A METHOD OF A PRIORI SOFTWARE RELIABILITY EVALUATION

Dmitry A. Maevsky, Svetlana A. Yaremchuk, Ludmila N. Shapa
•
Odessa National Polytechnic University, Odessa, Ukraine
e-mail: Svetlana397@yandex.ru

ABSTRACT

In the given paper a method of a priori evaluation of the amount of latent faults, the estimated latent defect density and their determination probability in Software before testing process is described. The possibility of the method usage for different schemes of the Software development process has been found. The a priori estimation of these reliability indexes makes the management decisions at the stage of testing more effective.

1 INTRODUCTION

The information system association with Software is the most important element of the modern society infrastructure and the necessary foundation of economical and socio-cultural activity of the mankind. The growing computer facilities output allows to meet the higher level requirements of the users to information systems. This promotes the increase of their Software complexity and the number of Software faults introduced by developers, and consequently the reliability reduction. It is of great importance for developers to a priori estimate the reliability indexes, i.e. before the beginning of testing process and at its intermediate stages which gives the opportunity to save time and material resources. That is why the development of a method of a priori software reliability estimation is an urgent and actual task.

2 PURPOSE AND TASKS OF THE RESEARCH

According to the standard [ISO 25010:2010], reliability is one of the eight major characteristics of Software quality. New quality and reliability indexes will be determined in the standards [ISO/IES CD 25022], [ISO/IEC 26023], [ISO/IEC CD 25024] which are being developed at present [www.iso.org 2013]. The functioning standards [ISO 9126-2:2003], [ISO 9126-3:2003] represent the reliability indexes used at the moment. The most demanded in Software Engineering are the following ones:

– the amount of latent faults;
– the fault density in the form of ratio:

$$FD = \frac{N_{open\_def}}{Size_{soft}} \tag{1}$$

where N_{open_def} – the number of faults found at the stage of testing, $Size_{soft}$ – the size of the initial program code including thousand lines;

– estimated latent fault density:

$$ELFD = \frac{N^*_{pred\_def} - N_{open\_def}}{Size_{soft}} \tag{2}$$

where $N^*_{pred\_def}$ – the predicted amount of faults contained in the Software;

– the fault revolution degree:

$$FR = \frac{N_{open\_def}}{N^{*}_{pred\_def}} \tag{3}$$

To calculate ELFD and FR indexes the predicted amount of faults in Software is necessary. In the paper [Maevsky at al. 2012(1)] a model of fault amount dependence-upon-the Software complexity has been proposed. The model is based on the following assumptions:

1. Complexity is the main objective reason of fault emergence in Software.
2. Software complexity can be numerically represented with vector $\mathbf{M} = \{M_1, M_2, ..., M_n\}$, which is stated on the basis of finite set value of the statistical complexity metrics of the initial code.
3. Vector $\mathbf{M}$ components are linked with the amount of faults in Software by the linear relationship with the help of relationship coefficients.
4. Other factors influencing on the amount of faults such as developer qualification, developer membership, development conditions, etc. are constant from project to project.

In accordance with these assumptions the predicting estimation of the amount of faults N^{*}_{d} in the researched project is defined by the linear relationship

$$N^{*}_{d} = x_1 \cdot M_1 + x_2 \cdot M_2 + ... + x_n \cdot M_n \tag{4}$$

where $x_1, x_2, ... x_n$ – unknown relationship coefficients. They can be obtained by solving the set of linear algebraic equations using the known values of fault amount and vectors $\mathbf{M}$ for the earlier developed Software projects.

In the paper [Maevsky at al. 2012(2)] the model verification on the experimental data basis of more than thirty Software projects is represented. The average deviation of predicted and actual amount of faults is 11% which is much less than deviation values of known a priori models by Akiyama, Gafniya, Holsted, TRW company (44% to 87%).

In spite of the success achieved there are several unresolved problems that do not give the opportunity to use the model as a basis for developing a practical a priori method of reliability estimation. In particular a developer is to realize: the quantity and types of metrics to be applied for estimating the amount of faults; the ways of increasing the value accuracy of the metrics chosen; the exact testing stages for calculating and using the ELFD and FR indexes.

The purpose of the given paper is to develop an a priori method of reliability estimation on the basis of the model mentioned above. To achieve the purpose a developer is to solve the following problems:

- to determine the most optimal number of metrics for estimated number of faults;
- to develop the sampling of the most informative metrics;
- to develop approaches to metric value calculation taking into account different Software development models and different fault presence probability in Software.
-

3 THE RESULTS OF THE RESEARCH

3.1. DETERMINATION OF OPTIMUM METRIC AMOUNT FOR ESTIMATION OF FAULT AMOUNT

To carry out the research we have chosen forty six object-oriented Software projects with the open initial code published in the Internet resource of the data repository of the International Scientific Conference PROMISE (PRedictOr Models in Software Engineering) [http://promisedata.googlecode.com 2013]. In order to provide the sample representativeness the following criteria have been applied to make the extraction: high frequency of quoting the project name in scientific papers; the functioning of projects in different branches; size tolerances (1000 to 500 000 thousand lines of the code; the number of classes in the project (50 to 13 000 classes). The total code volume of all the chosen projects is more than five million lines. The values of vector **M** components and the amount of faults for each of the classes of Software projects have been taken from the same repository.

With the help of the statistical analysis methods seven metrics – RSC, WMC, LOC, CE, NPM, LCOM and CBO – have been selected of twenty existing ones. They correlate with the amount of faults more than all others. We have used the statistical methods to prove the reliability of the assumption about the closeness to the linear relations between vector **M** components and the amount of faults (significance level 0,01). The chosen metrics have been researched by the factor and discriminant analysis methods, with the usage of the Software package STATGTAPHICS PLUS [http://www.statgtaphics.com 2013]. The investigation of correlation relations between the metric values and the amount of faults allows to decrease the number of metrics which are necessary for the vector **M**, variability reflection and to estimate the metric significance. The factor analysis results are represented in the Table below.

Table 1. The Results of the Factor Analysis

№	Metric's Name	Variability, %	Cumulation, %	Significance Index
1	WMC	83,762	83,762	6,700
2	CBO	5,724	89,486	0,457
3	RFC	4,540	94,026	0,363
4	LCOM	3,469	97,495	0,277
5	CE	1,202	98,697	0,096
6	NPM	0,841	99,537	0,067
7	LOC	0,301	100	0,024

The data in Table 1 show that just one metric has got the statistical significance index higher than 1. This metric reflects 84% variability of all metric value set. The next three metrics are less significant and reflect about 14% variability of the vector **M** values. The last three metrics reflect less than 2% variability and have got the minimum statistical significance. The analysis has allowed to reduce the amount of metrics which are necessary for prediction of the number of estimated faults **from seven to four**.

The discriminant analysis has been made for two groups of values. The first one includes the number of faults. The second one includes the values of the seven chosen metrics. The analysis has allowed to find multiple correlations between the amount of faults and metric values. The data represented in Table 2 show the reduction of the multiple correlation values and low multiple correlation value for three metrics.

Table 2. The Results of the Discriminant Analysis

Amount of Metrics	Multiple Correlation	Significance Index	P-value
1	**0,89875**	**43,7049**	**0,0000**
2	**0,73871**	**1,20115**	**0,0000**
3	**0,67443**	0,834368	**0,0000**
4	**0,55939**	0,455426	**0,0000**
5	0,36029	0,149174	0,5113
6	0,18535	0,355768	0,9065
7	0,16195	0,0269353	0,7505

The statistical significance index is more than 1 only for two metrics. P-value at P<0.05 reflects the statistical reliability of the result for 95% confident level which is demonstrated in four metrics.

Thus, on the basis of the results of the performed factor and discriminant analysis we can claim that two to four metrics are enough to be used as vector $\mathbf{M}$ components for the reliable estimation of the amount of faults.

3.2. THE SEQUENCE OF CHOICE PROCEDURE OF THE MOST INFORMATIVE METRICS

We cannot assert that it is possible to specify the most informative metrics for estimated fault complexity and amount in every specific case because in different Software there can exist their own types of complexity. To choose the metrics a developer needs to use statistical metric and fault data promoted in the earlier compiled Software projects according to the following technique:

Table 3. Correlation Matrix

Metrics	WMC	CBO	RFC	LCOM	CE	NPM	LOC	Amount of faults
WMC	1	0,7007	0,8978	0,8784	0,7958	0,9636	0,8570	0,8285
CBO	**0,7007**	1	0,7180	0,7019	0,7173	0,6637	0,7224	0,6994
RFC	0,8978	0,7180	1	0,8483	0,9202	0,8285	0,9671	0,8650
LCOM	0,8784	0,7019	0,8483	1	0,7193	0,8718	0,8905	0,8965
CE	0,7958	0,7173	0,9202	0,7193	1	0,7038	0,8677	0,7381
NPM	0,9636	0,6637	0,8285	0,8718	0,7038	1	0,7913	0,7935
LOC	0,8570	0,7224	**0,9671**	0,8905	0,8677	0,7913	1	0,8851
Amount of faults	0,8205	**0,6994**	0,8650	**0,8965**	0,7381	0,7935	0,8851	1

1. For every earlier compiled project:

1.1. Form Person's correlation coefficient matrix between $R_{M_i M_j}$ metric value and the amount of faults R_{N_d,M_i}. The example of such a matrix is represented in Table 3;

1.2. Analyze values R_{N_d,M_i}. Exclude metrics with $R_{N_d,M_i} < 0.2$, because there is no necessity to use such kind of metrics for obtaining the estimation of the amount of faults;

1.3. Analyze R_{M_i,M_1} remaining metrics. The metrics with $R_{M_i,M_1} > 0.8$ are multi-collinear. Then choose the collinear metric with the highest value R_{N_d,M_i} of each pair of ones. As a result a developer will obtain from two to four metrics with the highest R_{N_d,M_i} and the lowest R_{M_i,M_1}.

2. For each of the chosen metrics:

2.1. Rank for the choice frequency with assigning each of the metrics $rank^1_{M_i}$;

2.2. Rank for the average $\overline{R}_{N_d,M_i}$ with assigning each of the metrics $rank^2_{M_i}$;

2.3. Choose metrics with the highest total ranks $rank^{sum}_{M_i} = rank^1_{M_i} + rank^2_{M_i}$.

3. For the researched project:

3.1. Form the matrix $R_{M_i M_j}$ and analyze its multi-colinearity;

3.2. Finish the metric choice if the metric multi-colinearity was not established.

3.3. Choose a metric with the highest $rank_{M_i}^{sum}$ of each of the pair of collinear metrics if the metric multi-colinearity was established. Choose the missing metric according to p.2.3. with the next highest value rank $rank_{M_i}^{sum}$.

The proposed technique is easy to implement (for example with the help of Microsoft Excel analysis package) and allows to choose from two to four complexity metrics that are the most informative for estimating the amount of faults in the researched Software.

3.3. APPROACH TO METRIC CALCULATION FOR DIFFERENT SCHEMES OF SOFTWARE DEVELOPMENT PROCESS

The main peculiarity of the model of Software complexity-upon-fault amount dependence is that it is only to be used for a single a priori estimation of the amount of faults in Software. If in finding and correcting the faults the vector **M** component values do not change then the corresponding estimation of the amount of faults will not change as well. This peculiarity is to be taken into account in calculating the metrics for different schemes of Software development process. The most popular schemes are Waterfall, Iteration and Spiral ones.

The Waterfall Scheme means a strictly sequential execution of all development stages without repeating already passed stages and missing any subsequent stage. If a developer uses the Waterfall Scheme the calculating of vector **M** components for all Software project constituents (modules, classes) should be carried out after the completion of coding stage.

The Iteration Scheme using an agile technique divides the Software development process into separate iterations. At the end of each of the iterations a developer is to obtain an intermediate (but not necessarily functional) Software version and at the same time create a new or complete a previously created Software modules. As the amount of faults in a newly created and completed codes may be significantly different it is necessary to distinguish three types of iterations: 1) iteration of newly created Software code; 2) iteration of existing Software code completion; 3) iteration of simultaneous code creation and completion. Naturally for every iteration the relationship coefficient in a model of Software complexity- upon-fault amount dependence is different. That is why a developer is to use the data of the corresponding iterations of the previously created or newly developed project in order to calculate these coefficients. Assumption 4 is automatically performed for iterations of the developed project.

Spiral Scheme assumes at every spiral turn a developer creates a Software fragment or version using either Waterfall or Iteration model. In this case a developer is to combine the techniques of relationship coefficient calculation in the model of Software complexity-upon-fault amount dependence. But the obligatory condition is to calculate different coefficients for different iterations.

In using any Software development model a developer always creates a new Software code that can probably contain some faults. We call it a fault code. But he simultaneously uses the previously developed and already tested Software components as well as the code automatically generated by the medium of development. Such kind of a Software code has been called a faultless code. For the faultless code the probability of the fault presence tends to zero. So we can logically assume that in order to increase the accuracy of estimation of the amount of faults a developer is to calculate metric values just for a fault code.

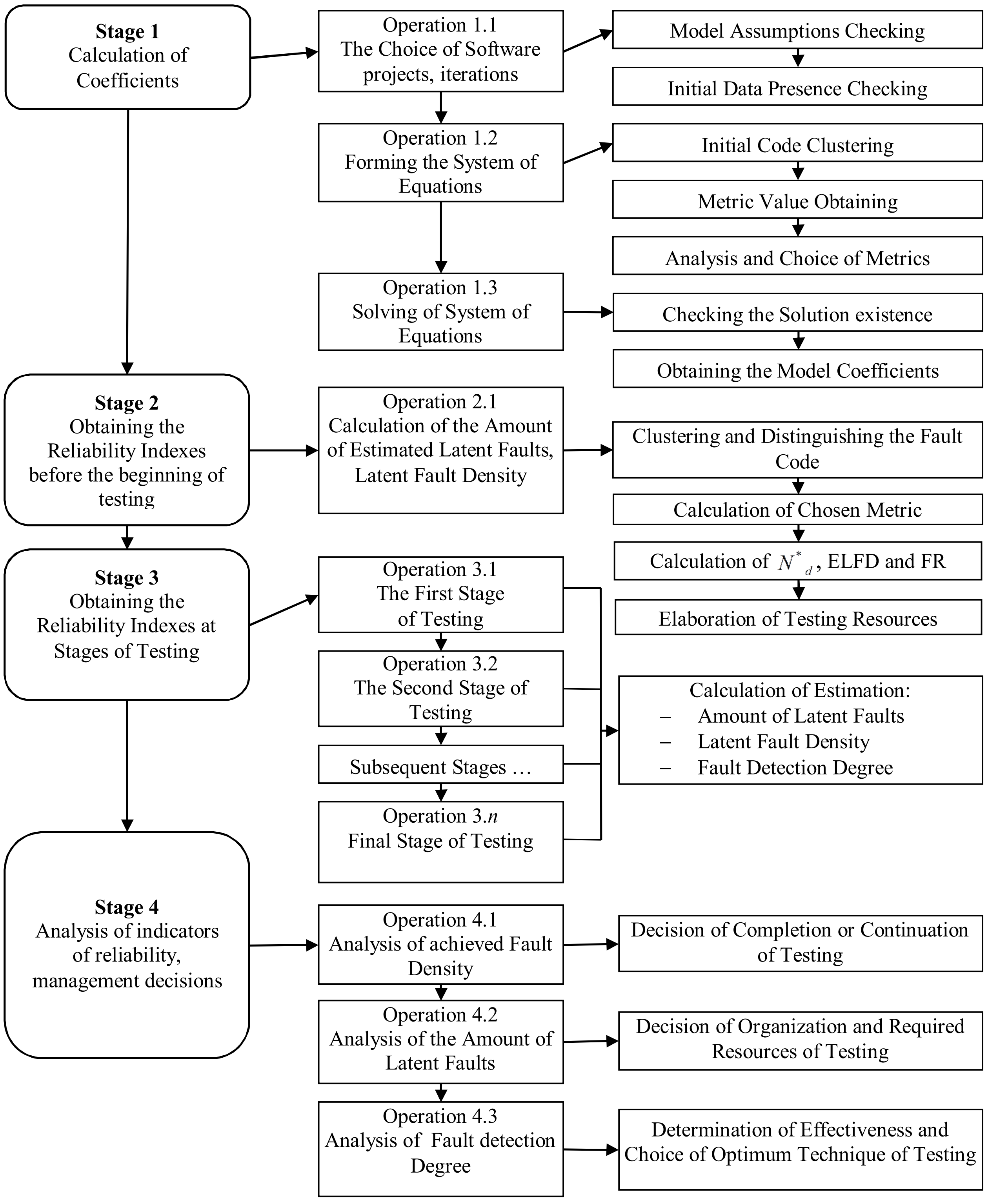
Stage 1
Calculation of Coefficients
Operation 1.1
The Choice of Software projects, iterations
Model Assumptions Checking
Initial Data Presence Checking
Operation 1.2
Forming the System of Equations
Initial Code Clustering
Metric Value Obtaining
Analysis and Choice of Metrics
Operation 1.3
Solving of System of Equations
Checking the Solution existence
Obtaining the Model Coefficients
Stage 2
Obtaining the Reliability Indexes before the beginning of testing
Operation 2.1
Calculation of the Amount of Estimated Latent Faults, Latent Fault Density
Clustering and Distinguishing the Fault Code
Calculation of Chosen Metric
Calculation of N^*_d, ELFD and FR
Elaboration of Testing Resources
Stage 3
Obtaining the Reliability Indexes at Stages of Testing
Operation 3.1
The First Stage of Testing
Operation 3.2
The Second Stage of Testing
Subsequent Stages …
Operation 3.n
Final Stage of Testing
Calculation of Estimation:
– Amount of Latent Faults
– Latent Fault Density
– Fault Detection Degree
Stage 4
Analysis of indicators of reliability, management decisions
Operation 4.1
Analysis of achieved Fault Density
Decision of Completion or Continuation of Testing
Operation 4.2
Analysis of the Amount of Latent Faults
Decision of Organization and Required Resources of Testing
Operation 4.3
Analysis of Fault detection Degree
Determination of Effectiveness and Choice of Optimum Technique of Testing

Figure 1. Method of a priori Software reliability evaluation

To do this a binary sign (fault or faultless) is to be assigned to each of the initial code components. The purpose of the sign can be formulated by developers knowing in detail the created Software component origin.

Thus as a result of the described analysis we have found the possibility to use the model of Software complexity-upon-fault amount dependence for different schemes of the development process. To increase the predicting estimation accuracy of the amount of faults the metric values should be calculated only for faultless code.

3.4. THE A PRIORI METHOD OF RELIABILITY EVALUATION

The essence of the method based on the model of Software complexity-upon-fault amount dependence and using the proposed techniques and approaches is the sequential performing of four stages (Fig.1).

At the **first stage** the model coefficient calculation is carried out. To do this in *Operation 1.1* the choice of the previously developed Software projects or their iterations is carried out taking into account the model tolerances. *Operation 1.2* produces the initial code division into fault and faultless ones in order to calculate metrics only for fault code. Then according to the technique proposed the analysis of correlation dependence of metric values and the amount of faults is carried out. From two to four the most informative metrics are chosen on its basis. Using the chosen metric values and the known amount of faults a system of equations which calculates the model coefficients is formed. *Operation 1.3* performs checking the existence of this system solution. The system solution existence conditions have been researched in the work [Yaremchuk 2013]. If the system has got a solution then the model coefficient calculation is produced. If it has not got a solution the return to Operation 1.1 occurs in order to choose other complexity metrics.

At the **second stage** performed before testing the division of the initial code of the researched Software project into fault and faultless ones is performed in *Operation 2.1*. Then the chosen metric value calculation only for the fault code, the predicted estimation of the amount of faults according to (4) and the estimated latent fault density are carried out. On the basis of the estimations obtained the previously planned resources of testing are defined more exactly.

At the **third stage** during the testing process *Operations 3.1 …3.n* the reliability metric calculation is produced taking into account the amount of detected faults: the amount of undetected faults, estimated latent fault density, degrees of fault detection. The obtained index monitoring allows to manage the process of achievement of the necessary Software reliability degree.

At the **fourth stage** on completion of testing process *Operation 4.1* performs comparing the achieved latent fault density to the required index and the decision about completion or continuation of testing process is made. In *Operation 4.2* on the basis of the predicted amount of the remaining latent faults the scheduling of the required resources for Software maintenance is made. *Operation 4.3* performs the detection fault degree analysis at the stages of testing. The analysis allows to estimate the effectiveness of testing processes of the given Software and choose the optimum testing techniques.

So the proposed method unites in a single procedure:

– the model of Software complexity-upon-fault amount dependence;

– the technique of determination of the optimum complexity metric amount;

– the technique of the choice of metrics which are the most informative for estimation of the amount of faults;

– approaches to metric value calculation which take into account both different Software development models and different fault presence probability in the Software components.

The method does not use experts' estimations and expensive Software. It uses simple algebraic and statistical calculations which are included in the standard set of functions of all electronic tables. All these characteristics make possible the usage of the method by the staff of software companies.

4 CONCLUSIONS

In the given paper a method of a priori Software reliability evaluation is proposed. It allows to obtain:

– estimation of latent fault amount and fault density before the beginning of testing process. These indexes promotes estimation and necessary resource volume attraction to perform the testing process within the scheduled date and development budget;

– estimation of latent fault amount, fault density and fault detection degree at the stages of testing. The analysis of obtained indexes and their dynamics allows to optimize the testing processes to achieve the required reliability within the scheduled date and development budget;

– reliability indexes after the completion of testing. Their analysis allows to estimate the achieved Software reliability, make a decision of testing completion or continuation, estimate the testing process effectiveness, organize the required resources for the Software maintenance.

The given method is used in Software developing by the Software companies in the city of Izmail, Ukraine. The estimated reliability indexes in various Software projects differed not less than 13% on the average from the actual values. The sufficient accuracy of reliability estimation provided timely decisions about the testing completion. Knowing the amount of latent faults allowed to determine the required number of specialists in the Software technical support group and schedule a new Software version issue.

During the testing process of one of the researched projects the estimation of latent fault amount exceeded 38% of the expected value performed on the basis of the previous projects. The reliability indexes were recalculated. In order to specify them a group consisting of the necessary number of testers was organized to achieve the required latent fault density within the scheduled date and development budget. The fault detection degree monitoring at the stages of testing allowed to determine and eliminate the defects of testing. The testing optimization promoted the developed Software improvement.

The practical usage of the method confirmed that it could be successfully embedded in Software Engineering. Along with the amount of faults a developer is to know their sites in modules/classes of Software. That is why the further research will concern the method of obtaining the Software module fault list ranked for the amount of faults.

5 REFERENCES

ISO/IEC 25010:2011. *Systems and Software engineering – Systems and Software Quality Requirements and Evaluations (SQuaRE) – System and Software Quality models*.

ISO/IEC CD 25022 *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of quality in use.*

ISO/IEC CD 25023 *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality.*

ISO/IEC CD 25024 *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of data quality.*

ISO standards. http://www.iso.org-20-11-2013.

ISO/IEC TR 9126-2:2003 *Software engineering – Product quality – Part 2: External metrics.*

ISO/IEC TR 9126-3:2003 *Software Engineering – Product Quality – Part 3: Internal metrics.*

Maevsky D.A., Yaremchuk S.A. (2012) *A priori Estimation of the Amount of Faults in Information System Software*. Radio Electronic and Computer Systems № 4(56) . – Kharkiv : KHAI , p. 73 – 80.

Maevsky D.A., Yaremchuk S.A. *The Estimation of the Amount of Software Faults on the Complexity Metric Basis*. Electrical Engineering and Computer Systems. – № 07(83). – Kiev: Tehnika, p. 113–120.

Yaremchuk S.A. (2013) *The Problems of A priori Reliability Index Estimation of Dependable Critical Purpose Software*. Radio Electronic and Computer Systems. – № 5(64). – Kharkiv: KHAI, p. 414–420.

The PROMISE Repository of empirical software engineering data. http://promisedata.googlecode.com – 20-11-2013.

Statgraphics Statistical Analysis and Data Visualization Software. http://www.statgraphics.com – 20-11-2013.