GA za svih pet korišćenih klastera (ANN-L36prim).

Gradient Descent (GA) for used datasets						
No. of Iteration	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**
Dataset_1	36	36	27	14	3	0
Dataset_2	36	35	23	17	5	0
Dataset_3	36	35	21	13	4	0
Dataset_4	36	35	25	15	7	0
Dataset_5	36	35	21	12	2	0

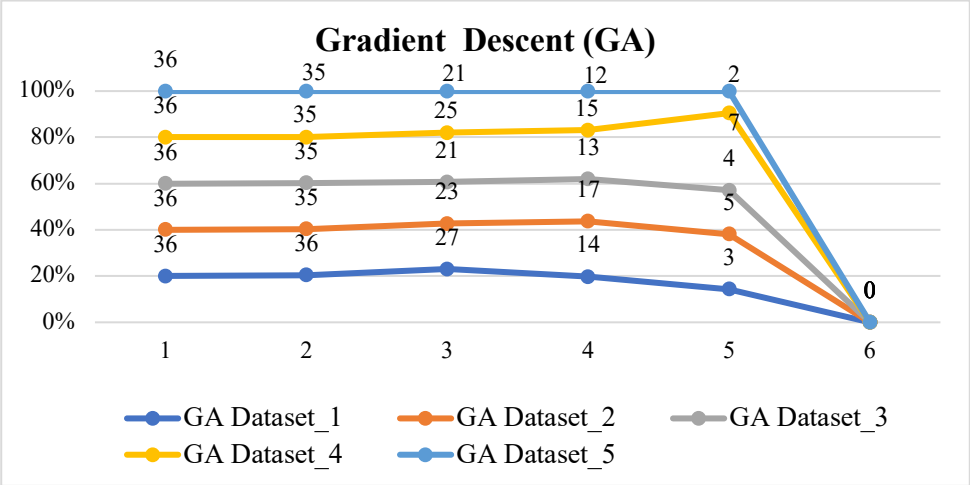


Figure 38. GA for all five clusters (ANN-L36prim).
Slika 38. Vrednost GA za svih pet korišćenih klaster (ANN-L36prim)

Table 51. shows the results obtained by training the second proposed ANN-L36prim architecture on the first cluster. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN1) is 32.1%, and the value of MMRE is 32.1%. The required GA criterion was met after six iterations.

Table 51. "Winner" MMRE vs. MMRE Dataset_1 (ANN-L36prim).
Tabela 51. Vrednost MMRE "Winner" mreže u odnosu na vrednost MMRE u Dataset_1 (ANN-L36prim).

	MMRE (%)					
No. of Iteration	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**
Dataset_1 Winner	34.8%	33.5%	32.8%	32.5%	32.1%	32.1%
Dataset_1	47.7%	37.6%	34.1%	33.1%	32.2%	32.1%

A graphical representation of the MMRE value for the Winner network relative to the MMRE value on Dataset_1 during six iterations is given in Figure 39.

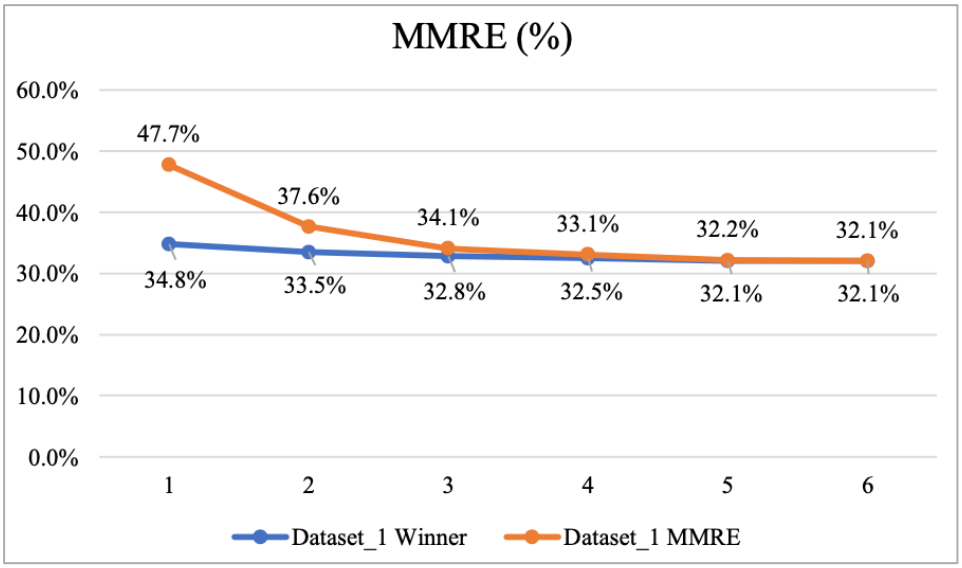


Figure 39. "Winner" MMRE vs. MMRE Dataset_1 (ANN-L36prim).
Slika 39. Grafička reprezentacija vrednosti MMRE "Winner" mreže u odnosu na vrednost MMRE u Dataset_1 (ANN-L36prim).

Table 52. shows the results obtained by training the second proposed ANN-L36prim architecture in the second cluster. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN36) is 32.0%, and the value of MMRE is 32.0%. The required GA criterion was met after six iterations.

Table 52. "Winner" MMRE vs. MMRE Dataset_2 (ANN-L36prim).
Tabela 52. Vrednost MMRE "Winner" mreže u odnosu na vrednost MMRE u Dataset_2 (ANN-L36prim).

	MMRE (%)					
No. of Iteration	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**
Dataset_2 Winner	35.2%	34.1%	33.8%	32.7%	32.2%	32.0%
Dataset_2	49.7%	38.6%	36.9%	35.0%	32.3%	32.0%

A graphical representation of the MMRE value for the Winner network relative to the MMRE value on Dataset_2 during six iterations is given in Figure 40.

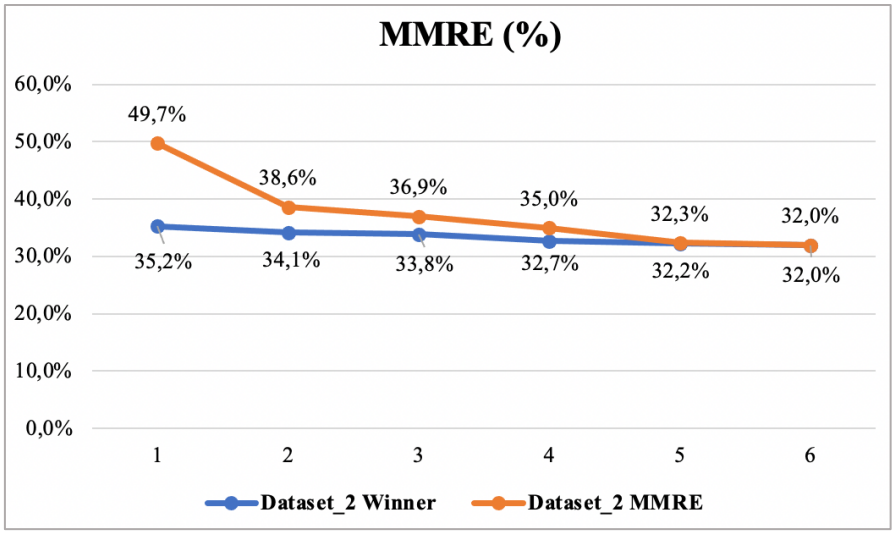


Figure 40. "Winner" MMRE vs. MMRE Dataset_2 (ANN-L36prim)
Slika 40. Grafička reprezentacija vrednosti MMRE "Winner" mreže u odnosu na vrednost MMRE u Dataset_1 (ANN-L36prim).

Table 53. shows the results obtained by training the second proposed architecture ANN-L36prim on the third cluster. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN12) is 16.4%, and the value of MMRE is 16.4%. The required GA criterion was met after six iterations.

Table 53. "Winner" MMRE vs. MMRE Dataset_3 (ANN-L36prim).
Tabela 53. Vrednost MMRE "Winner" mreže u odnosu na vrednost MMRE u Dataset_3 (ANN-L36prim).

	MMRE (%)					
No. of Iteration	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**
Dataset_3 Winner	16.7%	16.6%	16.5%	16.7%	16.4%	16.4%
Dataset_3	20.6%	18.5%	17.3%	16.9%	16.5%	16.4%

Graphical representation of the MMRE value for the "Winner" network relative to the MMRE value on Dataset_3 during six iterations is given in Figure 41.

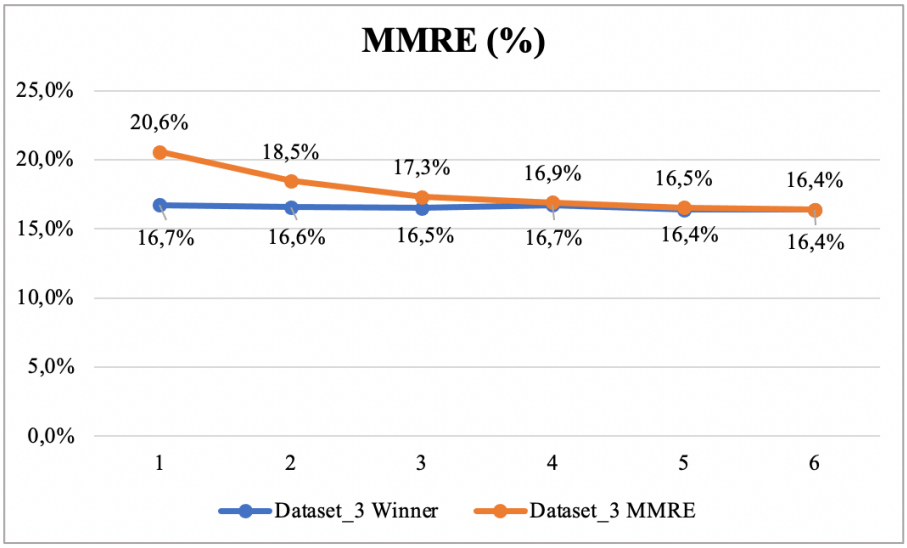


Figure 41. "Winner" MMRE vs. MMRE Dataset_3 (ANN-L36prim).
Slika 41. Grafička reprezentacija vrednosti MMRE "Winner" mreže u odnosu na vrednost MMRE u Dataset_3 (ANN-L36prim).

Table 54. shows the results obtained by training the second proposed architecture ANN-L36prim on the fourth cluster. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN5) is 29.7%, and the value of MMRE is 29.7%. The required GA criterion was met after six iterations.

Table 54. "Winner" MMRE vs. MMRE Dataset_4 (ANN-L36prim).
Tabela 54. Vrednost MMRE "Winner" mreže u odnosu na vrednost MMRE u Dataset_4 (ANN-L36prim).

	MMRE (%)					
No. of Iteration	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**
Dataset_4 Winner	37.5%	33.2%	32.1%	31.5%	30.4%	29.7%
Dataset_4	51.1%	42.7%	35.7%	32.2%	31.1%	29.7%

Graphical representation of MMRE value for "Winner" network relative to MMRE value on Dataset_4 during six iterations is given in Figure 42.

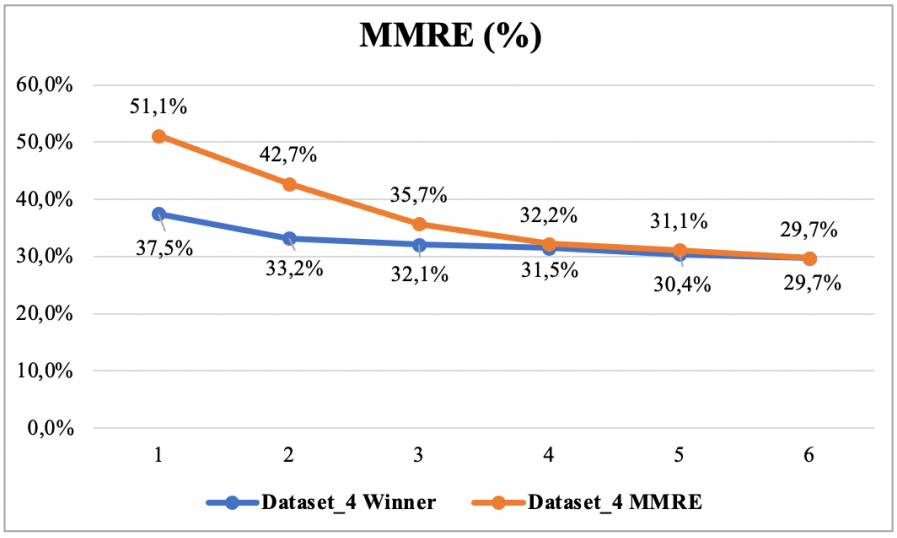


Figure 42. "Winner" MMRE vs. MMRE Dataset_4 (ANN-L36prim).
Slika 42. Grafička reprezentacija vrednosti MMRE "Winner" mreže u odnosu na vrednost MMRE u Dataset_4 (ANN-L36prim).

Table 55. shows the results obtained by training the second proposed architecture ANN-L36prim in the fourth cluster. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN5) is 19.0%, and the value of MMRE is 19.1%. The required GA criterion was met after six iterations.

Table 55. "Winner" MMRE vs. MMRE Dataset_5 (ANN-L36prim).
Tabela 55. Vrednost MMRE "Winner" mreže u odnosu na vrednost MMRE u Dataset_5 (ANN-L36prim).

	MMRE (%)					
No. of Iteration	**1.**	**2.**	**3.**	**4.**	**5.**	**6.**
Dataset_5 Winner	33.6%	29.4%	26.4%	22.5%	20.3%	19.0%
Dataset_5	40.3%	34.5%	29.8%	27.5%	22.4%	19.1%

Graphical representation of MMRE value for "Winner" network relative to MMRE value on Dataset_5 during six iterations is given in Figure 43.

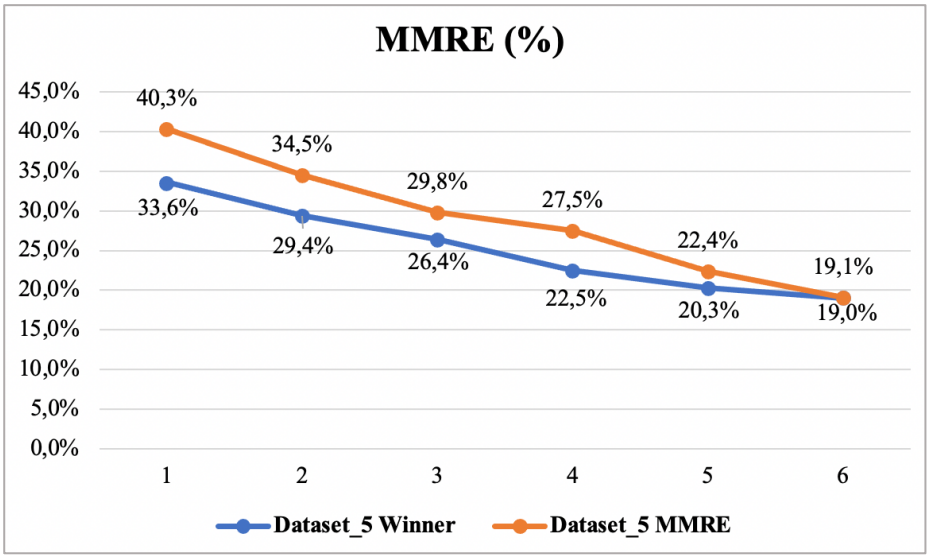


Figure 43. "Winner" MMRE vs. MMRE Dataset_4 (ANN-L36prim).
Slika 43. Grafička reprezentacija vrednosti MMRE "Winner" mreže u odnosu na vrednost MMRE u Dataset_5 (ANN-L36prim).

In this approach, the first proposed ANN-L12 architecture is with one hidden layer and two nodes, and then the second proposed ANN-L36prim architecture is with also one hidden layer and three nodes. The number of hidden layers indicates reducing the number of iterations to achieve the stop criteria. Choice of ANN architecture with one hidden layer did not significantly increase the number of factors. In the ANN-L12 architecture, this number is 11, and in the ANN-L36prim is 16, which was acceptable in COSMIC FFP approach, while experiments showed that the MMRE value was reduced by 0.9%. It is not necessary to introduce more complex architectures because this did not require more time for processing than other ANN architectures experimented with.

The result that was monitored during the execution of the required iterations is the value of MMRE, which, depending on the nature of the dataset, proved to be stable for the two proposed architectures in all three parts of the experiment.

Datasets with a small functional size have a higher value of MMRE (Dataset_1, Dataset_2) than datasets with a medium value of functional size (Dataset_3). Datasets with a large functional size value have a medium MMRE value (Dataset_4, Dataset_5). Validation datasets have different values of functional magnitude and have a higher MMRE value. The lowest value of MMRE is achieved on Dataset_3 for the proposed ANN-L12 architecture (17.0%), while for ANN-L36prim the value of MMRE is 16.4%. The mean MMRE value on all datasets in all three parts of the experiment was 29.7% for ANN-L12 and 28.8% for ANN-L36prim, see Figure 44, Table 56. Compared to the COCOMO approach, the value of MMRE was significantly reduced by 14.5%, see Table 57.

Table 56. MMRE value for each proposed ANN architecture (COSMIC FFP).
Tabela 56. Vrednost MMRE za svaku predloženu arhitekturu (COSMIC FFP).

Datasets	ANN-L12 MMRE(%)	ANN-L36prim MMRE(%)	Part of experiment
Dataset_1	36.1	36.2	Training
	36.7	35.6	Testing
Dataset_2	33.2	32.0	Training
	36.6	33.8	Testing
Dataset_3	17.0	16.4	Training
	18.1	17.2	Testing
Dataset_4	29.6	29.7	Training
	29.8	29.6	Testing
Dataset_5	24.2	19.0	Training
	27.0	24.8	Testing
Dataset_6	34.9	39.1	Validation1
Dataset_7	33.2	31.8	Validation2
AVERAGE(MMRE)	29.7	28.8	

Table 57. MMRE values COCOMO2000 vs. COSMIC FFP.
Tabela 57. Vrednosti MMRE u poređenju COCOMO2000 i COSMIC FFP.

COCOMO model			Functional point - COSMIC model	
ANN L-18 MMRE (%)	ANN L-27 MMRE (%)	ANN L-36 MMRE (%)	ANN L-12 MMRE (%)	ANN L-36prim MMRE (%)
59.7	45.3	43.3	29.7	28.8

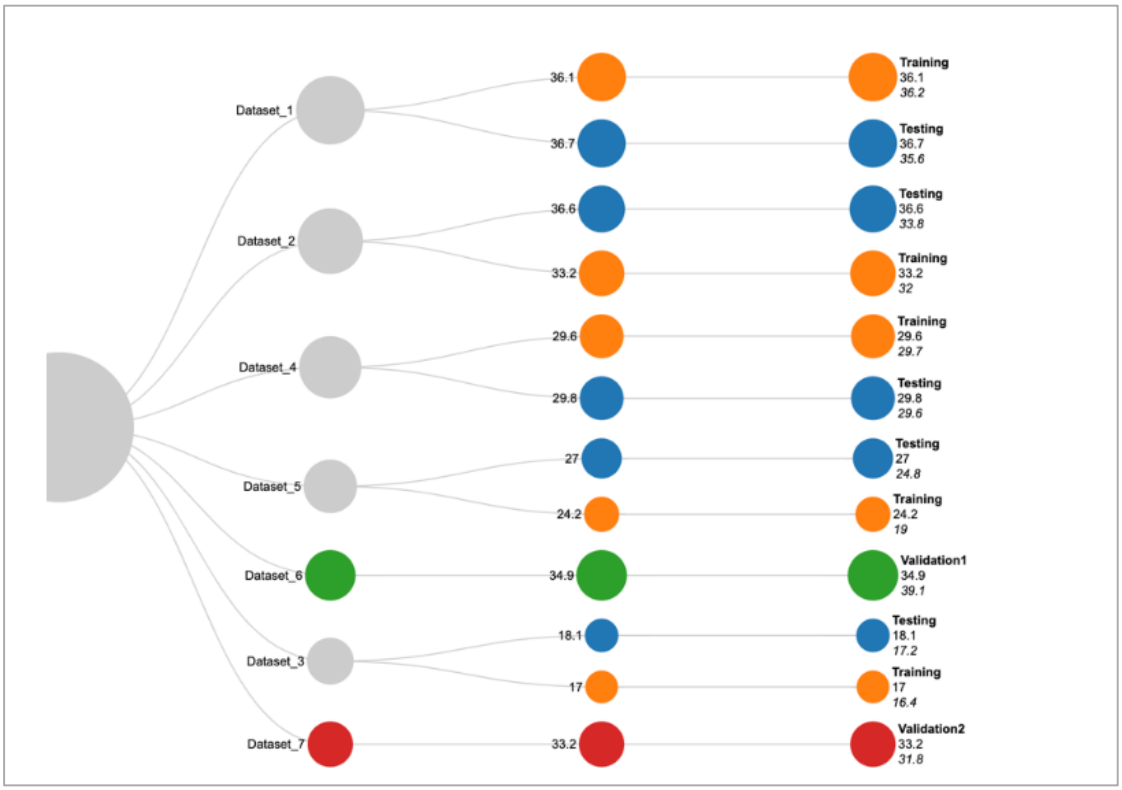


Figure 44. Graphical representation of MMRE value for each proposed ANN architecture (COSMIC FFP).

Slika 44. Grafička reprezentacija vrednosti MMRE svake predložene arhitekture (COSMIC FFP).

In addition to examining MMRE value, the convergence rate on all five datasets for the two ANN architectures was considered. It can be concluded that both architectures ANN-L12 and ANN-L36prim converge the fastest in Dataset_5 and the slowest in Dataset_, see Figure 45, Figure 46.

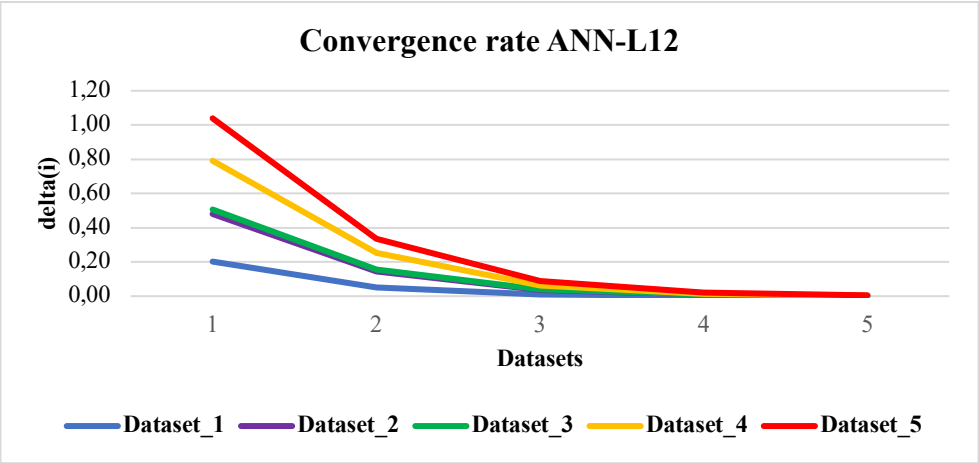


Figure 45. Convergence rate ANN-L12 (COSMIC FFP).
Slika 45. Brzina konvergencije ANN-L12 (COSMIC FFP).

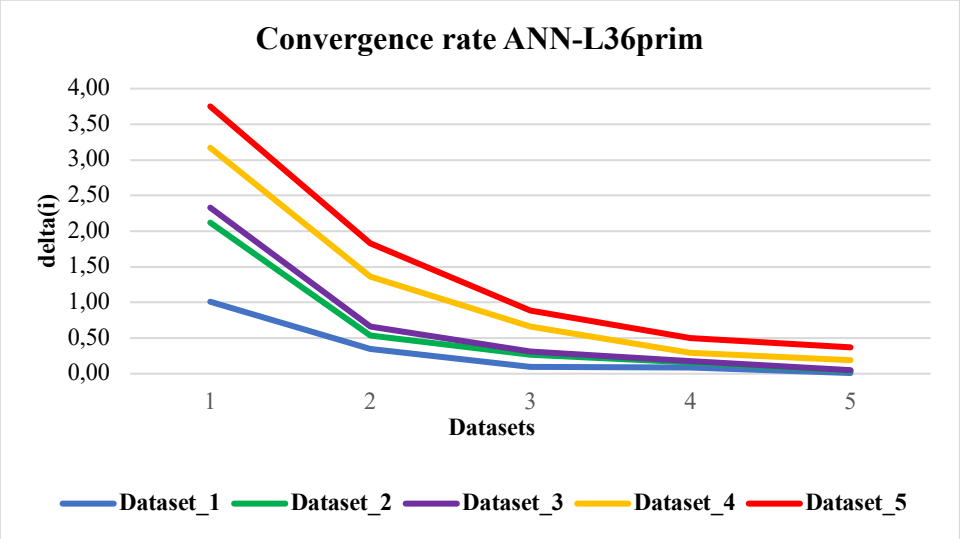


Figure 46. Convergence rate ANN-L36prim (COSMIC FFP).
Slika 46. Brzina konvergencije ANN-L36prim (COSMIC FFP).

The higher the value of the correlation coefficient, the more stable the relationship between the observed values. By calculating the correlation coefficient between the estimated and actual value, the reliability, accuracy, and precision of the proposed approach can be determined. Correlation coefficients, calculated according to Pearson and Spearman, indicate that the correlation is very high. The best result, based on the experiment performed, is achieved on Dataset_2 and Dataset_4, while the worst outcome is achieved on Dataset_6 and Dataset_7, see Table 58.

Table 58. Correlation coefficients (COSMIC FFP).
Tabela 58. Korelacioni koeficijenti (COSMIC FFP).

Correlation	Pearson		Spearman's rho	
Datasets	ANN-L12	ANN-L36prim	ANN-L12	ANN-L36prim
Dataset_1	0.859	0.756	0.884	0.785
Dataset_2	0.984	0.788	0.984	0.808
Dataset_3	0.734	0.688	0.745	0.678
Dataset_4	0.912	0.911	0.918	0.917
Dataset_5	0.767	0.618	0.715	0.918
Dataset_6	0.628	0.586	0.612	0.534
Dataset_7	0.674	0.622	0.654	0.618

By monitoring the prediction at three different values: 25%, 30%, and 50%, it is possible to conclude the appropriate degree of certainty of the proposed approach. This allows the best model of the presented ANN architectures to be selected. If the prediction value is high, the proposed model is good. In our approach, the highest prediction value of the 25% criterion is achieved on the Dataset_3 for ANN-L36prim architecture (84.1%). The highest prediction value of the 30% criterion is conducted on the Dataset_3 for ANN-L36prim architecture (86.4%). The prediction value of the 50% criterion is achieved again for the ANN-L36prim architecture but within the Dataset_1 (98.1%). We can show that in our work, prediction represents the percentage of projects that have an MRE value of less than 25%, 30, and 50%, respectively, see Table 59.

Table 59. Prediction measured on three criteria (COSMIC FFP).
Tabela 59. Merenje predikcije na sva tri predložena kriterijuma (COSMIC FFP).

PRED (%)	P(25)		P(30)		P(50)	
Datasets	ANN-L12	ANN-L36prim	ANN-L12	ANN-L36prim	ANN-L12	ANN-L36prim
Dataset_1	39.6	37.7	49.1	52.8	83.0	98.1
Dataset_2	33.3	31.7	41.3	44.4	69.8	82.5
Dataset_3	79.5	84.1	86.4	86.4	95.5	97.7
Dataset_4	42.3	42.3	51.3	51.3	78.2	80.8
Dataset_5	50.0	59.1	54.5	63.6	81.8	86.4
Dataset_6	28.6	28.6	28.6	35.7	78.6	78.6
Dataset_7	30.3	30.3	42.4	57.6	81.8	84.8

In addition to the presented criterions for monitoring the efficiency, effectiveness, and precision of the proposed approach, the influence of the input values of the COSMIC FFP method on the change in the MMRE value was also monitored. By omitting each value for the four input values (Entry, Exit, Read, and Write), the change in MMRE versus mean (MMRE) for the proposed architecture on each selected dataset was monitored. In

this way, it is possible to determine the impact of each input value on the final estimated value of the project.

When examining the influence of these input values for the ANN-L12 architecture, it can be concluded that the most significant influence has the second input value (Exit) on Dataset_1 (5.3%) and the smallest first input value (Entry) on Dataset_6 (-1.1%). It can be concluded that the Exit value on Dataset_1 increases the MMRE value by 5.3%, while Entry value decreases the MMRE value by 1.1%. The influence of all four input values on all seven datasets ranges in the interval [- 1.1, 5.3]. A value of 0.0% means that the input value does not affect the change in the MMRE value, see Figure 47, Table 60.

Table 60. Influence of input values on MMRE value (δ(%)) - ANN-L12 architecture.
Tabela 60. Uticaj ulaznih veličina na vrednost MMRE (δ(%)) - ANN-L12 arhitektura.

δ(%)	Dataset_1	Dataset_2	Dataset_3	Dataset_4	Dataset_5	Dataset_6	Dataset_7
Entry	3.3	3.7	0.2	0.0	0.3	-1.1	0.5
Exit	5.3	1.3	0.2	0.7	1.4	0.9	0.5
Read	4.1	-0.7	0.2	0.6	0.0	2.3	0.0
Write	0.6	1.3	-0.1	0.3	0.0	0.8	0.0

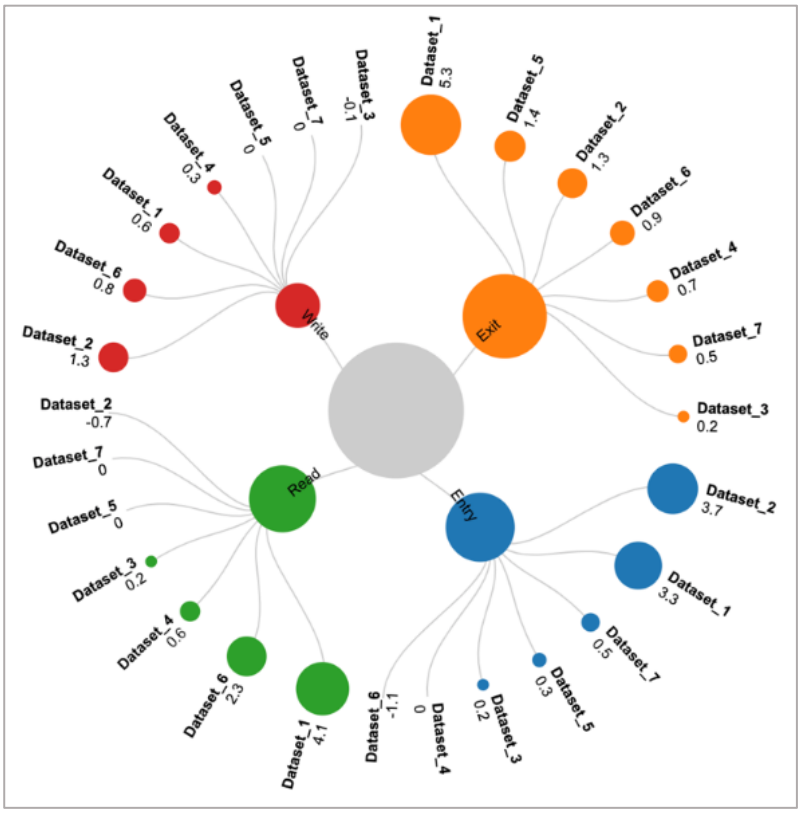


Figure 47. Graphical representation of the influence of input values on MMRE value (ANN-L12 COSMIC FFP).

Slika 47. Grafička reprezentacija uticaja ulaznih veličina na vrednost MMRE (ANN-L12 COSMIC FFP).

When examining the influence of the input values for ANN-L36prim architecture, it can be concluded that the first input value has the most significant influence (Entry) on Dataset_2 (3.7%). In comparison, the most negligible impact has the fourth input value (Write) in Dataset_7 (-2.0%). It can be concluded that the Entry value on Dataset_2 increases the MMRE value by 3.7%, while the Write value decreases the MMRE value by 2.0%. The influence of all four input values individually on all seven datasets take values from the interval [-2.0, 3.7], see Figure 48, Table 61.

Table 61. Influence of input values on MMRE value (δ(%)) - ANN-L36prim architecture.
Tabela 61. Uticaj ulaznih veličina na vrednost MMRE (δ(%)) - ANN-L36prim arhitektura.

δ(%)	Dataset_1	Dataset_2	Dataset_3	Dataset_4	Dataset_5	Dataset_6	Dataset_7
Entry	3.4	3.7	0.7	0.9	1.3	0.9	0.8
Exit	2.2	0.7	0.1	0.8	0.9	1.2	-0.6
Read	0.6	-0.7	0.1	1.2	1.6	1	-1.6
Write	0.4	1.1	0.4	1	0.2	1.7	-2.0

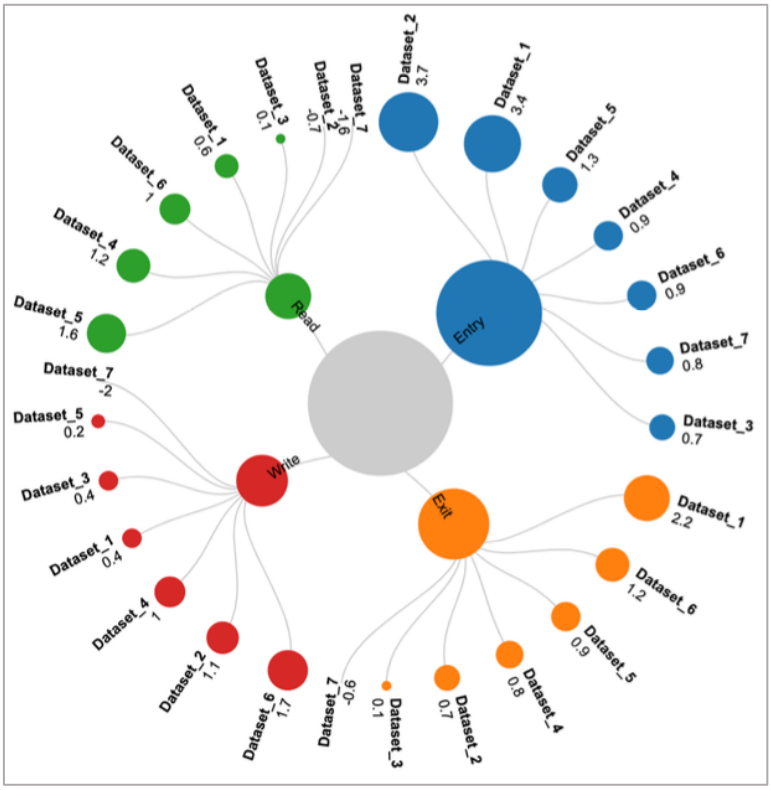


Figure 48. Graphical representation of the influence of input values on MMRE value (ANN-L36prim COSMIC FFP).
Slika 48. Grafička reprezentacija uticaja ulaznih veličina na vrednost MMRE (ANN-L36prim COSMIC FFP).

The COSMIC FFP measure defines a model of software functionality. By monitoring the influence of input values (Entry, Exit, Read, Write), the value of MMRE can be further improved depending on the user's needs, i.e. functional requirements, depending on the software being developed. By examining the individual effects of four different input values, it can be concluded that the most significant changes come from Entry and Exit input values. Therefore, it is necessary to re-perform the analysis and

additional testing of these two input values to establish if everything in their requirements is necessary. Entry value represents user messages to the system. It is desirable and possible to redesign this size and thus write it in a more suitable form to reduce its impact on the value of MMRE. The Exit value represents the messages that the system returns in response and can be read from files and can result from some logical, arithmetic and other mathematical operations. This size can be redesigned and modified to reduce its impact on MMRE. With these modifications and possible changes that will be adjusted to the needs of the users of the system, it is possible to increase the efficiency, accuracy, stability, and reliability of the proposed approach, and thus the success of the completion of the software project. The models and tools are currently consolidated to estimate software project development efforts based on functional points that express the functional size of that program. Recent research and many companies use the COSMIC FFP method because it is possible to assess the development effort in the initial stage compared to previously used FPA methods.

The obtained results of the used COSMIC FFP method show that it is possible with our approach and applied methodology to estimate the functional size quite accurately. Remarkably, the results found tend to suggest that in the presence of a set of projects, concerning the application domain, the nature of computation performed, and the implementation technology, it's possible to get more accurate estimates of the functional size (expressed either in FP or in CFP) on the premise of several base functional components (BFC) [12]. Before presenting the requirements specification in detail, it is possible to make an early if there is not enough time to apply some other standard methods.

The results shown in the previous tables and figures of this approach (COSMIC FFP) were processed in the R programming language (avaiable at: https://www.r-project.org/) and checked in the Python programming language (avaiable at: https://www.python.org/) within the RStudio environment (avaiable at: https://www.rstudio.com/products/rstudio/). The data required for statistical analysis were processed in the IBM SPSS Statistical 25 software tool (avaiable at: https://www.ibm.com/support/pages/downloading-ibm-spss-statistics-25).

4.3 Obtained results using UCP model and ANN

With the UCP model, it is possible to measure the size of the system similar as with the model of functional points. This model represents system characteristics and uses cases to estimate effort and cost needed for realization of software project. UCP is one of the most commonly used model due to the exceptional evaluation results that can be

achieved. The disadvantage of this model is that it does not consider the data structure in the system because such data are not contained in the use cases.

Table 62. shows the results obtained by training the first proposed ANN-L16 architecture on the used dataset. The number of iterations concerning the required GA criterion was monitored. The GA criterion was met after four iterations. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN6) is 6.7%, and the value of MMRE is 7.1%.

In addition to examining the MMRE value, the convergence rate on all training data for the two ANN architectures was examined. It can be concluded that the ANN-L36prim architecture quickly converges to the minimum knowledge of MMRE compared to the ANN-L16 architecture, see Figure 49.

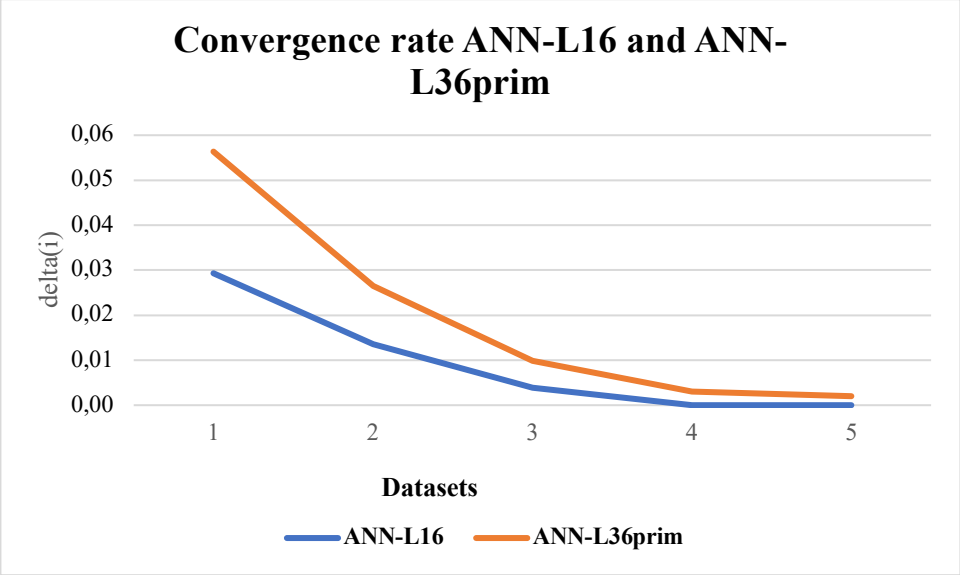


Figure 49. Convergence rate ANN-L16 vs. ANN-L36prim (UCP).
Slika 49. Brzina konvergencije ANN-L16 u odnosu na ANN-L36prim (UCP).

Table 62. ANN-L16 results of training part.
Tabela 62. Rezultati treniranja ANN-L16.

No. of Iter.	1.		2.		3.		4.	
ANN-L16	MRE	GA	MRE	GA	MRE	GA	MRE	GA
ANN1	0.084	0.084	0.080	0.004	0.076	0.004	0.072	0.004
ANN2	0.196	0.196	0.112	0.084	0.082	0.030	0.073	0.009
ANN3	0.188	0.188	0.113	0.076	0.081	0.031	0.072	0.009
ANN4	0.085	0.085	0.077	0.008	0.074	0.003	0.072	0.003
ANN5	0.161	0.161	0.105	0.056	0.080	0.025	0.073	0.008
ANN6	0.069	0.069	0.068	0.002	0.067	0.000	0.067	0.000
ANN7	0.078	0.078	0.073	0.006	0.071	0.002	0.070	0.001
ANN8	0.151	0.151	0.105	0.046	0.081	0.024	0.073	0.008
ANN9	0.191	0.191	0.120	0.071	0.076	0.044	0.072	0.004
ANN10	0.073	0.073	0.080	0.007	0.074	0.006	0.073	0.001
ANN11	0.078	0.078	0.080	0.001	0.075	0.005	0.072	0.003
ANN12	0.130	0.130	0.084	0.047	0.074	0.010	0.070	0.003
ANN13	0.113	0.113	0.083	0.030	0.074	0.009	0.071	0.002
ANN14	0.094	0.094	0.080	0.014	0.072	0.008	0.071	0.002
ANN15	0.094	0.094	0.080	0.015	0.073	0.007	0.071	0.002
ANN16	0.102	0.102	0.082	0.021	0.074	0.008	0.071	0.002
GA		**16**		**10**		**5**		**0**
Winner	**6.9%**		**6.8%**		**6.7%**		**6.7%**	
MMRE	**11.8%**		**8.9%**		**7.5%**		**7.1%**	

A graphical representation of GA values during four iterations is shown in Figure 50.

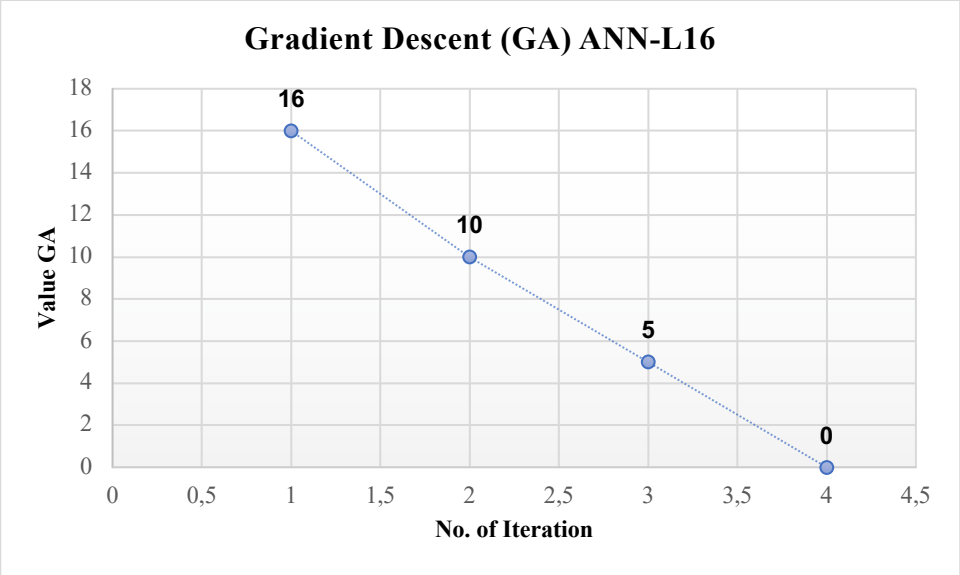


Figure 50. GA for ANN-L16 - training part (UCP).
Slika 50. Vrednost GA za ANN-L16 - treniranje (UCP).

A graphical representation of the MRE value for the "Winner" network relative to the MMRE value on the training dataset during the four iterations is given in Figure 51.

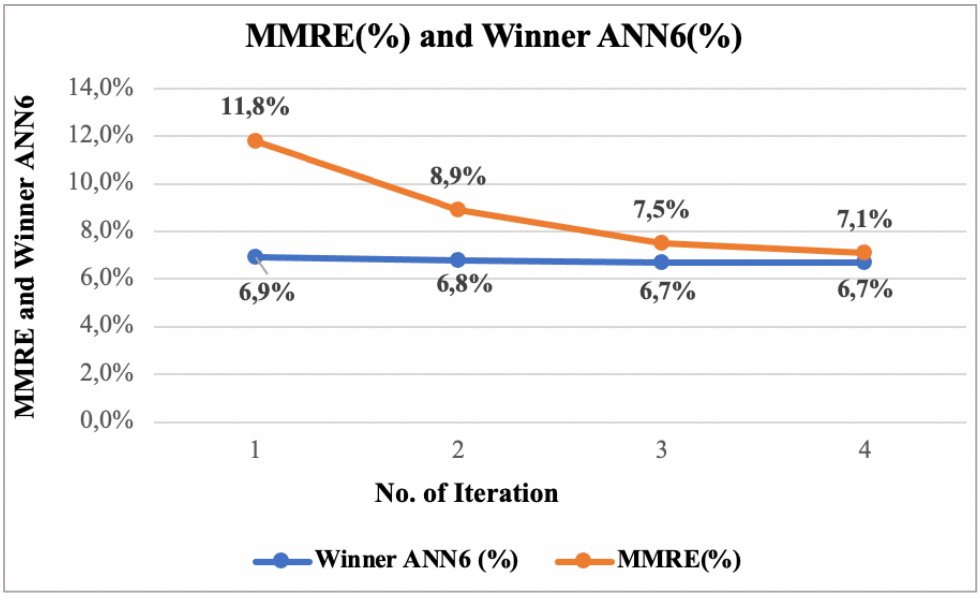


Figure 51. "Winner" MRE vs. MMRE on the training dataset (ANN-L16).
Slika 51. Vrednost MMRE "Winner" mreže u odnosu na vrednost MMRE na skupu podataka za treniranje (ANN-L16).

Table 63. shows the results obtained by training the second proposed ANN-36prim architecture on the used dataset. The number of iterations concerning the set GA criterion

was monitored. The GA criterion was met after six iterations. Based on all MRE values in each executed iteration, the "Winner" network is determined, i.e., the ANN network with the lowest MRE value. Additionally, the MMRE value was calculated for each iteration. The obtained value of the "Winner" network (ANN10) is 6.9%, and the value of MMRE is 7.0%.

Table 63. ANN-L36prim results of training part.
Tabela 63. Rezultati treniranja ANN-L36prim.

ANN-L36prim						
GA	36	35	23	14	3	0
Winner	7.3%	7.2%	7.1%	7.0%	7.0%	6.9%
MMRE	12.1%	9.4%	8.4%	7.5%	7.2%	7.0%

A graphical representation of GA values during six iterations is shown in Figure 52.

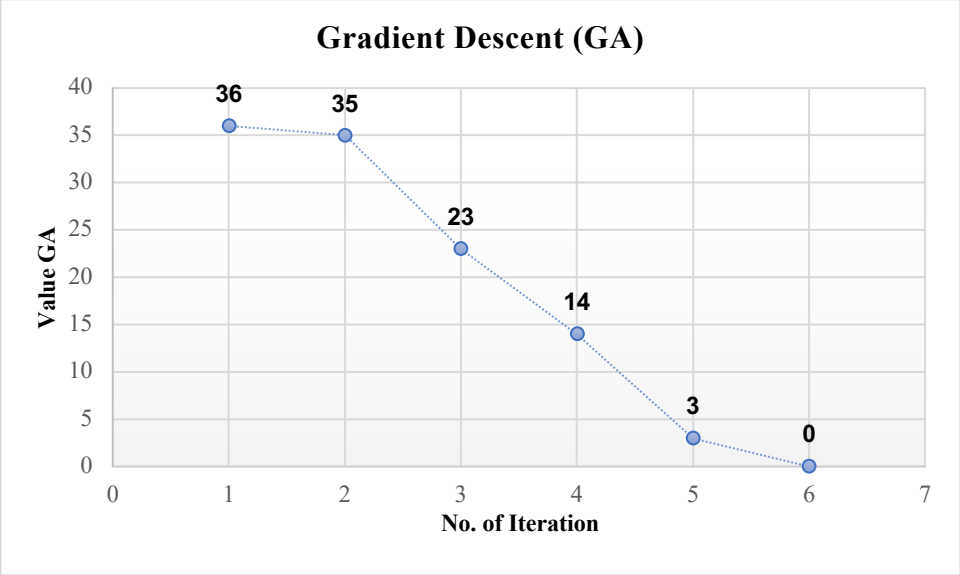


Figure 52. GA for ANN-L36prim - training part (UCP).
Slika 52. Vrednost GA za ANN-L36prim - treniranje (UCP).

A graphical representation of the MRE value for the "Winner" network relative to the MMRE value on the training dataset during six iterations is given in Figure 53.

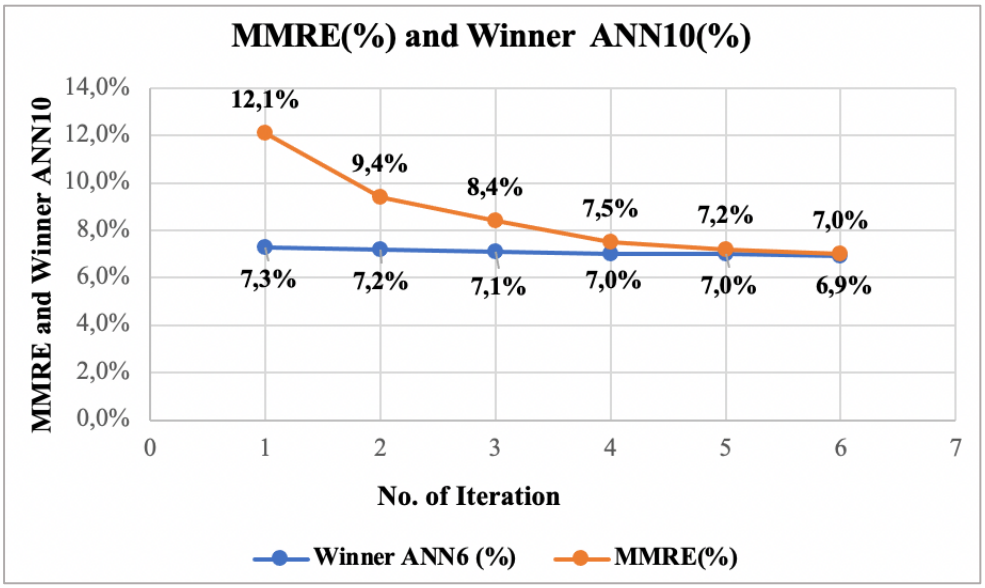


Figure 53. "Winner" MRE vs. MMRE on the training dataset (ANN-L36prim).
Slika 53. Vrednost MMRE "Winner" mreže u odnosu na vrednost MMRE na skupu podataka za treniranje (ANN-L36prim).

The obtained results for the two proposed architectures, ANN-L16 and ANN-L36, in all three parts of the experiment showed that the different nature of the data set does not affect the complexity of the architecture used. Furthermore, it does not depend on the value of the input values.

In the first proposed architecture, ANN-L16, all six input values were used (where four are linearly dependent and two linearly independent), and the MMRE value in all three parts of the experiment is 7.5%, see Table 64.

Using the second architecture ANN-L36prim with four independent input values, the same MMRE value was obtained in all three parts of the experiment i.e. 7.5%, see Table 64.

The error differences in individual parts of the experiment are not more than 0.5%, indicating the proposed model's reliability.

Table 64. MMRE value in all three parts of the experiment (UCP).
Tabela 64. Vrednost MMRE u sva tri dela eksperimenta (UCP).

Datasets	ANN-L16	ANN-L36prim	Part of experiment
	MMRE(%)	MMRE(%)	
Dataset_1	6.7	7.0	Training
	7.1	7.1	Testing
Dataset_2	8.0	7.5	Validation1
Dataset_3	8.3	8.4	Validation2
AVERAGE(MMRE)	7.5	7.5	

The huge values of the correlation coefficients (Pearson's and Spearman's rho) further show the consistency of the actual and estimated values obtained using the proposed models. In the ANN-L36prim architecture, Pearson's value is 0.983, which indicates an exceptional interrelationship between the observed values, see Table 65.

Table 65. Correlation coefficients (UCP).
Tabela 65. Korelacioni koeficijenti (UCP).

Correlation	ANN-L16	ANN-L36prim
Pearson's	0.875	0.983
Spearman's rho	0.784	0.962

Prediction represents the number of projects that have an error less than the value set by criterion. Prediction can further confirm the validity and reliability of the models used. For all three proposed criteria: PRED (25), PRED (30), and PRED (50), and in all three parts of the experiment: training, testing and validation, using both proposed architectures, the value is 100%, see Table 66.

Table 66. Prediction values (UCP).
Tabela 66. Vrednosti predikcije (UCP).

Training		
PRED(%)	ANN-L16(%)	ANN-L36prim(%)
PRED(25)	100.0	100.0
PRED(30)	100.0	100.0
PRED(50)	100.0	100.0
Testing		
PRED(25)	100.0	100.0
PRED(30)	100.0	100.0
PRED(50)	100.0	100.0
Validation1		
PRED(25)	100.0	100.0
PRED(30)	100.0	100.0
PRED(50)	100.0	100.0
Validation2		
PRED(25)	100.0	100.0
PRED(30)	100.0	100.0
PRED(50)	100.0	100.0

By examining the influence of dependent and independent variables on the change of MMRE value, it was shown that it is sufficient to use a four-dimensional vector instead of a six-dimensional vector. The error with dependent input values on the four datasets used is between -0.3 to 0.5, which is less than 1%. The most significant influence has the input value of AUCP (Dataset_3), and the change in the value of MMRE is, in that case, is increased by 0.5%. The slightest influence has the input value of UUCW (Dataset_4), and the change in the value of MMRE is, in this case, is decreased by 0.5%, which would mean that the error can be reduced/increased if the observed values are further analyzed.

It can be concluded that the architecture with six input sizes can be replaced with the architecture with four input sizes. That is, in the observed approach, the ANN-L16 architecture can be replaced with the ANN-L36prim architecture, see Table 67.

Table 67. Influence of the input values on the MMRE change (UCP).
Tabela 67. Uticaj ulaznih veličina na promenu vrednosti MMRE (UCP).

Dataset	MMRE	UAW	UUCW	UUCP	TCF	ECF	AUCP
Dataset_1	6.7%	7.1%	6.7%	7.0%	6.7%	6.7%	7.1%
Dataset_2	7.0%	7.1%	7.0%	7.2%	6.9%	7.1%	7.2%
Dataset_3	8.0%	7.9%	8.1%	7.9%	8.1%	8.1%	7.5%
Dataset_4	8.3%	7.9%	8.4%	7.9%	8.3%	8.2%	8.0%

From Table 68, it can be concluded that the dependent variable UUCP has less impact than the dependent variable AUCP. The most significant influence of AUCP (Dataset_3) on the change in MMRE value is 0.5%. The slightest influence of AUCP (Dataset_1) on the change in MMRE value is -0.3%.

Table 68. Influence of dependent variables (UUCP and AUCP) on the change of MMRE value.
Tabela 68. Uticaj zavisnih promenljivih (UUCP and AUCP) na promenu vrednosti MMRE.

Dataset	UUCP	g-UUCP=MMRE-UUCP	AUCP	g-AUCP=MMRE-AUCP
Dataset_1	6.9%	-0.1%	6.8%	-0.3%
Dataset_2	7.1%	-0.1%	7.1%	-0.1%
Dataset_3	8.0%	0.1%	8.0%	0.5%
Dataset_4	8.2%	0.2%	8.2%	0.2%
	max	0.2%	**max**	0.5%
	min	-0.1%	**min**	-0.3%

A graphical representation of the dependent input values of UUCP and AUCP with the values of their errors is given in Figure 54.

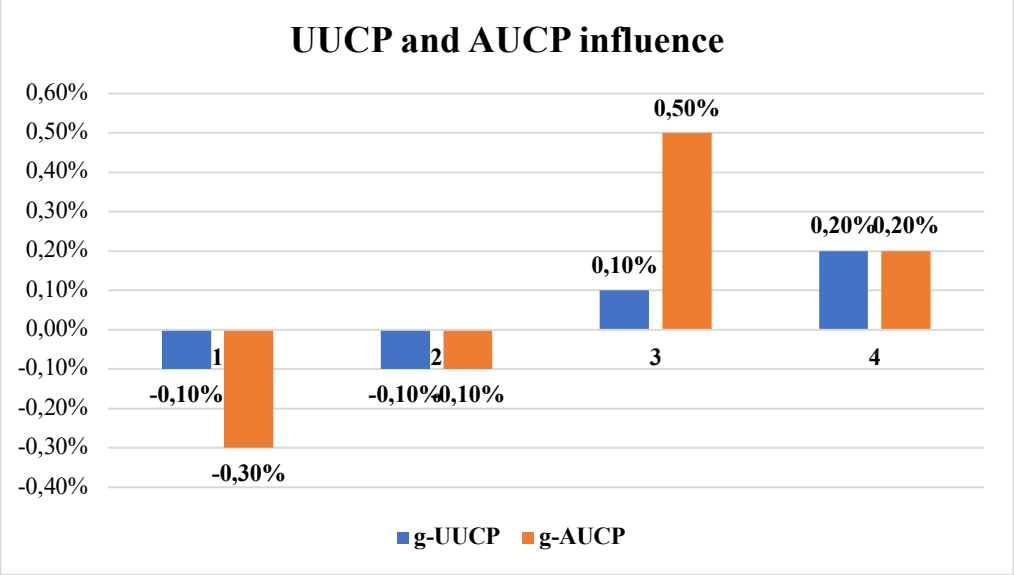


Figure 54. Influence of dependent variables (UUCP and AUCP) on the change of MMRE value.
Slika 54. Uticaj zavisnih promenljivih (UUCP and AUCP) na promenu vrednosti MMRE.

The results shown in the previous tables and figures for this approach (UCP) were processed in the R programming language (avaiable at: https://www.r-project.org/) and checked in the Python programming language (avaiable at: https://www.python.org/) within the RStudio environment (avaiable at: https://www.rstudio.com/products/rstudio/). The data required for statistical analysis were processed in the IBM SPSS Statistical 25 software tool (avaiable at: https://www.ibm.com/support/pages/downloading-ibm-spss-statistics-25).

4.4 Analysis of three proposed models

Comparative analysis of the parametric COCOMO2000 method with the improved COCOMO2000 by use of ANN leads us to the conclusion that reduction of the model error is 193.1/43.3=4.5 times. In the second proposed approach, comparing the parametric method COCOMO2000 and the improved COSMIC FFP with ANN, the model error reduction is 193.1/28.8=6.7 times. Compared to the parametric COCOMO2000 method with UCP and ANN, the model error reduction is 193.1/7.5=25.7 times Table 69, Figure 55.

In the first proposed approach, the lowest model error value is 43.3% for ANN-L36 architecture. In the second proposed approach, the lowest error value is achieved with ANN-L36prim and it is 28.8%. In the third proposed approach, both proposed architectures ANN-L16 and ANN-L36prim give the lowest model error value of 7.5%, see Table 69, Figure 55.

It can be concluded that the third proposed UCP approach achieves the lowest MMRE value. In addition, the ANN-L16 architecture in this approach quickly converges and reaches the "stop criterion" after the 4th iteration, which is also the lowest number of iterations performed concerning all used architectures in all three proposed approaches.

The influence of dependent variables on the change of MMRE value in the ANN-L16 architecture is less than 0.5%. It can be concluded that the improved UCP model using the ANN-L16 architecture is the best-proposed estimator of effort and cost for software project development.

Table 69. MMRE values for used approaches.
Tabela 69. Vrednosti MMRE u korišćenim pristupima.

MMRE(%)	**COCOMO2000 and ANN**				**COSMIC FFP and ANN**		**UCP and ANN**	
COCOMO2000	ANN-L9	ANN-L18	ANN-L27	ANN-L36	ANN-L12	ANN-L36prim	ANN-L16	ANN-L36prim
193.1%	72.0%	59.7%	45.3%	43.3%	29.7%	28.8%	7.5%	7.5%

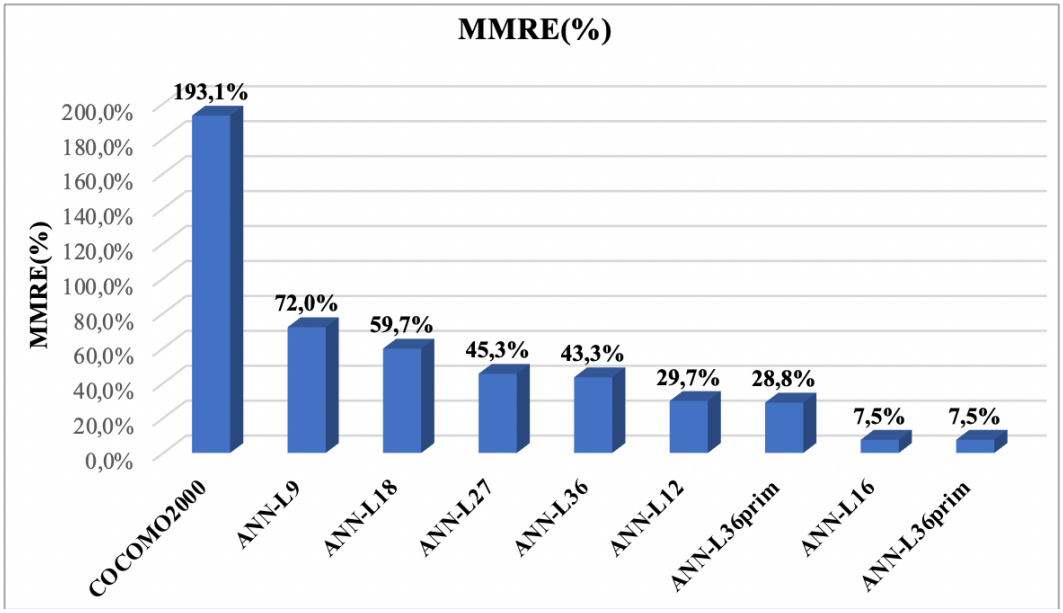


Figure 55. MMRE value for used approaches.
Slika 55. Vrednosti MMRE u korišćenim pristupima.

By selecting the best ANN architectures, which achieved the lowest MMRE value for each of the three proposed, improved models, it can be concluded that: COSMIC FFP with ANN is 43.3/28.8=1.5 times better than COCOMO2000 with ANN; UCP with ANN is 48.8/7.5=5.8 times better than COCOMO2000 with ANN; UCP with ANN is 28.8/7.5=3.8 times better than COSMIC FFP with ANN, see Table 70.

Table 70. COCOMO2000 and ANN vs. COSMIC FFP and ANN vs. UCP and ANN.
Tabela 70. Poređenje vrednosti MMRE za COCOMO2000 i ANN, COSMIC FFP i ANN i UCP i ANN.

	COCOMO2000 and ANN	**COSMIC FFP and ANN**	**UCP and ANN**	
MMRE(%)	ANN-L36	ANN-L36prim	ANN-L16	ANN-L36prim
	43.3%	28.8%	7.5%	7.5%

Chapter 5: The problem of generalization

During the software product development, various data are collected and used. By applying different machine learning tools, numerous analyses are performed, and specific conclusions are drawn based on which important decisions related to project implementation success are made [114], [115]. A huge number of tasks in the estimation process can be presented as machine learning problems. When it comes to artificial neural networks, the training procedure provides a large number of possibilities for solving the presented problem. It is essential to experiment with sufficiently available realistic projects to evaluate the new proposed solutions and projects accurately [119].

The information environment is constantly evolving and needs to be adjusted quickly and efficiently process of effort and cost estimation. Problems that remain insufficiently defined during ANN estimation are a division of the dataset into clusters, the optimal number of projects within each dataset, and various algorithms to check the size of the cluster concerning the number of weighting coefficients by various statistical analyses.

In addition to experiments with the three proposed new models and several different ANN architectures, it is necessary to experiment with new methods to achieve further improvements of cost and effort estimation. These are experimenting with: selecting other activation functions and encoding/decoding methods, monitoring and checking the influence of input variables on the change of relative error, and many others. Of course, the proposed models have more advantages than disadvantages comparing to similar traditional methods. However, each parameter that affects the estimation must be additionally analyzed and checked several times and applied in experiments in different ways.

5.1 Number of projects in the data set

Depending on the proposed approaches, the training process on different ANN architectures also involves use of specific datasets. The choice of appropriate datasets depends on the model and input sizes, as well as publicly available databases.

In the first proposed approach, for the ANN training procedure, the COCOMO2000 dataset was used. The same dataset was used for the ANN testing process on other projects, and the validation process used COCOMO81, NASA60, and Kemerer datasets. On each selected dataset, the number of projects needs to be greater than the number of weighting factors of the proposed ANN architecture.

For ANN-L9, ANN-L18 and ANN-L27 architectures, the set requirement for the minimum number of projects is met on all data sets used, see Table 71.

In the ANN-L36 architecture, which has 23 weighting coefficients, the number of testing projects was 20, so the required condition is not met (Dataset_2). In the ANN-L36 architecture, which has 23 weighting coefficients, the number of projects for the third validation was 15, so the required condition is not met (Dataset_5).

It can be concluded that in all parts of the experiment within the first proposed approach, the fulfillment of the conditions is **90%**, see Table 71.

Table 71. The number of projects greater than the number of weighting factors (COCOMO2000).

Tabela 71. Broj projekata veći od broja težinskih koeficijenata (COCOMO2000).

Datasets		**No. of projects**	**Experiment**	**Weighting factors (coefficients)**			
				ANN-L9 (4)	ANN-L18 (8)	ANN-L27 (13)	ANN-L36 (23)
Dataset_1	COCOMO2000	100	Training	+	+	+	+
Dataset_2	COCOMO2000	20	Testing	+	+	+	/
Dataset_3	COCOMO81	51	Validation1	+	+	+	+
Dataset_4	NASA	60	Validation2	+	+	+	+
Dataset_5	Kemerer	15	Validation3	+	+	+	/

In the second proposed approach, the ISBSG dataset divided into five selected clusters was used for the ANN training process. The same dataset in the five selected clusters was used for the ANN testing procedure. The number of projects in both procedures on each selected cluster is 70:30; 70 projects were used for training, and 30 projects for testing. Desharnais dataset and dataset combined from different, realistic industrial projects were used for the validation process. On each selected dataset, the number of projects needs to be greater than the number of weighting factors of the proposed ANN architecture.

In the ANN-L12 architecture, which has 11 weighting coefficients, only the number of projects to be tested was 7, so the required condition is not met for one data set (Dataset_5).

With the ANN-L36prim architecture, which has 16 weighting coefficients, the number of testing projects is 15, so the set condition is not met (Dataset_1). The number of projects for testing was 13 (Dataset_3), so within the third cluster, the required condition is not met. The number of training and testing projects (Dataset_5) was 14, 7, respectively. Within the Desharnais dataset (Dataset_6), the number of projects was 14,

and the set condition is not fulfilled. The number of data sets that fulfill the criteria was 7 of 12, see Table 72.

It can be concluded that in all parts of the experiment under the second proposed approach, the fulfillment of the conditions is **75%**, see Table 72.

Table 72. The number of projects greater than the number of weighting factors (COSMIC FFP).

Tabela 72. Broj projekata veći od broja težinskih koeficijenata (COSMIC FFP).

	Datasets	**Number of project**	**Experiment**	**Weighting factors (coefficients)**	
				ANN-L12 (11)	ANN-L36prim (16)
Dataset_1	ISBSG (Functional Size<10)	37	Training	+	+
	ISBSG (Functional Size<10)	15	Testing	+	/
Dataset_2	ISBSG (10<Functional Size<50)	45	Training	+	+
	ISBSG (10<Functional Size<50)	17	Testing	+	+
Dataset_3	ISBSG (50<Functional Size<100)	30	Training	+	+
	ISBSG (10<Functional Size<100)	13	Testing	+	/
Dataset_4	ISBSG (100<Functional Size<500)	60	Training	+	+
	ISBSG (10<Functional Size<500)	17	Testing	+	+
Datset_5	ISBSG (Functional Size>500)	14	Training	+	/
	ISBSG (Functional Size>500)	7	Testing	/	/
Dataset_6	Desharnais	14	Validation1	+	/
Dataset_7	Combined	33	Validation2	+	+

In the third proposed approach, for the ANN training process, the Benchmark (Mendeley) dataset divided into a 70:30 scale, 70% of projects for the training process, and 30% for the testing process were used. Combined real, industrial projects (Dataset_3, Dataset_4) were used for the validation procedure. On each selected dataset, the number of projects needs to be greater than the number of weighting factors of the proposed ANN architecture.

In the ANN-L16 architecture, which has 15 weighting coefficients, the condition is met that the number of projects for all parts of the experiments is greater than the number of weighting coefficients. With the ANN-L36prim architecture, which has 16 weighting coefficients, the condition is met that the number of projects for all parts of the experiment is greater than the number of weighting coefficients.

It can be concluded that in all parts of the experiment within the third proposed approach, the fulfillment of the conditions is **100%**, see Table 73.

Table 73. The number of projects greater than the number of weighting factors (UCP).
Tabela 73. Broj projekata veći od broja težinskih koeficijenata (UCP).

Dataset		Number of projects	Experiment	Weighting factors (coefficients)	
				ANN-L16 (15)	ANN-L36prim (16)
Dataset_1	UCP Benchmark Dataset	50	Training	+	+
Dataset_2	UCP Benchmark Dataset	21	Testing	+	+
Dataset_3	Combined	18	Validation1	+	+
Dataset_4	Combined Industrial projects	17	Validation2	+	+

5.2 Comparative analysis of proposed models with SVM algorithm

In order to confirm the correctness and reliability of the proposed approach and its comparison with other artificial intelligence tools, the Support Vector Machine (SVM) algorithm was used. SVM is a popular machine-learning (ML) algorithm and stems from the use of observed data for training. The SVM is a robust and proficient technique for both classification and regression. In addition, it minimizes the expected error, thus reducing the problem of overfitting [131], [132], [133]. The SVM machine algorithm divides the plane by the function f_{SVM} into two parts, so that the points (project input values) lie above or below the function f_{SVM}. Three different functions inside a kernel in SVM with radial basis function (RBF) were used:

- linear kernel function,
- quadratic kernel function,
- cubic kernel function.

The obtained results show that the estimated value in the training part of the experiment for ANN-L27 and ANN-L36 architectures, based on three input variables of E, PEM_i, and KLOC, have a very high degree of correlation (deterministic coefficient-R^2). Graphical representations of the actual and estimated values using the SVM algorithm and the corresponding kernel functions for both architectures are shown in Figure 56, Figure 57, and Figure 58.

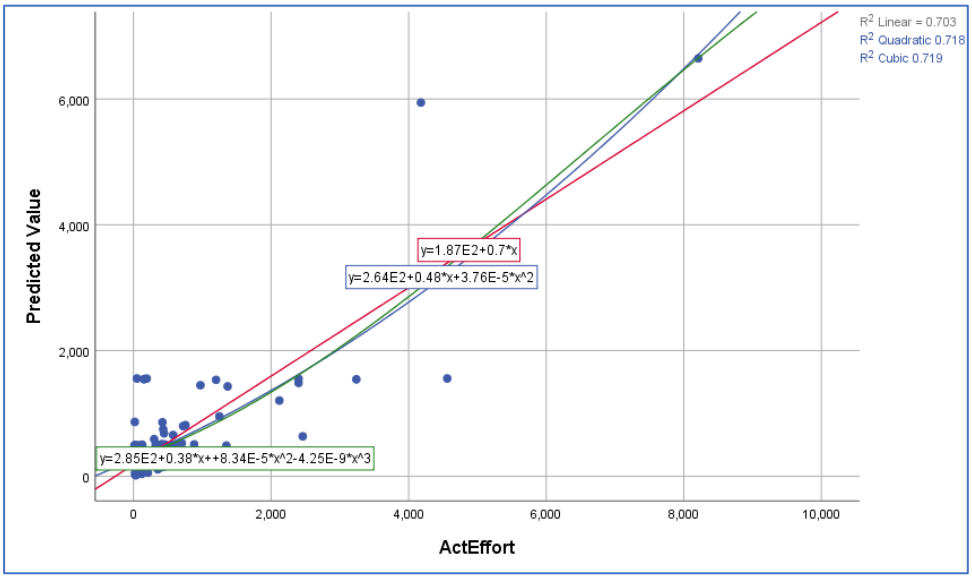


Figure 56. Graphical representation using different kernel functions based on SVM (RBF) for ActEffort on the training dataset (COCOMO2000).

Slika 56. Grafička reprezentacija ActEffort korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) u skupu podataka za treniranje.

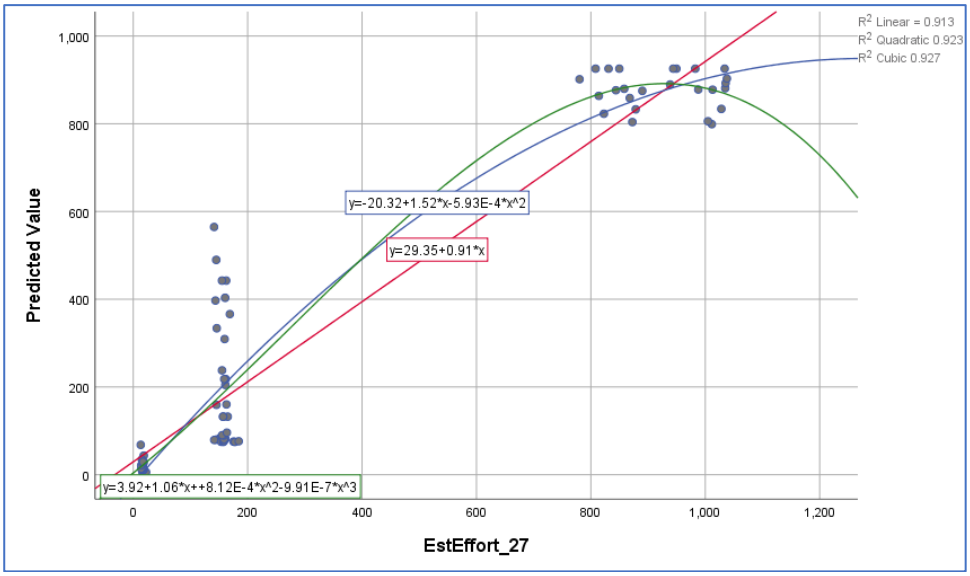


Figure 57. Graphical representation using different kernel functions based on SVM (RBF) for EstEffort on the training dataset (ANN-L27).

Slika 57. Grafička reprezentacija EstEffort korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) u skupu podataka za treniranje (ANN-L27).

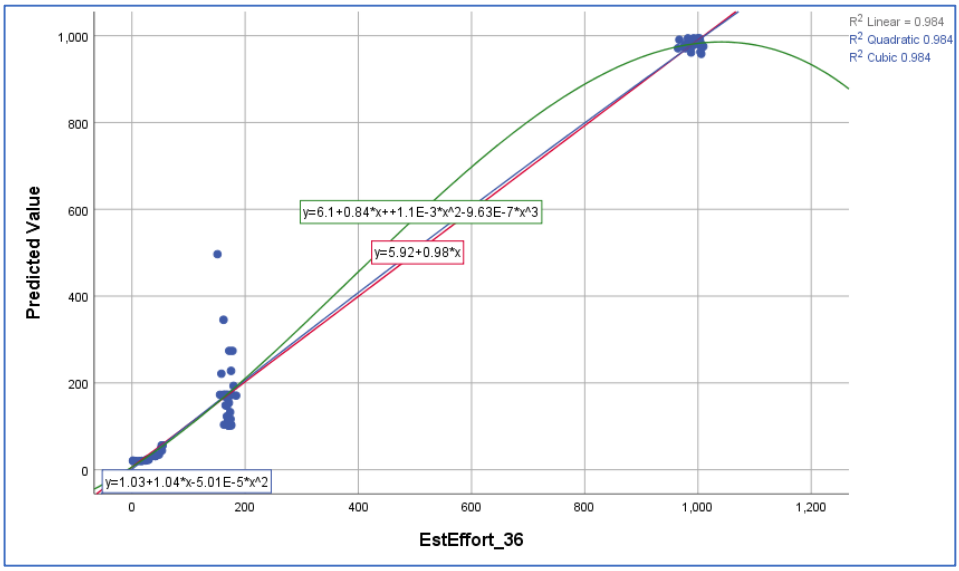


Figure 58. Graphical representation using different kernel functions based on SVM (RBF) for EstEffort on the training dataset (ANN-L36).

Slika 58. Grafička reprezentacija EstEffort korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) u skupu podataka za treniranje (ANN-L36).

The MMRE value for all three kernel functions used in the SVM algorithm is higher (60% for ANN-L27 architecture and 56.3% for ANN-L36 architecture) than the proposed approach using ANN and Taguchi Orthogonal Arrays, see Table 74. For ANN-L27 architecture using Taguchi Orthogonal Arrays and the COCOMO2000 model the MMRE value is 45.3%, and for ANN-L36 is 43.3%.

Additionally, the estimated value obtained using the improved COCOMO2000 model was confirmed by the high value of the deterministic coefficient - R^2, using three different kernel functions. This once again confirmed that the datasets were adequately divided into clusters.

Table 74. R^2 values using different kernel functions based on SVM (RBF) - COCOMO2000.

Tabela 74. Vrednosti R^2 korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) - COCOMO2000.

	Act Effort	EstEffort_27	EstEffort_36
R^2 Linear	0.703	0.913	0.984
R^2 Quadratic	0.718	0.923	0.984
R^2 Cubic	0.719	0.927	0.984
MMRE (%)		60.0	56.3

The obtained results show that the estimated value in the training part of the experiment for ANN-L12 and ANN-L36prim architectures, based on four input variables of Entry, Exit, Read, and Write, have a very high degree of correlation (deterministic coefficient - R^2). Graphical representations of the actual and estimated values using the SVM algorithm and the corresponding kernel functions for both architectures are shown in Figure 59, Figure 60, and Figure 61.

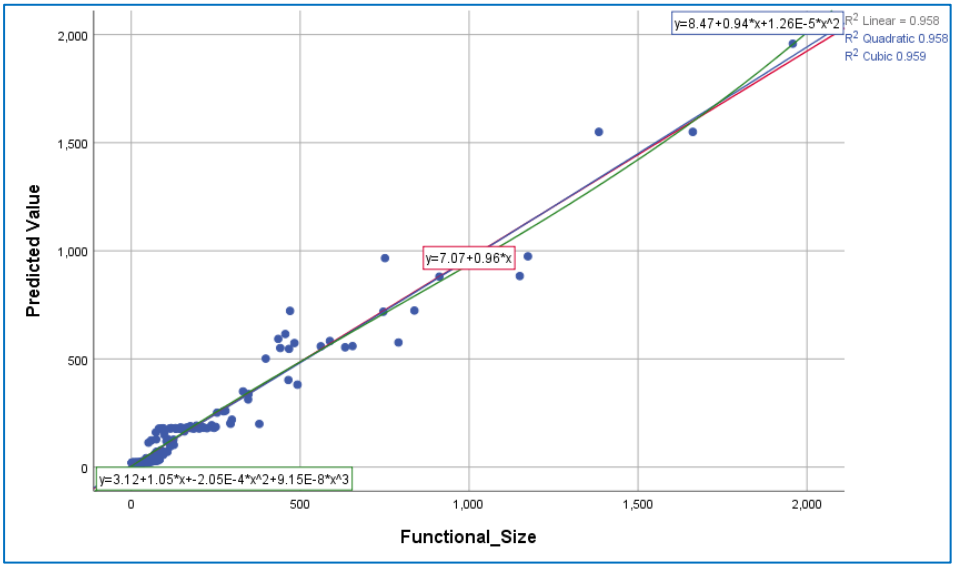


Figure 59. Graphical representation using different kernel functions based on SVM (RBF) for Functional Size on the training dataset (COSMIC FFP).

Slika 59. Grafička reprezentacija funkcionalnih tačaka korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) u skupu podataka za treniranje (COSMIC FFP).

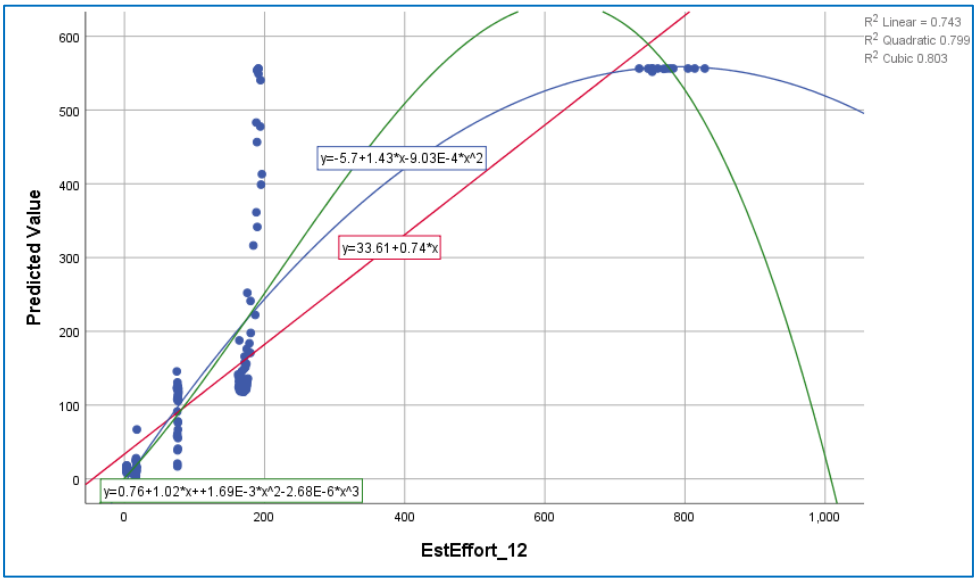


Figure 60. Graphical representation using different kernel functions based on SVM (RBF) for EstEffort on the training dataset (ANN-L12).

Slika 60. Grafička reprezentacija EstEffort korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) u skupu podataka za treniranje (ANN-L12).

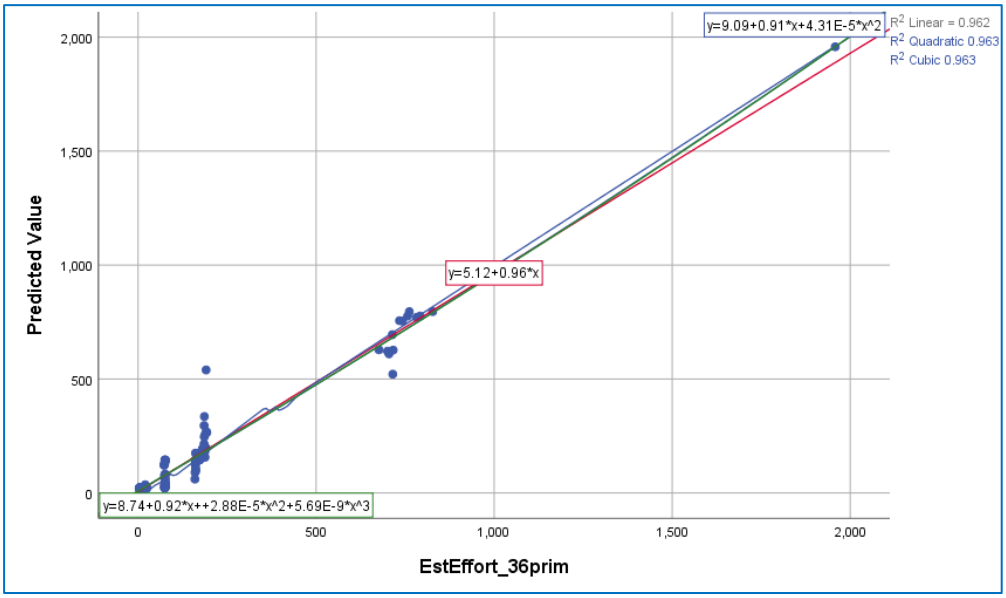


Figure 61. Graphical representation using different kernel functions based on SVM (RBF) for EstEffort on the training dataset (ANN-L36prim COSMIC FFP).

Slika 61. Grafička reprezentacija EstEffort korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) u skupu podataka za treniranje (ANN-L36prim COSMIC FFP).

The MMRE value for all three kernel functions used in the SVM algorithm is higher (38.1% for ANN-L12 architecture and 35.7% for ANN-L36prim architecture) than the proposed approach using ANN and Taguchi Orthogonal Arrays, see Table 75. For ANN-L12 architecture using Taguchi Orthogonal Arrays and the COSMIC FFP model the MMRE value is 29.7%, and for ANN-L36prim is 28.8%.

Additionally, the estimated value obtained using the improved COSMIC FFP model was again confirmed by the high value of the deterministic coefficient - R^2, using three different kernel functions. This once again confirmed that the datasets were adequately divided into clusters.

Table 75. R^2 values using different kernel functions based on SVM (RBF) - COSMIC FFP.

Tabela 75. Vrednosti R^2 korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) - COSMIC FFP.

	Act Effort	EstEffort_12	EstEffort_36prim
R^2 Linear	0.958	0.743	0.962
R^2 Quadratic	0.958	0.799	0.963
R^2 Cubic	0.959	0.803	0.963
MMRE (%)		38.1	35.7%

The obtained results show that the estimated value in the training part of the experiment for ANN-L16 and ANN-L36prim architectures, based on four input variables of UAW, UUCW, TCF, ECF, UUCP, and AUCP have a very high degree of correlation (deterministic coefficient - R^2). Graphical representations of the actual and estimated values using the SVM algorithm and the corresponding kernel functions for both architectures are shown in Figure 62, Figure 63, and Figure 64.

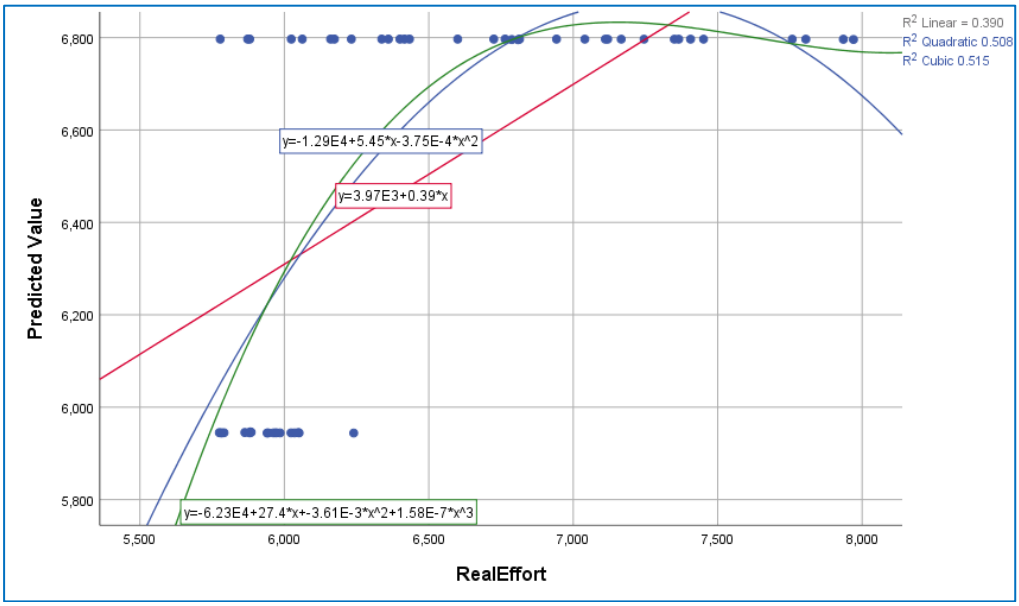


Figure 62. Graphical representation using different kernel functions based on SVM (RBF) for Real Effort on the training dataset (UCP).

Slika 62. Grafička reprezentacija stvarnog napora korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) u skupu podataka za treniranje (UCP).

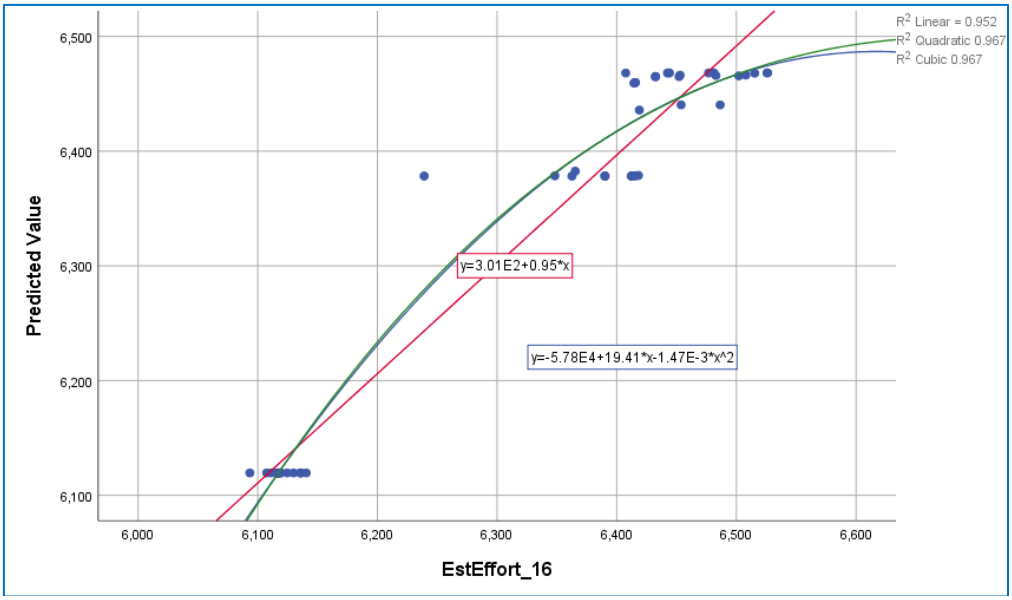


Figure 63. Graphical representation using different kernel functions based on SVM (RBF) for EstEffort on the training dataset (ANN-L16).

Slika 63. Grafička reprezentacija EstEffort korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) u skupu podataka za treniranje (ANN-L16).

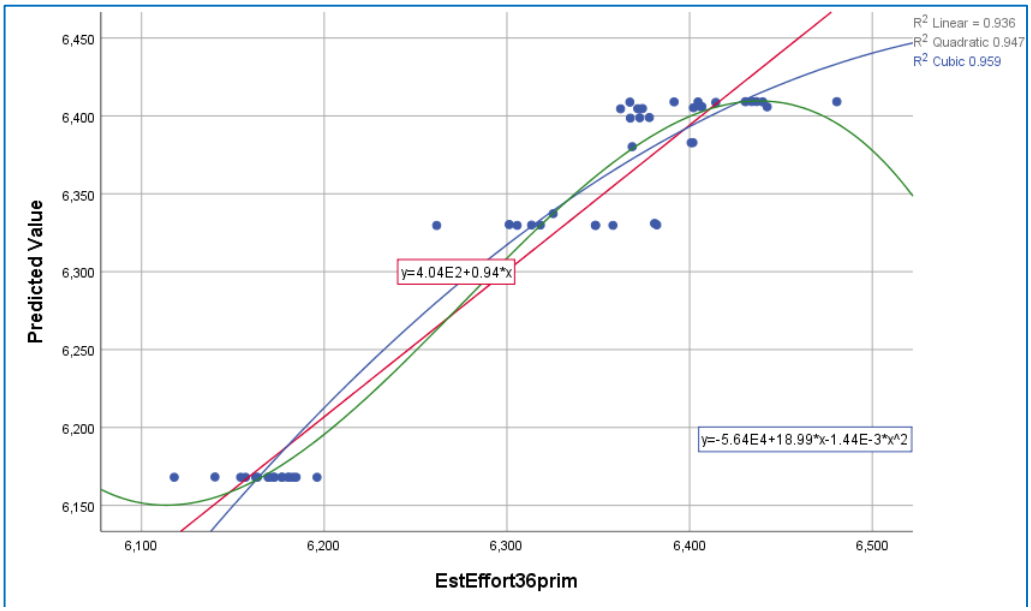


Figure 64. Graphical representation using different kernel functions based on SVM (RBF) for EstEffort on the training dataset (ANN-L36prim UCP).
Slika 64. Grafička reprezentacija EstEffort korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) u skupu podataka za treniranje (ANN-L36prim UCP).

The MMRE value for all three kernel functions used in the SVM algorithm is higher (13.2% for ANN-L16 architecture and 13.5% for ANN-L36prim architecture) than the proposed approach using ANN and Taguchi Orthogonal Arrays, see Table 76. For ANN-L16 architecture using Taguchi Orthogonal Arrays and the UCP FFP model the MMRE value is 7.5%, and for ANN-L36prim is 7.5%.

Additionally, the estimated value obtained using the improved UCP model was once again confirmed by the high value of the deterministic coefficient - R^2, using three different kernel functions. This once again confirmed that the datasets were adequately divided into clusters.

Table 76. R^2 values using different kernel functions based on SVM (RBF) - UCP.
Tabela 76. Vrednosti R^2 korišćenjem različitih funkcija jezgra zasnovana na SVM (RBF) - UCP.

	Act Effort	EstEffort_16	EstEffort_36prim
R^2 Linear	0.390	0.952	0.936
R^2 Quadratic	0.508	0.967	0.947
R^2 Cubic	0.515	0.967	0.959
MMRE (%)		13.2%	13.5%

Based on the obtained results, it can be concluded that our proposed approach is stable, reliable, and efficient even when compared by other machine-learning tools like SVM algorithm.

5.3 Construction and complexity of ANN architecture

One of the goals in all three proposed models is to select the simplest ANN architecture that meets the additionally set criteria:

- minimum number of iterations performed, reduced estimation time,
- the simplest ANN architecture that converges the fastest,
- minimum MMRE value,
- use of clustering and fuzzification methods and sigmoid function as activation function,
- stop criterion GA to stop the number of iterations.

The choice of architecture depends on the number of input values, the number of weight coefficients, and the corresponding orthogonal vector plan. Each proposed model has an architecture constructed based on two or three levels or a combined orthogonal vector plan. Experiments always use the simplest possible architecture, then add nodes to the hidden layer in and finally increase the number of hidden layers.

From Table 77, it can be concluded that the number of input values in the COCOMO2000 model is three, in the COSMIC FFP is four, while in the UCP model, it is six or four. The ANN-L9 architecture has no hidden layer but is constructed based on the orthogonal vector plan L9. Only the ANN-L36 architecture has two hidden layers: in the first hidden layer it has three nodes, while in the second hidden layer it has two nodes. All other used ANN architectures have one hidden layer. The number of nodes in the hidden layer is equal to zero in ANN-L9, 5 in ANN-L36, while in all other cases, it is equal to two or three. For each of the three proposed models, one output value is calculated, which represents the estimated value.

Table 77. Characteristics of ANN architecture.
Tabela 77. Karakteristike ANN arhitekture.

	COCOMO2000				COSMIC FFP		UCP	
Architecture	ANN-L9	ANN-L18	ANN-L27	ANN-L36	ANN-L12	ANN-L36prim	ANN-L16	ANN-L36prim
Input values	3	3	3	3	4	4	6	4
No. of hidden layers	0	1	1	2	1	1	1	1
No. of nodes in hidden layer	0	2	3	3+2	2	3	2	3
Output value	1	1	1	1	1	1	1	1

5.4 ANN convergence rate and the number of performed iterations

One of the goals of the new, improved models is to achieve the minimum number of iterations performed in order to reduce the time required for estimation. For each of the four listed ANN architectures, Table 78 gives the number of iterations that need to be performed to meet the set value for Gradient Descent criterion. The minimum number of iterations required to be performed is for the ANN-L27 and ANN-L36 architectures for small and medium clusters. The most significant number of iterations needs to be performed for the ANN-L18 architecture and it is nine iterations. It can be concluded that the number of required iterations is minimal (equal to nine), which leads to rapid estimation using the COCOMO2000 model.

The number of required iterations in the second COSMIC FFP approach for both proposed architectures is six, except for the ANN-L12 architecture, where for Dataset_1 and Dataset_3, it is five. It can be concluded that the number of required iterations is also minimal (equal to six), which leads to an even faster estimation using the COSMIC FFP model, see Table 79.

In the first proposed ANN-L16 architecture for the UCP model, it is necessary to perform four iterations to meet the set GA criterion. For ANN-L36rpim architecture, the number of iterations is equal to six. It can be concluded that the number of required iterations is the smallest comparing two other models (equal to six), which leads to an even faster estimation using the UCP model, see Table 80.

Table 78. Number of iterations performed for each ANN architecture - COCOMO2000.
Tabela 78. Broj izvršenih iteracija za svaku ANN arhitekturu - COCOMO2000.

COCOMO2000 model												
	ANN-L9			ANN-L18			ANN-L27			ANN-L36		
No. of Iteration	Small cluster	Medium cluster	Large cluster	Small cluster	Medium cluster	Large cluster	Small cluster	Medium cluster	Large cluster	Small cluster	Medium cluster	Large cluster
GA<0.01	7	6	8	7	6	9	5	5	8	5	5	7

Table 79. Number of iterations performed for each ANN architecture - COSMIC FFP.
Tabela 79. Broj izvršenih iteracija za svaku ANN arhitekturu - COSMIC FFP.

COSMIC FFP model										
	ANN-L12					ANN- L36prim				
No. of Iteration	Dataset_1	Dataset_2	Dataset_3	Dataset_4	Dataset_5	Dataset_1	Dataset_2	Dataset_3	Dataset_4	Dataset_5
GA<0.01	5	6	5	6	6	6	6	6	6	6

Table 80. Number of iterations performed for each ANN architecture - UCP.
Tabela 80. Broj izvršenih iteracija za svaku ANN arhitekturu - UCP.

UCP model		
No. of Iterattion	**ANN-L16**	**ANN-L36prim**
GA<0.01	Training 4	Training 6

In all three proposed approaches, the minimum number of iterations required to meet the GA criterion is 4 in the UCP model for the ANN-L16 architecture. It can be concluded that this architecture converges the fastest to the minimum value of MMRE.

5.5 Activation function choice and encoding/decoding method

During the research, for all experiments of all three proposed approaches, different activation functions in the hidden and output layers were used. Various functions have been experimented with, such as sigmoid function, hyperbolic tangent, Gaussian function, and others. The sigmoid function gave the best results in all experiments, i.e., the smallest MMRE value, and it was used in all three proposed improved models.

Different coding methods were used to homogenize the different nature of the input values: the fuzzification method, the logarithmic method, the combined method, and others. The best results were achieved in all experiments with the fuzzification method used in all three proposed models.

5.6 Threats to validity

In this chapter, the threats and validity of the proposed models in all three approaches will be explained in order to identify potential problems that can help to further improve future research in the field of software project estimation. The main validity aspects are: Internal validity, External validity, Construction validity, and Conclusion validity.

Internal validity

The choice of the set of methods used in all three proposed models represents the potential internal validity of the software effort . The proposed methods of clustering and fuzzification and the assignment of weighting coefficients to different ANN architectures are threats that have been addressed using hyperparametric optimization using the Taguchi method based on orthogonal vector plans. This method has proven to be an effective tool for robust design, i.e., as a good, new technique for optimizing software products. The used Taguchi method includes different input values depending on the three used models. The input values are combined with the weighting coefficients using the sigmoid activation function of the hidden and output layers to obtain the estimated value. From the simplest to the most complex, different ANN architectures were used until the obtained MRE value becomes less than 1%. The application of ANN constructed on Taguchi's orthogonal vector plans confirmed the reliability, stability, and efficiency of the proposed models. The proposed Orthogonal Array Tunning Method (OATM) gives good results while requiring a much smaller number of experiments to find the minimum value of MMRE. Using OATM, hyperparameters are represented by levels. In the three proposed models, two levels L1 and L2 or three levels L1, L2, and L3 were used, depending on the orthogonal vector plan used. The optimal set of parameters constructed in this way gives the best basis for each experiment.

External validity

The obtained results in all three proposed models can be used and applied in other areas, not only in the field of software project estimations. For all three parts of the experiment: training, testing, and validation, several publicly available data sets with real values of given projects were used, such as COCOMO2000, COCOMO81, NASA60, NASA93, Desharnais, Kemerer, ISBSG, UCP Benchmark (Mendeley) and other combined industrial made up of real projects. The obtained results showed the same efficiency even when using different ANN architectures in three different approaches. The method of halving the interval and reducing the values of the weight coefficients of

each subsequent iteration leads to rapid convergence that gradually tends towards the values of minimum costs and effort. This approach is simple and leads to finding the minimum MMRE value with fewer iterations performed being less than 10. Three orders of magnitude reduce the search interval by less than 10 steps. The obtained results can still be checked on new data sets.

Construction validity

Different ANN architectures were used in all three proposed improved models, based on Taguchi's orthogonal vector plans. Depending on the proposed model, the input values and weight coefficients of the corresponding orthogonal vector plan are combined. The algorithm by which each step is performed is presented in detail and methodology of each proposed and improved model is explained (see Chapter 3). Since one of the goals is to select the simplest architecture, which converges the fastest to the minimum value of MMRE, stability and reliable estimation accuracy are achieved, which is lower than other software development effort estimation models. The Gradient Descent criterion was also introduced to know when to run iterations when the ANN architecture converges to the final error value. In all experiments, for all ANN architectures, the "stop criterion" was achieved after a maximum of 9 iterations, while the best architecture in the UCP model has a maximum of 4 iterations.

Conclusion validity

The experiments presented in this dissertation have been repeatedly checked and validated on different data sets, using different ANN architectures and with a minimum number of iterations performed.

Chapter 6: Application of the proposed models and scientific contribution

The research realized in this doctoral dissertation has shown that better, improved models can be constructed based on existing methods of estimating effort and costs that give better results. The applied scientific methods that have been presented can help advance this area of software engineering. The new, applied methodology does not exclude the possibility of applying a subjective estimation of effort and costs, but can help the project team, pointing out potential problems due to discrepancies in the different methods. In this way, teams can perform additional analysis to better assess and correct the results in the shortest possible time. Otherwise, project teams would have to take over their own risk if the only is their subjective opinion.

The obtained results can help project teams, software engineers, and test engineers to make short-term plans with the high and long-term reliability of estimation. Therefore, the teams can can realize all phases of analysis and design and complete the project on time with great certainty. If additional realization is needed, it is possible to anticipate the effort and costs for each project task to be realized in quality and optimal way.

This dissertation presents three models for estimating the effort and costs of developing projects with an accuracy better than the accuracy of the previously existing original methods (see Chapter 3):

- new, improved COCOMO2000 with ANN,
- new, improved COSMIC FFP with ANN, and
- new, improved UCP with ANN.

Depending on clients' historical data, it is possible to choose one of the three proposed approaches and accurately, efficiently, and reliably assess the success of the implementation of the planned tasks. The application of scientific methods and proposed models can help software engineering reach a higher level of maturity. In particular, the probability of successful completion of software projects is not at a satisfactory level. It has been shown that software engineering can have much helpful information based on reliable scientific methods when it comes to the domain of software effort estimation.

The research presented in this dissertation has shown that new, improved models for estimating the effort and costs to implement software projects constructed using three different approaches can gain better results than previously used methods and models. It is also possible to combine the proposed models in order to estimate the software effort successfully.

All three proposed, improved models are used as a fundamental tool of an artificial neural network, so it is possible to solve many problems that are not only related to the

field of software engineering. It is possible to apply them in other areas and problems that cannot be solved by classical means of computer technology.

In addition to the application of the proposed methodology in the field of software engineering, it is also possible to apply the proposed models in other areas such as signal processing, image and speech recognition, recognition and processing of natural languages and different knowledge, recognition of printed texts, and others. They can also be successfully used in the medical sciences to construct various software solutions to diagnose a vast number of diseases. Also, they can be used in meteorology for forecasting weather conditions. Furthermore, they can be relevant in nuclear science, robotics, automatic control, telecommunications, financial, and banking services.

A company or individual can use three different proposed models depending on the data they collect and their needs. In addition, all three proposed approaches can be used together in the same time to obtain more stable, reliable, and accurate effort and cost estimation results.

Numerous new applications of the proposed models of artificial intelligence are expected in the future. Future research will focus on constructing models in solving the problem of cybercrime.

Chapter 7: Conclusion

In the first proposed, improved COCOMO2000 model, four different ANN architectures, five different datasets divided into three clusters, an activating sigmoid function, a fuzzification method, and a Taguchi method for estimating effort and cost were used. Based on all experiments, the approach constructed in this way ensures the reliability and stability of the obtained results, which was shown by monitoring the values of MMRE. It can be concluded that the convergence rate of each proposed architecture depends on the cost effect function and the nature of the projects in the different used datasets. The more complex the ANN architecture, the higher the convergence rate, the shorter the time it takes to perform the required iterations and achieve the minimum MMRE and vice versa. By dividing each data set into clusters and using the fuzzification method, the heterogeneous structure of the project can be partially mitigated, making the proposed model sufficiently stable, flexible, and reliable.

It can be concluded that the ANN-L36 architecture gave the best results, i.e., the lowest value of MMRE within the COCOMO2000 approach following all parts of the experiment.

The main advantages of this model are:

- the number of iterations is in the range of five to ten - which means shorter effort estimation time thanks to the convergence speed and simple architectures of each proposed ANN,
- use of simple ANN architectures,
- high actual effort coverage on used datasets, and
- minimal MMRE.

A possible drawback is finding new methods that could further reduce the value of MMRE. There are no particular limitations in applying this approach, precisely because of experimentation with several proposed ANN architectures and high coverage of project values.

Another proposed, improved COSMIC FFP model in this dissertation has shown that two different ANN architectures based on Taguchi Orthogonal Arrays can further reduce the MMRE value. The COSMIC FFP model belongs to a group of approaches based on user functional requirements using four input values. In this model, the ISBSG dataset, clustering method for input values and the fuzzification method were used to control and mitigate different project structures.

It was concluded that models constructed from the two proposed ANN architectures (ANN-L12 and ANN-L36prim) give a significantly lower MMRE value that differs by about 14.5% compared to the previous experiment realized on the improved

COCOMO2000 model. The obtained results show that the difference in MMRE of the more complex ANN-L36prim architecture is 0.9% lower than for the ANN-L12 architecture.

The efficiency and stability of the proposed model were confirmed by calculating two correlation coefficients. Monitoring the prediction on three different criteria further confirmed the accuracy and reliability of this model. In addition to calculating the MMRE value, the influence of the input values of the used COSMIC FFP model on the change of the MMRE value on each of the seven used datasets was examined.

The main advantages of the proposed approach are:

- the number of iterations is in the range of five to six - which significantly reduces the estimation time,
- use of simple ANN architectures,
- optimization of proposed ANN architectures using Taguchi Orthogonal Arrays,
- high coverage of different values of functional size of software projects,
- ISBSG repository of data on real projects collected from different companies.

There are no specific limitations in this approach, and it can be applied in various fields of business and science domains, such as medicine, recognition of patterns and images, nuclear science, and others.

The third proposed, improved UCP model uses two different ANN architectures and four different datasets, a sigmoid activation function, a fuzzification method, and a Taguchi method to estimate the effort and cost of software development. By monitoring the MMRE value and the convergence rate of each of these architectures, this model gives much better results compared to the previous two models.

Based on three performed parts of the experiment, it was concluded that the ANN-L16 architecture converges after the fourth iteration and gives 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 for the other proposed COSMIC FFP model. Following the prediction through all parts of the experiment, both ANN architectures of this model have a value of 100%, which means that the model is exact, accurate, and reliable. In addition to the MMRE value, the influence of the dependent variables UUCP and AUCP was monitored to check the influence on the change in the MMRE value. The resulting error is less than 0.5%, so it can be concluded that the ANN-L36prim architecture and vice versa can replace the ANN-L16 architecture.

The main advantages of this model are:

- the number of iterations is in the interval from four to six - which means reduced effort estimation time thanks to the exceptional convergence rate of both architectures,
- two simple proposed ANN architectures,
- high coverage of different real effort values and the lowest MMRE value of

7.5%.

A possible drawback is the finding of new methods that could further reduce the value of MMRE. However, there are no specific limitations in applying this approach. This model can be used alone or in combination with the previous two depending on the company's historical data for which the software is implemented. Although not as standardized as the previous two, it is increasingly used by software companies, software engineers, and teams to assess the effort required to implement software projects effectively.

The proposed, improved models can serve as an idea for constructing one or more tools that will accurately, efficiently, and reliably estimate the effort and costs during the various stages of software project development. In addition, these models can be used by software companies, software engineers, and technical project managers to obtain fast and accurate results for the reliable and stable estimation of all anticipated project implementation requirements. This can further reduce the problems most commonly encountered by professionals and teams in this area of software engineering.

Bibliography

[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] https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/GENREF/ChaosManifest_2011.pdf

[4] https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

[5] M. Ayat et al. Current trends analysis and prioritization of success factors: a systematic literature review of ICT projects. *International Journal of Managing Projects in Business*, 14 (3): 652-679, 2021.

[6] https://www.standishgroup.com/

[7] A. Stoica and J. Blosiu. Neural learning using Orthogonal arrays. *Advances in Intelligent Systems*, 41: 418, 1997.

[8] J. F. Khaw, B. Lim, and L. E.Lim. Optimal design of neural networks using the taguchi method. *Neurocomputing*, 7 (3): 225–245, 1995.

[9] A. BaniMustafa. Predicting software effort estimation using machine learning techniques. *In 8th International Conference on Computer Science and Information Technology (CSIT)*, pages 249–256, IEEE, 2018.

[10] 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.

[11] 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.

[12] P. S. Kumar and H. Behera. Role of soft computing techniques in software effort estimation: an analytical study. *Computational Intelligence in Pattern Recognition*. Advances in Intelligent Systems and Computing, pages 807–831, Springer, 2020.

[13] J. Popović. Enhancing methods for effort estimation in software projects. Doctoral dissertation, University of Belgrade, School of Electrical Engineering,

Belgrade, Serbia, 2016.

[14] S. J. Huang and N. H. Chiu. Optimization of analogy weights by genetic algorithm for software effort estimation. *Information and software technology*, 48 (11): 1034-1045, 2006.

[15] M. Shepperd, C. Schofield, and B. Kitchenham. Effort estimation using analogy. *In Proceedings of IEEE 18th International Conference on Software Engineering*, pages 170-178, IEEE, 1996.

[16] D. S. Cruzes and T. Dybå. Research synthesis in software engineering: A tertiary study. *Information and Software Technology*, 53 (5): 440-455, 2011.

[17] P. Pandey. Analysis of the techniques for software cost estimation. *In 3rd International Conference on Advanced Computing and Communication Technologies (ACCT)*, pages 16-19, IEEE, 2013.

[18] J. J. Ryan et al. Quantifying information security risks using expert judgment elicitation. *Computers & Operations Research*, 39 (4): 774-784, 2012.

[19] J. G. Borade and V. R. Khalkar. Software project effort and cost estimation techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(8): 730-739, 2013.

[20] O. Fedotova, L. Teixeira, and H. Alvelos. Software Effort Estimation with Multiple Linear Regression: Review and Practical Application. *Journal of Information Science and Engineering*, 29(5): 925-945, 2013.

[21] N. Condori-Fernandez and P. Lago. Characterizing the contribution of quality requirements to software sustainability. *Journal of Systems and Software*, 137: 289-305, 2018.

[22] C. Pacheco et al. Reusing functional software requirements in small-sized software enterprises: a model oriented to the catalog of requirements. *Requirements Engineering*, 22 (2): 275-287, 2017.

[23] T. Hovorushchenko and O. Pomorova. Methodology of evaluating the sufficiency of information on quality in the software requirements specifications. *In 9th International Conference on Dependable Systems, Services and Technologies (DESSERT),* pages 370-374, IEEE, 2018.

[24] A. Dar and N. Anuradha. Use of Orthogonal arrays and design of experiment via Taguchi L9 method in probability of default. *Accounting*, 4 (3): 113-122, 2018.

[25] M. Alshibli et al. A robust robotic disassembly sequence design using Orthogonal arrays and task allocation. *Robotics*, 8 (1): 20, 2019.

[26] G. Madhoo and M. Shilpa. Optimization of process parameters of stir casting technique using Orthogonal arrays. *International journal of advanced research methodology in engineering & technology*, 1 (2), 2017.

[27] P. Singh, V. Sarin, and N. Midha. Mixture Designs Constructed Using Taguchi's Mixed Element Orthogonal Arrays. Journal of Statistical Theory and Practice, 14 (4): 1-12, 2020.

[28] K. Suresh and R. Dillibabu. Designing a machine learning based software risk model using Naïve Bayes algorithm. *TAGA Journal*, 14: 3141-3147, 2018.

[29] P. Pospieszny, B. Czarnacka-Chrobot and A. Kobylinski. An effective approach for software project effort and duration estimation with machine learning algorithms. *Journal of Systems and Software*, 137: 184-196, 2018.

[30] M. Thrun, F. Pape, and A. Ultsch. Interactive machine learning tool for clustering \ in visual analytics. *In 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 479-487, IEEE, 2020.

[31] T. Tajti. Fuzzification of training data class membership binary values for neural network algorithms. *In Annales Mathematicae et Informaticae*, Eszterházy Károly Egyetem Líceum Kiadó, 52: 217-228, 2020.

[32] T. D. Khang et al. Fuzzy C-Means Clustering Algorithm with Multiple Fuzzification Coefficients. *Algorithms*, 13 (7): 158, 2020.

[33] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. arXiv preprint, arXiv:1710.05941, 2017.

[34] C. Nwankpa et al. Activation functions: Comparison of trends in practice and research for deep learning. arXiv preprint, arXiv:1811.03378, 2018.

[35] L. H. Chen and S. H. Nien. A new approach to formulate fuzzy regression models. *Applied Soft Computing*, 86 (01): 105915, 2020.

[36] van Smeden M et al. Sample size for binary logistic prediction models: beyond events per variable criteria. *Statistical methods in medical research*, 28 (8): 2455-2474, 2019.

[37] M. Nilashi et al. A recommender system for tourism industry using cluster ensemble and prediction machine learning techniques. *Computers & industrial engineering*, 109: 357-368, 2017.

[38] S. Scalabrino et al. Automatically assessing code understandability: How far are we? *In 32nd IEEE/ACM International Conference on Automated Software*

Engineering (ASE), pages 417-427, IEEE, 2017.

[39] V. G. Sychenko and D. V. Mironov. Development of a mathematical model of the generalized diagnostic indicator on the basis of full factorial experiment. *Archives of Transpor*t, 43, 2017.

[40] B. Durakovic. Design of experiments application, concepts, examples: State of the art. *Periodicals of Engineering and Natural Sciences (PEN)*, 5 (3): 421-439 2017.

[41] J. Nunez Ares. and P. Goos. An integer linear programing approach to find trend-robust run orders of experimental designs. *Journal of Quality Technology*, 51 (1): 37-50, 2019.

[42] S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107: 3-11, 2018.

[43] Sharma S. and Sharma S. Activation functions in neural networks. *Towards Data Science*, 6 (12): 310-316, 2017.

[44] G. Mourgias-Alexandris et al. An all-optical neuron with sigmoid activation function. *Optics express*, 27 (7): 9620-9630, 2019.

[45] R. Kapelko. Asymptotic formula for sum of moment mean deviation for order statistics from uniform distribution. Discrete Mathematics, *Algorithms and Applications*, 11 (02): 1950015, 2019.

[46] B. W. Boehm. Safe and simple software cost analysis. *IEEE software*, 17 (5): 14-17, 2000.

[47] 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.

[48] https://www.ifpug.org/about-function-point-analysis/?lang=de

[49] C. R. Symons. Function point analysis: difficulties and improvements. *IEEE transactions on software engineering*, 14 (1): 2-11, 1988.

[50] N. Board. Software Measurement in the Netherlands - The 25th Anniversary of Nesma. *In 2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*, pages 125-126, IEEE, 2014.

[51] https://www.totalmetrics.com/function-point-resources/downloads/COSMIC-Versus-IFPUG-Similarities-and-Differences.pdf

[52] 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.

[53] L. C. Briand, K. El Emam, and F. Bomarius. COBRA: a hybrid method for software cost estimation, benchmarking, and risk . *In Proceedings of the 20th international conference on Software engineering*, pages 390-399, IEEE, 1998.

[54] S. Diev. Use cases modeling and software estimation: applying use case points. ACM SIGSOFT *Software Engineering Notes*, 31 (6): 1-4, 2006.

[55] L. Lavazza and S. Morasca. Software effort estimation with a generalized robust linear regression technique. *In 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012),* pages 206-215, IET, 2012.

[56] V. Nguyen, B. Steece, and B. W. Boehm. A constrained regression technique for COCOMO calibration. *In Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 213-222, ACM-IEEE, 2008.

[57] 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.

[58] N. Mittas and L. Angelis. Ranking and clustering software cost estimation models through a multiple comparisons algorithm. *IEEE Transactions on software engineering*, 39 (4): 537-551, 2012.

[59] E. Kocaguneli et al. Exploiting the essential assumptions of analogy-based effort estimation. IEEE transactions on software engineering, 38 (2): 425-438, 2011.

[60] M. P. Fay and Y. Malinovsky. Confidence intervals of the Mann-Whitney parameter that are compatible with the Wilcoxon-Mann-Whitney test. *Statistics in medicine*, 37 (27): 3991-4006, 2018.

[61] http://info.usherbrooke.ca/llavoie/enseignement/References/CocomoII_Reference_Manual.pdf

[62] L. H. Putnam and W. Myers. Measures for excellence: reliable software on time, within budget. *Prentice Hall Professional Technical Reference*, First Edition, 1991. ISBN-10: 0135676940

[63] D. D. Galorath and M. W. Evans. Software sizing, estimation, and risk management: when performance is measured performance improves, *Auerbach*

Publications, First Edition 2006. ISBN-10: 0849335930.

[64] B. Boehm et al. Software engineering economics. *New York*, 197, 1981.

[65] B. W. Boehm et al. Software cost estimation with COCOMO II. First Edition, *Prentice Hall Press*, 2009.

[66] A. A. Fadhil, R. G. Alsarraj, and A. M. Altaie. Software cost estimation based on dolphin algorithm. IEEE Access, 8: 75279–75287, 2020.

[67] G. Kumar and P. K. Bhatia. Automation of software cost estimation using neural network technique. *International Journal of Computer Applications*, 98 (20): 11-17, 2014.

[68] S. Goyal and A. Parashar. Machine learning application to improve cocomo model using neural networks. *International Journal of Information Technology and Computer Science (IJITCS)* 3: 35–51, 2018.

[69] M. Nandini, P. Bhargavi, and G. R. Sekhar. Face recognition using neural networks. *International Journal of Scientific and Research Publications*, 3 (3): 1-5, 2013.

[70] S. Mukherjee and R. K. Malu. Optimization of project effort estimate using neural network. *In: 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pages 406–410, IEEE, 2014.

[71] N. Couellan. Probabilistic robustness estimates for feed-forward neural networks. *Neural Networks*, 142: 138-147, 2021.

[72] M. Pandey, R. Litoriya, and P. Pandey. Validation of existing software effort estimation techniques in context with mobile software applications, Wireless Personal Communications, 110 (4):1659–1677, 2020.

[73] G. Boetticher, An of metric contribution in the construction of a neural network-based effort estimator. *In: Second International Workshop on Soft Computing Applied to Software Engineering*, Enschade, NL, 2001.

[74] K. Kamaraj, C. Arvind, and K. Srihari. A weight optimized artificial neural network for automated software test oracle. *Soft Computing*, 24 (17):13501–13511, 2020.

[75] M. Albarqi, R. Alsulami, and J. Graham. Automated data processing of neutron depth profiling spectra using an artificial neural network. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 953: 163217, 2020.

[76] M. A. Hamada. Neural Network Estimation Model to Optimize Timing and Schedule of Software Projects. *In 2021 IEEE International Conference on Smart Information Systems and Technologies (SIST)*, pages 1-7, IEEE, 2021.

[77] C. L. C. Roxas. Modeling road construction project cost in the philippines using the artificial neural network approach. *In: 2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, pages 1–5, IEEE, 2019.

[78] J. Popović and D. Bojić. A Comparative Evaluation of Effort Estimation Methods in the Software Life Cycle. *Computer Science and Information Systems*, 9 (1): 455-484, 2012. ISSN: 1820-0214.

[79] G. De Vito, F. Ferrucci, and C. Gravino. Design and automation of a COSMIC measurement procedure based on UML models. *Software and Systems Modeling*, 19 (1): 171-198, 2020.

[80] C. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30 (5): 416-429, 1987.

[81] V. T. Ho and A. Abran. A Framework for automatic function point counting from source code. *In International Workshop on Software Measurement (IWSM'99)*, 1999.

[82] C. Symons. Software Sizing and Estimating, Mk II Function Point Analysis. *John Wiley & Sons*, Chichester, England, 1991.

[83] https://cosmic-sizing.org/wp-content/uploads/2015/07/Measurement_Patterns_Guideline_v1.01-1.pdf

[84] M. Salmanoglu, T. Hacaloglu, and O. Demirors. Effort estimation for agile software development: comparative case studies using COSMIC functional size measurement and story points. *In Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement (IWSM Mensura '17), pages 41-49, Association for Computing Machinery*, New York, USA, 2017.

[85] I. Hussain, L. Kosseim, and O. Ormandjieva. Approximation of COSMIC functional size to support early effort estimation in Agile. *Data & Knowledge Engineering*, 85: 2-14, 2013.

[86] Z. Sakhrawi, A. Sellami, and N. Bouassida. Investigating the Impact of Functional Size Measurement on Predicting Software Enhancement Effort Using Correlation-Based Feature Selection Algorithm and SVM Method. In: Ben Sassi S., Ducasse S., Mili H. (eds) Reuse in Emerging Software Engineering Practices. ICSR 2020. *Lecture Notes in Computer Science*, 12541. Springer, Cham, 2020.

[87] T. Fehlmann and L. Santillo. From story points to cosmic function points in agile software development–a six sigma perspective. *In Metrikon-Software Metrik Kongress*, page 24, 2010.

[88] M. Ochodek, S. Kopczyńska, and M. Staron. Deep learning model for end-to-end approximation of COSMIC functional size based on use-case names. *Information and Software Technology*, 123 (07):106310, 2020.

[89] S. Bagriyanik and A. Karahoca. Using Data Mining to Identify COSMIC Function Point Measurement Competence. *International Journal of Electrical and Computer Engineering*, 8 (6): 5253, 2018.

[90] W. Xia et al. A new calibration for Function Point complexity weights. *Information and Software Technology*, 50 (7-8): 670-683, 2008.

[91] W. Xia, D. Ho, and L. F. Capretz. A neuro-fuzzy model for function point calibration. *arXiv preprint*, arXiv:1507.06934, 2015.

[92] https://www.cs.cmu.edu/~jhm/DMS%202011/Presentations/Cohn%20%20Estimating%20with%20Use%20Case%20Points_v2.pdf

[93] M. Kassab, C. Neill, and P. Laplante. State of practice in requirements engineering: contemporary data. *Innovations in Systems and Software Engineering*, 10:(4): 235-241, 2014.

[94] M. Ochodek, J. Nawrocki, and K. Kwarciak. Simplifying effort estimation based on Use Case Points. *Information and Software Technology* archive, 53 (3): 200-213, 2011.

[95] E. R. Carroll. Estimating software based on use case points. *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 257-265, San Diego, CA, USA, 2005.

[96] A.B. Nassif, L.F. Capretz, and D. Ho. Enhancing Use Case Points Estimation Method using Soft Computing Techniques. *Journal of Global Research in Computer Science*, 1 (4): 12- 21, 2010.

[97] A.B. Nassif. Software Size and Effort Estimation from Use Case Diagrams Using Regression and Soft Computing Models. Doctoral dissertation, Western University, London, Ontario, Canada, 2012.

[98] M. Azzeh. Fuzzy Model Tree for Early Effort Estimation Machine Learning and Applications. *12th International Conference on Machine Learning and Applications*, pages 117-121, Miami, Florida, USA, IEEE, 2013.

[99] T. Urbanek et al. Using Analytical Programming and UCP Method for Effort

Estimation. Modern Trends and Techniques in Computer Science, *Advances in Intelligent Systems and Computing*, 285: 571-581, 2014.

[100] A. Kaur and K. Kaur. Effort estimation for mobile applications using use case point (UCP). *In Smart Innovations in Communication and Computational Sciences*, pages 163-172, Springer, Singapore, 2019.

[101] Z. C. Ani, S. Basri, and A. Sarlan. A reusability of UCP-based effort estimation framework using object-oriented approach. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9 (3-5): 111-114, 2017.

[102] Y. Mahmood, N. Kama, and A. Azmi. A systematic review of studies on use case points and expert-based estimation of software development effort. *Journal of Software: Evolution and Process*, 32 (7), e2245, 2020.

[103] K. K. Gebretsadik and W. T. Sewunetie. Designing Machine Learning Method for Software Project Effort Prediction. *Computer Science and Engineering*, 9(1): 6-11, 2019.

[104] R. Alves, P. Valente, and N. J. Nunes. Improving software effort estimation with human-centric models: a comparison of UCP and iUCP accuracy. *In Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems*, pages 287-296, 2013.

[105] E. Guresen and G. Kayakutlu. Definition of artificial neural networks with comparison to other networks. *Procedia Computer Science*, 3: 426-433, 2011.

[106] J. Zupan. Introduction to artificial neural network (ANN) methods: what they are and how to use them. *Acta Chimica Slovenica*, 41: 327-327, 1994.

[107] S. Lek and J. F. Guégan. Artificial neural networks as a tool in ecological modelling, an introduction. *Ecological modelling*, 120 (2-3): 65-73, 1999.

[108] A. R. Venkatachalam. Software cost estimation using artificial neural networks. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93) Nagoya*, Japan, pages 987-990, IEEE, 1993.

[109] M. H. Sazli. A brief review of feed-forward neural networks. Communications Faculty of Sciences University of Ankara Series A2-A3 *Physical Sciences and Engineering*, 50 (01), 2006.

[110] I. A. Basheer and M. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, 43 (1): 3-31, 2000.

[111] B. J. Yum et al. The Taguchi robust design method: Current status and future directions. *Journal of Korean Institute of Industrial Engineers*, 39 (5): 325-341, 2013.

[112] K. L. Tsui. An overview of Taguchi method and newly developed statistical methods for robust design. *IIE Transactions*, 24 (5): 44-57, 1992.

[113] J. D. Kechagias et al. A comparative investigation of Taguchi and full factorial design for machinability prediction in turning of a titanium alloy. *Measurement*, 151: 107213, 2020.

[114] N. Rankovic. et al. A new approach to software effort estimation using different Artificial Neural Network architectures and Taguchi Orthogonal Arrays. *IEEE Access*, 9: 26926-26936, 2021.

[115] N. Rankovic et al. Improved effort and cost estimation model using Artificial Neural Networks and Taguchi method with different activation functions. *Entropy*, 23 (7), 854, 2021.

[116] A. Jeang and C. L. Chang. Combined robust parameter and tolerance design using Orthogonal arrays. *The International Journal of Advanced Manufacturing Technology*, 19 (6), 442-447, 2002.

[117] E. Kolaiti and C. Koukouvinos. On the use of three level Orthogonal arrays in robust parameter design. *Statistics & probability letters*, 76 (3): 266-273, 2006.

[118] R. Fontana, F. Rapallo, and H. P. Wynn. Circuits for robust designs. *arXiv preprint*, arXiv:2106.11213, 2021.

[119] D. Rankovic et al. Convergence rate of Artificial Neural Networks in software development projects. *Information and Software Technology Journal*, 138 (10), 2021.

[120] http://promise.site.uottawa.ca/SERepository/

[121] H. K. Lam. A review on stability analysis of continuous-time fuzzy-model-based control systems: From membership-function-independent to membership-function-dependent analysis. *Engineering Applications of Artificial Intelligence*, 67: 390-408, 2018.

[122] L. Ghosh, S. Saha, and A. Konar. Decoding emotional changes of android-gamers using a fused Type-2 fuzzy deep neural network. *Computers in Human Behavior*, 116: 106640, 2021.

[123] A. B. Nassif et al. Software development effort estimation using regression fuzzy models. *Computational intelligence and neuroscience*, 2019.

[124] M. V. Boldin. On the power of Pearson's test under local alternatives in autoregression with outliers. *Mathematical Methods of Statistics*, 28(1): 57-65, 2019.

[125] D. Blum and H. Holling. Spearman's law of diminishing returns. A meta-analysis. *Intelligence*, 65: 60-66, 2017.

[126] W. Li. A new approach to solve uncertain multidisciplinary design optimization based on conditional value at risk. *IEEE Transactions on Automation Science and Engineering*, 18 (1): 356-368, 2020.

[127] L. Qiao. Deep learning based software defect prediction. *Neurocomputing*, 385: 100-110, 2020.

[128] M. A. Shah. Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction. *IEEE Access*, 8: 58402-58415, 2020.

[129] https://www.isbsg.org/software-project-data/

[130] https://data.mendeley.com/datasets/2rfkjhx3cn/1

[131] P. Manali et al. Long-lead prediction of ENSO modoki index using machine learning algorithms. *Scientific reports*, 10 (1): 1-13, 2020.

[132] H. Liang. Dynamic evaluation of drilling leakage risk based on fuzzy theory and PSO-SVM algorithm. *Future Generation Computer Systems*, 95: 454–466, 2019.

[133] Y. Liu, L. Wang, and K. Gu. A support vector regression (SVM)-based method for dynamic load identification using heterogeneous responses under interval uncertainties. *Applied Soft Computing*, 107599, 2021.

Prošireni Izvod

Predmet istraživanja

Procena napora i troškova je od izuzetnog značaja za uspešnu realizaciju softverskih projekata. Vreme izrade projekta je bitan faktor, kako za naručioce projekta tako i za realizatore projekta. Količina novca koja je potrebna da se uloži u projekat utiče na odluku da li će projekat početi ili ne, odnosno da li će se uspešno završiti ili ne. Cena koštanja projekta se u praksi najčešće upoređuje sa cenom koštanja sličnih projekata, koji su uspešno završeni. Vreme i potreban novac nisu jedini faktori koji definišu početak realizacije projekta, već se u obzir moraju uzeti i ostali parametri kao što su: kvalitet, složenost, prekovremeni rad tima i drugo. Neadekvatne procene napora najčešće mogu dovesti do kašnjenja u realizaciji projekta, zahtevima za dodatnim novčanim sredstvima, prekovremenim radom eksperata i slično. Takođe, one mogu da utiču direktno na kvalitet softvera. Usled ne sagledavanja svih potrebnih parametara za procenu realizacije softvera, često se dešava da se pojedine aktivnosti, kao što su: dodatna testiranja, kompletiranje dokumentacije i dodatno definisanje zahteva korisnika svode na minimum napora. Sve ovo može dovesti do velikog broja projekata koji su nerazjašnjeni i koji predstavljaju trenutno ogledalo softverske industrije.

Dosadašnji načini procene napora i troškova tokom realizacije softverskih projekata su se zasnivali na nepouzdanim i nepreciznim metodama, tehnikama i modelima. Rezultat ovakvih nepotpunih procena je veliki broj neuspešnih i nerealizovanih projekata. Sva istraživanja ukazuju da se uspešno završi samo oko 30% projekata, dok preostali procenat projekata ostaju nerešeni ili potpuno propali. Dosadašnje najčešće korišćene metode su: metoda sličnosti, metoda analize i sinteze, procena zasnovana na znanju eksperata i razne parametarske (algoritamske) metode. Softverske kompanije svakako pored raznih metoda procene, koriste i različite pomoćne softverske alate i servise kako bi ispunile zahteve kupaca. Da bi konstruisali pouzdani softver visokih standarda i performansi, mnogi istraživači su predlagali različite kombinacije parametarskih i ne-parametarskih modela procene napora i troškova. Neophodno je analizirati i eksperimentalno proveriti dosadašnje najuspešnije metode i modele, kako bi se one adekvatno mogle unaprediti.

Predmet istraživanja ove doktorske disertacije jeste analiza dosadašnjih najboljih praktično korišćenih pristupa i modela u proceni napora i troškova. Potom su na osnovu njih konstruisana, predstavljena i eksperimentalno potvrđena tri nova, poboljšana modela u okviru tri različita pristupa koji će dati bolje rezultate.

Cilj istraživanja

Cilj ove disertacije je konstruisanje tri nova, poboljšana modela zasnovana na različitim arhitekturama veštačkih neuronskih mreža (*engl.* Artificial Neural Network/ANN). Unapređenje postojećih metoda i modela bi se realizovalo primenom veštacke inteligencije, koja bi poslužila kao moćan alat za postizanje boljih rezultata. Korišćenjem različitih arhitektura veštačkih neuronskih mreža konstruisanih na osnovu Tagučijevih ortogonalnih vektorskih planova moguće je optimizovati predložene modele. Realna i pravilna procena je od izuzetnog značaja u upravljanju softverskim projektima, kako ne bi došlo do otkazivanja, propadanja ili probijanja vremena ili budžeta projekta. Poboljšanjem postojećih metoda procene bi se smanjili rizici i potencijalne greške projekta kako za softverske kompanije koje ga realizuju tako i za klijente koji očekuju proizvod unutar okvira budžeta i vremena.

Glavni cilj ove disertacije je eksperimentalno proveriti i utvrditi najbolji model za pouzdanu, efikasnu, brzu i tačnu procenu napora i troškova za realizaciju softverskih projekata. Potrebno je identifikovati najbolji model koji ispunjava zadate kriterijume: ima najjednostavniju ANN arhitekturu, minimalan broj iteracija, ANN arhitektura treba najbrže da konvergira ka minimalnoj vrednosti MMRE - zbog smanjenja potrebnog vremena za estimaciju. Dodatno se bira odgovarajuća metodologija za konstrukciju modela i u okviru nje se definiše najbolja optimizaciona metoda (metoda Tagučijevih ortogonalnih vektorskih planova bazirana na latinskim kvadratima, što znači da se u velikoj meri smanjuje broj eksperimenta). Tagučijeva metoda za razliku od potpunog ortogonalnog faktorijalskog plana (*engl.* Full Factorial Plan/FFP), smanjuje broj eksperimenata na minimalni, pri čemu ostaje potpuna pokrivenost svih jednako zastupljenih parametara koji utiču na procenu. Definiše se i odgovarajuća aktivaciona funkcija koja daje najmanju grešku modela. Zatim se uvodi i metoda klasterizacije koja u velikoj meri doprinosi smanjenju heterogene strukture različite prirode realnih projekata iz različitih domena primene.

Dodatno, testiranje i validacija se vrše na drugim realnim projektima i različitim ANN arhitekturama, zavisno od odabranog pristupa i predloženog modela. Dodatne statističke provere i potvrde preciznosti, pouzdanosti i efikasnosti predloženih novih rešenja se uvode na svakom predloženom modelu. Na kraju se porede predloženi modeli sa SVM (*engl.* Support Vector Regression) algoritmom.

Novi, poboljšani modeli

Iz svakog od tri najčešće korišćena pristupa, koji se zasnivaju na različitim metodama procene napora i troškova, odabran je i konstruisan novi, poboljšani model.

Prethodno korišćeni modeli predloženih pristupa nisu davali dovoljno dobre rezultate da bi se značajno poboljšala uspešnost projekata.

Tri nova, poboljšana modela su konstruisana pomoću različitih ANN arhitektura koje su zasnovane na različitim Tagučijevim ortogonalnim vektorskim planovima. ANN predstavljaju dobru tehniku za obradu informacija i mogu umnogome doprineti konstruisanju novih modela za procenu softvera. Zbog sposobnosti ANN-a da uče iz različitih skupova podataka, moguće je generisati tačne i pouzdane rezultate kako bi se izbegle nepredviđene situacije. Različite arhitekture ANN, se koriste u cilju identifikovanja najjednostavnije koja ispunjava dodatno postavljene kriterijume.

Kada se koriste tri ulazne veličine na primer, kod ANN-L27 arhitekture, prema FFP planu je potrebno izvršiti 3^{13}=1 594 323 iteracija (eksperimenata). Međutim, korišćenjem Tagučijevog ortogonalnog vektorskog plana sa 13 parametara (težinskih koeficijenata), potrebno je samo 3^3=27 iteracija (eksperimenata). Korišćenjem ove metode robusnog dizajna eksperimenata se smanjuje broj iteracija za 99.99830649% (0.9999830649 = 1 – (27/1 594 323)). Predlog redukovanog FFP plana dao je Dr Genichi Taguchi iz Japana i zasnovan je na izuzetnom skupu latinskih kvadrata. Tagučijev ortogonalni vektorski plan omogućava uzimanje odabranog podskupa kombinacija bez ponavljanja. Na ovaj način se svi faktori uzimaju u obzir podjednako i mogu se procenjivati nezavisno jedni od drugih. Ovo omogućava prikupljanje dovoljnog broja podataka kako bi se došlo do informacija koji faktori najviše utiču na kvalitet proizvoda koji se razvija. Pri tom se uzima minimalan broj eksperimenata, štedi vreme i potrebni resursi. Izbor odgovarajućeg Tagučijevog ortogonalnog vektorskog plana zavisi od broja težinskih faktora i broja ulaznih veličina. Opšti cilj ove metode je stvaranje visoko kvalitetnog proizvoda uz moguće smanjenje troškova razvoja. Kombinovane metode kao što su klasterizacija, metoda fazifikacije i dodatne metode provere daju bolje rezultate od dosadašnjih korišćenih modela i pristupa, što i jeste cilj ove disertacije.

1. Novi, poboljšani COCOMO2000 model

COCOMO2000 (*engl.* Constructive Cost model) je parametarski model koji veličinu sistema računa kao kombinaciju matematičkih modela. Osnovni podaci su parametri koji se dobijaju eksperimentalnim putem, merenjem realnih vrednosti tokom izrade projekta. Merenje na osnovu broja linija izvornog koda služi da se utvrdi veličina i kompleksnost softverskog projekta. Najšešće se koristi da se utvrdi napor i vreme potrebno za realizaciju projekta. Svakako najznačajnija metoda za procenu napora iz ove grupe je COCOMO2000 parametarska metoda. Na ovaj način pomoću COCOMO2000 moguće je proceniti i zahtevano vreme izrade. Broj linija koda je jednostavan način za procenu napora i troškova, ali ima i dosta nedostataka, kao na primer, razlike u korišćenom programskom jeziku (C++, Java, C# i sl.) i uspostavljanju ekvivalencije

određenih baza podataka. Stvarni napor (*engl.* Actual Effort) predstavlja stvarnu vrednost projekta, baziranu na broju linija koda izraženih u čovek-mesecima (*engl.* person-months/PM). Koristi 22 parametara kao ulazne veličine, podeljena u dve grupe: prvih 5 čine faktori skaliranja (*engl.* Scale factors) i drugu grupu čini 17 faktora multiplikatora napora (*engl.* Effort Multipliers). Oni se korišćenjem parametarske metode mogu svesti na tri ulazne veličine koje će se koristiti u novom predloženom modelu.

U novom, poboljšanom modelu korišćene su četiri različite arhitekture ANN, bazirane na različitim Tagučijevim ortogonalnim vektorskim planovima. Najjednostavnija arhitektura ANN-L9, koristi ortogonalni plan L9 na 3 nivoa i 4 težinska koeficijenta. Složenija ANN-L18, koristi ortogonalni plan L18 na 2 i 3 nivoa i 8 težinskih koeficijenta. Nešto složenija ANN-L27, koristi ortogonalni plan L27 na 3 nivoa sa 13 težinskih koeficijenata. Najsloženija korišćena ANN-L36 arhitektura u ovom modelu koristi ortogonalni plan L36 na 2 i 3 nivoa i 23 težinska koeficijenta. Pored različitih arhitektura, metoda klasterizacije u velikoj meri doprinosti kontrolisanju različitih vrednosti realnih projekta iz prakse, što za cilj ima realnu procenu i smanjenje minimalne relativne greške modela. Korišćena aktivaciona funkcija je sigmoidna funkcija skrivenog i izlaznog sloja. Ova funkcija omogućava dodatnu homogenizaciju ulaznih veličina, ali i doprinosi brzini konvergencije svake od navedenih ANN arhitektura. Eksperimenti su pokazali da je broj iteracija u svakoj ANN arhitekturi manji od 10, što izuzetno skraćuje vreme procene. Pouzdanost, preciznost i efikasnost novog predloženog modela je više puta proveravana i potvrđivana na pet različitih skupova podataka i proveravanjem uz metode statističke analize. Na osnovu dobijene procenjene vrednosti računate su različite metrike kao što su: MAE (*engl.* Magnitude Absolute Error), MRE (*engl.* Magnitude Relative Error), MMRE (*engl.* Mean Magnitude Relative Error).

2. Novi poboljšani COSMIC FFP model

Analiza funkcionalnih tačaka (*engl.* Function Point Analysis) je pristup koji je nastao u cilju prevazilaženja nedostataka prethodnog pristupa, koji se zasniva na merenju veličine sistema na osnovu broja linija koda. U ovom pristupu, funkcionalnost sistema se meri na osnovu veličina, izraženih u funkcionalnim tačkama. Različiti sistemi mogu imati slične funkcionalnosti, ali mogu koristiti različite tehnologije ili programske jezike, pa se zbog toga razlikuju u broju izvornih linija koda. Pristup zasnovan na broju funkcionalnosti je razvio veliki broj modela kako bi najefikasnije i najtačnije procenili funkcionalnu veličinu. U ovoj disertaciji će biti prikazana najmlađa metoda iz familije funkcionalnih tačaka, COSMIC FFP. Ova metoda se koristi u procesu procene napora i troškova funkcionalne veličine softverskih projekata i zasnovana je na 14 parametra koji se svode na 4 ulazne veličine. Kod ranijih metoda funkcionalnih tačaka korišćeno je pet ulaznih

veličina. Korišćene 4 ulazne veličine u predloženom COSMIC FFP modelu je svedeno na:

1. **Entry** - poruke koje korisnici šalju sistemu ili poruke koje jedan sistem šalje drugom u cilju prenosa neophodnih podataka. Ove poruke ne moraju biti sistemski unosi;
2. **Exit** - poruke koje sistem ili modul vraća kao odgovor korisniku u vidu podataka koji se mogu čitati iz datoteka i ove poruke mogu biti i u obliku aritmetičkih i logičkih operacija;
3. **Read** - poruke koje ažuriraju podatke u sistemu i mogu biti ražlicite datoteke, tabele i drugi podaci;
4. **Write** - poruke koje šalju podatke iz sistema i mogu biti u obliku tabela, datoteka i slično.

Može se zaključiti da funkcionalna veličina sistema predstavlja ukupan broj svih korišćenih poruka. Sistem se može posmatrati kao četvorodimenzioni vektorski prostor, koji predstavlja ukupan broj poruka. Poruke čine podaci koji se unose, podaci koji izlaze iz sistema, podaci koji se zapisuju ili podaci koji se čitaju iz datoteka. Ova metoda omogućava otkrivanje i utvrđivanje uticaja i najmanje promene na funkcionalnu veličinu. Prednost ove metode je što je nezavisna od tehnologije i što ne postoji gornja granica funkcionalne veličine. Samim tim nema ni zasićenja, jer složenost funkcionalnosti može neograničeno da raste u zavisnosti od broja poruka u sistemu.

U novom, poboljšanom COSMIC FFP modelu korišćene su dve različite ANN arhitekture, bazirane na različitim Tagučijevim ortogonalnim vektorskim planovima. Najjednostavnija arhitektura ANN-L12, koristi ortogonalni plan L12 na 2 nivoa i 11 težinskih koeficijenta. Složenija ANN-L36prim, koristi ortogonalni plan L36prim na 2 i 3 nivoa i 16 težinskih koeficijenta. Korišćen je najpoznatiji repozitorijum industrijskih projekta ISBSG, na kome su primenjivane metoda klasterizacije i podele skupova podataka u razmeri 70:30, od kojih je 70% projekata korišćeno za treniranje, a 30% preostalih za testiranje. Metoda klsterizacije u velikoj meri doprinosi ublažavanju heterogene strukture različitih realnih projekta iz prakse, što za cilj ima realnu procenu i smanjenje minimalne relativne greške modela. Korišćena aktivaciona funkcija je sigmoidna funkcija skrivenog i izlaznog sloja. Pomoću ove funkcije moguće je kontrolisati različite vrednosti ulaznih veličina, ali i doprineti brzini konvergencije svake od navedenih ANN arhitektura. Eksperimenti su pokazali da je broj iteracija u svakoj od njih manji od 7, što u mnogome skraćuje vreme procene. Pouzdanost, preciznost i efikasnost novog predloženog modela je više puta proveravana i potvrđivana na sedam različitih skupova podataka i proveravanjem pomoću metoda statističke analize. Na osnovu dobijene procenjene vrednosti računate su različite metrike kao što su: MAE, MRE, MMRE, uticaj ulaznih veličina na promenu vrednosti MMRE.

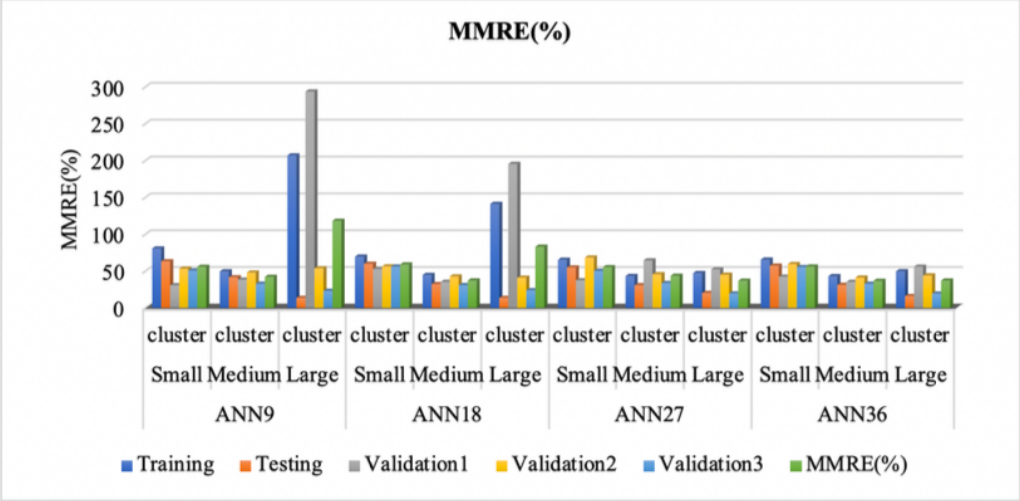
3. Novi poboljšani UCP model

UCP (*engl.* Use Case Point Analysis) metoda je najnovija, ali i najčešće korišćena metoda za procenu napora i troškova za realizaciju softverskih proizvoda. Iako nije standardizovana u okviru ISO standarda poput COCOMO2000 i COSMIC FFP metoda, ovom metodom se dobijaju greške procene između 20% i 35%. Najbolji rezultat koji je postignut ovom metodom je vrednost greške oko 10%. Za određivanje funkcionalne veličine UCP metodom koriste se korisnici sistema i slučajevi korišćenja. Korisnici sistema se dele na tri grupe: jednostavne - u zavisnosti od interakcije sa sistemom dodeljuje im se težinski faktor 1, srednje - u zavisnosti od interne/eksterne komunikacije dodeljuje im se težinski faktor 2, i složene - u zavisnosti od složenosti interakcija dodeljuje im se težinski faktor 3. Postoje i tri kategorije slučajeva korišćenja koje se definisu na osnovu izvršenog broja transakcija i broja razmena poruka korisnika i sistema: jednostavne - za manje od 3 transakcije i dodeljuje im se težinski faktor 5, srednje - od 4 do 7 transakcija i dodeljuje im se težinski faktor 10, i složene - više od 8 transakcija i dodeljuje im se tezinski faktor 15. Veličina sistema se definiše na osnovu četvorodimenzionog ili šestodimenzionog vektora čiji elementi predstavljaju složenost prethodno navedenih korisnika i slučajeva korisnika u sistemu. To su: **UAW** (*engl.* Unadjusted Actors Weight), **UUCW** (*engl.* Unadjusted Use Case Weight), **UUCP** (*engl.* Unadjusted Use Case Point Weight), **TCF** (*engl.* Technical Complexity Factor), **ECF** (*engl*. Environmental Complexity Factor), **AUCP** (*engl.* Adjusted Use Case Point Weight), od kojih su **UUCP** i **AUCP** zavisne promenljive, a **UAW**, **UUCW**, **TCF** i **ECF** nezavisne ulazne veličine. Stvarni napor (*engl.* Real Effort) se dobija kao norma vektora jednog od dva vektora i predstavlja realnu funkcionalnu veličinu ili broj tačaka slučajeva korišćenja.

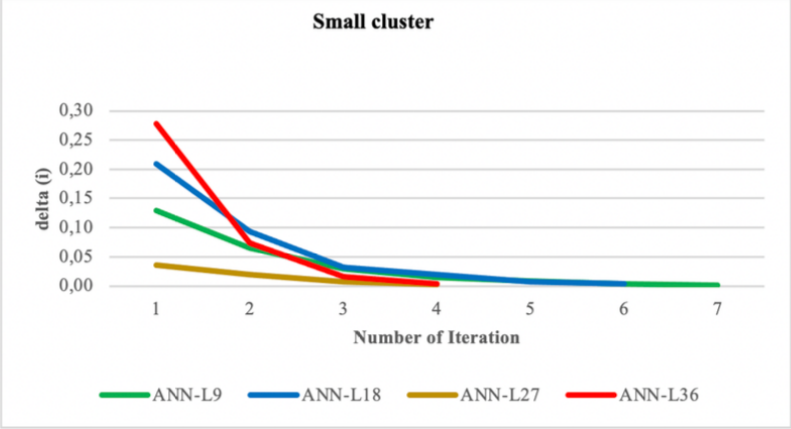
U novom, poboljšanom UCP modelu korišćene su dve različite ANN arhitekture, bazirane na različitim Tagučijevim ortogonalnim vektorskim planovima. Najjednostavnija arhitektura ANN-L16, koristi ortogonalni plan L16 na 2 nivoa i 15 težinskih koeficijenata. Složenija ANN-L36prim, koristi ortogonalni plan L36prim na 2 i 3 nivoa i 16 težinskih koeficijenta. Pored različitih arhitektura, metoda klasterizacije u velikoj meri doprinosi ublažavanju različite strukture realnih projekta iz prakse, što za cilj ima realnu procenu i smanjenje minimalne relativne greške modela. Korišćena aktivaciona funkcija je sigmoidna funkcija skrivenog i izlaznog sloja koja doprinosi dodatnoj homogenizaciji ulaznih veličina, ali i brzini konvergencije svake od navedenih arhitektura ANN. Eksperimenti su pokazali da je broj iteracija u svakoj od njih manji od 7, što značajno skraćuje vreme procene. Pouzdanost, preciznost i efikasnost novog predloženog modela je više puta proveravana i potvrđivana na četiri različita skupa podataka i proveravanjem korišćenjem metoda statističke analize. Na osnovu dobijene procenjene vrednosti računate su različite metrike kao što su: MAE, MRE, MMRE, uticaj zavisnih promenljivih, ulaznih veličina na promenu MMRE i druge.

Dobijeni rezultati

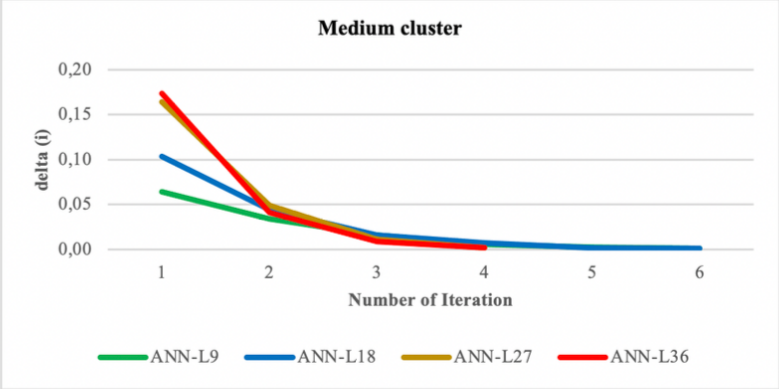
Iz dobijenih rezultata za prvi novi, poboljšani COCOMO2000 model se može zaključiti da arhitektura ANN-L36 postiže najbolje rezultate sa prosečnom vrednosti MMRE od 43.3% u sva tri dela eksperimenta. Nešto veću vrednost MMRE, oko 2%, postiže arhitektura ANN-L27 (45.3%). Slabiji rezultat, veća vrednost MMRE greške se postiže kod ANN-L18 arhitekture (59.7%), a najslabiji rezultat se postiže kod ANN-L9 arhitekture (72%). Analiziranjem sve četiri ANN arhitekture na svakom klasteru na pet različitih skupova podataka, pokazalo se, da sa povećanjem broja skrivenih slojeva procena vrednosti MMRE je pouzdanija, videti Slika 34. Dodatno je praćena i brzina konvergencije svake arhitekture ka minimalnoj relativnoj greški i pokazalo se da u sva tri klastera najbrže konvergira ANN-L36 arhitektura. Broj iteracija potrebnih za svaki deo eksperimenta je u opsegu od 5 do 9, videti Slika 31, Slika 32, Slika 33.



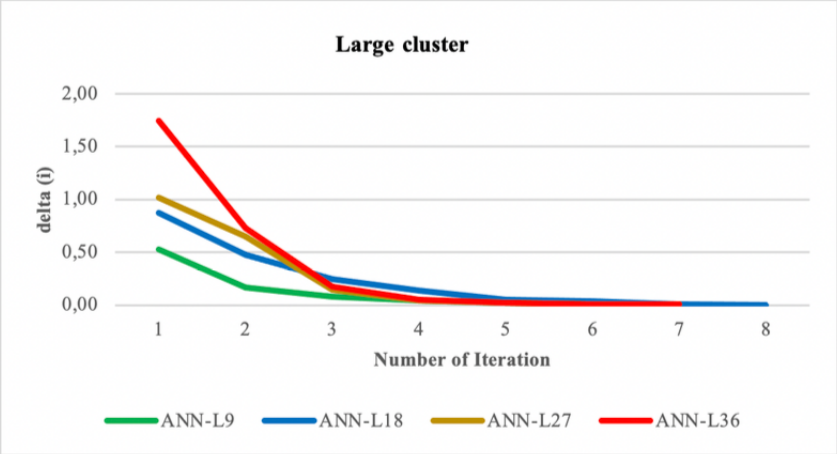
Slika 34. Vrednosti MMRE za svaku predloženu arhitekturu u svakom delu eksperimenta (COCOMO2000).



Slika 31. Brzina konvergencije četiri predložene ANN arhitekture nad malim klasterom.



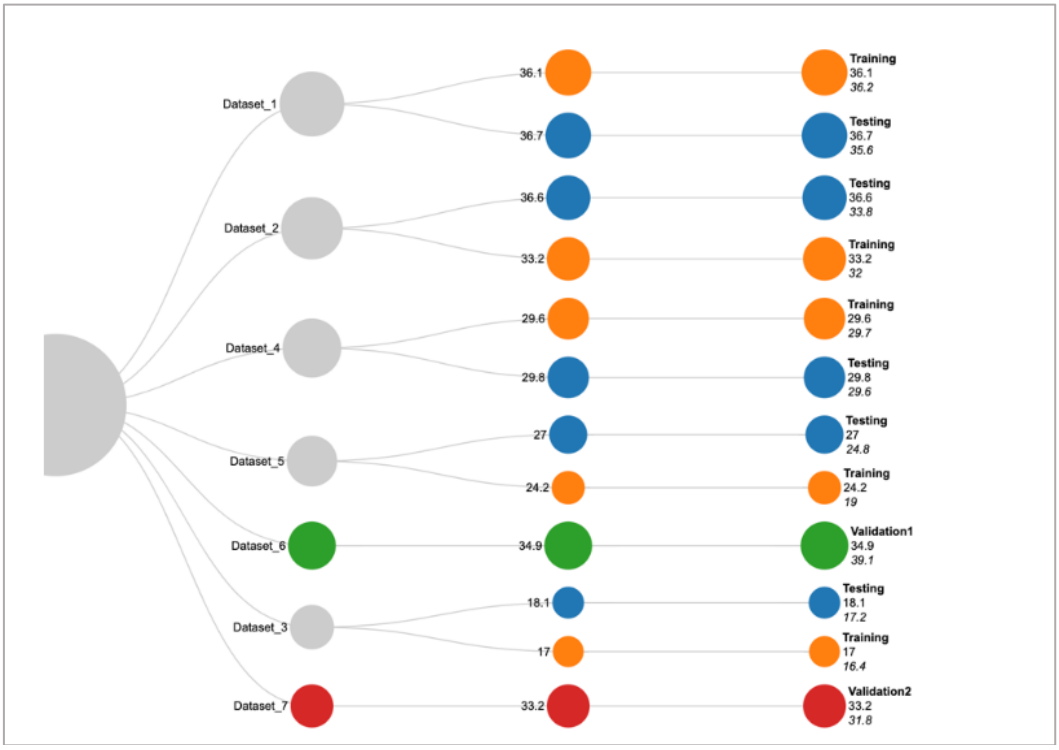
Slika 32. Brzina konvergencije četiri predložene ANN arhitekture nad srednjim klasterom.



Slika 33. Brzina konvergencije četiri predložene ANN arhitekture nad velikim klasterom.

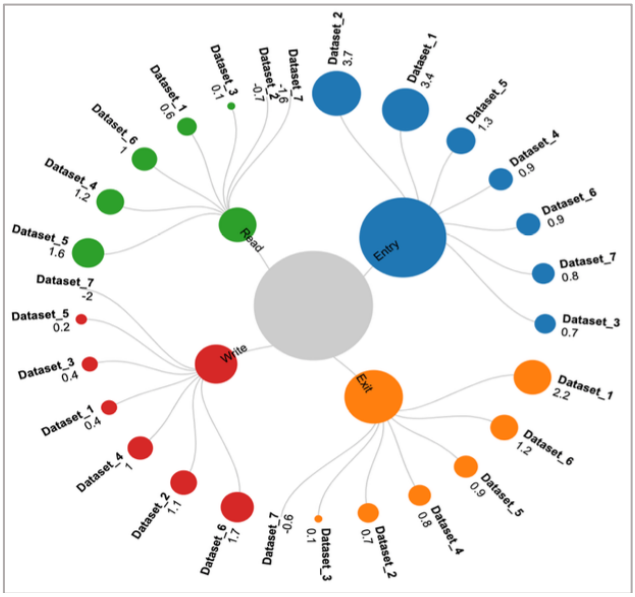
Rezultat koji je praćen tokom izvršavanja potrebnih iteracija u drugom predloženom COSMIC FFP modelu, je vrednost MMRE, koja se, u zavisnosti od prirode skupa podataka, pokazala stabilnom za dve predložene arhitekture u sva tri dela eksperimenta. Skupovi podataka male funkcionalne veličine imaju veću vrednost MMRE (Dataset_1, Dataset_2) od skupova podataka srednje vrednosti funkcionalne veličine (Dataset_3). Skupovi podataka sa velikom vrednosti funkcionalne veličine imaju srednju vrednost MMRE (Dataset_4, Dataset_5). Skupovi podataka za proveru validnosti imaju različite vrednosti funkcionalne veličine i imaju veću vrednost MMRE. Najniža vrednost MMRE postignuta je na Dataset_3 za predloženu arhitekturu ANN-L12 (17.0%), dok je za ANN-L36prim arhitekturu vrednost MMRE 16.4%. Srednja vrednost MMRE u svim skupovima podataka u sva tri dela eksperimenta iznosila je 29.7% za ANN-L12 i 28.8%

za ANN-L36prim, videti Slika 44, Tabela 56. U poređenju sa COCOMO pristupom, vrednost MMRE je značajno smanjena, za 14.5%. Dodatno je praćen uticaj ulaznih veličina Entry, Exit, Read i Write na promenu vrednosti MMRE. Kod arhitekture ANN-L12 najveći uticaj na promenu greške ima ulazna veličina Exit, dok kod arhitekture ANN-L36prim najveći uticaj ima ulazna veličina Entry, videti Slika 47, Slika 48.



Slika 44. Grafička reprezentacija vrednosti MMRE svake predložene arhitekture (COSMIC FFP).

Slika 47. Grafička reprezentacija uticaja ulaznih veličina na vrednost MMRE (ANN-L12 COSMIC FFP).



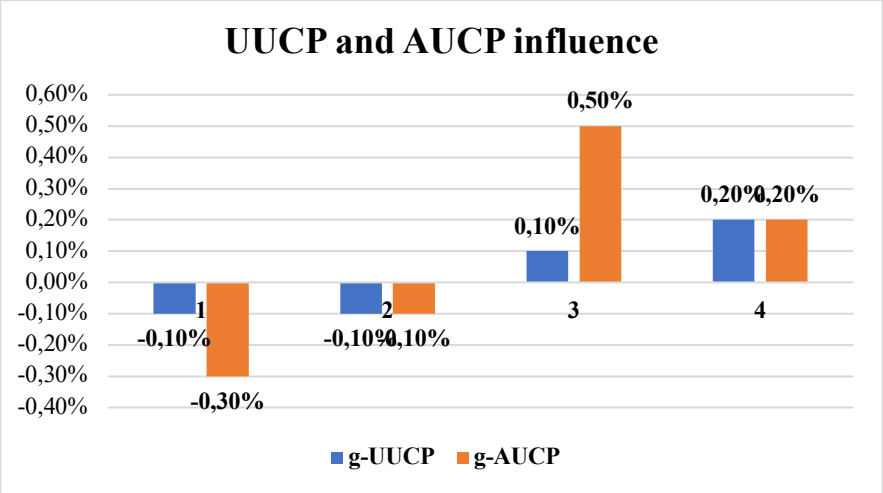
Slika 48. Grafička reprezentacija uticaja ulaznih veličina na vrednost MMRE (ANN-L36prim COSMIC FFP).

Primenom novog, poboljšanog UCP modela za dve predložene arhitekture ANN-L16 i ANN-L36 u sve tri faze eksperimenta, pokazano je da različita priroda skupa podataka ne utiče na složenost arhitekture koja se koristi. Odnosno, nezavisna je od vrednosti ulaznih veličina. U prvoj predloženoj arhitekturi ANN-L16 je korišćeno svih

šest ulaznih veličina (gde su četiri linearno zavisne, a dve linearno nezavisne) i vrednost MMRE u sva tri dela eksperimenta iznosi 7.5% Korišćenjem druge arhitekture ANN-L36prim sa četiri nezavisne ulazne veličine dobijena je ista vrednost MMRE u sva tri dela eksperimenta i iznosi 7.5%, videti Tabela 64. Razlike grešaka u pojedinim delovima eksperimenta nisu veće od 0.5%, što ukazuje na pouzdanost i preciznost predloženog modela. Ovo je trenutno jedan od najčešće korišćenih modela zbog izuzetnih rezultata procene koji se mogu postići njegovom primenom.

Tabela 64. Vrednost MMRE u sva tri dela eksperimenta (UCP).

Datasets	ANN-L16 MMRE(%)	ANN-L36prim MMRE(%)	Part of experiment
Dataset_1	6.7	7.0	Training
Dataset_2	7.1	7.1	Testing
Dataset_3	8.0	7.5	Validation1
Dataset_4	8.3	8.4	Validation2
AVERAGE(MMRE)	**7.5**	**7.5**	



Slika 54. Uticaj zavisnih promenljivih (UUCP and AUCP) na promenu vrednosti MMRE.

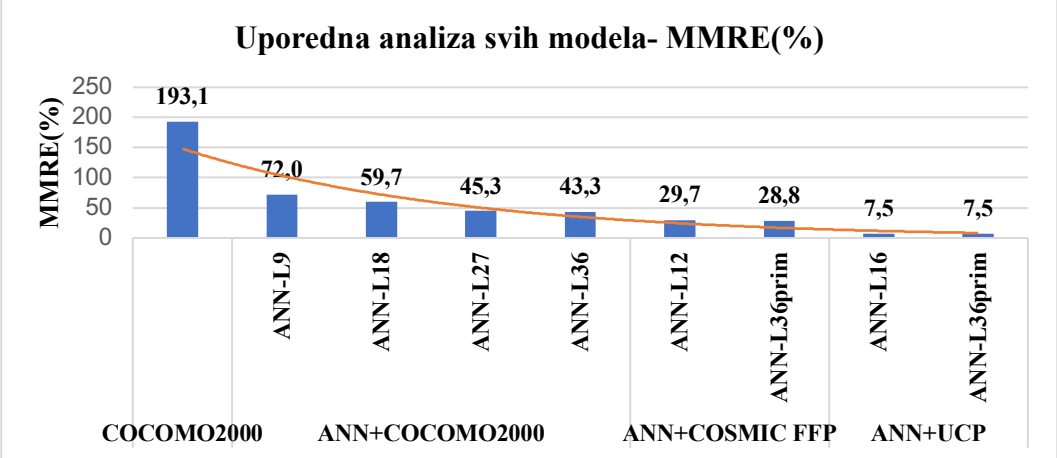
Dodatno je ispitivan uticaj dve zavisne promenljive UUCP i AUCP na promenu vrednosti MMRE, videti Slika 54. Kod obe ANN arhitekture ona je u intervalu od -0.3 do 0.5%.

Poređenjem dobijenih rezultata korišćenjem sva tri nova predložena poboljšana modela može se zaključiti:

- Najveća vrednost greške u proceni napora i troškova je u parametarskoj COCOMO2000 metodi i iznosi 193.1%;
- Najbolji rezultat, najniža vrednost MMRE u prvom predloženom modelu iznosi 43.3% što je 4.5 puta bolji rezultat u poređenju sa parametarskom metodom;
- U drugom predloženom modelu, greška na arhitekturi ANN-L36prim je 28.8% što je 6.7 puta bolji rezultat u poređenju sa parametarskom metodom;
- Kod trećeg modela, što je i najbolji rezultat u svim izvedenim eksperimentima, vrednost MMRE za obe predložene ANN arhitekture iznosi 7.5%, što je 25.7 puta bolji rezultat od parametarske metode.

U poređenju modela međusobno, videti Slika 55:

- Drugi predloženi model ANN+COSMIC FFP ima 1.6 puta manju grešku od prvog modela ANN+COCOMO2000;
- Treći predloženi model ANN+UCP ima 5.8 puta nižu grešku od prvog predloženog modela ANN+COCOMO2000;
- Treći predloženi model ANN+UCP ima 3.8 puta nižu vrednost MMRE od drugog predloženog modela ANN+COSMIC FFP.



Slika 55. Vrednosti MMRE u korišćenim pristupima.

Zaključak

U zavisnosti od veličine projekta za koji treba vršiti procenu napora i troškova, broja ulaznih veličina i drugih specifičnih faktora moguće je koristiti sva tri nova, poboljšana modela. Posle više realizovanih eksperimenata i analiza dobijenih rezultata, može se zaključiti da treći predloženi model daje najmanju vrednost relativne greške i ona iznosi 7.5%. Ako se uzme u obzir da je najmanji broj iteracija potrebnih za dostizanje

minimalne MMRE samo četiri, može se identifikovati najbolja ANN arhitektura, a to je ANN-L16 u okviru UCP modela.

Predloženi, novi, poboljšani modeli mogu poslužiti kao ideja za izgradnju jednog ili više alata koji će tačno, efikasno i pouzdano proceniti napor i troškove tokom različitih faza razvoja softverskog projekta. Pored toga, ove modele mogu da koriste softverske kompanije, softverski inženjeri i tehnički rukovodioci projekata, kako bi dobili brže i tačnije rezultate za pouzdanu, stabilnu i efikasnu procenu svih predviđenih zahteva za realizaciju projekata. Ovim se mogu dodatno smanjiti problemi sa kojima se stručnjaci i timovi najčešće susreću u ovoj oblasti softverskog inženjerstva.

Disertacija se sastoji iz 7 poglavlja, organizovanih na sledeći način:

Prvo poglavlje **1. Uvod** ima tri dela. U prvom delu **1.1** su prikazani problemi i dosadašnji načini u proceni napora i troškova kada je pitanju realizacija softverskog projekta. Drugi deo **1.2** prikazuje predmet istraživanja ove doktorske disertacije. Potom se detaljno opisuje šta je cilj i koji modeli će biti predstavljeni, na koji način će biti konstruisani i eksperimentalno proveravani. Treći deo **1.3** objašnjava značaj uvođenja i predstavljanja novih poboljšanih modela i zbog čega je to važno za ovu oblast softverskog inženjerstva. Potom se objašnjava i šira primena novih poboljšanih modela u drugim oblastima, ne samo u proceni napora i troškova za realizaciju softverskih projekata.

Drugo poglavlje **2. Dosadašnje metode procene napora i troškova na softverskim projektima** sadrži sedam delova i detaljno prikazuje različite metode i modele dosadašnjih procena napora i troškova. U prvom delu **2.1** objašnjava se primena različitih, do sada korišćenih, parametarskih metoda i navodi primer COCOMO2000 parametarske metode, sa kojom će se porediti i rezultati novih poboljšanih, predloženih modela. U drugom delu **2.2** prikazano je korišćenje i nekih drugih parametarskih i kombinovanih metoda kao što su linearna regresija i druge statističke procene i analize. Treći deo **2.3** je posvećen detaljnijem objašnjenju dosadašnjeg COCOMO2000 pristupa, čija je realna vrednost zasnovana na broju linija koda i različitim kombinovanim metodama. U delu **2.3.1** su objašnjeni svi parametri ovog pristupa i njihove kombinacije sa drugim metodama, algoritmima i tehnikama, koje bi dale bolje rezultate. Takođe se detaljno prikazuje koji se parametri mere, a bitno utiču na procenu napora i troškova, kako se oni računaju i šta predstavljaju ulazne veličine na osnovu kojih se izračunava stvarna vrednost projekta. Deo **2.3.2** predstavlja i opisuje dosadašnja istraživanja vezana za korišćenje COCOMO2000 modela i pristupa u mnogim istraživačkim i naučnim radovima. Četvrti deo ovog poglavlja **2.4** Analiza funkcionalnih tačaka, prikazuje dosadašnji pristup u korišćenju analize funkcionalne veličine sistema. Za razliku od prethodnog pristupa veličina sistema se ovde računa na osnovu određenih funkcionalnih tačaka koje se mere. U delu **2.4.1** se detaljno prikazuju ulazni parametri koji bitno utiču

na procenu napora i troškova, kako se koriste i šta predstavljaju ulazne veličine na osnovu kojih se izračunava stvarna funkcionalna veličina sistema. Deo **2.4.2** predstavlja i opisuje dosadašnja istraživanja vezana za korišćenje ovog modela i pristupa, kao i sve druge modele nastale kao rezltat poboljšanih prethodnih pristupa. Ističe se i značaj ovih modela kako u praksi tako i u mnogim istraživačkim, stručnim i naučnim radovima. U delu **2.5** je predstavljen treći, najčešće korišćeni pristup koji se zasniva na analizi korisnika i slučajeva korišćenja. U delu **2.5.1** se prikazuju svi parametri koji se mere kako bi se izračunala realna vrednost projekta. Detaljno su opisani i predstavljeni parametri koji utiču na realnu vrednost projekta i način na koji se ona adekvatno može izračunati. Deo **2.5.2** je posvećen dosadašnjim istraživanjima koja su vezana za predstavljanje i korišćenje ovog modela i pristupa i sve druge modele nastale kao rezltat poboljšanih prethodnih pristupa. Takođe, ističe se značaj ovih modela kako u praksi tako i u mnogim istraživačkim i naučnim radovima. U delu **2.6** Primena veštačkih neuronskih mreža je detaljno objašnjena uloga i mogućnosti primene savremenog alata veštačke inteligencije - veštačkih neuronskih mreža. Ovaj alat je pogodan za korišćenje kada nema određenih pravila po kojima bi se izračunavala konačna izlazna vrednost. Veštačke neuronske mreže imaju važnu ulogu u konstruisanju sva tri nova poboljšana predložena modela za procenu napora i troškova za realizaciju softverskih projekata, koji će biti predstavljeni u ovoj disertaciji. Deo **2.7** Tagučijev metod robusnog dizajna je posvećen izuzetno važnoj metodi optimizacije, koja značajno doprinosi unapređenju novih poboljšanih modela. Strategija robusnog dizajna podrazumeva i korišćenje ortogonalnih vektorskih planova za prikupljanje pouzdanih informacija o parametrima projekta sa izuzetno malim brojem eksperimenata. Ovaj metod se do sada koristio u drugim oblastima, a veoma retko u ovoj oblasti softverskog inženjerstva.

U poglavlju **3. Novi poboljšani modeli u okviru tri pristupa procene softvera** koje se sastoji iz tri dela detaljno su objašnjena i predstavljena sva tri nova konstruisana modela iz tri različita pristupa procene napora i troškova za razvoj softverskih projekata. U delu **3.1** prikazan je prvi, novi, poboljšani COCOMO2000 model, koji koristi četiri različite ANN arhitekture. Svaka arhitektura konstruisana je na osnovu odgovarajućeg Tagučijevog ortogonalnog vektorskog plana. Izbor vektorskog plana zavisi od broja ulaznih veličina i broja težinskih koeficijenata. Potom je odabrano pet različitih skupova podataka koji su opisani u delu **3.1.1** na kojima će se eksperiment izvršavati. Deo **3.1.2** prikazuje korišćenu metodologiju u sva tri dela eksperimenta: treniranje, testiranje i validacija i izvršava se korak po korak na osnovu datog algorima. Deo **3.2** prikazuje drugi, novi, poboljšani COSMIC FFP model, koji koristi dve različite arhitekture ANN. Svaka arhitektura konstruisana je na osnovu odgovarajućeg Tagučijevog ortogonalnog vektorskog plana. Izbor vektorskog plana zavisi od broja ulaznih veličina i broja težinskih koeficijenata. Potom je odabrano sedam različitih skupova podataka predstavljenih u delu **3.2.1** na kojima će se eksperiment izvršavati. Deo **3.2.2** prikazuje korišćenu metodologiju

u sva tri dela eksperimenta: treniranje, testiranje i validacija i izvršava se korak po korak na osnovu datog algorima. U delu **3.3** prikazan je treći, novi, poboljšani UCP model, koji koristi dve različite arhitekture ANN. Svaka arhitektura konstruisana je na osnovu odgovarajućeg Tagučijevog ortogonalnog vektorskog plana. Izbor vektorskog plana zavisi od broja ulaznih veličina i broja težinskih koeficijenata. Potom je odabrano četiri različita skupa podataka predstavljenih u **3.3.1** na kojima će se eksperiment izvršavati. Deo **3.2.2** prikazuje korišćenu metodologiju u sva tri dela eksperimenta: treniranje, testiranje i validacija i izvršava se korak po korak na osnovu datog algorima.

Četvrto poglavlje **4. Analiza dobijenih rezultata tri nova poboljšana modela** sadrži četiri dela. U delu **4.1** za novi poboljšani COCOMO2000 model, predstavljeni su rezultati svake iteracije svakog klastera za svaku od četiri predložene ANN arhitekture. Potom su analizirane vrednosti relativne greške, dobijene korišćenjem prikazanih ANN. Urađena je i statistička provera pouzdanosti modela kroz predikciju na tri kriterijuma i proverom sa dva koeficijenta korelacije. Dobijeni rezultati su bolji od prikazanih u prethodnim istraživanjima. Praćen je broj izvršenih iteracija i brzina konvergencije i rezultat ovog modela je do sada najbolji, što kao posledicu ima izuzetno brzo vreme estimacije. Na osnovu dobijenih rezultata moglo se zaključiti da je ANN-L36 arhitektura koja daje najnižu vrednost greške od 43.3%, a konvergira posle pete iteracije. Deo **4.2** za novi poboljšani COSMIC FFP model, prikazuje rezultate svake iteracije svakog klastera za svaku od dve predložene ANN arhitekture. Analizirane su vrednosti relativne greške dobijene korišćenjem dve različite ANN arhitekture. Urađena je i statistička provera pouzdanosti modela kroz predikciju na tri kriterijuma i proverom sa dva koeficijenta korelacije. Dobijeni rezultati su bolji od prethodno predstavljenih u sličnim istraživanjima. Najniža vrednost MMRE je 28.8% na arhitekturi ANN-L36prim. Praćen je i broj izvršenih iteracija i brzina konvergencije, i rezultati ovog modela su do sada najbolji u poređenju sa dobijenim rezultatima iz dosadašnjih istraživanja. Ovo za posledicu ima izuzetno brzo vreme estimacije. i konvergenciju posle pete iteracije. U delu **4.3** za novi poboljšani UCP model, predstavljeni su rezultati svake iteracije svakog klastera za svaku od dve predložene ANN arhitekture. Potom su analizirane vrednosti relativne greške dobijene korišćenjem različitih ANN. Urađena je i statistička provera pouzdanosti modela kroz predikciju na tri kriterijuma i proverom sa dva koeficijenta korelacije. Dobijeni rezultati su najbolji u poređenju sa do sada prikazanim u bilo kom prethodnom eksperimentalnom i naučnom istraživanju. Vrednost MMRE je 7.5%, dok arhitektura ANN-L16 konvergira posle četiri iteracije, što kao posledicu ima izuzetno brzo vreme estimacije.

Konačno, u delu **4.4** na osnovu analiza i poređenja dobijenih rezultata u različitim eksperimentima, identifikovan je najbolji model - treći, novi, poboljšani UCP model, koji daje najbolje rezultate. Potom je identifikovana i najbolja ANN-L16 arhitektura koja postiže minimalnu vrednost greške nakon samo četiri iteracije.

Poglavlje **5. Problem generalizacije**, se sastoji iz šest delova i predstavlja uopštavanje problema koji su se javljali tokom realizacije svih prethodno navedenih eksperimenata. Deo **5.1** daje odgovor na pitanje koliko projekata je potrebno za svaki klaster svakog skupa podataka u svakom delu eksperimenta. Upoređivanjem sa prethodnim istraživanjima ovaj broj treba da bude veći od broja težinskih koeficijenata korišćenog ortogonalnog vektorskog plana, što je u svim korišćenim skupovima podataka ove disertacije u velikom procentu ispunjeno. U delu **5.2** je rešen problem ispravno korišćene podele skupa podataka na klastere, korišćenjem uporedne analize svakog novog poboljšanog modela sa SVM (*engl.* Support Vector Regression) algoritmom koristeći proveru još jednim detrminističkim koeficijentom korelacije - R^2. Dobijeni rezultati pokazuju da je vrednost R^2 kod sva tri modela upoređivanjem sa tri različite funkcije jezgra RBF (*engl.* Radial Basis Function): linearnom, kvadratnom i kubnom veća od 0.9. Ovim se još jednom potvrđuje preciznost, pouzdanost i efikasnost tri nova, predložena, poboljšana modela. Deo **5.3** daje odgovor na pitanje kako i koju odgovarajuću ANN arhitekturu treba izabrati za sva tri dela eksperimenta. Polazi se od najjednostavnije ANN arhitekture bez skrivenog sloja, potom se uvodi jedan skriveni sloj sa najmanje dva čvora, zatim se povećava broj čvorova u skrivenom sloju arhitekture i na kraju se koristi arhitektura sa dva skrivena sloja. Odabir nove, složenije arhitekture se zaustavlja kada razlika vrednosti MMRE uzastopnih iteracija bude oko 1%. U delu **5.4** analizira se i prikazuje broj potrebnih iteracija svakog modela i ispituje brzina svih korišćenih ANN arhitektura. U delu **5.5** prikazuje se važnost odabira aktivacione funkcije koja daje najnižu vrednost greške, a pri tome konvergira ka najnižoj vrednosti sa minimalnim brojem iteracija.U delu **5.6** predstavljaju se pretnje po validnost modela, kao što su interna, eksterna i zaključna validnost.

Poglavlje **6. Primena predloženih modela i naučni doprinos** objašnjava koje su mogućnosti primene novih, predloženih modela ne samo u oblasti procene napora i troškova u softverskom inženjerstvu, već i drugim oblastima nauke i industrije. Novi, poboljšani modeli mogu preciznijom i efikasnijom procenom u velikoj meri povećati procenat uspešne realizacije projekata, a samim tim značajno unaprediti ovu oblast softverskog inženjerstva.

Poglavlje **7. Zaključak** prikazuje glavne zaključke svih eksperimenata, dobijenih rezultata, njihov značaj i primenu u praksi.

Kratka Biografija

Nevena Ranković je rođena 17.03.1994. godine u Valjevu. Srednju Matematičku gimnaziju je završila 2012. godine u Beogradu. Osnovne akademske studije Informatike, smer Informacione Tehnologije, upisuje 2012. godine, na Departmanu za Matematiku i Informatiku, Prirodno-matematičkog fakulteta, Univerziteta u Novom Sadu. U periodu od 2015. godine do 2016. godine pohađa predavanja master studija modula: Softversko Inženjerstvo na Tehničkom fakultetu u Štutgartu, Nemačka. Master akademske studije Informatike, smer Softversko Inženjerstvo, upisuje 2016. godine, na Departmanu za Matematiku i Informatiku, Prirodno-matematičkog fakulteta, Univerziteta u Novom Sadu. Master rad pod nazivom „Analiza rizika u upravljanju softverskim projektima" u saradnji sa istraživačkom stanicom "Petnica" u Valjevu brani 2018. godine. Doktorske akademske studije upisuje 2018. godine na istoj visokoškolskoj instituciji. Trenutno je student treće godine doktorskih studija. U periodu od 2018. godine do 2019. godine je birana u zvanje saradnika u nastavi i asistenta na Departmanu za matematiku i informatiku, Prirodno-matematičkog fakulteta, Univerziteta u Novom Sadu, za užu naučnu oblast Računarske Nauke. Trenutno, Nevena Ranković je izabrana u zvanje asistenta na Katedri za Računarske Nauke, Računarskog fakulteta, Univerziteta Union u Beogradu. Istraživač je na internom projektu Računarskog fakulteta, Univerziteta Union u Beogradu, pod nazivom: "Primena veštačke inteligencije u softverskom inženjerstvu". U svom naučno-istraživačkom radu bavi se: primenom veštake inteligencije u softverskom inženjerstvu, poslovnom inteligencijom, agilnim razvojem, veb tehnologijama, kvalitetom softvera, inženjerstvom zahteva, upravljanjem softverskim projektima, testiranjem softvera i optimizacionim problemima. Autor/koautor je 16 naučnih radova od kojih je 5 objavljeno u uglednim međunarodnim časopisima.

Short Biography

Nevena Rankovic was born in Valjevo, Serbia, in 1994. She received the B.Sc. and M.Sc. degrees in Informatics, Software Engineering from the University in Novi Sad, Faculty of Sciences, Department of mathematics and informatics, in 2015 and 2018, respectively. From 2015 to 2016, she attended the master study module lectures: Software Engineering at the Technical Faculty in Stuttgart, Germany. From 2018 to 2019 she worked as a teaching assistant at the University of Novi Sad, Department of mathematics and informatics, and also enrolled in Ph.D. studies. Currently, she is a teaching assistant at Union University, School of Computing Belgrade on subjects like Software Testing, Software Engineering, and Software Project Management. Her research areas are Software Engineering, Applied Artificial Intelligence, Business intelligence, Agile Development, Web Technologies, Software Quality, Requirements Engineering, Software Project Management, Software Testing, and Software Metrics. She is the author/co-author of 16 scientific papers (5 in respective scientific journals).

Овај Образац чини саставни део докторске дисертације, односно докторског уметничког пројекта који се брани на Универзитету у Новом Саду. Попуњен Образац укоричити иза текста докторске дисертације, односно докторског уметничког пројекта.

План третмана података

Назив пројекта/истраживања
Estimation of effort and costs in the development of software projects using artificial neural networks based on Taguchi's orthogonal vector plans (Процена напора и трошкова за развој софтверских пројеката помоћу вештачких неуронских мрежа заснованих на Тагучијевим ортогоналним векторским плановима)
Назив институције/институција у оквиру којих се спроводи истраживање
а) Универзитет у Новом Саду, Природно-математички факултет
Назив програма у оквиру ког се реализује истраживање
1. Опис података
1.1 Врста студије *У овој студији су коришћени јавно доступни скупови података.* *Линкови:* [1] доступно на следећем репозиторијуму: http://promise.site.uottawa.ca/SERepository/datasets-page.html [2] доступно на следећем репозиторијуму: https://www.isbsg.org/ за приступ је потребно чланство [3] доступно на следећем репозиторијуму: https://data.mendeley.com/datasets/2rfkjhx3cn/1
2. Прикупљање података
3. Третман података и пратећа документација
4. Безбедност података и заштита поверљивих информација

5. Доступност података
6. Улоге и одговорност