Intelligence Method of Software Quality Evaluation and Prediction

Oksana Pomorova
Head of System Programming Department
Khmelnitsky National University
Khmelnitsky, Ukraine
o.pomorova@gmail.com

Tatyana Hovorushchenko
Department of System Programming
Khmelnitsky National University
Khmelnitsky, Ukraine
tat_yana@ukr.net

***Abstract*— Research is dedicated to development of artificial neural net's method of software quality evaluation and prediction, which provides the realization of comparative analysis of different project versions and selection of the best of them accordance its characteristics on the basis of design stage metrics analysis.**

Keywords-software, software metric, Safety Case methodology, artificial neural network.

I. INTRODUCTION

Safety Case methodology (Safety computer-aided software engineering) is developed over 20 years [1], [2]. The primary object of Safety Case methodology is to minimize software security and commercial risks by constructing a report, which should provide evidences, reasons and arguments that software is safe, and all requirements for the software is properly implemented. Currently, this methodology is generally accepted, but the level of its automation is still low.

The process of software developing for the Safety Case methodology depends on a large number of documents, source code, software evaluation methods and analysis of their results and software testing results. The main task of Safety Case methodology [3] is the automating of the creation of:

1) software requirements profile (including standards for software development, subject domain standards and customer requirements) - functional, inverse and non-functional requirements for software systems are analyzed; requirements for software safety and security are researched and completeness of all kinds of requirements is estimated;

2) software analysis results profile - metric analysis results, source code and software test results are studied and analyzed;

3) evaluation of results profile accordance to requirements profile - obtained software accordance to its requirements is evaluated.

One of the main tools of software quality analysis and evaluation is metric analysis. The metric is defined as a measure of degree of possessing property that has a numerical

National Safeware Engineering Network of Canters of Innovative Academia-Industry Handshaking 158886-TEMPUS-2009-UK-JPCR

value [4]. Generally, software metric is a measure, which provides obtaining of numerical values of some software properties or its specifications. The modern software industry has accumulated a large number of metrics, which evaluate some industrial and exploitation software characteristics. But aspirations of its universality, ignoring of type and application domain of design software, ignoring of software life cycle stages and its groundless using in industrial decisions making procedures significantly undermined the developers and users confidence to metrics. These circumstances require the careful selection of metrics for design software certain type and application domain, considering their restrictions at different software life cycle stages, establishing order for their common use, storage and integration of heterogeneous metric information to make timely industrial decisions.

Currently, only collection, registration and calculation of metric information are automated. The technique of metric analysis results processing [4] has the following unsolved tasks: 1) absence of unified standards for metrics, which leads to subjective selection of quality evaluation methods; 2) difficulty of interpretation the metrics values, which is caused by individual projects features and absence of metrics standard values; 3) absence of criterion to compared essentially new and previous projects, which leads to the impossibility of interpretation of obtained metrics for new project; 4) basic parameters in the selection of software realization versions are the design cost and time and software company reputation, but the decisions, taken on the basis of these parameters, not guarantee software quality.

On the basis of the above the need and actuality of scientific research in development of new effective methods of software quality evaluation and prediction arises. The intelligent methods, in particular artificial neural net's method of software quality evaluation and prediction (NMEP) [4], [5], are perspective today.

II. Software Design Results Evaluation and Quality Characteristics Prediction

Today, software is a defining component of many systems, including systems for critical applications, integrated (embedded) and specialized systems for various purposes. For such systems software bugs presence and low quality threaten the catastrophes that result to human victims, ecological cataclysms, and economic losses.

The problem of bugs exposure and removal is aggravated as increasing software complexity. Development of modern software design technologies requires the dynamic development of software quality evaluation tools, especially at the stage design (from the viewpoint of financial and time expedience).

The modern software industry is characterized by high competition. For success in this market the software company must to develop, to implement and to accompany the software considering the lead-times and providing the satisfactory quality. Therefore, many software companies are investing in software development technologies upgrading, on the basis that such facilities investment is necessarily covered a cost.

Nine stage design metrics with the exact values and 15 stage design metrics with the predicted values were selected as the basic metrics to development of intelligence method of design results evaluation and software quality characteristics prediction. The basic software design stage metrics and algorithms to measure were described in [4]-[6].

To determine the used design stage metrics ranges, we introduce some assumptions and impose certain limits on software and projects [7], that will be analyzed using artificial neural net's method of design results evaluation and software quality prediction (NMEP): 1) maximum 100 variables to calculate and output, 100 modified or created in the module variables, 100 managing variables and 100 unused in the module ("parasitic") variables for each software module; 2) software consists of maximum 50 modules; 3) maximum quantity of intermodals relations is 2450; 4) maximum 50 procedures for the data structure updating, 50 procedures for reading of information from the data structure and 50 procedures for the data structure reading and updating contain in each module, 5) each module actually gets access to a global variable as many times as it can get such access; 6) software contains maximum 50000 lines of source code; 7) the problem statement takes 10% of the time, the design - 35% of the time, the programming - 35% of the time and software testing, debugging and quality audit - 20% of the time; 8) models creation and modification take about 25% of the design stage time; 9) maximum quantity of models and prototypes errors for the one module should not exceed 100 errors; 10) quantity of software unique operators along with the subprogram names does not exceed 25000, the total quantity of software operands does not exceed 50000, the quantity of software unique operands does not exceed 400; 11) quantity of logical or cycle operators does not exceed 50000; 12) quantity of software operators does not exceed 50000; 13) each variable of software module is passed on its interface; 14) the quantity of source code lines, divided into two, is the maximum software design expected cost in U.S. dollars; 15) software quality audit takes 50% of the time of testing, debugging and quality audit stage; 16) quantity of external inputs of each module is 49, quantity of external output of each module is 49, quantity of external requests to each module is 49, 17) each module has maximum 50 internal logical files and uses maximum 50 external logical files.

The ranges of software design stage metrics values are determined in Table 1.

TABLE 1. SOFTWARE DESIGN STAGE METRICS VALUES RANGES

№	**Design Stage Metrics with the Exact Values**	**Design Stage Metrics with the Predicted Values**
1	Chepin's metric: -1, 0..32500	Expected Lines Of Code (LOC): -1, 0..50000
2	Jilb's metric (absolute modular complexity): -1, 0..2450	Halstead's metric: -1, 0..1562500
3	McClure's metric: -1, 0..120050	McCabe's metric: -1, 0..2402
4	Kafur's metric: -1, 0..497500	Jilb's metric (relative logical complexity): -1, 0..1
5	Cohesion metric: -1, 0..10	Expected quantity of program statements: -1, 0..50000
6	Coupling metric: -1, 0..9	Expected estimate of interfaces complexity: -1, 0..1
7	Metric of the global variables calling: -1, 0..1	Software design total time: -1, 0..520 (working days)

8	Time of models modification: -1, 0..46	Design stage time: -1, 0..182 (working days)
9	Quantity of found bugs during the models and prototype inspection: -1, 0..5000	Software design expected cost: -1, 0..25000 (usd)
10		Software quality audit expected cost: - 1, 0..2500 (usd)
11		Software realization productivity: -1, 0..5 (minutes for 1 line of code)
12		Program code realization expected cost: -1, 0..8750 (usd)
13		Expected functional points (FP): -1, 0..2945
14		Effort applied by Boehm's model: -1, 0..394 (man-months)
15		Expected development time by Boehm's model: -1, 0..520 (working days)

The design results evaluation and software complexity and quality characteristics prediction are the results of processing of metrics from Table 1 using artificial neural network.

III. Artificial Neural Net's Method of Software Quality Evaluation and Prediction

NMEP provides an evaluation of the project and prediction designed software complexity and quality on the basis of exact and predicted values of complexity and quality metrics of the design stage. NMEP uses artificial neural network (ANN), which provides metrics processing and approximation.

Input data for ANN are the set of the design stage metrics with the exact values and the set of the design stage metrics with the predicted values.

The results of ANN functioning are 4 characteristics: 1) project complexity estimate; 2) project quality evaluation; 3) designed software complexity prediction; 4) designed software quality prediction (Fig.1)

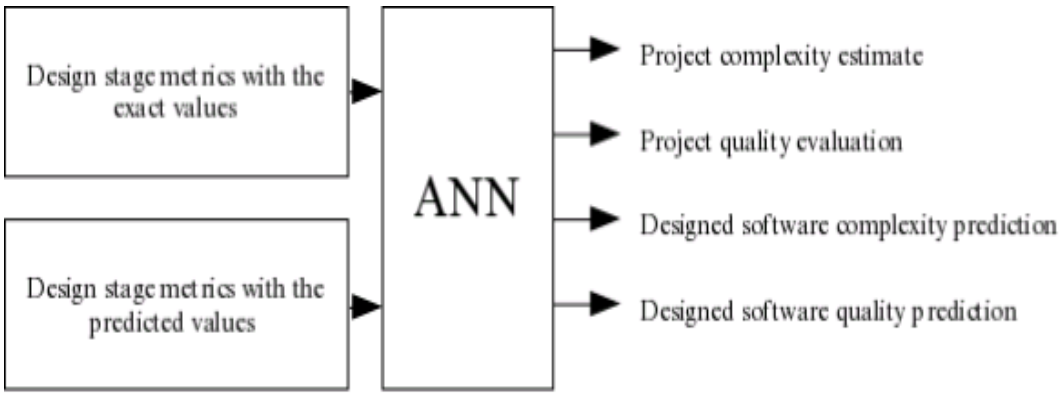

Figure 1. NMEP concept

ANN has 9 one type inputs for the quantitative values of design stage exact metrics and 15 another type inputs for the quantitative values of design stage predicted metrics. If a certain metric is not determined, then -1 is given on the proper ANN input.

Project complexity estimate is value in the range [0, 1], where 0 - design stage complexity metrics with the exact values were not determined, approximately 0 - the project is complicated to realization and 1 - the project is simple to realization; project quality evaluation is value in the range [0, 1], where 0 - design stage quality metrics with the exact values were not determined, approximately 0 - is a low quality project and 1 - the project satisfies the customer requirements in quality; software complexity prediction is value in the range [0, 1], where 0 - design stage complexity metrics with the predicted values were not determined, approximately 0 - designed software will has significant complexity and 1 - designed software is expected simple; software quality prediction is value in the range [0, 1], where 0 - design stage quality metrics with the predicted values were not determined, approximately 0 - designed software will has low quality and 1 - high quality software is expected. The complexity characteristics include not only the simplicity or complexity of designed software from the viewpoint of its size, cost and design time, but also the maintenance difficulty or simplicity, usability and the effectiveness of the methods chosen to solve the task.

The conclusion about the project quality and complexity and the expected quality and complexity of designed software is based on an analysis of 4-th obtained results.

IV. ANN REALIZATION

Analysis of the artificial neural network's architecture was conducted [6] to select the architecture for the design stage software metrics analysis and software quality characteristics prediction. General regression neural networks (GRNN) are radial basis function (RBF) networks. GRNN are used to analyze the numerical series. Probabilistic neural networks (PNN) are radial basis function networks. PNN are intended for solving probability tasks, in particular, for the classification tasks. Kohonen self-organizing network (map) is designed for data clustering. Learning vector quantization networks (LVQN) are used for clustering and classification. Elman and Hopfield networks are networks with dynamic feedback, focuses on processing of dynamic models that considering processes pre-history.

Because the task of metrics analysis and prediction of software quality characteristics doesn't have the properties of the numerical series, the feedbacks and the necessary for data classification and clustering, multilayer perceptron is chosen to solve this task. Using another type of neural network to solve this task will artificially distort its nature, and the ANN results are inadequate.

Multilayer perceptron (Fig. 2) is ANN for metrics analysis and software quality characteristics prediction.

ANN from Fig.2 is realized in the Matlab. To creation of ANN template the function *network* is used. Number of input vectors (*net.numInputs = 4*), layers (*net.numLayers = 4*), elements of each ANN input vector (*net.inputs {1}. size = 4; net.inputs {2}. size = 5; net. inputs {3}. size = 6; net.inputs {4}. size = 9*) were determined.

Bias connection matrix: *net.biasConnect=[1;1;1;1]*.

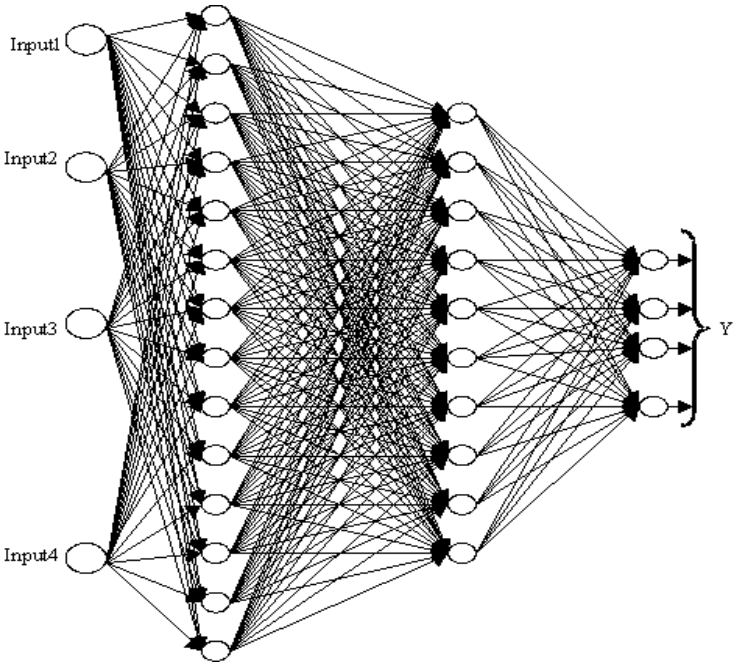

Figure 2. ANN architecture for metrics analysis and software quality characteristics prediction

Input connection matrix, layer connection matrix, output connection matrix and target connection matrix are defined as follows: *net.inputConnect = [1 1 1 1; 0 0 0 0; 0 0 0 0; 0 0 0 0]; net.layerConnect = [0 0 0 0; 1 0 0 0; 0 1 0 0; 0 0 1 0]; net.outputConnect = [0 0 0 1]; net.targetConnect = [0 0 0 1].*

Ranges of input vectors values: *net.inputs{1}.range = [0 32500; 0 2450; 0 120050; 0 497500]; net.inputs{2}.range = [0 10; 0 9; 0 1; 0 46; 0 5000]; net.inputs{3}.range = [0 50000; 0 1562500; 0 2402; 0 1; 0 50000; 0 1]; net.inputs{4}.range = [0 520; 0 182; 0 200000; 0 20000; 0 5; 0 70000; 0 2945; 0 394; 0 24].*

Number of neurons in ANN layers: *net.layers{1}.size=24; net.layers{2}.size=14; net.layers{3}.size=8; net.layers{4}.size=4.*

Nguyen-Widrow function (*initnw*) is an initialization function for each ANN layer. Hyperbolic tangent is the activation (transfer) function of 1-st, 2-nd and 3-rd layers, linear function is activation (transfer) function of 4-th layers.

ANN is represented in the Simulink (Fig. 3, Fig.4).

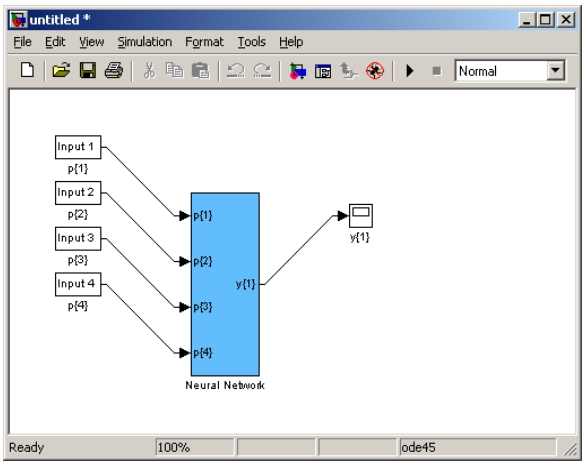

Figure 3. ANN architecture in Simulink

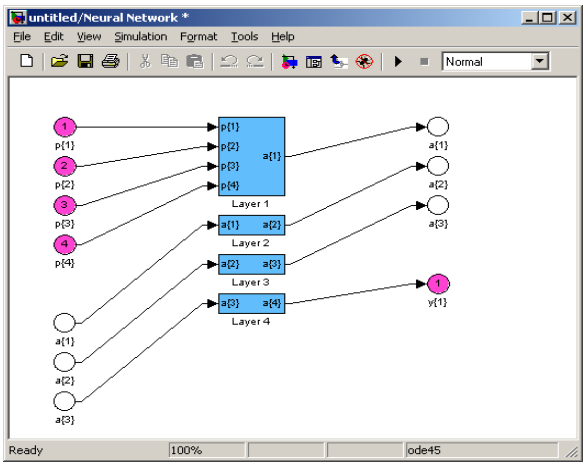

Figure 4. Structural scheme of ANN layers

V. Analysis of The ANN Training Results

The sequence of training vectors (training sample) for obtained ANN training is defined as:

c={[32500;0;0;0] [30875;0;0;0] […]; - training vectors for Input1;

[0;0;0.15;0;0] [0;0;0.2;0;0] […]; - training vectors for Input2;

[3450;0;0;0;0;0] [5900;0;0;0;0;0] […]; - training vectors for Input3;

[0;0;0;0;1.7;0;0;0;0] [0;0;0;0;1.8;0;0;0;0] […]}; - training vectors for Input4.

Target vector is defined as:

m={[0.05;0.02;0.01;0.01] [0.1;0.04;0.02;0.02] […]}.

ANN training method was selected on the basis of ANN training results with the training sample of 1935 vectors by different training methods: GDA - gradient descent with adaptive learning rule backpropagation method, LM - Levenberg-Marquardt backpropagation method, OSS - One step secant backpropagation method, RP - Resilient backpropagation (Rprop) method.

The research demonstrates that training performance did not differ significantly for different training methods (approximately 0,102197), therefore the select of training method was influenced by "the number of epochs" and "the training time". As best training method OSS was selected.

ANN testing was performed on the basis of testing sample of 324 vectors.

The process of ANN training and testing by GDA method is shown on Fig.5, by OSS method - on Fig.6, by LM method - on Fig.7, by RP method - on Fig.8. The bottom line shows the ANN training and the top line - ANN testing.

ANN training and testing charts show that the network is trained with the performance that is caused by insufficient quantity of training sample vectors.

Combinatory formula for the calculating of number of combinations (selections) is used to calculate the training sample maximum size: $C(24,4)=(24!)/(4!*(24-4)!)=10626$.

So, if more than 10 thousand training vectors are, then statistical methods can be applied to the task of design results evaluation and software quality characteristics prediction.

ANN is used at input information incomplete. According to the formula of ANN training sample size [8] we get that training sample of $N>(24*24)/(10^{-1})=5760$ vectors is required for ANN training, which consists of 24-th input neurons, 24-th hidden layers neurons (14-th neurons - the first hidden layer (approximating) and 10-th neurons - the second hidden layer (adjustment)), with accuracy 10^{-1}.

ANN results analysis was performed on the basis of projects developed by software company "STU-Electronics", Khmelnitsky (Ukraine). Examples of results for 5 projects are shown in Table 2.

According to the results: 1-st project is evaluated as rather simple and high-quality; the designed software will have the same characteristics; 2-nd project - complicated and low quality; the designed software will have the same characteristics; 3-rd project has medium complexity and quality; the designed software will have the same characteristics; 4-th project is simple and has high quality, the designed software is also expected a simple and high quality; 5-th project is complicated and has low quality, the designed software is expected very difficult and low quality.

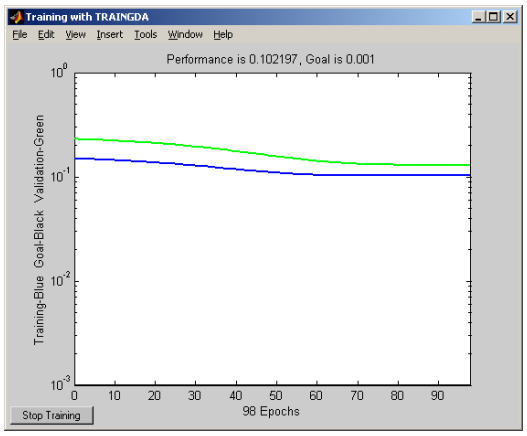

Figure 5. ANN training and testing by GDA method

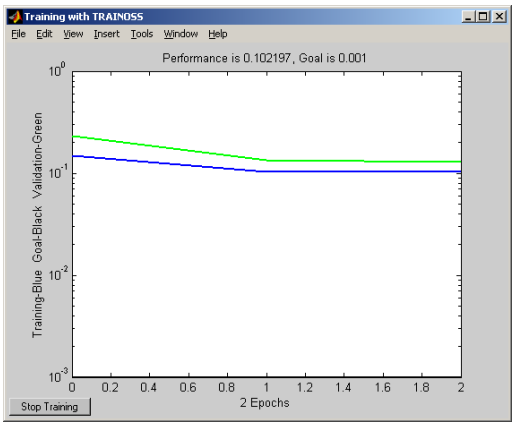

Figure 6. ANN training and testing by OSS method

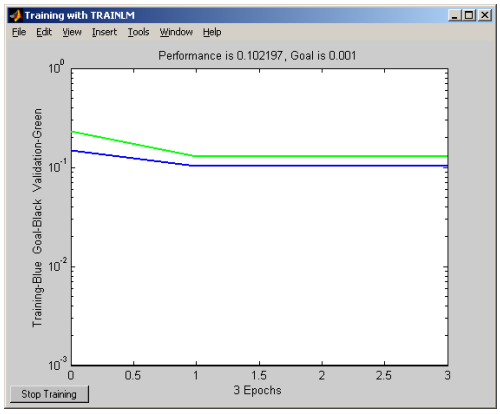

Figure 7. ANN training and testing by LM method

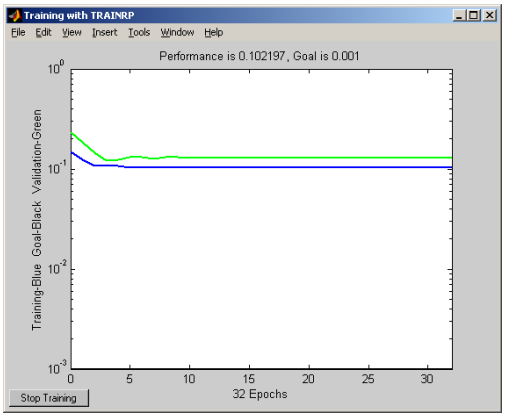

Figure 8. ANN training and testing by RP method

TABLE 2.PROCESSING OF STAGE DESIGN METRICS USING ANN

Project №	**Stage Design Metrics**	**ANN Result**
1	Chepin's metric - 1700 Cohesion metric -10 Coupling metric - 1 Expected Lines Of Code (LOC) - 3300 Halstead's metric - 73400 Software design total time - 27 working days Software design expected cost - 975 usd Expected functional points (FP) - 120	**Y_1=0,95** **Y_2=1** **Y_3=0,95** **Y_4=0,96**
2	McClure's metric – 90000, Kafur's metric - 376900 Chepin's metric – 24530, Cohesion metric – 3, Coupling metric - 7 Expected Lines Of Code (LOC) - 40135 Halstead's metric – 124928, McCabe's metric - 1903 Software design total time - 396 working days Software design expected cost -19000 usd Expected functional points (FP) - 2220	**Y_1=0,25** **Y_2=0,3** **Y_3=0,2** **Y_4=0,26**

3	Chepin's metric - 14538 , Cohesion metric - 7 Coupling metric - 4 Expected Lines Of Code (LOC) - 25530 Halstead's metric - 781232 Software design total time - 218 working days Software design expected cost - 10762 usd Expected functional points (FP) - 1210 Software quality audit expected cost - 1100 usd	**Y_1=0,55** **Y_2=0,6** **Y_3=0,51** **Y_4=0,57**
4	McClure's metric – 12000, Kafur's metric - 64238 Chepin's metric – 3241, Cohesion metric - 9 Coupling metric - 3 Metric of the global variables calling - 0,089 Expected Lines Of Code (LOC) - 6240 Halstead's metric - 162251 Jilb's metric (relative logical complexity)- 0,12 McCabe's metric - 298 Software design total time - 69 working days Software design expected cost - 3660 usd Effort applied by Boehm- 60 man-month	**Y_1=0,9** **Y_2=0,92** **Y_3=0,89** **Y_4=0,86**
5	Chepin's metric - 27625 Jilb's metric (absolute modular complexity) - 2090 Cohesion metric – 1, Coupling metric - 9 Quantity of found bugs during the models and prototype inspection - 4321 Expected Lines Of Code (LOC) - 47550 Halstead's metric - 1484376 Jilb's metric (relative logical complexity)- 0,95 Software design total time - 488 working days Software design expected cost - 23375 usd Effort applied by Boehm- 377 man-month	**Y_1=0,15** **Y_2=0,11** **Y_3=0,05** **Y_4=0,06**

On the basis of ANN results, design cost and time (Table 3) the choice of project version was performed.

TABLE 3. THE CRITERIONS OF CHOICE OF PROJECT VERSION

Version	**Values Y_1, Y_2**	**Values Y_3, Y_4**	**Design cost**	**Design time**
1	Y_1=0,51 Y_2=0,56	Y_3=0,60 Y_4=0,57	10875 usd	200 working days
2	Y_1=0,32 Y_2=0,35	Y_3=0,38 Y_4=0,37	11125 usd	210 working days

The characteristics of project versions from Table 3 evidence that both versions have approximately the same design cost and time, but significantly different estimates of project complexity and quality and prediction of designed software complexity and

quality. On the basis of only cost and time software company can make a false conclusion about selection of the project version. ANN evaluations help to make the right selection.

ACKNOWLEDGMENT

The necessity and actuality of scientific research in software quality evaluation and prediction comes from the results of the software metric evaluation methods analysis.

The main parameters in the selection of software project version are the design cost and time and designing company reputation, but decisions on the basis of these parameters are not always guarantee the proper software quality. Predicted evaluations of designed software complexity and quality give the prediction about complexity and quality of concrete project version realization and allow comparing the different project versions, when the cost and time is approximately equal. The proposed approach provides the motivated and grounded decision about selection of the project version on the basis not only cost and time, but also considering quality characteristics.

Research showed that in future attention should be paid to the solution of: 1) the problem of metric information lack to increasing of the training and testing samples size; 2) the development of designed software complexity evaluation metrics from the viewpoint of the maintenance difficulty or simplicity, usability and the effectiveness of the methods chosen to solve the task; 3) ANN architecture optimization and ANN performance function selection.

REFERENCES

[1] Bishop P. A Methodology for Safety Case Development / P. Bishop. - 1998

[2] Kelly T. Arguing Safety – A Systematic Approach to Managing Safety Cases / T. Kelly. - 1998

[3] A. Gordeyev, V. Kharchenko, A. Andrashov, B. Konorev, V. Sklyar, A. Boyarchuk. Case-Based Software Reliability Assessment by Fault Injection Unified Procedures // Proceedings of the 2008 International Workshop on Software Engineering in East and South Europe – Germany, Leipzig, 2008. – pp. 1-8

[4] Pomorova O.V., Hovorushchenko T.O. Analysis of Methods and Tools of Software Systems Quality Evaluation // Radioelectronic and Computer Systems – Kharkiv: KhAI, 2009 – № 6, pp.148-158

[5] Pomorova O.V., Hovorushchenko T.O. Intelligence Method of Design Results Evaluation and Software Quality Characteristics Prediction // Radioelectronic and Computer Systems – Kharkiv: KhAI, 2010 – № 6, pp.211-218

[6] Pomorova O.V., Hovorushchenko T.O., Tarasek S.Y. Analysis and Processing of Software Quality Metrics on the Design Stage // Transactions of Khmelnitsky National University – Khmelnitsky: KhNU, 2010. - № 1, pp.54-63

[7] Eric J. Braude. Software Engineering: An Objected-Oriented Perspective - New York: Wiley Computer Publishing, 2004. -655 p.

[8] F. Wasserman, Neurocomputer Techniques: Theory and Practice [Russian translation]. Moscow: Mir, 1992. - 240 p.