Method of Developing the Defect-Free Medical Software by Establishing the Presence of Residual Defects

Tetiana Hovorushchenko[a], Peter Popov[b]

[a] *Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, 29016, Ukraine*
[b] *City University of London, Northampton Square, London, EC1V 0HB, United Kingdom*

Abstract

The actual task is to develop the defect-free medical software by establishing the presence of residual defects, and the purpose of this study is to develop an intelligent method & system of developing the defect-free medical software by establishing the presence of residual defects. The paper develops a method of developing the defect-free medical software by establishing the presence of residual defects, the essence of which is to identify the set of defects of different levels of criticality and analysis of this set for the presence or absence of residual defects. The method differs from the known ones in that the input information about the results of the basic testing is processed by an artificial neural network. The proposed system of developing the defect-free medical software by establishing the presence of residual defects allows the user to obtain a conclusion about establishing the presence or absence of residual defects (indicating the level(s) of criticality of residual defects in case of their presence) based on the processing of information on the number and types of defects detected during the basic testing contained in the basic test report. On the basis of heuristic estimates, the threshold values of the number of defects of each criticality level are defined, after exceeding which a conclusion is made about establishing the presence or absence of residual defects (indicating the criticality level(s) of residual defects in case of their presence). In general, the proposed system allows the development of defect-free software by detecting residual defects in medical software after the basic testing, thereby increasing the veracity of testing and, accordingly, increasing the quality of medical software.

Keywords

Software, medical software, software defects, minor software defects, moderate software defects, serious software defects, catastrophic software defects, residual software defects, defect-free medical software.

1. Introduction & State-of-the-art

The density of defects in software of different sizes is represented by a diagram (Figure 1), which shows the typical density of defects per 1000 lines of code for software of different sizes [1-6]. Figure 1 shows that software, which has millions of lines of code, cannot be defect-free at this time. The task of the developers in this case is to ensure the required quality and reliability of the software, taking into account that the software has some unknown number of residual defects that should be identified and blocked or their impact should be reduced to an acceptable level.

Types of software defects: defects in the formation of the software requirements [7-9], defects in software coding [10], defects in software testing and verification [11], defects in software development planning [12], defects in software design [13, 14], defects in software development management [15].

Defects remain in the software even after testing due to objective (imperfection of testing methods and incompleteness of tests, lack of funds for testing, etc.) and subjective (insufficient qualification of

IDDM-2021: 4th International Conference on Informatics & Data-Driven Medicine, November 19–21, 2021, Valencia, Spain
EMAIL: tat_yana@ukr.net (T. Hovorushchenko), p.t.popov@city.ac.uk (P. Popov)
ORCID: 0000-0002-7942-1857 (T. Hovorushchenko), 0000-0002-3434-5272 (P. Popov)

software developers and testers, the impact of their inherent subjective shortcomings, etc.) factors [16-19]. Such defects are referred to as residual defects.

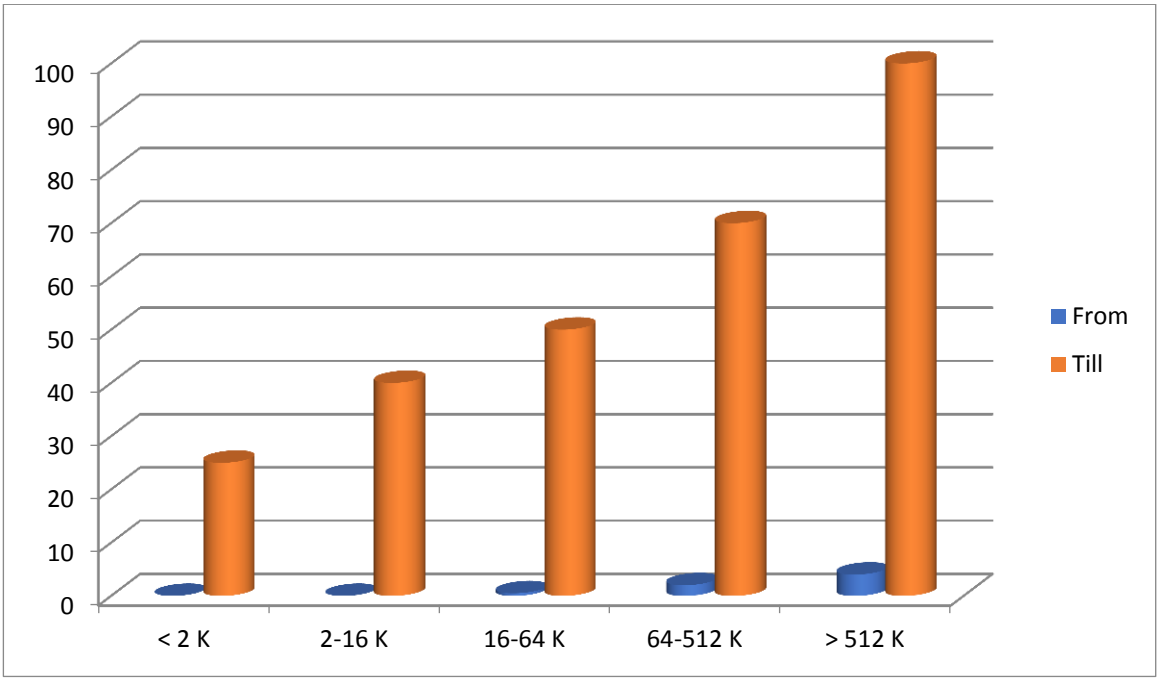

Figure 1: Diagram of typical defect density per 1000 lines of code for software of different sizes

Residual defect is any software defect that remains in the software product after its testing and debugging.

Residual defects differ from those detected in that they at a certain point in time after testing and debugging the software exist and have not yet been identified.

The presence of residual defects in the software can lead to faults and failures, accidents and catastrophes, the consequences of which are financial losses, information losses, reputational losses, and even human losses.

As a result of the manifestation of residual defects, the following negative events have been occurred:

1. Due to a large number of residual defects in Shadow's automated internal voting system software, the US Democratic Party was unable to vote in 2020, that completely undermining its reputation and credibility in the US market [20];

2. Residual defects in the software led to failures in British Airways' information systems for online check-in and flight services during flights in 2019, as a result of which British Airways canceled 117 flights at Heathrow Airport, 10 – at Gatwick Airport; the loss exceeded $ 200 million [21];

3. Due to residual software defects in the automated system timing, Boeing's first Starliner CST-100 spacecraft in 2019 started engines ahead of schedule and burned excess fuel, because of what the spacecraft from entering orbit and docking to the International Space Station [22];

4. Residual defects in the software led to the inability to collect debts on loans and could lead to a loss of revenue of Provident Financial in the amount of $ 158 billion in 2018, due to which the value of shares of the British financial company Provident Financial fell by a total of $ 2.2 billion [23];

5. In 2018, in the United Kingdom, 709000 medical records were not delivered to patients or their therapists due to residual defects in the software, resulting in 1,700 cases where untimely data led to a serious deterioration in patients' health [23];

6. Residual defects in the software led to an increase in prison terms ranging from 0.5 to 1.5 years in the United States in 2018, as a result of which hundreds of prisoners filed lawsuits [23];

7. In 2017, a residual defect in the software was found in Fiat Chrysler trucks, which temporarily disabled the airbags and seat belts, resulting in a fatal accident, and Fiat Chrysler withdrew 1.25 million trucks already sold.

On the one hand, the annual increase in the number of lines of program code [25], and on the other – the increase in the number of news about software defects [26], gives the right to argue the potential increase in the number of residual software defects.

Even more residual defects remain in multidisciplinary medical software, which is developed at the intersection of the medical domain and software engineering domain, when there is a different understanding of the context of information, the need to consider both software development standards and standards of the medical field [2, 6].

Since both the veracity of testing and the quality of software depend on the number of defects found in it, including residual, the increase of the veracity of testing and, accordingly, the quality of software can also be achieved by developing the defect-free software by detecting residual defects in software after basic testing.

Then *actual task* is to develop the defect-free software by establishing the presence of residual defects, and *the purpose of this study* is to develop an intelligent method & system of developing the defect-free software by establishing the presence of residual defects.

2. Concept of Developing the Defect-Free Medical Software by Establishing the Presence of Residual Defects

Let's clarify the distribution of the type of software defects to form a conclusion about the presence or absence of residual software defects. All software defects are divided into minor, moderate, serious and catastrophic.

Minor software defects will be such defects that do not affect the user's actions, the software with their presence is suitable for use.

Moderate software defects are considered to be such defects that affect the user's actions, but the software with their presence is still suitable for use with partial loss of functionality.

Serious software defects will be considered such defects that lead to erroneous results, as a result of which the software is already unusable.

Catastrophic software defects will be considered such defects that lead to distortion of information (data), as a result of which the software is unusable and an attempt to use it leads to system failure.

Minor defects are assigned the lowest criticality level – the first. Moderate defects are assigned, respectively, the 2nd criticality level; serious – the 3rd criticality level. The highest criticality level is assigned to catastrophic defects – 4th level.

The accumulation of a certain number of minor defects leads to the appearance of moderate defects, the accumulation of a certain number of which, in turn, leads to the appearance of serious defects, the accumulation of a certain number of which, accordingly, leads to the appearance of catastrophic defects. This is justified by the fact that the longer the defect is in the software development cycle, the more it penetrates into all software components and the more damage it causes at all stages to all components. Thus, defects of one level of criticality can be the cause of residual defects not only of the next level of criticality, but also of residual defects of higher levels of criticality. In this case, we introduce a threshold of the allowable number of defects, above which it is necessary to conclude that the presence of residual defects of the next level of criticality.

The initial data for forming a conclusion about the presence or absence of residual defects of a certain level of criticality is information about the number and types of defects detected during the basic testing, i.e. the report on basic software testing.

Finding as many residual defects as possible will increase the veracity of the testing process and, accordingly, the quality of the software, which, in turn, will make it possible to develop the defect-free software.

The concept of developing the defect-free medical software by establishing the presence of residual defects using an artificial neural network (for solving the difficult-to-formalize problem of determining the importance and criticality of software defects and their mutual influence, vagueness of initial data about existing defects, etc.) is presented in Figure 2.

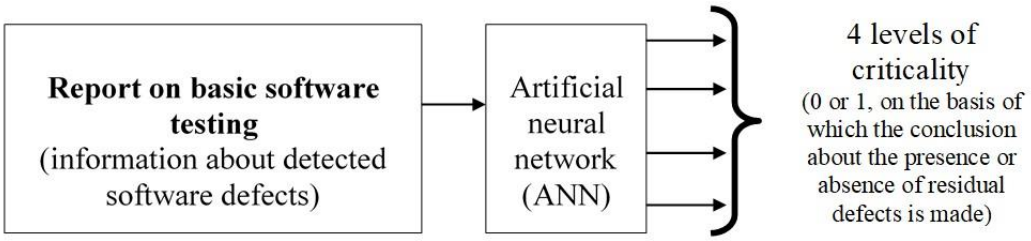

Figure 2: Concept of developing the defect-free medical software by establishing the presence of residual defects

3. Method & System of Developing the Defect-Free Medical Software by Establishing the Presence of Residual Defects

For describing the rules of forming a conclusion about establishing the presence of residual defects, we enter the threshold $r_i \in R$ ($R=\{r_j \mid j=1..k\}$, where r_j – the threshold of the allowable number of defects, exceeding of which leads to the conclusion is formed about the presence of residual defects, j – the number of criticality types varying from 1 to k, k – the number of criticality levels of defects (i.e. now $k=4$)).

The general rule. If the ratio of the total value of defects of the i-th criticality level to the total number of defects detected during the basic testing (or the total value of defects of the i-th criticality level) exceeds the threshold r_i, it is concluded about establishing the presence of residual defects $(i+1)$-th (next) criticality level.

The following *specific rules of forming the conclusion about establishing the presence of residual defects* follow from the general rule:

1. If the ratio of the total value of defects of the 1st level of criticality (minor defects) to the total number of defects detected during the basic testing is equal to or exceeds the threshold r_1, it is concluded about establishing the presence of residual defects of the 2nd criticality level (moderate defects).

2. If the ratio of the total value of defects of the 2nd level of criticality (moderate defects) to the total number of defects detected during the basic testing is equal to or exceeds the threshold r_2, it is concluded about establishing the presence of residual defects of the 3rd criticality level (serious defects).

3. If the total value of defects of the 3rd criticality level (serious defects) exceeds the threshold r_3, the conclusion is made about establishing the presence of residual defects of the 4th criticality level (catastrophic defects).

4. If the total value of defects of the 4th criticality level (catastrophic defects) is equal to or exceeds the threshold r_4, then a conclusion is made about the unsuitability of the software and the possible failure of the system.

Method of developing the defect-free medical software by establishing the presence of residual defects consists of the following steps:

1. Based on the results of the artificial neural network (ANN) the set $D=\{d_i \mid i=1..4\}$ is formed, where d_i – the total values of the defects of each of the criticality levels.

2. Based on the set D the set $DN=\{dn_i \mid i=1..4\}$ of ratio $dn_i=\frac{d_i}{n}$ is formed, where n – the total number of software defects detected during the basic testing.

3. All the rules of forming the conclusion about establishing the presence of residual defects are analyzed, and based on known facts (elements d_3 and d_4 of the set *D;* elements dn_1 and dn_2 of the set *DN*) conclusions are sought, which follow from these facts. The analysis of the rules of forming the conclusion is as follows. In the above set of rules the search for a rule for elements d_3 and d_4 of

the set *D* and elements dn_1 and dn_2 of the set *DN* is conducted. If the value of the elements satisfies the condition of the left part of the rule, then this rule, in fact, becomes one of (or the only) conclusions of the method. If none of the rules worked, then a conclusion is made about establishing the absence of residual defects.

Based on the developed rules and method the structure of *system of developing the defect-free medical software by establishing the presence of residual defects* is developed (Figure 3).

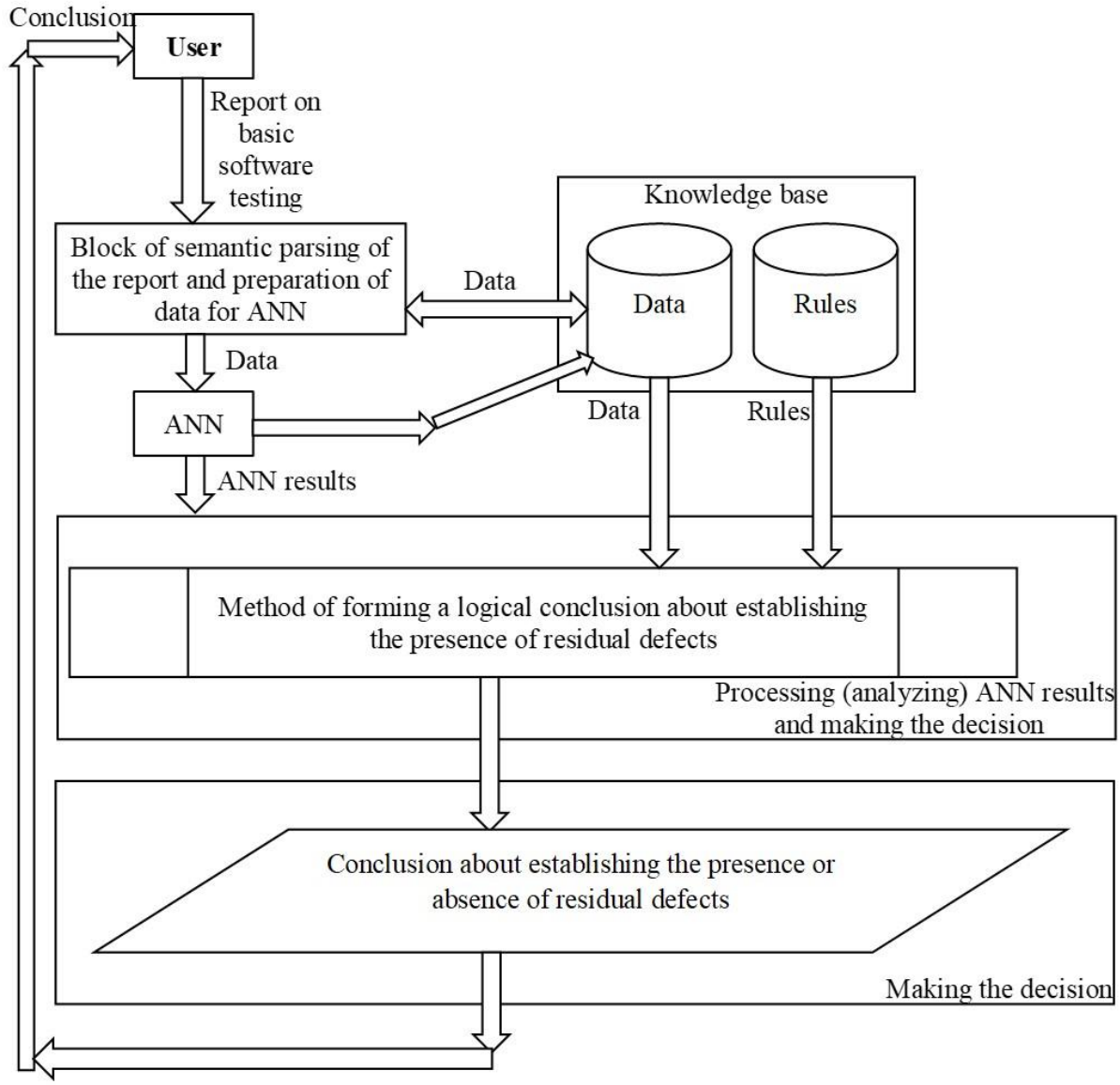

Figure 3: Structure of the system of developing the defect-free medical software by establishing the presence of residual defects

System of developing the defect-free medical software by establishing the presence of residual defects functions as follows. The user of the system submits a report on basic testing of the medical software. Block of semantic parsing of the report and preparation of data for ANN performs semantic parsing of the report, selecting from it information about software defects found during the basic testing, and then converts this information from the linguistic form of presentation to quantitative form using data tables of the knowledge base. Knowledge base contains tables for assigning numbers to the types of detected software defects. ANN processes all obtained quantitative information about software defects detected during the basic testing, and then provides information about the levels of criticality of detected defects (0 or 1 for each of the 4 levels, where 1 means that the defect belongs to

this level of criticality). The results of ANN are recorded in the data table of the knowledge base. The knowledge base also contains the above rules for forming the conclusion about establishing the presence of residual defects. Further, according to the above method of developing the defect-free software by establishing the presence of residual defects, a conclusion about establishing the presence or absence of residual defects is formed (indicating the criticality level(s) of residual defects in case of their presence) and issued to the user of the system.

Therefore, the proposed system of developing the defect-free medical software by establishing the presence of residual defects allows the user, based on the report on the results of the basic testing, to obtain a conclusion about establishing the presence or absence of residual defects (indicating the criticality level(s) of residual defects in case of their presence).

Since the clarification of the distribution of the type of software defects to form a conclusion about the presence of residual software defects was carried out in this study, the threshold values of the number of defects of each criticality level, exceeding which the conclusion is made about the presence of the residual defects (indicating the criticality level(s) of residual defects in the case of establishing their presence), are not described in known literature references.

For the establishment of these thresholds, a study of the number of the medical software defects was conducted, which was developed by software companies in Khmelnytskyi and consisted of a different number of lines of code, taking into account and without taking into account the impact of defects of one criticality level on the occurrence of the defects of the next criticality level. The results of this study are shown in Table 1.

Table 1

The number of the medical software defects, taking into account and without taking into account the impact of defects of one criticality level on the occurrence of defects of the next criticality level

	The number of detected defects without taking into account the impact of defects of one criticality level on the occurrence of defects of the next criticality level			The actual number of defects		
Number of lines of code	100	1000	10000	100	1000	10000
The total number of defects	10	25	68	16	36	85
Minor defects	8	19	51	8	19	51
Moderate defects	2	5	16	5	14	33
Serious defects	0	1	1	2	2	1
Catastrophic defects	0	0	0	1	1	0

As a result of the analysis of Table 1, the following conclusions can be drawn:

1. For the medical software source code from 100 operators, the number of minor defects is 80% (more than 75%) of the total number of detected defects without taking into account the interaction of defects of one criticality level on the occurrence of defects of the next criticality level, and because of this there 3 moderate defects appearanced; the number of moderate defects is 50% of the total number of detected defects without taking into account the interaction of defects of one criticality level on the occurrence of defects of the next criticality level, and because of this there 2 serious defects appearanced that caused 1 catastrophic defect;

2. For the medical software source code from 1000 operators, the number of minor defects is 76% (more than 75%) of the total number of detected defects without taking into account the interaction of defects of one criticality level on the occurrence of defects of the next criticality level, and because of this there 9 moderate defects appearanced; the number of moderate defects is 56% (more than 50%) of the total number of detected defects without taking into account the interaction of defects of one criticality level on the occurrence of defects of the next criticality level, and because of this there 1 serious defects, 2 serious defects appearanced, that caused 1 catastrophic defect;

3. For the medical software source code from 10000 operators, the number of minor defects is 75% of the total number of detected defects without taking into account the interaction of defects of one criticality level on the occurrence of defects of the next criticality level, and because of this there 17 moderate defects appearanced; the number of moderate defects is 49% (not more than 50%) of the total number of detected defects without taking into account the interaction of defects of one criticality level on the occurrence of defects of the next criticality level, therefore, serious defects due to the accumulation of moderate defects did not occur; the same serious defect did not cause a catastrophic defect.

In this case, *the threshold values of the number of defects of each criticality level, in excess of which the conclusion is made about the presence or absence of residual defects (indicating the criticality level(s) of residual defects in case of their presence),* are entered on the basis of heuristic estimates (Table 1) as follows:

1. If the ratio dn_1 of the total value of defects of the 1st level of criticality (minor defects) to the total number of defects detected during the basic testing is equal to or exceeds 75%, it is concluded about establishing the presence of residual defects of the 2nd criticality level (moderate defects).

2. If the ratio dn_2 of the total value of defects of the 2nd level of criticality (moderate defects) to the total number of defects detected during the basic testing is equal to or exceeds 50%, it is concluded about establishing the presence of residual defects of the 3rd criticality level (serious defects).

3. If the total value d_3 of defects of the 3rd criticality level (serious defects) exceeds 2, the conclusion is made about establishing the presence of residual defects of the 4th criticality level (catastrophic defects).

4. If the total value d_4 of defects of the 4th criticality level (catastrophic defects) is equal to or exceeds 1, then a conclusion is made about the unsuitability of the software and the possible failure of the system.

So, elements of the set *R* (thresholds) have the following values: $r_1 = 0.75$; $r_2 = 0.5$; $r_3 = 2$; $r_4 = 1$.

The analysis of Table 1 shows that in excess of such values there are residual defects of higher criticality levels, so these values are used as thresholds in forming a conclusion about establishing the presence or absence of residual defects (indicating the level(s) of critical residual defects in case of establishment of their presence).

4. Results & discussion

Several experiments were conducted with the proposed system of developing the defect-free medical software by establishing the presence of residual defects. During the *first experiment*, the user of the system submitted a report on basic testing of the software module for automation of medical processes of the family medicine outpatient clinic of Ozerna microdistrict (Khmelnytskyi). Block of semantic parsing of the report and preparation of data for ANN performed semantic parsing of the report, selecting from it information about software defects found during the basic testing, and then converted this information from a linguistic form of presentation to a quantitative form using data tables of the knowledge base. ANN processed information about software defects detected during the basic testing, and then provided the following information:

{[0;0;1;0] [1;0;0;0] [1;0;0;0] [0;1;0;0] [0;1;0;0] [0;1;0;0] [1;0;0;0] [1;0;0;0]}

After deciphering these data, it becomes clear that one defect of the 3rd level of criticality (serious), three defects of the 2nd level of criticality (moderate) and four defects of the 1st level of criticality (minor) were diagnosed.

Further, according to the developed method of developing the defect-free software by establishing the presence of residual defects, sets $D_1 = \{4;3;1;0\}$ and $DN_1 = \{0.5;0.375;0.125;0\}$ are formed. All the rules of forming a conclusion about establishing the presence of residual defects are analyzed, and conclusions are found on the known facts, which of these facts follow: $dn_1 < r_1$, i.e. rule 1 does not work; $dn_2 < r_2$, i.e. rule 2 does not work; $d_3 < r_3$, i.e. rule 3 does not work; $d_4 < r_4$, i.e. rule 4 does

not work too. Therefore, none of the rules worked, so the conclusion is made about establishing the absence of residual defects for the 1st experiment.

During the *second experiment*, the user of the system submitted a report on basic testing of the software module "Medical card of an outpatient" of the family medicine outpatient clinic of Ozerna microdistrict (Khmelnytskyi). Block of semantic parsing of the report and preparation of data for ANN performed semantic parsing of the report, selecting from it information about software defects found during the basic testing, and then converted this information from a linguistic form of presentation to a quantitative form using data tables of the knowledge base. ANN processed information about software defects detected during the basic testing, and then provided the following information:

{[0;0;1;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [0;0;1;0] [1;0;0;0] [0;0;0;1] [0;1;0;0] [0;1;0;0] [0;1;0;0] [1;0;0;0] [0;0;0;1] [0;0;1;0] [1;0;0;0] [1;0;0;0] [0;0;0;1] [1;0;0;0] [0;1;0;0] [0;1;0;0] [0;0;0;1]}

After deciphering these data, it becomes clear that four defects of the 4th level of criticality (catastrophic), three defects of the 3rd level of criticality (serious), five defects of the 2nd level of criticality (moderate) and eight defects of the 1st criticality level (minor) have been diagnosed.

Further, according to the developed method of developing the defect-free software by establishing the presence of residual defects, sets $D_2 = \{8;5;3;4\}$ and $DN_2 = \{0.4;0.25;0.15;0.2\}$ are formed. All the rules of forming a conclusion about establishing the presence of residual defects are analyzed, and conclusions are found on the known facts, which of these facts follow: $dn_1 < r_1$, i.e. rule 1 does not work; $dn_2 < r_2$, i.e. rule 2 does not work; $d_3 > r_3$, i.e. rule 3 works; $d_4 > r_4$, i.e. rule 4 works too. Thus, the developed system forms a conclusion about establishing the presence of residual defects of the 4th level of criticality (catastrophic defects) and about the unsuitability of the software and the possible failure of the software module.

During the *third experiment*, the user of the system submitted a report on basic testing of the software module "Laboratory" of the family medicine outpatient clinic of Ozerna microdistrict (Khmelnytskyi). Block of semantic parsing of the report and preparation of data for ANN performed semantic parsing of the report, selecting from it information about software defects found during the basic testing, and then converted this information from a linguistic form of presentation to a quantitative form using data tables of the knowledge base. ANN processed information about software defects detected during the basic testing, and then provided the following information:

{[1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [0;0;1;0] [1;0;0;0] [1;0;0;0] [0;1;0;0] [0;1;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [0;1;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [1;0;0;0] [0;1;0;0] [0;1;0;0] [1;0;0;0]}

After deciphering this data, it becomes clear that one defect of the 3rd level of criticality (serious), five defects of the 2nd level of criticality (moderate) and twenty-four defects of the 1st level of criticality (minor) have been diagnosed.

Further, according to the developed method of developing the defect-free software by establishing the presence of residual defects, sets $D_3 = \{24;5;1;0\}$ and $DN_3 = \{0.8;0.17;0.03;0\}$ are formed. All the rules of forming a conclusion about establishing the presence of residual defects are analyzed, and conclusions are found on the known facts, which of these facts follow: $dn_1 > r_1$, i.e. rule 1 works; $dn_2 < r_2$, i.e. rule 2 does not work; $d_3 < r_3$, i.e. rule 3 does not work; $d_4 < r_4$, i.e. rule 4 does not work too. Thus, the developed system forms a conclusion about establishing the presence of residual defects of the 2nd level of criticality (moderate).

During the *fourth experiment*, the user of the system submitted report on basic testing of the software module "Accounting of the functional diagnostics room" of the family medicine outpatient clinic of Ozerna microdistrict (Khmelnytskyi). Block of semantic parsing of the report and preparation of data for ANN performed semantic parsing of the report, selecting from it information about software defects found during the basic testing, and then converted this information from a linguistic form of presentation to a quantitative form using data tables of the knowledge base. ANN processed information about software defects detected during the basic testing, and then provided the following information:

{[0;1;0;0] [1;0;0;0] [0;1;0;0] [0;1;0;0] [0;1;0;0] [1;0;0;0] [0;1;0;0] [0;1;0;0] [1;0;0;0] [1;0;0;0] [0;1;0;0] [1;0;0;0] [1;0;0;0] [0;1;0;0] [0;1;0;0]}

After deciphering this data, it becomes clear that nine defects of the 2nd level of criticality (moderate) and six defects of the 1st level of criticality (minor) have been diagnosed.

Further, according to the developed method of developing the defect-free software by establishing the presence of residual defects, sets $D_4 = \{6;9;0;0\}$ and $DN_4 = \{0.4;0.6;0;0\}$ are formed. All the rules of forming a conclusion about establishing the presence of residual defects are analyzed, and conclusions are found on the known facts, which of these facts follow: $dn_1 < r_1$, i.e. rule 1 does not work; $dn_2 > r_2$, i.e. rule 2 works; $d_3 < r_3$, i.e. rule 3 does not work; $d_4 < r_4$, i.e. rule 4 does not work too. Thus, the developed system forms a conclusion about establishing the presence of residual defects of the 3rd level of criticality (serious).

During the *fifh experiment*, the user of the system submitted report on basic testing of the software module "Accounting of the office of radiological, fluorographic studies and magnetic resonance imaging" of the family medicine clinic of Ozerna district (Khmelnytskyi). Block of semantic parsing of the report and preparation of data for ANN performed semantic parsing of the report, selecting from it information about software defects found during the basic testing, and then converted this information from a linguistic form of presentation to a quantitative form using data tables of the knowledge base. ANN processed information about software defects detected during the basic testing, and then provided the following information:

{[1;0;0;0] [1;0;0;0] [1;0;0;0] [0;1;0;0] [0;1;0;0] [0;1;0;0] [1;0;0;0] [1;0;0;0] [0;1;0;0] [0;1;0;0]}

After deciphering these data, it becomes clear that five defects of the 2nd level of criticality (moderate) and five defects of the 1st level of criticality (minor) were diagnosed.

Further, according to the developed method of developing the defect-free software by establishing the presence of residual defects, sets $D_5 = \{5;5;0;0\}$ and $DN_5 = \{0.5;0.5;0;0\}$ are formed. All the rules of forming a conclusion about establishing the presence of residual defects are analyzed, and conclusions are found on the known facts, which of these facts follow: $dn_1 < r_1$, i.e. rule 1 does not work; $dn_2 = r_2$, i.e. rule 2 works; $d_3 < r_3$, i.e. rule 3 does not work; $d_4 < r_4$, i.e. rule 4 does not work too. Thus, the developed system forms a conclusion about establishing the presence of residual defects of the 3rd level of criticality (serious).

During the *sixth experiment*, the user of the system submitted report on basic testing of the software module "Accounting for the endoscopic office" of the family medicine outpatient clinic in the Ozerna district (Khmelnytskyi). Block of semantic parsing of the report and preparation of data for ANN performed semantic parsing of the report, selecting from it information about software defects found during the basic testing, and then converted this information from a linguistic form of presentation to a quantitative form using data tables of the knowledge base. ANN processed information about software defects detected during the basic testing, and then provided the following information: {[0;0;1;0] [0;0;1;0] [0;0;1;0] [0;0;0;1] [0;0;0;1]}

After deciphering these data, it becomes clear that three defects of the 3r level of criticality (serious) and two defects of the 4th level of criticality (catastrophic) were diagnosed.

Further, according to the developed method of developing the defect-free software by establishing the presence of residual defects, sets $D_6 = \{0;0;3;2\}$ and $DN_6 = \{0;0;0.6;0.4\}$ are formed. All the rules of forming a conclusion about establishing the presence of residual defects are analyzed, and conclusions are found on the known facts, which of these facts follow: $dn_1 < r_1$, i.e. rule 1 does not work; $dn_2 < r_2$, i.e. rule 2 does not work; $d_3 > r_3$, i.e. rule 3 works; $d_4 > r_4$, i.e. rule 4 works too. Thus, the developed system forms a conclusion about establishing the presence of residual defects of the 4th level of criticality (catastrophic defects) and about the unsuitability of the software and the possible failure of the software module.

The results of all 6 experiments will be presented in the form of a diagram of the detected medical software defects of different levels of criticality – Figure 4.

Thus, as shown by the conducted experiments, the proposed system of developing the defect-free medical software by establishing the presence of residual defects as a result of its operation issues a conclusion about establishing the presence or absence of residual defects (indicating the level(s) of critical residual defects in case of their presence) based on the processing of information on the number and types of defects detected during the basic testing, which is contained in the report of the basic testing.

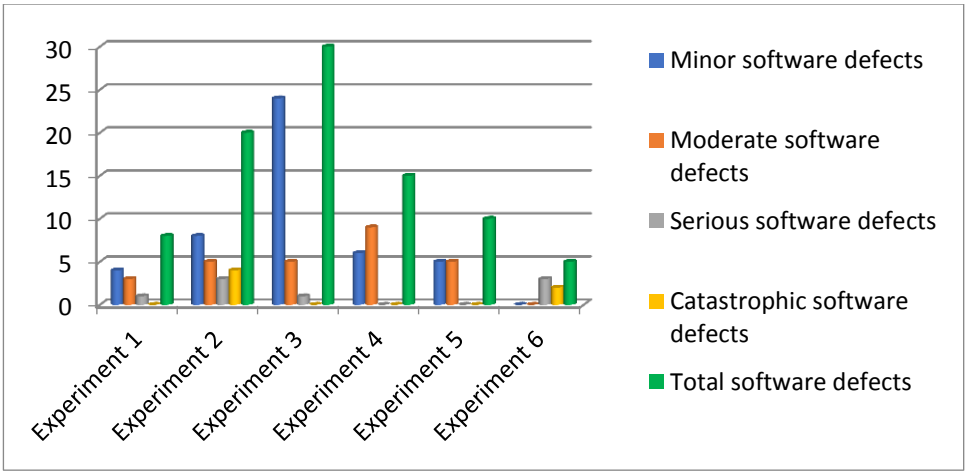

Figure 4: Diagram of detected software defects of different levels of criticality as a result of 4 experiments

Therefore, the proposed system allows developing the defect-free medical software by detecting the residual defects in the software after the basic testing, thereby increasing the reliability of testing and, accordingly, increasing the quality of the medical software.

5. Conclusions

On the one hand, the annual increase in the number of lines of software source code, and on the other – the increase in the number of news about software defects, give the right to rightly affirm a potential increase in the number of residual software defects.

The paper develops the method of developing the defect-free medical software by establishing the presence of residual defects, the essence of which is to identify the set of defects of different levels of criticality and analysis of this set for the presence or absence of residual defects. The method differs from the known ones in that the input information about the results of the main testing is processed by an ANN.

The structure of the system of developing the defect-free medical software by establishing the presence of residual defects is proposed, which allows the user on the basis of the report on the results of the basic testing to get a conclusion about establishing the presence or absence of residual defects (indicating the level(s) of critical residual defects in case of their presence). In general, the proposed system allows the development of defect-free medical software by detecting residual defects in software after the basic testing, thereby increasing the reliability of testing and, accordingly, increasing the quality of the medical software.

6. References

[1] S. McConnell, Code complete, Microsoft Press, Redmond, 2013.

[2] T. Hovorushchenko, Methodology of evaluating the sufficiency of information for software quality assessment according to ISO 25010. Journal of Information and Organizational Sciences 42 1 (2018) 63-85. doi: 10.31341/jios.42.1.4.

[3] T. Hovorushchenko, O. Pavlova, M. Bodnar, Development of an intelligent agent for analysis of nonfunctional characteristics in specifications of software requirements. Eastern-European Journal of Enterprise Technologies 1 2 (2019) 6-17. doi: 10.15587/1729-4061.2019.154074.

[4] T. Hovorushchenko, O. Pavlova, Evaluating the software requirements specifications using ontology-based intelligent agent, in: Proceedings of 2018 IEEE International Scientific and Technical Conference “Computer Science and Information Technologies”, CSIT’2018, Lviv, 2018, vol.1, pp.215-218. doi: 10.1109/STC-CSIT.2018.8526730.

[5] T. Hovorushchenko, O. Pomorova, Evaluation of Mutual Influences of Software Quality Characteristics Based ISO 25010:2011, in: Proceedings of 2016 International Scientific and

Technical Conference “Computer Science and Information Technologies”, CSIT’2016, Lviv, 2016, pp. 80-83. doi: 10.1109/STC-CSIT.2016.7589874.
[6] T. Hovorushchenko, O. Pomorova, Information Technology of Evaluating the Sufficiency of Information on Quality in the Software Requirements Specifications. CEUR-WS 2104 (2018) 555-570.
[7] I. Margarido, J. Faria, R. Vidal, M. Vieira, Classification of Defect Types in Requirements Specifications: Literature Review, Proposal and Assessment, in: Proceedings of 2011 Iberian Conference on Information Systems and Technologies, CISTI’2011, Chaves, 2011, pp. 1-6.
[8] M. Felderer, A. Beer, Using Defect Taxonomies for Testing Requirements. IEEE Software 32 (2015) 94-101. doi: 10.1109/MS.2014.56.
[9] K. Gopal, S. Jadoo, J. Ramgoolam, V. Devi, Software Quality Problems in Requirement Engineering and Proposed Solutions for an Organization in Mauritius. International Journal of Computer Applications 137 (2016) pp. 23-31. doi: 10.5120/ijca2016908698
[10] B. Marjerison, The Software Bug Book: Thoughts on Software Quality, Scotts Valley, Scotland, 2013.
[11] K. Naik, P. Tripathy, Software Testing and Quality Assurance: Theory and Practice, Wiley Publishing, New York, 2011.
[12] G. Blokdyk, Nintendo Software Planning & Development: Second Edition, CreateSpace Independent Publishing Platform, New York, 2018.
[13] S. Loveland, G. Miller, Jr. Prewitt, M. Shannon, Software Testing Techniques: Finding the Defects that Matter, Mandevilla Press, Florida, 2014.
[14] J. Capers, Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies, McGraw-Hill Education, Orlando, 2010.
[15] J. Huang, Software Error Detection through Testing and Analysis, Wiley Publishing, New York, 2009.
[16] A. Drozd, M. Drozd, V. Antonyuk, Features of Hidden Fault Detection in Pipeline Components of Safety-Related System. CEUR-WS 1356 (2015) 476–485.
[17] O. Drozd, V. Antoniuk, V. Nikul, M. Drozd, Hidden faults in FPGA-built digital components of safety-related systems, in: Proceedings of the 14th International Conference “Modern problems of radio engineering, telecommunications and computer science”, TCSET’2018, Lviv-Slavsko, 2018, pp. 805-809. doi: 10.1109/TCSET.2018.8336320.
[18] O. Drozd, I. Perebeinos, O. Martynyuk, K. Zashcholkin, O. Ivanova, M. Drozd, Hidden fault analysis of FPGA projects for critical applications, in: Proceedings of IEEE International Conference “Modern problems of radio engineering, telecommunications and computer science”, TCSET’2020, Lviv-Slavsko, 2020, paper 142. doi: 10.1109/TCSET49122.2020.235591.
[19] O. Mishchuk, R. Tkachenko, I. Izonin, Missing data imputation through SGTM neural-like structure for environmental monitoring tasks. Advances in Intelligent Systems and Computing 938 (2019) 142-151. doi: 10.1007/978-3-030-16621-2_13
[20] How tech firm Shadow sought to revolutionize Democratic campaigns – but stumbled in Iowa, 2020. URL: https://www.washingtonpost.com/technology/2020/02/04/how-tech-firm-shadow-sought-revolutionize-democratic-campaigns-stumbled-iowa.
[21] British Airways passengers stranded after IT failures, 2019. URL: https://www.bbc.com/news/uk-49261497.
[22] Starliner anomaly to prevent ISS docking, 2019. URL: https://spacenews.com/starliner-anomaly-to-prevent-iss-docking.
[23] The 2018 Software Fail Watch Awards, 2018. URL: https://www.tricentis.com/blog/software-fail-awards-2018.
[24] Fiat Chrysler is recalling more than 1.25 million pickup trucks worldwide over a software error that "may be related" to a death and two injuries, 2017. URL: https://www.bbc.com/news/technology-39898319.
[25] What is the cost of poor software quality in the U.S.?, 2021. URL: https://www.synopsys.com/blogs/software-security/poor-software-quality-costs-us.
[26] Software Fails Watch, 5th edition, 2019. URL: https://www.tricentis.com/wp-content/uploads/2019/01/Software-Fails-Watch-5th-edition.pdf.