SSRG International Journal of Computer Science and Engineering
ISSN: 2348–8387 / https://doi.org/10.14445/23488387/IJCSE-V10I9P102

Volume 10 Issue 9, 10-18, September 2023

Original Article

Software Engineering Project Life Cycle Modeling Based on Neural Network Technologies

Amirali Kerimovs

Independent Researcher, Riga, Latvia.

Corresponding Author : amir.academicinquiry@gmail.com

Received: 24 July 2023 Revised: 26 August 2023 Accepted: 14 September 2023 Published: 30 September 2023

Abstract *- The article presents a classification and analysis of teams in the field of Software Engineering and the feasibility of their use, depending on the conditions of an externally dynamically changing environment. The role of the project life cycle automation system in the field of Software Engineering is given. It indicates how important attention to resource management is to ensure the effective, timely, and high-quality implementation of project tasks. The article presents a complex socio-technical safety-oriented system, describes its purpose, and the main functions for controlling the occurrence of errors in the execution of tasks. The importance of neural network risk management in such projects in which they operate under conditions of uncertainty due to the neglect of initiation processes and other phases of the life cycle. This research introduces the novel application of neural networks in software engineering project evaluation, highlighting their adaptability and precision. By strategically integrating neural networks into project stages, the study aims for more accurate assessments and offers customizable analysis options. This innovative approach aligns with the modern digital landscape, emphasizing the synergy between neural network technologies and well-structured project lifecycles, ultimately improving project management and decision-making in software engineering.*

Keywords *- Project life cycle, Software engineering, Neural network, Project control software, Data formalization.*

1. Introduction

New external challenges and the integration of the Software Engineering system into artificial intelligence implementation standards encourage the implementation of innovative projects (automation of activities, process reengineering, infrastructure development), as well as the use of flexible, adaptive management to improve the interaction between departments, efficient distribution, and management of resources. An analysis of well-known project management methodologies (IPMA, PMI, P2M, PRINCE) showed that a characteristic feature of modern methodologies is the use of limited project life cycle models. Since project management in the field of Software Engineering is one of the most significant and, at the same time, poorly formalized processes, the development of new life cycle models is relevant. An important component in implementing the content of new projects, programs, and project portfolios in the field of Software Engineering is the formation of project resource portfolios, the structure of which should meet the goals and objectives of the project.

2. Literature Review

2.1. Project-Oriented Management

The problems of project-oriented management in complex systems have been studied by many scientists [7-11]. The life cycle and group dynamics of the design cycle in the field of Software Engineering have also been studied by foreign researchers [1-6].

In [5], the management and methodology of resource management of projects and programs were analyzed with an assessment of the scale of the Harrington desirability function. These methods do not fully describe the life cycle of using resource blocks, which have their own characteristics and are a dynamically complex system.

The work [1] reflects the important issues of the phases and groups of periodization of project knowledge management, which, to a large extent, affects the achievement of success in projects and programs. However, attention is not focused on the study of the relationships of the life cycle of stakeholders.

In addition, taking into account the insights from reference [3] holds significant relevance. The author prominently underscores the pivotal nature of the stakeholder identification phase within a project, in tandem with methodologies geared towards effective team management and judicious allocation of project resources. A comprehensive delineation of the various phases

encompassing the project life cycle within the realm of Software Engineering is thoughtfully provided. Moreover, the author substantiates these concepts through illustrative instances showcasing the prosperous operational dynamics of distinct organizations.

The insights encapsulated within this referenced work stand as a potent foundation, warranting further in-depth investigation. The current study recognizes the merits of this contribution and systematically analyzes its pertinence in elucidating the intricacies of the project life cycle. The aforementioned work thus serves as an invaluable underpinning for the framework developed in this study, enriching the discourse around life cycle modeling.

2.2. The use of Neural Networks for the Correct Planning and Execution of a Project in the Field of Software Engineering

The use of neural networks to predict the results of the correct planning and execution of a project in the field of Software Engineering, according to [4], has several advantages. First, neural networks can analyze large amounts of data and identify hidden dependencies that may be missed by other data analysis methods.

Secondly, neural networks can learn from data and adapt to environmental changes, making achieving high prediction accuracy possible. Thirdly, using neural networks can simplify the forecasting process since knowledge of specific mathematical methods is not required.

However, these studies have not solved the scientific problem of modeling weakly formalized phases of the life cycle of project development in the field of Software Engineering.

3. Materials and Methods

Each element of the project in the field of Software Engineering has a different typology but is aimed at achieving a common goal and tasks. Project management functions *f(pm*), which involve a variety of resources, include:

- *f1* – human resource planning function;
- *f2* – control function of technical resources of the project;
- *f3* - the function of implementation and development of the project team;
- *f4* – execution control function;
- *f5* - project management function.

To build a neural network designed to create software (software) to predict the results of a project in the field of Software Engineering at each stage of the life cycle, it is necessary to take into account the scheme of standard attributes of its implementation:

- Operand Type: Specifies the type of operand or characteristic used to analyze or predict the results of the system.
- Operand value: Represents the numeric or text value of the operand, which is used in neural network algorithms to predict results.

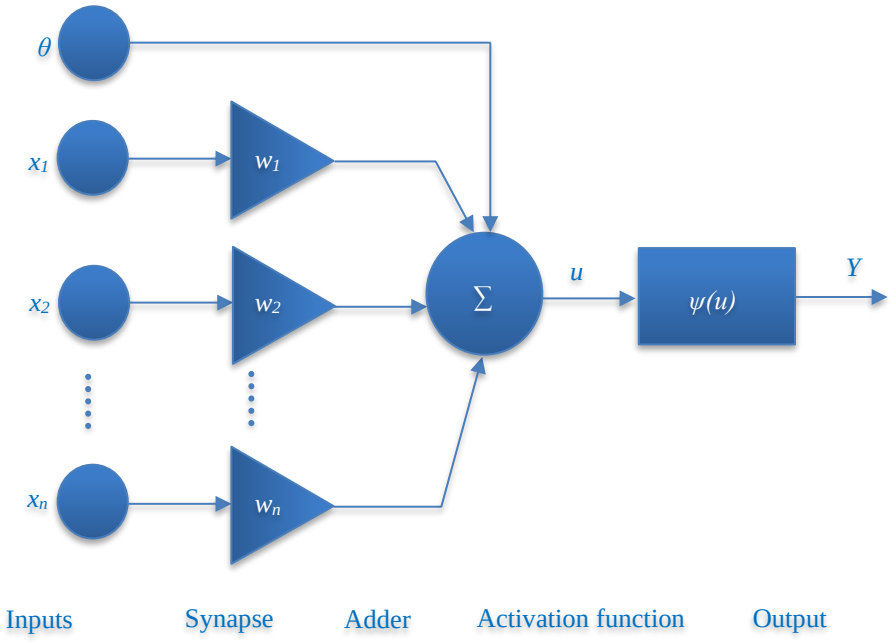

Fig. 1 The general structure of an artificial neuron for analyzing the results of a project in the field of Software Engineering at each stage of the life cycle

Using these attributes and operands, you can develop a data structure and prepare a training set that includes historical data about projects, resources, and other related attributes. This data can then be used to train a neural network to predict the results of a Software Engineering project at each stage of the life cycle.

The neural network can be configured to analyze relationships between project attributes and life cycle operands in order to predict different outcomes and optimize the contract management processes of a heat and power organization.

For example, a neural network can analyze and predict project resource consumption based on data from various systems, such as control and monitoring systems. It can analyze the relationship between these data and determine the best solutions for managing the consumption of human and technical resources. Then, its integral part is an artificial (formal) neuron, the structure of which is generally shown in Fig. 1.

To create a mathematical model for evaluating the results of a project in the field of Software Engineering at each stage of the life cycle, it is necessary to determine a set of input parameters (X) and develop a forecasting model (F(X)). Based on the predicted value, management decisions can be made, for example, to optimize the processes for completing the tasks of each cycle, to improve the quality of the use of technical resources, etc.

The mathematical model of a neuron can be represented as:

$$Y = \psi\left(\sum_{i=1}^{n} w_i x_i + \theta\right) \quad (1)$$

Where x_i is the value of *the i* -th signal coming to the input of the neural network; w_i is the weight coefficient *of i* - th synapse; n is the number of inputs; θ – shift signal (threshold value); *Y is* the value of the output signal.

Then, the formula for the neural network might look like this:

$$Y = f(w_1 x_1 + w_2 x_2 + w_3 x_3 + ... + wn\,xn + \theta) \quad (2)$$

Where:

- x_1, x_2, x_3,..., x_n are the values of the project stage process specification ID attributes.
- w_1, w_2, w_3,..., w_n - weight coefficients the neural network will determine in the learning process.
- θ - shift signal (threshold value), which determines the neuron's activation threshold.
- *f* is an activation function that is applied to the total input of the neuron to generate an output value of *Y*.

The rationale for integrating neural networks into evaluating individual stages within an engineering project is rooted in their inherent capacity for adaptability and contextual understanding. The selection of an optimal activation function, the meticulous determination of hidden layers, and the precise configuration of neurons within the neural network are contingent upon the nuanced exigencies of the specific project and its distinctive demands.

A meticulous mathematical model is meticulously crafted by harnessing the neural network's intrinsic ability to decipher complex patterns and relationships. This model is an intellectual apparatus adept at tackling multifaceted challenges inherent to the project's progression. At its core, this endeavor hinges upon the meticulous construction of a neural network model – a model purposefully devised to execute a nonlinear approximation of the intricate mapping function, F(X), linking the ensemble of input variables, denoted as X, to the ultimate output, encapsulated as Y.

In its dynamic and adaptable nature, the neural network emerges as a powerful tool capable of transcending linear limitations. Its deployment within each stage of the engineering project infuses a layer of sophisticated intelligence. By facilitating a multifaceted analysis, the neural network encapsulates the intricate interplay between diverse variables, thereby offering insights that transcend the confines of traditional methodologies. This infusion of AI-driven prowess significantly elevates the efficacy and comprehensiveness of evaluating each distinct facet of the project lifecycle.

As such, the strategic integration of neural networks into the evaluation of engineering project stages is underpinned by a quest for enhanced accuracy, holistic comprehension, and the nuanced capacity to decipher intricate relationships within the complex tapestry of project dynamics.

In the literature, various approaches have been employed to estimate software development efforts accurately. In recent years, attempts have been made to apply machine learning approaches, made possible by the availability of datasets from a large number of completed projects. Among different machine learning methods, models based on neural networks are emerging as novel approaches.

While some researchers [7-11] have worked on neural network-based models, there is still a need for further research and attempts to find the most suitable model for software effort prediction in terms of accuracy, reconfigurability, and model suitability rationale. A comprehensive review of 21 articles related to software effort and cost estimation using neural networks is presented in Lopez-Martin [12].

Wen and colleagues [13] conducted a comprehensive systematic review that delved into the realm of machine learning applied to Software Development Effort Estimation (SDEE) spanning the years from 1991 to 2010. Their findings resoundingly affirm the promising potential of machine learning models within the domain of SDEE. A particularly notable revelation from their study was the prominent standing of artificial neural network models, securing the second position in terms of the number of publications selected for analysis.

Now, let's dissect the pivotal significance of software effort and cost estimation within the context of software development projects. This process is undeniably a linchpin of software project management, holding paramount importance. Precisely estimating the effort required for software development is not merely a managerial chore; it is the compass guiding sound decision-making. It becomes especially critical as the intricacies and complexities of software projects increase, typically leading to escalating development costs. Consequently, the software engineering community has long been on a quest to formulate apt models for accurate software effort estimation.

Despite a wealth of research in this domain, achieving pinpoint accuracy in predicting development costs has remained an elusive goal. The software community has forged a repertoire of unique tools and techniques dedicated to tackling the challenges inherent in managing software development projects.

These tools and techniques come into play at various phases of software development, commencing with the software requirements specification. As the demand for software applications continues to soar, coupled with the ever-increasing complexity of software, the dire need for precise project estimates becomes more apparent than ever.

Indeed, the importance of reliable software effort estimates extends beyond the confines of software companies. They serve as foundational elements in diverse aspects of project management, including Request for Proposals (RFPs), contract negotiations, project planning, monitoring, and control. However, it is worth noting a sobering reality – as reported by Molokken and Jorgensen in their estimation studies review, software projects typically demand 30-40% more effort than initially estimated [14].

Comparing neural networks for the software engineering project lifecycle involves evaluating and contrasting the effectiveness of different neural networks when applied at various stages of the software engineering project's lifecycle. Data formalization in this context refers to the process of transforming and structuring information used within neural networks to ensure more accurate and organized analytical utilization of this information within the project. To train a neural network to evaluate the results of a Software Engineering project at each stage of the life cycle, various algorithms are used, such as backpropagation (backpropagation) and support vector machines (support vector machines). After training the neural network, you can use the resulting mathematical model to predict the work of the contracts department of a heat and power organization based on new input data.

A block diagram of a generalized algorithm for constructing a neural network model for the contracting department of an organization is shown in fig. 2 with the following designations: 1 – transition to the parametric synthesis stage; 2 – transition to the stage of structural synthesis; 3 – stage of preliminary data preparation.

The model's intricate technical attributes revolve around the seamless amalgamation of diverse systems and cutting-edge technologies, establishing a synergistic ecosystem of unparalleled functionality. A prime exemplification of this symbiotic integration is witnessed in the seamless interaction with the SAP Unified Information System – a feat achieved through a judicious implementation of an array of protocols and standards, including but not limited to RFC, SOAP, REST, and the like. This intricate interplay of established communication frameworks serves as the linchpin for cohesive data exchange and optimal system synchronization.

Furthermore, the model's technical blueprint is artfully shaped by a nuanced cognizance of the idiosyncrasies underpinning databases and data storage systems. The orchestration of machine learning algorithms and neural networks, pivotal in the model's operational prowess, finds its manifestation through the robust canvas of the Python programming language. This language, revered for its versatility and expansive capabilities, is a formidable conduit for executing advanced data analysis and machine learning endeavors.

Within this technological tapestry, the seamless union of the model with specialized libraries engenders a harmonious convergence of intellectual prowess and computational finesse. The likes of NumPy, Pandas, Scikit-learn, and TensorFlow are harnessed to their fullest potential, endowing the model with an exceptional arsenal of tools designed to navigate the complex intricacies of data manipulation, algorithmic inference, and neural network training.

The technical landscape thus emerges as an intricate mosaic where each component is delicately interwoven, meticulously calibrated to actualize a model that transcends conventional boundaries. This symbiotic marriage of systems and technologies encapsulates a relentless pursuit of technical excellence, forging a model that stands as a testament to the harmonious integration of cutting-edge methodologies and time-honored standards.

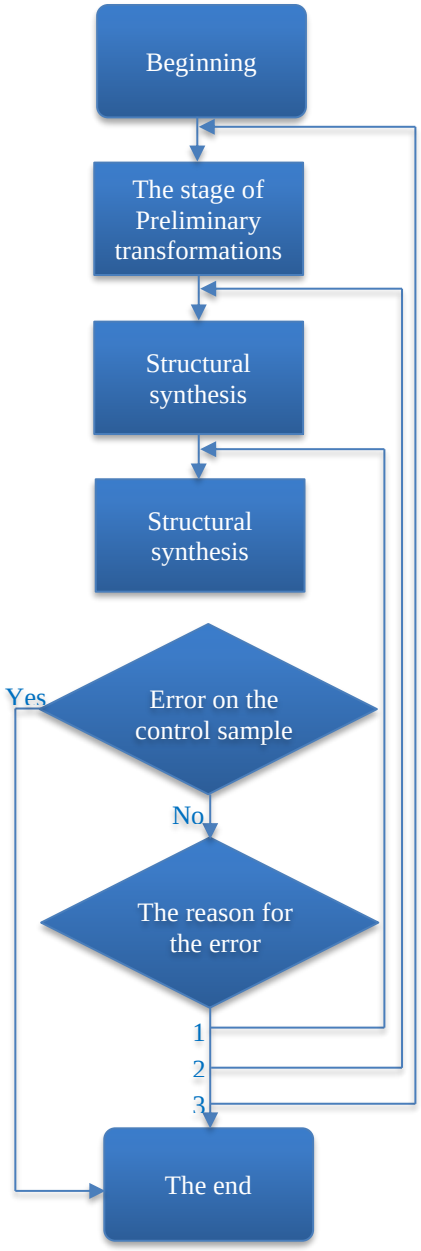

Fig. 2 Basic block diagram of the algorithm for building a neural network model for analyzing the results of a project in the field of Software Engineering at each stage of the life cycle

4. Results and Discussion

4.1. Empirical Data

There are two main types of teams in Software Engineering: cross-functional and intact teams (Table 1). The life cycle of the formation of the CK team should be flexible and adaptive to external factors and influences. From the point of view of the system approach, adaptation is the process of changing the parameters and structure of the system, in particular, control actions, based on current information to achieve a certain, usually optimal, adaptive one.

However, human resource planning is fully effective only if integrated into the joint strategic planning process and is not aimed at meeting only current needs (Fig. 3).

Fig. 3 Gantt chart of project lifecycle management in software engineering

The combination of the work structure (WBS) and the hierarchical structure of the CK team allows you to plan, control the work, and measure its performance by individual departments and the organization as a whole. Each manager in this hierarchy has his own set of plans and reports according to his area of responsibility. Therefore, to build a model, BPMN (Business Process Model and Notation) tools can be used, which allows you to describe business processes in a graphical form and determine their relationship and dependencies. Based on expert assessments, we will create data on software development for this project. Calculation of the generalized criterion of the desirability function based on the data of the project for evaluating the results of a project in the field of Software Engineering at each stage of the life cycle and matrices of correspondence of performance indicators to the Harrington desirability function scale are presented in Table 1.

Table 1. Calculation of the generalized criterion of the desirability function

Resource types	1	2	3	4	5
A) labor resources	0.550	0.550	0.715	0.285	0.715
B) information resources	0.715	0.550	0.550	0.285	0.900
C) material resources	0.715	0.715	0.715	0.550	0.550
D) financial resources	0.900	0.900	0.715	0.715	0.285
E) intellectual resources	0.900	0.900	0.715	0.715	0.285

The initiation of the Analysis Package becomes a pivotal step for the effective utilization of a software suite designed for conducting regression analysis. This foundational activation serves as the gateway to a realm where data-driven insights unravel and project dynamics are meticulously scrutinized through the lens of advanced modeling.

Within this analytical domain, an array of supplementary configurations stands at the ready, offering an avenue for tailoring the analysis process to the specific contours of the project at hand. These configurations extend to encompass pivotal parameters, including the assignment of labels, the calibration of confidence levels, the consideration of constant-zero paradigms, and the visualization of a normal probability plot, among other versatile functionalities.

However, it is noteworthy that a substantial majority of instances necessitate minimal alterations to these configuration settings. The default parameters are adeptly engineered to seamlessly accommodate a wide array of scenarios, engendering a frictionless analytical voyage.

Yet, amidst this ensemble of settings, a prudent focal point emerges—directed toward the outcome parameters. These outcome variables serve as the crux upon which the efficacy of the analysis hinges. Notably, by default, the resultant analysis findings manifest on a distinct sheet, a deliberate design choice to ensure uncluttered representation.

However, a potent customization avenue exists, poised to cater to divergent analytical preferences. By merely manipulating a toggle, the outcome of the analysis can be elegantly poised within a designated range on the same sheet that harbors the foundational dataset. This not only streamlines the analytical presentation but can also extend to crafting an independent archival record residing in a novel file.

4.2. Software Implementation Data

In essence, the orchestration of these software-driven maneuvers harmonizes with the overarching theme of engineering project life cycle modeling entrenched in neural network technologies. This juxtaposition of analytical prowess and sophisticated customization embodies the ethos underpinning the innovative exploration of software engineering within the contemporary digital landscape.

The culmination of the regression analysis endeavor manifests in a structured tableau, meticulously positioned as per the designated configuration settings. At the heart of this analytical revelation resides a paramount metric - the R-square coefficient. Serving as the vanguard of model evaluation, this numerical emblem meticulously gauges the efficacy of the meticulously constructed model. In our specific context, this pivotal coefficient emerges as 0.705, which equates to approximately 70.5%.

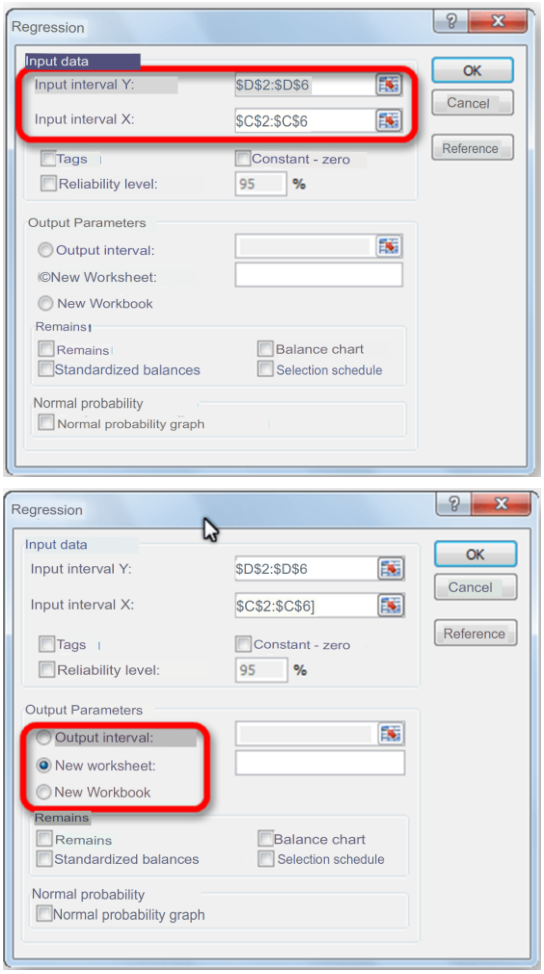

Fig. 4 Calculation of regression *X* for the attributes of the neural network model

This discerning evaluation crystallizes into an acceptable echelon of quality, signifying a commendable alignment between the modeled predictions and the empirical actualities. Importantly, it underscores the magnitude of the relationship held within the intricate contours of the data. A value surpassing the 0.5 thresholds is emblematic of an efficacious modeling endeavor, while anything below this benchmark denotes a shortfall in the quality of the modeled relationship.

Yet, this analytical odyssey unveils another linchpin revelation, adroitly positioned at the intersection of the "Y-intercept" lineage and the "Coefficients" column. This spatial nexus proffers a tangible insight into the imminent projection of Y – an emblem of the dependent variable. Within the ambit of our current tableau, this consequential cell yields a numeric value of 58.04, signifying the anticipated manifestation of the dependent variable amidst the grand tapestry of the project.

The tableau's scrutiny extends further, inviting a gaze upon the juncture where the "Variable X1" intersects the

formidable "Coefficients" column. This interface bespeaks numerical verification and heralds a narrative of dependence, unraveling the intricate dance between Y and its interdependent variable X1. A coefficient of 1.31 presides as a beacon of influence, eloquently delineating the potency of X1 in sculpting the contours of Y's manifestation. This commendable magnitude affirms a substantial level of impact, underscoring the gravitas of X1 in the realm of this meticulously engineered model.

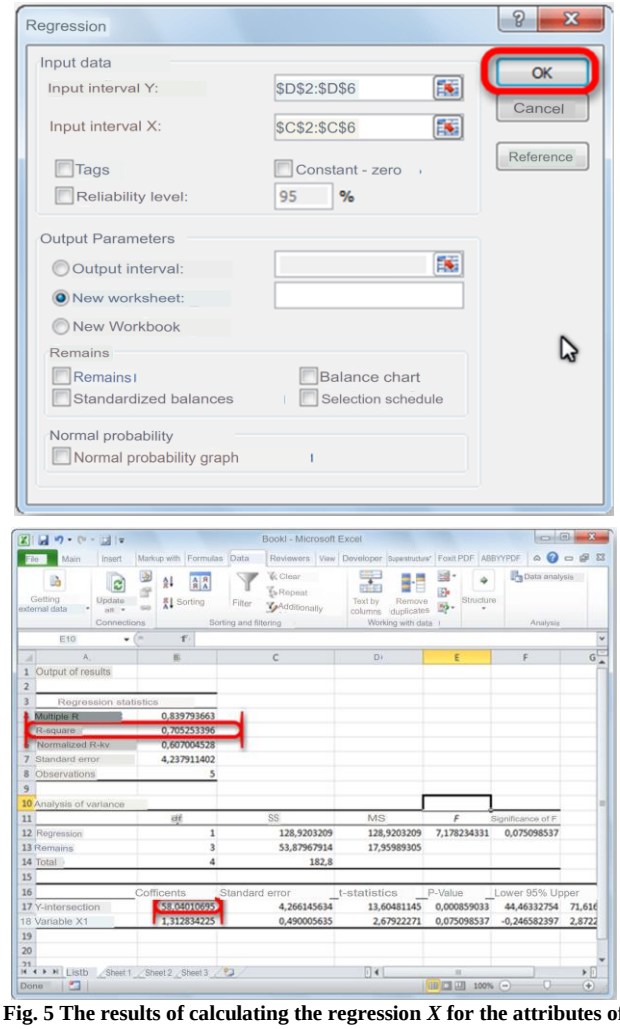

Fig. 5 The results of calculating the regression *X* for the attributes of the neural network model

In essence, this nuanced analysis is a testament to the pivotal synergy between the chosen neural network technologies and the meticulously orchestrated life cycle of the software engineering project. It encapsulates the very essence of the theme, unfurling a panoramic vista into the symbiotic fusion of cutting-edge methodologies and analytical finesse.

When comparing the work of the versions of the integration functionality of the CIS of the heat supply organization and the system, it was found that the average time for the conclusion of the project of the old version of the integration, taking into account the correction of various kinds of errors is approximately 64 hours (8 working days). At the same time, the average time to conclude a project for a new version of integration, taking into account the addition of a new stage in the process of concluding projects and correcting various kinds of errors, is approximately 39 hours (5 working days).

5. Conclusion

An important stage of the project is its quality planning. A project resource management plan should be developed, taking into account its labor intensity, resource limitations, even distribution of tasks among performers, and complexity. The article considers a neural network approach to describing the life cycle of complex systems of a safety-oriented system. Thanks to this research, we have developed a schematic life cycle model in complex systems for evaluating the results of a Software Engineering project at each life cycle stage, which will allow further research to automate the processes of formation and management of resources.

The research described in the provided text focuses on the novel use of neural networks for project planning and execution in the field of software engineering. The study aims to highlight the advantages of integrating neural networks into project evaluation at different stages of the project lifecycle. It emphasizes the adaptability and contextual understanding inherent in neural networks.

Key highlights of the research and its achievements compared to existing solutions include:

- Adaptive Neural Networks: The research harnesses the adaptive capabilities and contextual understanding of neural networks. This adaptability allows for the selection of optimal activation functions, careful determination of hidden layers, and precise neuron configuration tailored to the specific requirements of each project.
- Enhanced Precision: The strategic integration of neural networks into project stage evaluation aims to achieve higher accuracy in project assessment. Neural networks excel at deciphering complex relationships within the dynamic project landscape, contributing to a more accurate analysis.
- Structural Algorithm: The research proposes a structural algorithm for building neural network models to analyze project results at various stages of the software engineering lifecycle. This algorithm combines the Work Breakdown Structure (WBS) and the hierarchical team structure to plan, monitor, and measure departmental and organizational performance. This integration leverages BPMN tools for creating a generalized criteria function for project result assessment.

- Customizable Analysis: The research highlights the ability to customize the analysis results, offering flexibility to meet various analytical preferences. The option to align the analysis results within a specified range on the same sheet as the basic dataset simplifies analytical representation and supports the creation of independent archival documents.
- Orchestration of Software-Driven Maneuvers: The research aligns with the overarching theme of modeling the lifecycle of engineering projects using neural network technologies. This combination of analytical expertise and complex configuration embodies the essence of innovative software development research in the modern digital landscape.
- Regression Analysis and R-Squared Coefficient: The culmination of the research is a structured table arranged according to specified configuration parameters. The primary metric used is the R-squared coefficient, which rigorously evaluates the model's effectiveness. In this specific context, the coefficient's value of 0.705 indicates a commendable level of correspondence between modeled predictions and empirical facts.
- Y-Intercept and Coefficients: The research reveals a significant revelation at the intersection of the "Y-Intercept" and "Coefficients" columns. This spatial relationship provides insight into the inevitable projection of the dependent variable emblem, illustrating the synergy between neural network technologies and a well-organized software development project lifecycle.

In summary, this research represents an innovative approach to project evaluation in software engineering. Integrating neural networks achieves higher precision, adapts to project-specific requirements, and leverages advanced analytical techniques. The study's results and methodologies open new avenues for improving project management and decision-making processes within the software engineering domain.

References

[1] Pedro José Bernalte Sánchez, Fausto Pedro García Márquez, and Mayorkinos Papaelias, "Life Cycle Sustainability Assessment of ENDURUNS Project: Autonomous Marine Vehicles," *E3S Web of Conferences*, vol. 409, pp. 1-11, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[2] Illia Gorbatiuk, "Peculiarities of Project Management in the Context of Software Engineering," *Scientific Papers of Berdiansk State Pedagogical University Series Pedagogical Sciences*, vol. 3, no. 3, pp. 267-274, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[3] Kalinina Aksinya Vasilievna, and Petrochenko Marina Vyacheslavovna, "An Integrated Approach to the Assessment of Construction Life Cycles Using Software Packages at the Design Stage," *Stroitel'stvo: Nauka I Obrazovanie*, vol. 12, no. 1, pp. 88-100. [Google Scholar] [Publisher Link]

[4] V.V. Liubchenko, "Some Aspects of Software Engineering for AI-Based Systems," *Problems in Programming*, no. 3, pp. 99-106, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[5] Christine Lucas-Lamouroux et al., "Model-Based System Engineering, an Industrialization Path for Decommissioning Projects by ASSYSTEM," *EPJ Nuclear Sciences and Technologies*, vol. 9, no. 24, pp. 1-7, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] G.O. Mirskikh et al., "Life Cycle of the Software of Engineering Objects," *Енергетика І Автоматика*, no. 5, pp. 115-128, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] V. Venkatachalam, "Function Point Analysis: An Empirical Study," *IEEE Transactions on Software Engineering*, vol. 19, no. 10, pp. 985-996,1993.

[8] R. Tadayon, "Software Effort Estimation Using a Neural Network-Genetic Algorithm Hybrid Model," *In Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 314-321, 2005.

[9] A. Ivata, J.R. Amorim, and R.A. Falbo, "A Genetic Algorithm-Based Method for Software Effort Estimation Using Fuzzy Logic," *In Proceedings of the 2009 International Conference on Information Reuse and Integration*, pp. 315-320, 2009.

[10] R. Dave, and S. Dutta, "A Neural Network Approach to Software Cost Estimation," *International Journal of Computer Applications*, vol. 24, no. 9, pp. 29-35, 2011.

[11] M. Shepperd, and J. McDonnell, "A Critique of Software Defect Prediction Models," *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1216-1228, 2012.

[12] C. Lopez-Martin, "Software Effort Prediction Based on Computational Intelligence Techniques: A Systematic Review," *Information and Software Technology*, vol. 56, no. 10, pp. 1246-1265, 2014.

[13] Jianfeng Wen et al., "Systematic Literature Review of Machine Learning Based Software Development Effort Estimation Models," *Information and Software Technology*, vol. 54, no. 1, pp. 41-59, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[14] Magne Jørgensen, "Forecasting of Software Development Work Effort: Evidence on Expert Judgment and Formal Models," *International Journal of Forecasting*, vol. 23, no. 3, pp. 449-462, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[15] Andrew Quansah et al., "Requirement Engineering Problems Impacting the Quality of Software in Sub-Saharan Africa," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 1, pp. 350-355, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] Shilkina Svetlana, and Ivanova Olga, "The Choice of Software for the Implementation of Projects Based on Information Modeling Technologies," *Construction and Architecture*, vol. 11, no. 2, 2023. [CrossRef] [Publisher Link]

[17] K.S. Sharanya Shrivathsa et al., "An Intranet-Based Project Life-Cycle Management," *International Journal of Computer and Organization Trends*, vol. 7, no. 4, pp. 27-31, 2017. [Publisher Link]

[18] Shulga, Tatiana & Khramov, Dmitrii. (2023). Life cycle ontology of software engineering. Vestnik of Astrakhan State Technical University. *Series: Management, computer science and informatics.* 2023. 66-74. 10.24143/2072-9502-2023-2-66-74.