UDC 004.891.3: 004.3

T.O.Govorushchenko

PROBLEMS OF SOFTWARE HIDDEN BUGS IDENTIFICATION NEURONET TECHNIQUE REALIZATION

Khmelnitskiy National University

Essence of category software retesting process model and software hidden bugs identification neuronet technique, realization of software hidden bugs identification system and problems of technique realization are elucidated in article.

Preamble

Software testing on different life cycle stage is one of basic means with the help of which functioning high reliability computer engineering correct and application task solution accuracy are achieved.

Realization of software analysis how diagnosis object give a chance to draw next conclusions:

1) new architecture development and introduction, computers and systems complication this time take the lead over system and application software testing techniques and means development;

2) being diagnostic programs not always completely allow for increasing requirements to software development;

3) diagnostic program reduce low quality effectiveness and reliability of being computer system software.

One of basic software testing constituent is testing how finding program bugs process. Its part increase because of modern computer system software is sufficiently complicated and cannot be free of defects. Cause of not-finding software bugs is tests imperfection, but not program zero-defects.

From software techniques analysis it was cleared, that none of the make not universal and have certain defects.

Finding of software testing defects, including of hidden software bugs identification, is pressing task of software testing techniques development which software reliability is raised.

For software testing process reliability increasing conception of software testing reliability increasing, software retesting neuronet category model, software hidden bugs identification neuronet technique and software hidden bugs identification system were developed [1-3].

Developed software testing reliability increasing conceptual model [1] is provide for software retesting neuronet category model choice and substantiation.

Research goal and tasks

The research goal is software testing process reliability increasing at the expense of hidden bugs identification by software retesting.

Science task is software testing process reliability increasing by neuronet information technologies using.

Software retesting neuronet category model

Choice of ANN instrument motivated by ANN enables to allow for importance (weights) every type of detected non-hidden errors, admissible thresholds of boundary quantity of errors for each category and interference of the different category hidden bugs [4].

We used ANN, in which a multi-layer-perceptron (MLP) composed with simple Rosenblatt perceptron (fig.1).

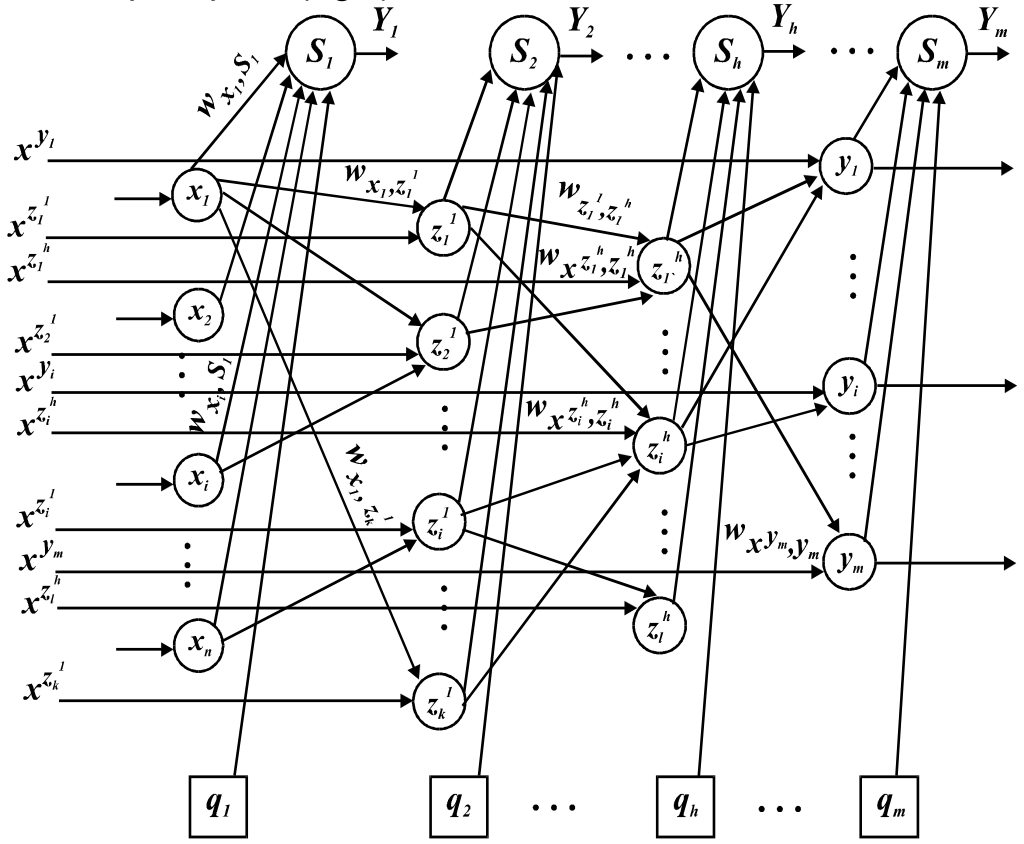

Fig.1. ANN, which take account interference of the different categories program errors

Functional Y_1 is determined as: $Y_1 = F_{S_1}\left(\sum_{i=1}^{n} x_i w_{x_i} - q_1 w_{q_1}\right)$.

Functional Y_h:

$$Y_h = F_{S_h}\left(\sum_{i=1}^{h}\left(f(z^h)\cdot\left(\sum_{i=1}^{h} z_i^{h-1} w_{z_i^{h-1},z_i^h} + \sum_{i=1}^{\ell} x^{z_i^h} w_{x^{z_i^h},z_i^h}\right)\right)\cdot w_{z_i^h,S_h}\right) - q_h w_{q_h,S_h},$$

where $f(z^h)$ - activation function of neurons of latent layer h; z_i^{h-1} - starting value of activity of i-th neurons of latent layer $(h-1)$; $w_{z_i^{h-1},z_i^h}$ - weight coefficient of connection between i-th neurons of ANN latent layer $(h-1)$ and latent layer h; $x^{z_i^h}$ - i-th ANN entrance, which is directly connected with i-th neuron of layer h; $w_{x^{z_i^h},z_i^h}$ - weight coefficient of connection between entrance $x^{z_i^h}$ and i-th neurons of layer h;

$w_{z_i^h, S_h}$ - weight coefficient of connection between i-th neurons of layer h and neuron S_h.

Activation function of hidden (associative) layer is hyperbolic tangent function. Activation function of effectors layers is linear function, which gives results in the range [-1,1] . We transformation (round-up) results at the following way: $Y_i = 1$, if $Y_i > 0$; $Y_i = 0$, if $Y_i < 0$ or $Y_i = 0$.

The ANN simulation model was developed in Matlab [2, 4].

Input data were coded for ANN training. According to statistics, tester tests program using no more than four methods and four operations of the same basic software testing methods [5].

On each of ANN inputs q_1 - q_4 "0" or "1" must be transmitted.

On input x_i numbers of errors types it was finding during basic testing were transmitted. According to statistics maximum 14-15 mistakes were present in program, therefore at this input can be transmitted no more than 16 errors types.

On Input2 (x^{Z_1}) Input3 (x^{Z_2}), Input4 (x^{Z_3}), Input5 (x^{Z_4}) basic testing operations numbers were transmitted. On Input2 operations numbers, which first category level bugs (minor) [6] were discovered, were transmitted. On Input3 operations numbers, which second category level bugs (moderate) were discovered, were transmitted. On Input4 operations numbers, which third category level bugs (serious) were discovered, were transmitted. On Input5 operations numbers, which fourth category level bugs (catastrophic) were discovered, were transmitted.

Each output Y_i is correlated with i-th error's category and is possess the value "1" if ANN prognoses presence of i-th category error, else output Y_i is possess the value "0".

Choice of ANN training algorithm was investigate on base sample of the 2250 vectors. Different training algorithms use different criterions of training quality evaluation.

As a result of investigation next was specified: training algorithm CGB on base conjugate-gradient method with back propagation and restarts in modification of Pauel-Biele, onestep algorithm of secant method, training algorithm SCG and threshold back-ropagation error algorithm with using of quality composite test are the most effective, convenient and expedient. Performance of simulated ANN training depends on criterion of training quality evaluation. Minimum training error, which reached with training sample of 2250 vectors, is 0.11209.

Software hidden bugs identification neuronet technique [2,3]

Input data for software retesting realization are information about bugs types, techniques and operations of basic testing. This information was taken from basic testing reports, which were given from testers in the form of journal "Testing technique – Testing operation – Operation result (finding bag type)". With the help of serialization matrixes input report transformation from linguistic to numerical form is taken place. Input data numerical matrix is given to ANN inputs. From output ANN data out-

put vectors numerical matrix is formed. On ANN outputs is 4 category level, 0 on which signify this category level bag absent, 1 – this category level bag present. ANN output vectors numerical matrix is transformed into linguistic form with the help of serialization matrix.

Software hidden bugs identification neuronet technique is differed from known thereby, that input information about basic testing results is processed by ANN (software retesting neuronet category model).

Proposed technique should used on incoming inspection stage, because this technique helps to assess software quality to customer and points to software hidden bugs presence.

Software hidden bugs identification system

Structure schem of software hidden bugs identification system is showed in [3].

On the block of data connection user file with results of the basic testing represented as a testing journal «Testing method – Testing operation – Finding mistake type» is fed. The file data are processed by the encoder. Encoder transforms input data from a linguistic form in a quantitative form, fills the knowledge base by input data and forms entrances vectors for decision maker. Knowledge base contains tables with input data of system, auxiliary tables and tables with rules for the forming deduction about a necessity and method(s) of the repeated testing. Solution of tasks of hidden mistakes finding is based on category model of process of repeated testing [3], in which considering of importance of each type mistakes, interference of mistakes types, fuzzy input data about existent mistakes is allowed, and is possible with the artificial neuron network (ANN) using. Therefore by decision maker it is used an artificial neuron network on the entrances of which information about methods and operations of the basic testing and types of finding during the basic testing software mistake(s) is given, and category level of hidden mistakes is decision maker results. Results of decision maker are given to encoder, which fills knowledge base by result data, transforms of resulting vectors in a linguistic form and are transmitted on the decision maker interpretation module. Decision maker interpretation module on the basis of rules [2] generates a deduction about a necessity and method(s) of the repeated testing, which is transmitted through dialog component to the user. Dynamic guide gives to user information about input file format, known basic software testing techniques and operations, finding mistake types and transmits all messages some system components to user. The result of system functioning is deduction about repeated testing necessity and advisable repeated testing method(s).

Software hidden bugs identification system was realized in Borland C++ Builder 6.0. Dialog box is showed on fig.2.

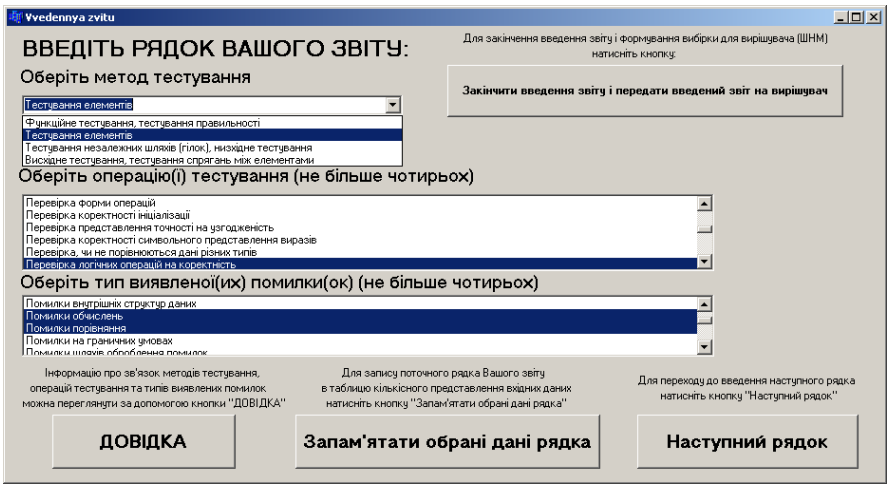

Fig.2. Entering the report line

Problems of software hidden bugs identification neuronet technique realization

Advantages of proposed decision:

- no limitaion on software type and size, on software programming language; single requirement is based on testing report presence;
- taking into account bugs interferences give rise to reliability increasing on 15-28% [7];
- possibility of ANN afterlearning by insufficiency or rise new techniques-operations-bag types in process of software hidden bugs identification system running.

But during technique realization problem sequence, which demand alteration and partial revision of software hidden bugs identification neuronet technique.

First and major problem is correct construction ANN learning sample, which allows for all probable correct combinations of techniques-operations-bugs types and contains uncorrect combinations and boundary (thtreshold) combinations. ANN training sample of 2250 vectors was constructed during processing of data of 9 experts – leading programmers Sitronics Telecom Solutions (Khmelnitskiy branch). The most difficult task for experts was bag category level decision. The second task by complexity was development of uncorrect and boundary (threshold) combination “techniques-operations-bugs types”. Therefore training sample contains majority correct combinations and minority uncorrect and threshold combinations. For this problem solution it is necessary that take the data from experts of different firms-software authors and training sample will be more rich and correct. But the problem of experts finding is also serious. It is impossible to take into account all experts knowledges by means of training sample.

The second problem of proposed techniques using is problem, that majority firm-software authors don’t compose the basic testing reports on the base of their analysis software hidden bugs identification system proposes the conclusion about necessary and technique software retesting, in other words about software hidden bugs presence. Collectives, who compose the report, not always use the classical methods-operations-software bugs types, on the base of them described system was constructed. Therefore by using of reports with non-classical techniques-operations-software bugs

types ANN will be in need of aftertraining with innovation techniques-operations-bugs types processing.

The third problem is the problem of unwillingness of proposed system using by software engineers. This system uses on incoming inspection stage, which customer accomplishes with the purpose of valuation of receptee software development and testing quality and indication of receptee software hidden bugs. Earlier software engineers try hard to devise a software the most qualitative. But market dictates other conditions. Customer pays in advance and demands of fast software development. This situation demands quality losses from the software engineers. It is clear that software engineer doesn't want to realize the development and testing software retesting and doesn't want so that customer knows about hidden bugs presence and tests imperfection. This problem solving may be the developer goal to create quality software.

The next problem is absence of select ANN structure clear rules. Software retesting neuronet category model is generalized complicated structure ANN (multilayer-perceptron and simple Rozenblatt perceptron), which is enough for software retesting tasks solution. With using other types ANN for retesting task solution task nature will be artificially misrepresent and ANN results will be incorrectly interpret.

Conclusions

In this article software testing process reliability increasing conception of software testing reliability increasing, software retesting neuronet category model, software hidden bugs identification neuronet technique and software hidden bugs identification system were described. The basic attention was given the problems of software hidden bugs identification neuronet technique realization and possible solving paths.

Literature

1. Lokazyuk V.M., Panteleeva (Govorushchenko) T.O. Category model of software defects repeated testing process // Transaction of Khmelnitsky National University, 2004, W1, p.53-58. (In Ukrainian)
2. Govorushchenko T.O. Research of application software retesting system decision maker model // Transaction of Khmelnitsky National University, 2007, W3, p.236-244. (In Ukrainian)
3. Govorushchenko T.O. Realization and functioning of application software retesting system // Transaction of Khmelnitsky National University, 2007, W2, p.113-120. (In Ukrainian)
4. Lokazyuk V.M., Pomorova O.V., Govorushchenko T.O. Imitation model of retesting software system // Transaction of Khmelnitsky National University, 2006, W6, p.65-72. (In Ukrainian).
5. Robert Culbertson, Chris Brown, Gary Cobb, Rapid testing – Prentice hall PTR, 2001. - 384 p.
6. V.Lokasyuk, O.Pomorova, T.Govorushchenko. Neural Nets Method for Estimation of the Software Retesting Necessity // Proceedings of the 2008 international workshop on Software Engineering in east and south Europe – Germany, Leipzig, 2008. – pp. 9-14. ISBN 978-1-60558-076-0. (*http://doi.acm.org/10/1145/1370868.1370871*)
7. Govorushchenko T.O. Software hidden bugs finding effectiveness valuation // Transaction of Khmelnitsky National University, 2005, W1, p. 190-195 (In Ukrainian)