UDC 004.891.3 : 004.3

T.O. GOVORUSCHENKO

Khmelnitskiy National University, Ukraine

DETERMINATION OF NECESSITY AND ADVISABLE METHOD(S) OF REPEATED APPLICATION SOFTWARE TESTING

In article the repeated software testing system, which prognoses hidden mistakes presence in software after basic testing and proposes repeated application software testing method(s), is presented. Results of proposed system functioning are represented.

Repeated software testing, Repeated software testing system, Technique of prognostication of hidden mistakes presence, Forming logical deduction rules, Forming logical deduction technique and algorithm.

Preamble

Software testing is one of the basic methods guaranteeing software reliability. In general, there are a lot of conceptions and interpretations of software testing. One of the most common is interpretation, when by testing is implied denials finding process caused by program mistakes presence [1]. The interpretation of testing as starting of source code with testing data and research software system output data and software product performance capabilities for control of system functioning accuracy [2] is worthy of respect. The most widespread interpretations of testing is interpretation, when the testing essence lies in functioning programs controlling by results of their realization using special neat input data (tests) [3]. It, per se, is the method of finding of software mistakes presence by test data processing and referencing testing results with predicted results.

The basic testing is conducted on the different software life cycle phases, especially, on the planning phase, design phase, encoding phase. On the planning phase analysis and evaluation of software requirements, descriptions and compatibility option is conducted. On the design phase there is no source code of program yet, completely formalized and detailed declared ideas are tested. Project conformance to documentation requirements, description by project of all correlations and data transfer between modules requires a special attention. On the encoding phase complete program is tested.

Problem formulation

Presently the problems of test programs development are actual: 1) application software size increases, caused by corporative problems and large analytical problems solving, huge data arrays processing, functioning in great different directions; 2) existing test software correctly functions for previous versions application software, but is not effective for modern application software, because does not overtake development dynamics and does not take into account of it special features.

In addition, software failures may be raised by hidden mistakes. This way such mistakes could be found in rare cases. Therefore such mistakes were found only in process of long software running. The hidden mistakes are the most dangerous. So, *main research task* is development of techniques and means of software testing effectiveness increasing using the hidden software mistakes finding in repeated testing process. Repeated testing is realized as individual technological process after software development and debugging.

1. Problem solving
1.1. Conception of hidden mistakes category levels

As for software mistakes distribution into their kinds and influence on computer system functioning, their distribution by the priorities and categories is known in literature [4]. Mistakes distribution by the priorities is done in compliance with mistakes influence level determination norms.

Accurate definition of that viewpoint of hidden mistakes description was produced [5]. All hidden mistakes were divided by kinds on insignificant (I), moderate (M), serious (S) and catastrophic (C).

By Insignificant (I) *hidden mistakes* will think such mistakes, which do not influence on user actions, program product with their presence suitable for use.

By Moderate (M) *hidden mistakes* will think such mistakes, which influence on user actions, but program product with their presence will be suitable for use with loss of some functionality.

By Serious (S) *hidden mistakes* will think such mistakes, which gives rise to false results, by reason of what program product is unusable.

By Catastrophic (C) *hidden mistakes* will think such mistakes, which gives rise to disfigurement over information (data), by reason of what program product is unusable and results of its work gives rise to failure of computer system.

Insignificant hidden mistakes were classified as a lowermost level category – first (1). Moderate hidden mistakes were classified, accordingly, a level second; serious - level third. Highest by level thought a catastrophic – level fourth. Classified by rank, levels of hidden mistakes are four.

Certain quantity of the insignificant mistakes (I′) gives rise to appearance of several moderate mistakes types (M′), which, in one's turn, gives rise to appearance of certain serious mistakes quantity (S′), and they, accordingly, gives rise to catastrophic mistakes (C′).

1.2. Repeated software testing system

For solving of problem of software testing efficiency increasing was developed repeated software testing system [6], on entrance of which report of basic testing in the form of register "Testing method – Testing operation – Finding mistake type".

Composition of reports about basic testing process and results is made not always, but firms and collectives, who seriously works on software testing improving, so fulfil.

The structure chart of repeated software testing system is presented on figure 1.

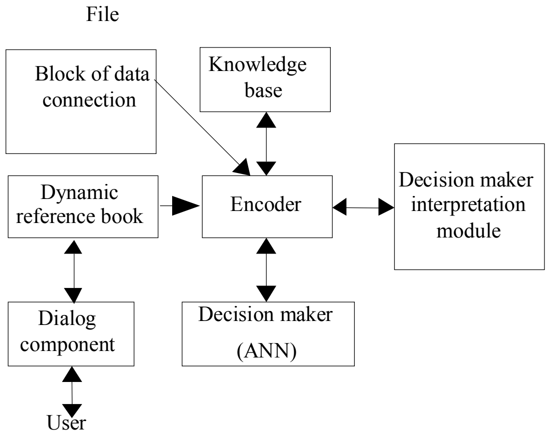


Fig.1. Structure chart of repeated software testing system

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. Encoder checks up file data reliability and completeness during forming entrances vectors. If data not authentic or incomplete, encoder passes on a dynamic reference book the report with suggestion to form another file of the same form, as previous, with additional results which transform into a quantitative form like to data of basic file, after that they are added to the knowledge base. 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 [5], 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 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 table and table «Decision maker results» generates a deduction about a necessity and method(s) of the repeated testing, which is transmitted through dialog component to the user.

The result of system functioning is deduction about repeated testing necessity and advisable repeated testing method(s).

Offered repeated software testing system allows to the user, giving in this system a report about the results of the basic testing, to solve a problem of decision making about the necessity of the repeated testing, in other words presence of the hidden mistakes, and to take recommendation about a method(s), which must carry out the repeated testing.

1.3. Technique of prognostication of hidden mistakes presence on basis of basic testing results

On base of offered approach to distribution of hidden mistakes for their categories will enter set $A = \{a_h \mid h = 1..s\}$, where a_h – threshold of admissible mistakes quantity and importance of mistakes of different type of the same kind, in the excess of which necessary to realize a repeated testing for the purpose of finding of such kind hidden mistakes, h – quantity of thresholds types, which changes from 1 to s, s – quantity of hidden mistakes category levels ($s = 4$). This will raises, for one’s part, testing process effectiveness at all, and also quality of program product.

Have in mind, that the mistakes of the some category level can be causing of appearance of mistakes of not only following category level, but also appearances of higher category levels mistakes.

Assertion 1. If summary quantity and importance of mistakes of h-th category level exceeds a threshold $a_h \in A$, then cause of presence of others (higher) category levels mistakes is mistakes of h-th category level.

This assertion is consequence of offered conception of hidden mistakes categories and task raising of repeated testing.

1.4. Technique of forming logical deduction about necessity and advisable method(s) of repeated testing

For description of rules for forming deduction about necessity of the repeated testing [6] we will enter the threshold $a_i \in A$, at exceeding of which it is necessary to carry out the repeated testing with the purpose of hidden mistakes of this level finding. Then rules for forming deduction about necessity of the repeated testing look like: «if ratio of total quantity of mistakes of the i-th category level to the common quantity of finding during the basic testing mistakes exceeds a threshold a_i, then the repeated testing is necessary ».

For description of rules for forming deduction about advisable method(s) of the repeated testing we will enter the threshold $b_j \in B$, where $B = \{b_p \mid p = 1..z\}$ – set of thresholds of allowed mistake(s) of each type quantity finded during the basic software testing, p – quantity of thresholds types, which changes from 1 to z, z – mistakes types quantity of known by system (z=22). Then rules for forming deduction about method(s) of the repeated testing look like: «if quantity of finding during the basic software testing mistake of type j more than 0, then the repeated testing is recommended to conduct by a method which finds the mistakes of type j ».

On the basis of rules for making decision about the repeated testing technique of forming logical deduction about a necessity and method(s) of the repeated testing [6] was developed.

On the basis of ANN's work results the set $K = \{k_i \mid i = 1..4\}$ was formed, where k_i are total values mistakes of each category level. Using the set K and set of category level of hidden mistakes $R = \{rk_k \mid k = 1..n\}$, the set $KR = \{kr_i \mid i = 1..4\}$ of ratios $kr_i = \frac{k_i}{n}$ is formed. On the basis of quantitative form of input data about types of finding during basic testing mistakes we form set $TP = \{tp_k \mid k = 1..nq\}$, where tp_k is quantity of mistake(s) found by method k, nq is quantity of testing methods, in other words $nq = 7$.

Order of revision and application of rules on the basis of the received results (method of search) is determined. Procedure of choice reduces to the determination of search direction and method of its realization. In this research the method of search realization in width in the straight direction is used [7], that is at first the decision maker interpretation module analyzes all rules for forming deduction about the necessity of the repeated testing and using the known facts (elements of vector KR) finds a conclusion, which from these facts follows, and only then, if a conclusion about the necessity of the repeated testing will be formed, analyses the rules for forming deduction about advisable method(s) of the repeated testing and using the known facts (elements of vector TP) a conclusion which from these facts follows will be found.

The analysis of rules for forming deduction about the necessity of the repeated testing is executed as follows. In the set of rules as «if-then» $PR = \{pr_h \mid h = 1..m\}$ a rule for each of elements of set KR is searched. If the value of set element meets the condition of rule left part, this rule is added to the set of selected rules $OPR = \{opr_y \mid y = 1..g\}$. Criterion of choice of single rule for the set OPR not actual, because all rules have identical right part (result), in which conclusion about the necessity of the repeated testing is formed. Consequently, if quantity of the selected rules $g > 0$, then deduction that repeated testing is needed is formed.

After the forming deduction about the necessity of the repeated testing, the rules for forming deduction about advisable method(s) of the repeated testing are analyzed. In the set of rules PR a rule for each of elements of set TP is searched. If the value of set element meets the condition of rule left part, this rule is added to the set of selected rules $OP = \{op_e \mid e = 1..s\}$. Criterion of choice of single rule for the set OP not actual, because if in the set of select rules got s rules, then repeated testing must be conducted by s methods. Consequently, the union of right parts of rules of set OP forms a conclusion about advisable method of repeated testing.

1.5. Algorithm of forming logical deduction about necessity and advisable method(s) of repeated testing

On basis of described above technique was developed algorithm of forming logical deduction about necessity and advisable method(s) of repeated testing (fig. 2).

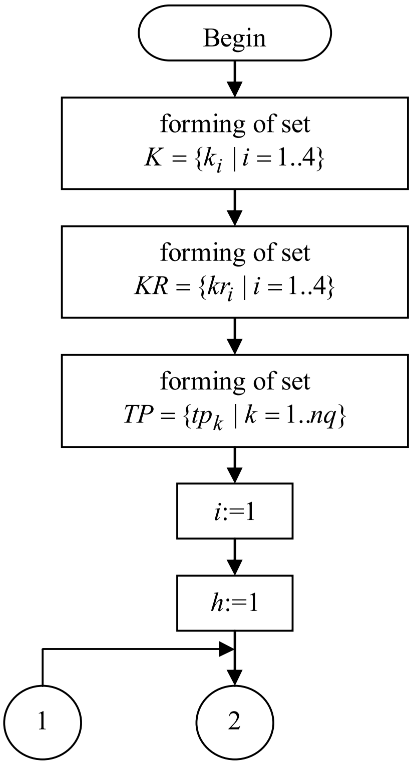
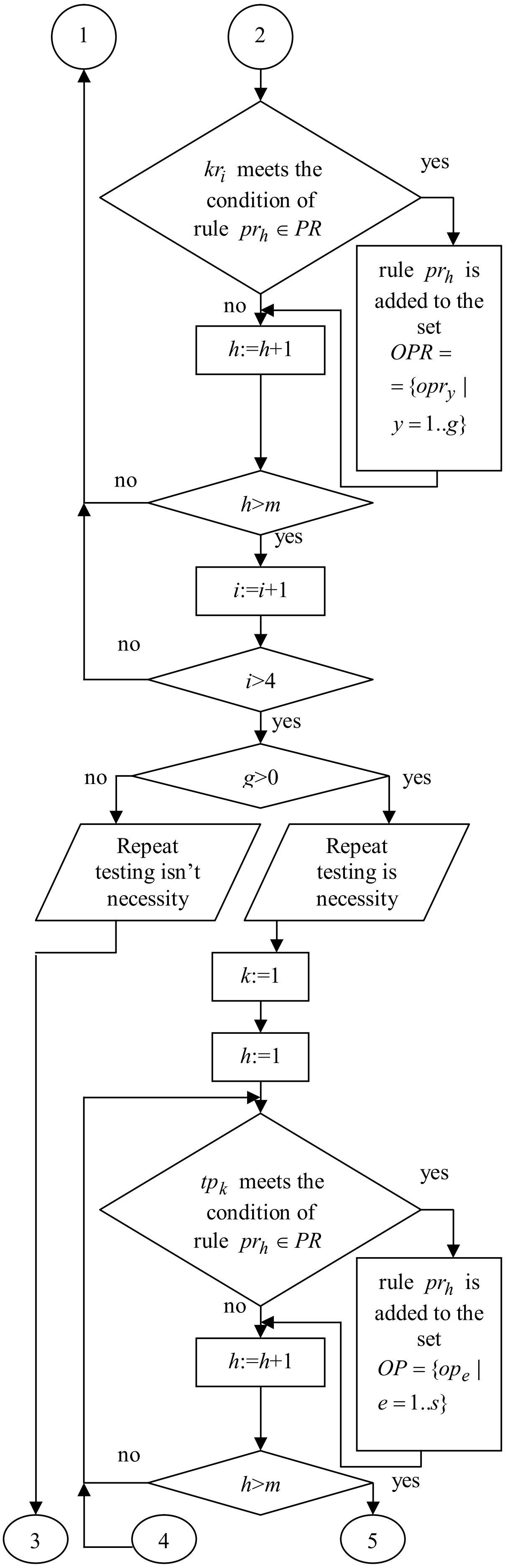

Begin
forming of set
$K = \{k_i \mid i = 1..4\}$
forming of set
$KR = \{kr_i \mid i = 1..4\}$
forming of set
$TP = \{tp_k \mid k = 1..nq\}$
i:=1
h:=1
1
2

Fig. 2. Algorithm of forming logical deduction (part 1)


1
2
kr_i meets the condition of rule $pr_h \in PR$
yes
no
rule pr_h is added to the set $OPR =$ $= \{opr_y \mid$ $y = 1..g\}$
h:=h+1
no
h>m
yes
i:=i+1
no
i>4
yes
no
g>0
yes
Repeat testing isn't necessity
Repeat testing is necessity
k:=1
h:=1
tp_k meets the condition of rule $pr_h \in PR$
yes
no
rule pr_h is added to the set $OP = \{op_e \mid$ $e = 1..s\}$
h:=h+1
no
h>m
yes
3
4
5

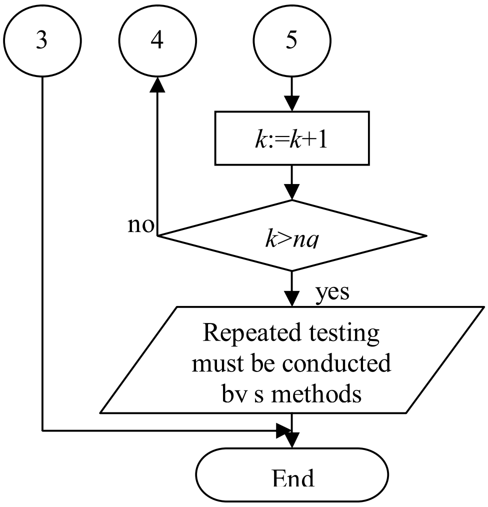


Fig. 2. Algorithm of forming logical deduction (part 2)

2. Results of repeated software testing system functioning

For ANN learning and for input system data forming application programs with open source code from collection of program Examples in package Borland C++ Builder 5.0 were researched, for example, for forming of report about basic testing games Football, Swat, EarthPong from said collection were used. To this programs mistakes of different known for system types were artificially inserted, next program was tested by known for system testing methods and operation, in result of which training sample for ANN (on basis of table 1) and report about basic testing results (table 2) for system functioning examination were formed. Training sample contains 740 training vectors. Vectors of training sample were formed in the following way. Data of each line of table 1 were transformed from a linguistic form in a quantitative form. On forming i-th vector of training sample "1" were got to ANN's inputs q_{mn_i} (mn_i – number of software testing method of i-th line of table 1), x'_{on_i} (on_i – number of software testing operation of i-th line of table 1), x_{pn_i} (pn_i – number of type of finding mistake of i-th line of table 1). "0" were got to all other ANN's inputs.

Report about basic testing results, which gave to system input, contained 80 line.

Table 1

Training sample for ANN

Software testing method	**Software testing operation**	**Type of finding mistake**	**Category level**
Top-down testing	Control of "gag" functioning correctness	Incorrect "gag" functioning	Moderate
Bottom-up testing	Correctness of union of modules to common structure control	Mistakes of union of modules to common structure	Serious
Accuracy testing	Control of correspondence known program functions to received functions	Known program functions mismatch to received functions	Serious
…	…	…	…

Table 2

Report about basic software testing results

№	**Software testing method**	**Software testing operation**	**Type of finding mistake**
1	Testing of independent paths (branches)	Control of accuracy of True and False branches for all logical decisions	Logical conditions mistakes
2	Elements testing	Control of hold data integrity	Mistakes of internal data structures
3	Functional testing	Control of execution of prospective functions by program	Program and its functioning mismatch in advance known program functions
…	…	…	…

After report processing decision maker (ANN) prognosed, that in program 2 hidden mistakes of first category level (insignificant), 4 hidden mistakes of second category level (moderate), 7 hidden mistakes of third category level (serious), 5 hidden mistakes of fourth category level (catastrophic). Then ANN's results gave to decision maker interpretation module, which formed deduction about necessity repeated testing. This deduction depended on ratio of total value of each category level mistakes to the common quantity of finding during basic testing mistakes. As that ratios for third and fourth category levels exceeded threshold of admissible mistakes quantity and importance of own category level mistakes accordingly, then deduction was: "Repeated software testing is necessity".

Conclusions

Offered conception of hidden mistakes category levels allows to prognose hidden mistakes presence on basis of basic testing results.

Offered repeated application software testing system permits allows to make decision about necessity and choice of method(s) of the repeated testing on basis of report about basic testing results.

References

1. A Practical Guide to Testing Object-Oriented Software / John D.McGregor, David A. Sykes. – New York, 2001. – 432 p.
2. Sommerville I. Software Engineering, 6-th edition: Addison Wesley, 2002. – 624 p.
3. Локазюк В.М., Савченко Ю.Г. Надійність, контроль, діагностування та модернізація ПК: Посібник. – К.: Видавничий центр "Академія", 2003. – 376 с.
4. Robert Culbertson, Chris Brown, Gary Cobb. Rapid testing – Prentice hall PTR, 2001. – 384 p.
5. Локазюк В.М., Пантелєєва (Говорущенко) Т.О. Категорійна модель процесу повторного тестування дефектів програмного забезпечення // Вісник Технологічного університету Поділля – Хмельницький: ТУП, 2004. – Ч. 1, т. 1. – С. 53-58.
6. Говорущенко Т.О. Система повторного тестування програмного забезпечення // Радіоелектронні і комп'ютерні системи. – Х.: НАУ "ХАІ", 2005. – № 4. – С. 120-126.
7. Базы знаний интеллектуальных систем / Т.А. Гаврилова, В.Ф. Хорошевский. – С.-Пб.: Питер, 2001. – 384 с.

Надійшла до редакції 27.01.2006

Рецензент: д-р техн. наук, проф. А.О. Саченко, Тернопільський державний економічний університет, Тернопіль.