Method for determining the security level of software

Tetiana Hovorushchenko[1,*,†], Yurii Voichur[1,*,†], Dmytro Medzatyi[1,†], Artem Boyarchuk[2,†] and Alina Hnatchuk[1,3,†]

[1] *Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, 29016, Ukraine*
[2] *Tallinna Tehhnikaülikool, Ehitajate tee, 5, Tallinn, 12616, Estonia*
[3] *Prague University of Economics and Business, Winstona Churchilla str., 1938/4, Prague, 13067, Czech Republic*

Abstract

Currently, there is an increase in the complexity of software, an increase in the responsibility assigned to it, and tightening requirements for software quality and security on the part of users, so predicting and determining the level of software security (as one of the characteristics of software quality) based on requirements using AI components is an urgent task, the solution of which is the purpose of this study. The analyzed AI-based methods and tools for predicting the level of software security and quality have great potential, but they do not establish the dependence of software security on quality attributes, do not form a predicted numerical value of software security based on attributes, and do not provide a prediction of the level of software security based on the obtained numerical value. The developed method for determining the security level of software establishes the dependence of software security on quality attributes, forms a predicted numerical value of software security based on attributes, provides a prediction of the level of software security based on the obtained numerical value, and provides a comparison of software requirements specifications by predicted level of security of developed software (of course, if the bugs are not made at the next lifecycle stages) and a possibility of rejection form unsuccessful specifications.

Keywords

Software security, software security level, initial level of security, medium level of security, sufficient level of security, high level of security, software quality, software quality characteristics, software security subcharacteristics, software quality attributes

1. Introduction

Now various business and industrial enterprises use software. The need of software is emerged through the existing technological breakthroughs. The need of software increase for a wide range of enterprises and economy sectors. “Statista” shows that software spending is currently estimated at USD 491 billion, and the size of the USA software market is USD 285 billion [1, 2].

Under such circumstances, when software engineering is crucial and software becomes more complex, creating high-quality software is the most crucial undertaking for the software

IntelITSIS’2024: 5th International Workshop on Intelligent Information Technologies and Systems of Information Security, March 28, 2024, Khmelnytskyi, Ukraine
* Corresponding author.
† These authors contributed equally.
tat_yana@ukr.net (T. Hovorushchenko); voichury@khmnu.edu.ua (Y. Voichur); medza@ukr.net (D. Medzatyi); a.boyarchuk@taltech.ee (A. Boyarchuk); alinahnatchuk@ukr.net (A. Hnatchuk)
0000-0002-7942-1857 (T. Hovorushchenko); 0000-0003-3085-7315 (Y. Voichur); 0000-0002-1879-2945 (D. Medzatyi); 0000-0001-7349-1371 (A. Boyarchuk); 0000-0003-0155-9255 (A. Hnatchuk)
© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

domain's expansion and high reputation, as software quality is a growing concern for users. The stakeholder satisfaction is associated with software quality assurance, because if the consumer is pleased with the product, then this product is quality. According to standard, software quality shows the degree to which the software meets the user's needs in the different conditions [3].

Nowadays, software has become one of the most expensive industries, and any bottlenecks in the development process can lead to undesirable consequences. Multiple software faults and failures still occur uninvited due to errors and defects remaining in the software. For every invested $1 billion, software companies lose an average of $97 million due to poor software quality [4]. The price of low-quality software in 2020 amounted to 260 billion US dollars (compared to 177.5 billion US dollars in 2018) [5]. In general, about 10-20% of all software projects are not completed, 40-60% of projects are completed 150-200% late, 40-55% of projects require additional costs, 25-40% of projects do not fully realize their objectives, 20% of projects do not take into account all changes on the part of the customer [6].

Because it is impossible to find and fix all software defects, it is important that the negative effect of defects will be detected, resolved, and minimized as early as possible. The early finding and fixing the software defects provide minimizing the damage caused by software defects, because the amount of time and money increases when there are defects in the software. Although software defects monitoring and repair are also both costly and time-consuming to complete procedures.

So, the software quality assurance is usually results in more money and time. The software developers generally consider software quality assurance as an additional lengthy and documentation-intensive operation with little value to the client, however, they are wrong, because clients are interested in the high quality of the product they will have to work with, on which their health and even life may depend.

Now we have not only increasing scale of software, but the rapid development of artificial intelligence also. Artificial intelligence is the best tool for analysis of the vast amount of data and saving of human effort, in particular for significantly reducing the time and increasing the efficiency in the development of complicated software. The world quality report estimates that 64% of the companies implement artificial intelligence for the software quality assurance processes [7].

So, now software quality is the key aspect and priority of functioning every software organization. Software quality prediction is performed at various stages of software projects. As the size of software is constantly growing, software quality prediction and assessment is becoming more complex. The accurate software quality prediction and assessment will help software developers and software engineers to develop the high-quality software. Early (based on the specification of software requirements) prediction of software quality is used to select certain preventive measures to reduce the number of software failures and malfunctions during its operation. High-quality requirements engineering leads to an increase in software quality and security and reduces the risk of software project failure (exceeding development time, exceeding development cost, lack of planned functions).

According to the ISO 25010:2011 standard [3], software security is one of the characteristics of software quality and, like other characteristics, is determined on the basis of certain quality attributes from the ISO 25023:2016 standard [8]. Measurement of such attributes is mainly performed for the finished source code, but all these attributes, along with their values, should

be specified in the software requirements specification so that developers are obliged to ensure the presence and value of each attribute in their software for its further verification and validation. Thus, based on the values of the attributes contained in the requirements, it is realistic to predict the level of software security.

The ontology of software security as a software quality characteristic based on the ISO 25010 standard is shown in Fig. 1 [9, 10].

Fig. 2 [9, 10] shows a weighted ontology of software security as a software quality characteristic based on the ISO 25010 standard (an ontology in which all attributes have certain weighting factors). The weighting factors for software quality attributes were determined according to the method for estimating factor weights proposed in [11].

Thus, at present, there is an increase in the complexity of software, an increase in the responsibility assigned to it, and tightening requirements for software quality and security on the part of users, so predicting and determining the level of software security (as one of the characteristics of software quality) based on requirements using AI components is an *urgent task*, the solution of which will be *the purpose of this study*.

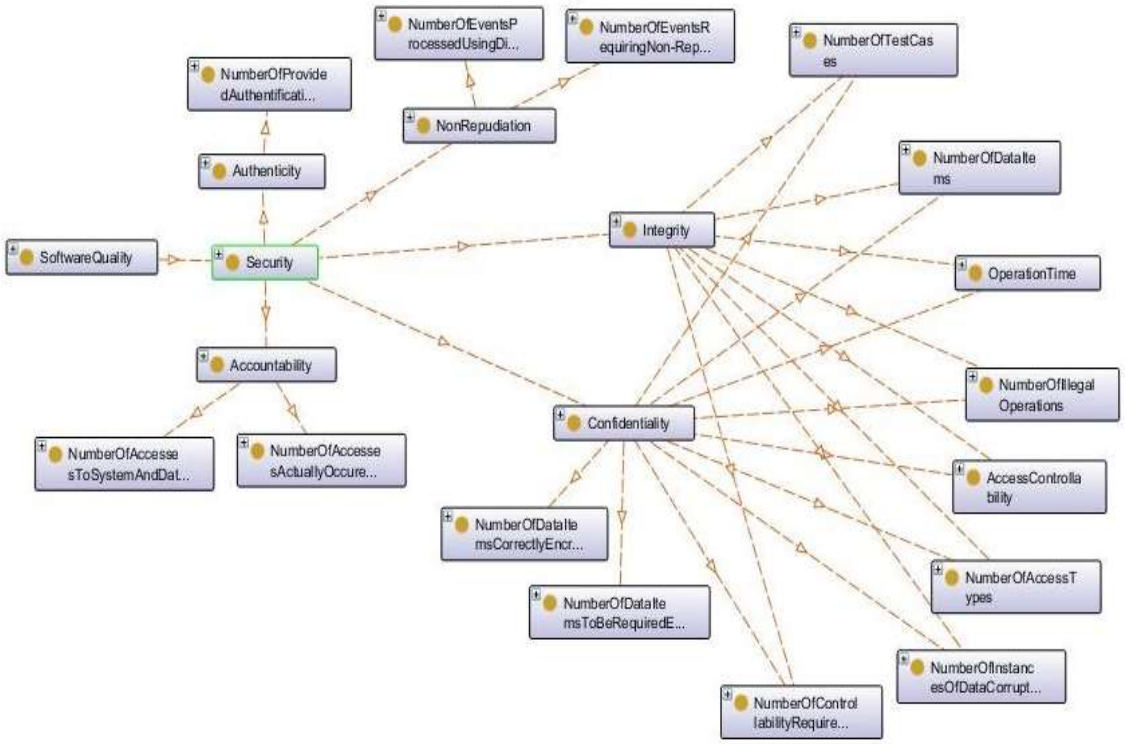

Figure 1: Ontology of software security as a software quality characteristic based on the ISO 25010 standard [9, 10].

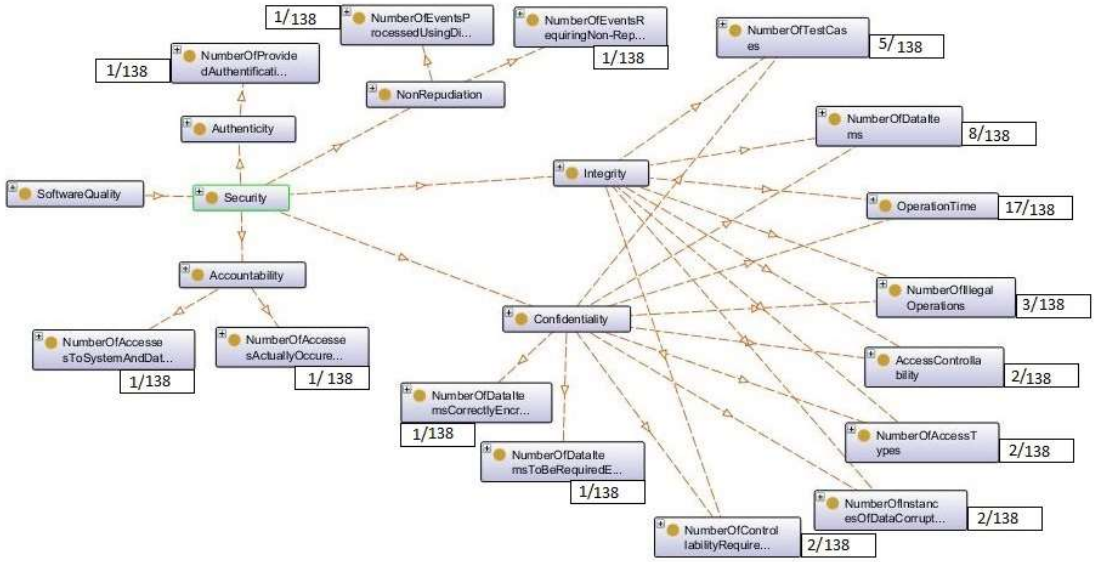

Figure 2: Weighted ontology of software security as a software quality characteristic based on the ISO 25010 standard [9, 10].

2. Survey of Research

Let's consider known AI-based methods and tools for predicting the level of software security and quality.

Paper [12] proposes the software structure and function protection using recurrent neural networks with good protection effect, that can be applied to information misuse, information anomaly and security response.

Paper [13] proposes the deep learning-based method for vulnerability detection, that can learn and automatically generate the vulnerability pattern, and the graph neural network-based method for slice-level vulnerability detection and interpretation. These methods normalize the source code, extract slices to reduce the interference of redundant information, and the vulnerability slices are fed into the vulnerability interpreter to obtain the concrete lines of vulnerability code. These methods correctly detect 59 real vulnerabilities in the four open-source software.

Authors of [14] develop the methodology for analyzing the software security and detecting security incidents, the deep learning-based and artificial immune-based model for security incidents identification, the artificial immune-based and convolutional neural network-based method for optimization and classification of security incidents, and the software package for detecting the software security incidents.

Paper [15] propose the various machine learning models for predicting the software faults, the performance of which depends on quality of set of data, on data issues (data dimensionality, class overlapping, class imbalance, missing data) and can be enhanced by enhancing the dataset quality, including data quality, data pre-processing, data modeling, data performance. etc.

The authors of [16] investigates the prediction for software efficiency and quality analysis, for evaluation of each software component efficiency parameters and for analysis of basic aspects before the software design stage using enhanced feed-forward neural network machine learning classification with CatBoost.

Paper [17] discuss some models, methods, tools and standards of software quality and quality assurance by using machine learning-based approaches for forecasting, optimization, features identifying and enhancing the effectiveness of software defects prediction.

The authors of [18] develop the optimized machine learning-based model for software fault prediction with the purpose of the software quality improving. The software important features are selected with ant colony optimization technique, after that the selected features are fed to support vector machine as its inputs.

The goal of study [19] is the prediction of the software quality with higher accuracy than previous methods and tools. The authors of this study prove that machine learning algorithms with data pre-processing and feature extraction on datasets with the software metrics provide more accurate results in the software quality prediction.

Paper [20] research the impact of software domain and software quality attributes in software quality prediction by deep learning methods with using different datasets. The value of this research is in raising the identifying the quality attributes in requirements preparation and help requirements engineers understand what requirements' issues to focus.

Authors of [21] analyze the relationship between the improvement of software requirements and the software quality. Analysis shows that software quality depends on the measures of metrics from ISO/IEC 25010, IS\IEC 25023 standards. The study empirically shows that the improvement of the requirements leads to the improvement of the software quality.

Authors of [22] develop the software defect prediction model and software maintainability prediction model, which based on the decision tree as a more efficient classifier. In addition, authors develop the framework on the basis of the set of guidelines for improving the software quality.

Authors of [23] use the generalized regression neural network with the improved cuckoo search algorithm for mapping the nonlinear relationship between software metrics and software quality characteristics, and propose the GRNN-based software quality prediction model for improving the accuracy of the software defects prediction.

Paper [24] proposes the framework of the single-layer radial basis function network with thin-plate spline RBF (as its activation function) for software quality prediction. The proposed network was verified for five unknown software samples and was demonstrated that the predicted quality is very close to the actual software quality.

Authors of [25] develop the seven-ensemble machine learning model for software defect prediction based on the Cat boost. The obtained results prove that the proposed Cat boost model provides the high performance for all the three defects datasets though decreasing the overfitting and reducing the training time.

Paper [26] empirically demonstrates performance of defects prediction by ten ensemble predictors. It used 15 software projects from PROMISE repository and results of

experiments demonstrate that ensemble predictors improve the performance of defects detection.

Authors of [27, 28] use the firefly optimization methodology and propose the objective function for prediction of the software quality with superior results on MATLAB with actual data.

The analyzed AI-based methods and tools for predicting the level of software security and quality have great potential, but they do not establish the dependence of software security on quality attributes, do not form a predicted numerical value of software security based on attributes, and do not provide a prediction of the level of software security based on the obtained numerical value.

3. Method for Determining the Security Level of Software

From Figs. 1, 2, it can be seen that software security (as a characteristic of software quality) depends on 5 subcharacteristics (Confidentiality, Integrity, Non-Repudiation, Accountability, Authenticity), each of which depends on certain quality attributes. Thus, according to ISO 25010 [3] and ISO 25023 [8], software security (as a characteristic of software quality) depends on 23 quality attributes, but on 15 different quality attributes.

For determining the level of software security based on quality attributes from software requirements, we should first calculate the predicted numerical value of software security based on the values of the 15 attributes defined in Figs. 1, 2, taking into account their interdependencies, which is a difficult formalized task. According to the Hecht-Nielsen theorem, to establish and take into account the interdependencies between the values of quality attributes from software requirements and the value of software security (as a characteristic of software quality), let's use the artificial neural network (ANN) of the "multilayer perceptron" type, which will receive as inputs the values of 15 quality attributes on which software security depends (Figs. 1, 2) and, after their approximation, will determine the predicted numerical value of software security in the interval [0; 1].

The described concept of determining the software security based on quality attributes from software requirements is based on the concept of predicting the software quality characteristics based on quality attributes using ANN, which was developed by the authors in [29] and is shown in Fig. 3.

The ANN is trained in such a way that, based on the values of the relevant attributes, it generates a predicted numerical value of software security *pnvss* in the interval [0;1], where the value "0" means the worst level of software security, and the value "1" means the best level of software security as a characteristic of software quality. However, it is difficult for both the customer and even the developer to correctly interpret the obtained numerical value of software security, and therefore it is difficult to correctly assess the level of software security based on the obtained from ANN value.

Therefore, in order to simplify unambiguous interpretation of the predicted numerical value of software security, it is first necessary to determine the thresholds by which the conclusion about the level of software security (as a quality characteristic) will be generated.

For establishing such thresholds, we analyzed 230 available specifications of software requirements to find the values of quality attributes on which software security depends, for which ANN determined the predicted numerical value of software security, as well as 230

corresponding finished programs written according to these requirements, for which the security level (one of four – initial, medium, sufficient, high) was determined during certification. The specifications of software requirements and finished programs were provided for analysis by software companies in Khmelnytskyi (Ukraine) as part of scientific cooperation with the Department of Computer Engineering and Information Systems of Khmelnytskyi National University.

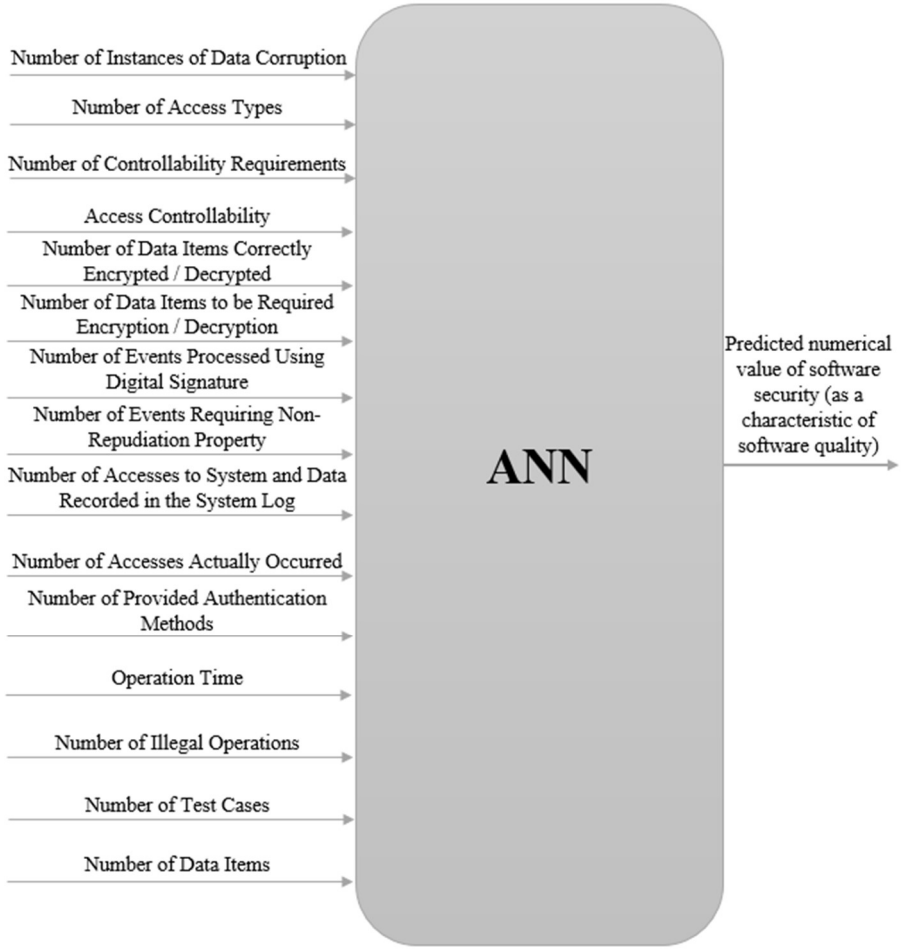

Figure 3: Concept of determining the software security based on quality attributes from software requirements.

As a result of the conducted analysis, let's form the threshold values of the predicted numerical value of software security *pnvss* for determining the level of software security (as a characteristic of software quality):

1. initial level of security – $pnvss \in [0; 0{,}22)$.
2. medium level of security – $pnvss \in [0{,}22; 0{,}49)$.
3. sufficient level of security – $pnvss \in [0{,}49; 0{,}89)$.
4. high level of security – $pnvss \in [0{,}89; 1]$.

Taking into account the concept of determining the software security based on quality attributes from software requirements and the formed thresholds of the predicted numerical value of software security *pnvss* for determining the level of software security, the *method for determining the security level of software* consists of the following steps:

1. preprocessing of software requirements – representation of the requirements specification in a form suitable for analysis for finding the values of quality attributes
2. analysis of software requirements to find the values of 15 quality attributes on which software security depends (Fig. 3)
3. preparation of the found values of 15 quality attributes, on which software security depends, for submitting them to the ANN input – at this stage, the ANN input vectors are prepared taking into account the fact that security subcharacteristics depend on 23 attributes, including 15 different attributes (Fig. 1, 2), and the ANN inputs are formed as 5 sets (for 5 software security subcharacteristics) of 1, 2, 8, 10, and 2 attributes according to the ontologies presented in Figs. 1, 2
4. processing of attribute values by an artificial neural network
5. analysis of the result of the ANN – the predicted numerical value of the software security *pnvss*
6. forming a conclusion about the predicted level of software security based on the following rules:

- if $pnvss \in [0; 0{,}22)$, then the software is predicted to have an initial level of security;
- if $pnvss \in [0{,}22; 0{,}49)$, then the software is predicted to have a medium level of security;
- if $pnvss \in [0{,}49; 0{,}89)$, then the software is predicted to have a sufficient level of security;
- if $pnvss \in [0{,}89; 1]$, then the software is predicted to have a high level of security.

The ANN was implemented in the Matlab. The *gensim(net)* operator generated a visualization of the developed ANN in the Simulink (Fig. 4-8).

For training the resulting ANN, a training sample of 4750 vectors and a testing sample of 867 vectors were formed based on the analysis of existing software requirements and corresponding off-the-shelf programs with a known level of security provided by software companies from Khmelnytskyi (Ukraine). For calculating the required training sample size for the ANN to be trained with an error of about 0,1, we use the formula: $N > \frac{h \cdot g}{e_0} = \frac{20 \cdot 23}{0{,}1} = 4600$, where g is the number of input neurons of the ANN ($g=23$); h is the number of neurons of the hidden layers of the ANN ($h=12+8=20$), e_0 is the permissible training error ($e_0=0{,}1$). Thus, training sample vectors are enough to train an ANN to recognize possible situations with a given accuracy. The process of training and testing the ANN is shown in Figs. 9, 10, where the blue curve is the ANN training schedule, the green curve is the ANN testing schedule, and the black line is the ANN training goal, which, as can be seen from Figs. 9, 10 was achieved.

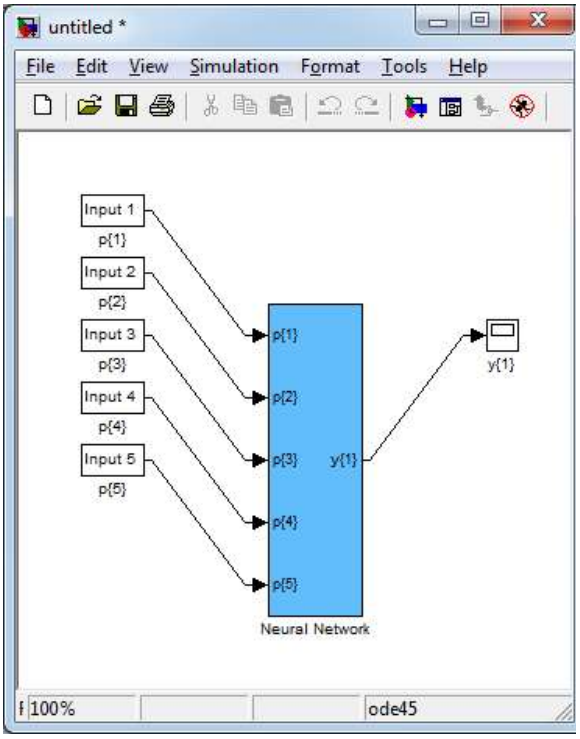

Figure 4: ANN's architecture.

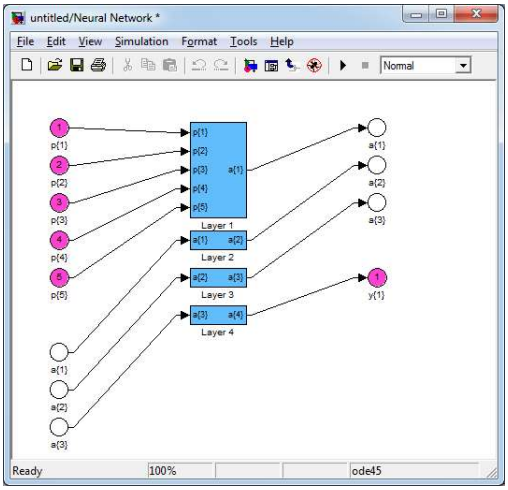

Figure 5: ANN's layers structure.

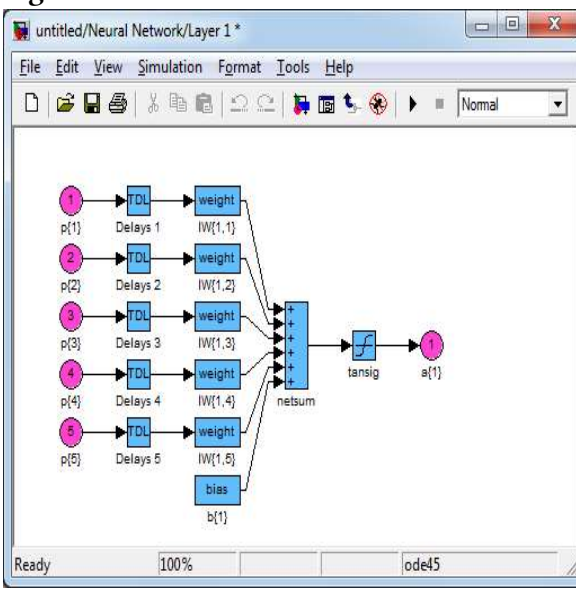

Figure 6: ANN's first layer.

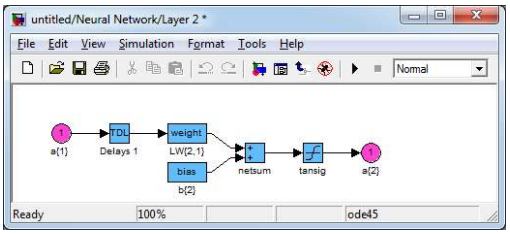

Figure 7: ANN's second layer (ANN's third layer is similar to the second layer)

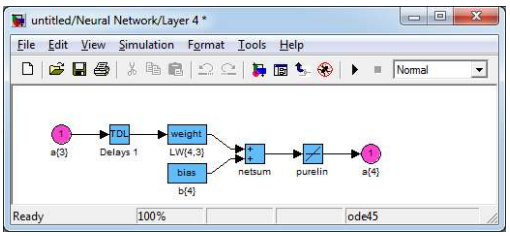

Figure 8: ANN's fourth layer.

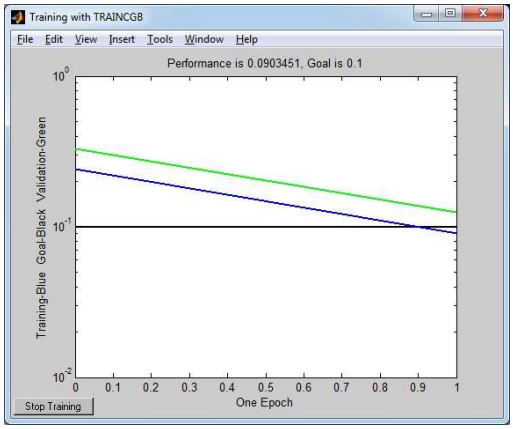

Figure 9: ANN's training and testing using *traincgb* algorithm with *msereg* quality criterion.

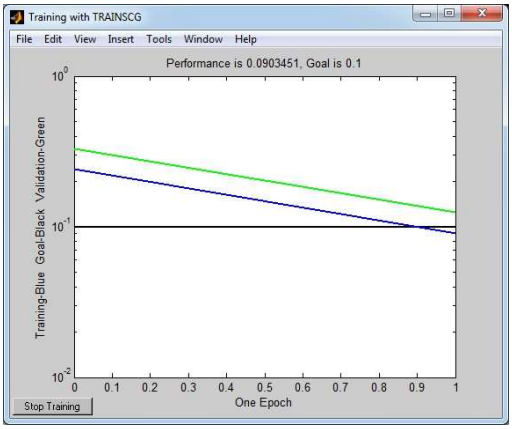

Figure 10: ANN's training and testing using *trainscg* algorithm with *msereg* quality criterion.

For selecting the optimal ANN training algorithm, we analyzed the ANN's training process by the different algorithms using different quality criteria. The results of the analysis are shown in Table 1.

Table 1
Analysis of the ANN's training process

ANN's training algorithm	ANN's training quality criterion	ANN's training error
trainbfg	*mse*	0,100291
trainoss		0,100291
traincgb		0,100291
traingda		0,100291
trainlm		0,100291
trainrp		0,100291
trainscg		0,100291
trainbfg	*msereg*	0,0964035
trainoss		0,0964035
traincgb		0,0903451
traingda		0,0995513
trainlm		0,0903451
trainrp		0,0952321
trainscg		0,0903451
trainbfg	*mae*	0,396053
trainoss		0,396053
traincgb		0,251652
traingda		0,262794
trainlm		0,264611
trainrp		0,251225
trainscg		0,395961

The conducted analysis has shown that the worst ANN's training result is obtained by the training quality criterion *mae* in combination with all training algorithms, and the best result is obtained by the training quality criterion *msereg*. The analysis of the training results using *msereg* training quality criterion makes it possible to determine that the most accurate result is obtained by the training algorithms *traincgb, trainlm, trainscg*. The analysis of the ANN training and testing results proved that the network trained with the specified accuracy.

4. Results & Discussion

Let's consider 10 specifications of software requirements prepared by 10 different software companies in Khmelnytskyi (Ukraine) as a result of the stage of collecting and analyzing requirements for the same software. Each of the specifications went through a preprocessing stage, during which it was prepared for analysis to find the values of quality attributes. Next, each specification was analyzed to find the values of 15 quality attributes on which software security depends. After that, the input vectors of the ANN were formed and transferred to the artificial neural network for processing.

Based on the processing of the attributes' values, the ANN provided the following values of the predicted numerical value of software security *pnvss* for the 10 analyzed specifications (Fig. 11): $pnvss_1 = 0,12$; $pnvss_2 = 0,93$; $pnvss_3 = 0,45$; $pnvss_4 = 0,67$; $pnvss_5 = 0,34$; $pnvss_6 = 0,09$; $pnvss_7 = 0,78$; $pnvss_8 = 0,98$; $pnvss_9 = 0,41$; $pnvss_{10} = 0,53$.

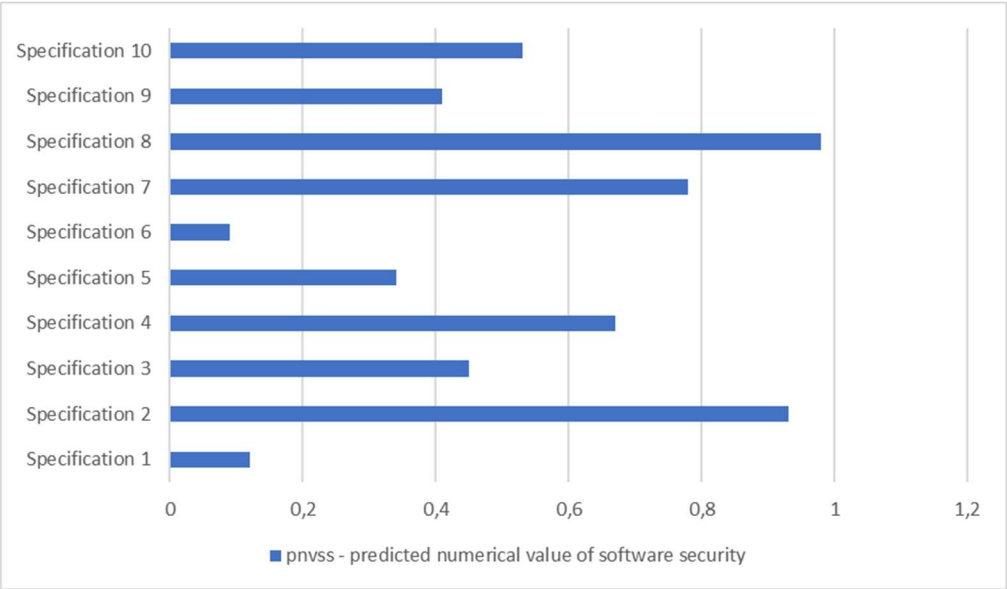

Figure 11: Predicted numerical value of software security *pnvss* for the 10 analyzed specifications.

Next, we analyzed the obtained predicted numerical values of software security $pnvss_1$-$pnvss_{10}$, which resulted in the conclusions about the predicted level of software security for the 10 analyzed specifications: 1) since $pnvss_1 \in [0; 0,22)$, the software that will be developed according to specification 1 is predicted to have an initial level of security; 2) since $pnvss_2 \in [0,89; 1]$, the software that will be developed according to specification 2 is predicted to have a high level of

security; 3) since $pnvss_3 \in [0{,}22;\ 0{,}49)$, the software that will be developed according to specification 3 is predicted to have a medium level of security; 4) since $pnvss_4 \in [0{,}49;\ 0{,}89)$, the software that will be developed according to specification 4 is predicted to have a sufficient level of security; 5) since $pnvss_5 \in [0{,}22;\ 0{,}49)$, the software that will be developed according to specification 5 is predicted to have a medium level of security; 6) since $pnvss_6 \in [0;\ 0{,}22)$, the software that will be developed according to specification 6 is predicted to have an initial level of security; 7) since $pnvss_7 \in [0{,}49;\ 0{,}89)$, the software that will be developed according to specification 7 is predicted to have a sufficient level of security; 8) since $pnvss_8 \in [0{,}89;\ 1]$, the software that will be developed according to specification 8 will predictably have a high level of security; 9) since $pnvss_9 \in [0{,}22;\ 0{,}49)$, the software that will be developed according to specification 9 is predicted to have a medium level of security; 10) since $pnvss_{10} \in [0{,}49;\ 0{,}89)$, the software that will be developed according to specification 10 is predicted to have a sufficient level of security.

Thus, the software developed according to specifications 2 and 8 is predicted to have a high level of security (of course, if the bugs are not made at the next lifecycle stages), so the customer is recommended to order software development from software companies that have prepared requirements' specifications 2 and 8.

Taking into account the results of the experimental studies, it was concluded that the developed method for determining the security level of software establishes the dependence of software security on quality attributes, forms a predicted numerical value of software security based on attributes, provides a prediction of the level of software security based on the obtained numerical value, and provides a comparison of software requirements specifications by predicted level of security of developed software (of course, if the bugs are not made at the next lifecycle stages) and a possibility of rejection form unsuccessful specifications.

5. Conclusions

Currently, there is an increase in the complexity of software, an increase in the responsibility assigned to it, and tightening requirements for software quality and security on the part of users, so predicting and determining the level of software security (as one of the characteristics of software quality) based on requirements using AI components is an urgent task, the solution of which is the purpose of this study.

The analyzed AI-based methods and tools for predicting the level of software security and quality have great potential, but they do not establish the dependence of software security on quality attributes, do not form a predicted numerical value of software security based on attributes, and do not provide a prediction of the level of software security based on the obtained numerical value.

The developed method for determining the security level of software establishes the dependence of software security on quality attributes, forms a predicted numerical value of software security based on attributes, provides a prediction of the level of software security based on the obtained numerical value, and provides a comparison of software requirements specifications by predicted level of security of developed software (of course, if the bugs are not made at the next lifecycle stages) and a possibility of rejection form unsuccessful specifications.

References

[1] Software, 2022. URL: https://www.statista.com/markets/418/topic/484/software/.

[2] M. Chornobuk, V. Dubrovin, L. Deineha, Cybersecurity: Research on Methods for Detecting DDOS Attacks, Comput. Syst. Inf. Technol. № 4 (2023) 6–9. doi:10.31891/csit-2023-4-1.

[3] ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models. 2011.

[4] Success Rates Rise, 2017. URL: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf.

[5] The Cost of Poor Software Quality in the US: A 2020 Report, 2020. URL: https://www.it-cisq.org/pdf/CPSQ-2020-report.pdf.

[6] Pulse of the Profession 2023: Power Skills, Redefining Project Success, 2023. URL: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pmi-pulse-of-the-profession-2023-report.pdf?v=7933da8f-304b-4fe3-a655-78dace54174a&rev=427949fcdb684485a020cc72ea219f32.

[7] S. Ramchand, S. Shaikh, I. Alam, Role of Artificial Intelligence in Software Quality Assurance. Lecture Notes in Networks and Systems (2021) 125–136. doi:10.1007/978-3-030-82196-8_10.

[8] ISO 25023:2016. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Measurement of system and software product quality. 2016.

[9] T. Hovorushchenko, O. Pomorova, Methodology of evaluating the sufficiency of information on quality in the software requirements specifications, in: Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies DESSERT-2018, Kyiv, 2018, pp. 385-389. doi:10.1109/dessert.2018.8409161.

[10] T. Hovorushchenko, Information Technology for Assurance of Veracity of Quality Information in the Software Requirements Specification. Advances in Intelligent Systems and Computing 689 (2018) 166–185. doi:10.1007/978-3-319-70581-1_12.

[11] E. Zaitseva, T. Hovorushchenko, O. Pavlova, Y. Voichur, Identifying the Mutual Correlations and Evaluating the Weights of Factors and Consequences of Mobile Application Insecurity, Systems 11 5 (2023) 242. doi:10.3390/systems11050242.

[12] Y. Mi, E. Gao, Information Sharing Security Protection System Based on Artificial Intelligence, in: Proceedings ofv2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications, IEEE, 2022. doi:10.1109/icmnwc56175.2022.10031986.

[13] Y.-T. Hu, S.-Y. Wang, Y.-M. Wu, D.-Q. Zou, W.-K. Li, H. Jin. A Slice-level vulnerability detection and interpretation method based on graph neural network. Journal of Software 34 6 (2023) 2204 – 2221.

[14] A. Tanwar, K. Sundaresan, P. Ganesan, S. Ravi, R. Karthik, Proximal Instance Aggregator networks for explainable security vulnerability detection, Future Gener. Comput. Syst. (2022). doi:10.1016/j.future.2022.04.008.

[15] K. Bhandari, K. Kumar, A. L. Sangal, Data quality issues in software fault prediction: a systematic literature review, Artif. Intell. Rev. (2022). doi:10.1007/s10462-022-10371-6.

[16] D. Sudharson, P. S. Kailas, K. Vignesh, T. Senthilnathan, V. Poornima, S. Vijay, Software Quality Prediction by CatBoostFeed-Forward Neural Network in Software Engineering. System Reliability and Security (2023) 207–218. doi:10.1201/9781032624983-11.

[17] J. Mona, R. Al-Sagheer, S. Alghazali. Software Quality Assurance Models and Application to Defect Prediction Techniques. International Journal of Intelligent Systems and Applications in Engineering 11 1 (2023) 169 – 178.
[18] M. Shafiq, F. H. Alghamedy, N. Jamal, T. Kamal, Y. I. Daradkeh, M. Shabaz, Scientific programming using optimized machine learning techniques for software fault prediction to improve software quality. IET Software (2023). doi:10.1049/sfw2.12091.
[19] A. A. Ceran, Y. Ar, Ö. Ö. Tanrıöver, S. Seyrek Ceran, Prediction of software quality with Machine Learning-Based ensemble methods, Mater. Today (2022). doi:10.1016/j.matpr.2022.11.229.
[20] G. Airlangga, A. Liu. Investigating Software Domain Impact in Requirements Quality Attributes Prediction. Journal of Information Science and Engineering 38 2 (2022) 295 – 316. doi: 10.6688/JISE.202203_38(2).0002.
[21] L. Canchari, P. Angeleri, A. Dávila, Requirements Validation in the Information System Software Development Lifecycle: A Software Quality in Use Evaluation, Program. Comput. Software 49 8 (2023) 610–624. doi:10.1134/s0361768823080054.
[22] B. Desai, R. K. Sungkur, Software Quality Prediction Using Machine Learning, Int. J. Softw. Innov. 10 1 (2022) 1–35. doi:10.4018/ijsi.297997.
[23] L. Liu, P. Han, Application of improved cuckoo algorithm to optimize generalized regression neural network in software quality prediction, in: Proceedings of R. Tiwari, International Conference on Neural Networks, Information, and Communication Engineering NNICE 2022, SPIE, 2022. doi:10.1117/12.2639204.
[24] Ritu, O. Sangwan. Radial Basis Function Network Based Intelligent Scheme for Software Quality Prediction. Communications in Computer and Information Science 1572 (2022) 327 – 340. doi: 10.1007/978-3-031-05767-0_26.
[25] Y. K. Saheed, O. Longe, U. A. Baba, S. Rakshit, N. R. Vajjhala, An Ensemble Learning Approach for Software Defect Prediction in Developing Quality Software Product. Communications in Computer and Information Science (2021) 317–326. doi:10.1007/978-3-030-81462-5_29.
[26] F. Yucalar, A. Ozcift, E. Borandag, D. Kilinc, Multiple-classifiers in software quality engineering: Combining predictors to improve software fault prediction ability. Eng. Sci. Technol. Int. J. 23 4 (2020) 938–950. doi:10.1016/j.jestch.2019.10.005.
[27] D. Pankwar, G. L. Saini, P. Agarwal, P. Singh, Firefly Optimization Technique for Software Quality Prediction. Soft Computing: Theories and Applications (2022) 263–273. doi:10.1007/978-981-19-0707-4_25.
[28] B. Iegorov, Y. Kravchyk, S. Rybalko, I. Ivashkiv, A. Chub. The Methodical Approach of the Substantiation of the Evaluation Indicators System of the Agro-Industrial Complex Development. Universal Journal of Agricultural Research 9 5 (2021) 191 - 199. doi: 10.13189/ujar.2021.090506.
[29] T. Hovorushchenko, D. Medzatyi, Y. Voichur, M. Lebiga, Method for forecasting the level of software quality based on quality attributes, J. Intell. & Fuzzy Syst. (2022) 1–15. doi:10.3233/jifs-222394.