METHODS OF DETERMINATION OF QUALITY OF SOFTWARE

Developed modern software tool for determining the quality of software (SW) techniques metric analysis. The software allows you to use quality metrics to calculate the corresponding metric and determine the value of the complex index of quality software product. Clarified the quality assessment process, software analyzes the concept of the quality of the software product as an object of standardization and quality levels of performance models of the software. This allowed the opportunity to improve the quality of software by generating the relevant requirements of the criteria for quality evaluation. It is also possible to make the improvement of the metric analysis of models of its quality and its quantitative measurement methods in all phases of a software project.

It was revealed that the driving force behind the success of software projects is the desire of their leaders to develop such software, which would have a certain value. It should be important for certain tasks or to achieve tactical and strategic objectives. The value of the software can be expressed in the form of its value, or in some other form. The customer usually has their own idea of the maximum cost of investing in the development of software. These funds profit it expects to achieve in the case of the main goals of using the software. It can also have a vision of the functionality of software and certain expectations of its quality.

The features of the use of the metric analysis for determining the quality of the software, revealed the lack of uniform standards for the metric. Therefore, each supplier of its measurement system offers its own methods of evaluating the quality of software and associated metrics. Also it is challenging to interpret the evaluation metrics, which for the majority of users of its software metrics and their values are not absolutely clear and informative. It was found that the main parameters of the choice of an embodiment of the software is its cost, the duration of the development process and the reputation of the designer of the company. But the decisions taken on the basis of these parameters, not always guarantee proper quality of the software.

**Keywords:** information technology; software design; quality standards; software requirements; requirements specification; quality assessment criteria; Software quality metrics; a comprehensive indicator of quality.

**Introduction.** The quality of software is the main characteristic of it in various spheres of information technology usage [52], which indicates its degree of compliance with the requirements [27, 33, 45, 60]. The quality of software means actions that determine how the software meets its purpose [46]. Such actions are interpreted in several different ways, which leads to several independent definitions of this term. Mostly, the software quality is understood as a set of properties of a software product that characterizes its ability to meet the given or predicted needs of the customer, which he expressed in the form of user requirements at the initial stages of software development [55, 56].

Standard ISO/IEC 9126 [36] governs the internal and external quality assurance features. External characteristics reflect the requirements for the operation of software and internal characteristics used to prepare plans to achieve the required values as its external characteristics [38, 39]. Quality characteristics reflect the properties of the software as well as customer and developer looks at it [23, 24, 42, 43, 61, 66]. However, direct user software mainly interested in operating its quality [37], that the cumulative effect of achieving the required characteristics of the program, the value of which is measured by the speed and reliability of the results, not its property. This concept is much broader than any single characteristic quality assurance, for example, ease of use or reliability [40].

The quality of software is determined by using the quality model [4, 6, 10, 17, 24, 37, 63, 66], which becomes particularly important along the development and improvement of technology for evaluating the quality of software, namely, the methods of metric analysis. The essence of this technique is the following: based on the software quality indicators, the values of the corresponding quality metrics and values of the integrated quality indicator for the future software product are calculated [15, 65]. According to ISO 24765: 2010 [41], the metric is defined as the property possession degree of a certain product that has a numerical value.

Typically, when evaluating the quality of the software to understand the steps that define how the software meets its purpose [46]. Software Quality assessed using a quality model [37]. This assessment is particularly important to the development and improvement of software quality definition technologies, namely – metric analysis methods. All this has led to the need for developing appropriate means for...
determining the appropriate quality assurance methods.

Analysis of recent research and publications. Today the evaluation of software quality standard ISO 25010:2011 [34] Carried out in the order based on attributes as specified in ISO 25023:2016 [35], subcharacteristics and evaluate quality characteristics which, together, provide a comprehensive assessment of software quality. Evaluation is software quality metric using the results of the analysis comes down to this: on the basis of quality indicators calculated values of the metrics of quality and value of the future complex quality software. According to the ISO 24765:2010 [41], the metric is defined as a measure of the degree of ownership of the property of a product that has a numerical value.

ISO 9126 [36] standard defines quality as "the totality of features and characteristics of a product or service that bears on its ability to satisfy given needs" [2]. The development and improvement of software and quality, respectively, are essentially organizational in nature, not technical [20, 42]. The ISO 9126 model [37] consists of a four-part standard for "Software engineering – product quality," which includes quality model, external metrics, internal metrics, and quality-in-use metrics, respectively [3, 11, 19]. A recently proposed model, McCall's model, describes software quality as the characteristics of a software product that represent its capability to satisfy both explicit and implicit requirements [11, 26, 60]. It proposes six high-level, independent quality measures, namely, Reuse based on Object-oriented Technology, Dromey's Quality Model, Software Assurance Technology Center Quality Model, Qualidade Model for Object-oriented Design, Metric-based Quality Model for Object-oriented Design, and Software Metrics. These measures comprise a set of software features by which the product quality is depicted and appraised [51]. Software quality has also been depicted [12] according to its product characteristics: internal quality (i.e., mode of product development), such as software complexity and configuration; external quality (i.e., product functionality), such as serviceability and reliability [21]. The three most common software quality definitions are as follows:

1) software quality is determined by a set of quality factors [31, 32];
2) software quality is determined by user satisfaction [16];
3) software quality is determined by unexpected software performance or errors [14, 47].

However, the software quality determination procedure and appropriate methods and tools for ensuring this quality [43, 62], as well as the process of software development itself, remain without the fundamental theory and effective methodology provided [7]. Most of the researches in the field of software quality evaluation, especially in the early stages of its life cycle, are chaotic and unsystematic [13]. At the same time, as proved in investigations [54, 55, 56], it is possible to identify and eliminate up to 55% of all the issues of the future software product at the end of the software architecture design stage [9, 64]. Therefore, the problem of software quality evaluation requires meticulous attention to prevent unexpected losses and unpleasant incidents caused by errors in its work [26].

Object of study – metric analysis of software quality.
Subject of study – methods and means of software quality metric analysis, which will give an opportunity to calculate the software quality metrics, using the relevant indicators calculated at the early stages of the project development.

The purpose of the study is to develop an adequate tool to determine the software quality by using methods of metric analysis, which will give an opportunity to calculate relevant metrics with using of quality indicators and determine the integrated quality indicator of the whole software product.

To realize this goal, you must do the following main objectives of the study:

1) find out the features of the quality assessment software to analyze its quality as a subject for standardization and presentation level of software quality model;
2) determine the features using metric analysis for determining the quality of software that will enable them to explain the reasons for inefficient use, and different interpretations of the values of these metrics;
3) develop a software tool for determining the quality of the software, which provides the ability to predict the further development of its process efficiency and create the appropriate set of data to determine the value of complex quality software;
4) make appropriate conclusions and make recommendations for the practical use of means for determining the quality metric analysis software methods.

The scientific novelty of the research – a method for using metric analysis to determine the quality of software was first developed. According to this method systematically available to metric standards, so each measuring system provider must use only approved methods of quality assessment software and the corresponding metrics. Also offered a unified approach to the interpretation of the values of the metrics, which allowed for most users these metrics and their values are equally informative to understand and apply.

The practical significance of the research results obtained is designed as a modern software tool for determining the quality metric analysis software methods allows using quality indicators relevant metric to calculate and determine the value of complex quality software.

Peculiarities of the software quality evaluation process. For any software product, there are many interpretations of its quality. Negotiated software quality indicators may be required to follow by its performers during all stages of software development. There can be only a few of such indicators or they may reflect certain properties of the future software that their immediate users and other interested parties would like to see [5]. Often these indicators can be get as a result of a some compromise. This approach fully coincides with the notion of "acceptable quality of software", which is a less rigid point of view on the quality of software as a guaranteed achievement of its perfection [45, 57].

Modern software development approaches have achieved such perfection that there appeared a need to use engineering methods for evaluating the design results during all stages of the project implementation, monitoring the degree of the planned quality indicators achievement and their metric analysis, determining the potential risk and the degree of finished components usage to reduce the cost of a new project implementation [1, 7, 50, 59]. The basis of engineering methods for evaluating the software quality is the possibility of quality improving by creating appropriate requirements for the quality assessment criteria [12, 45, 60], improving the models of metric analysis of its quality and methods for its quantitative measurement at all stages of the project implementation [53, 54, 57, 58].
The cost to ensure software quality can be differentiated to the cost of defect prevention, the cost of the work effectiveness evaluation, the cost of internal and external failures during the operation of software [44]. The driving force for the successful implementation of software projects is the desire of their executives to develop a software that would have a certain value, that is, it would be significant for solving certain tasks or achieving objectives – tactical and strategic. The value of the software can be expressed in the form of its price or in some other form. Often the customer has his own view on the maximum cost of investing, the return of which he expects in the event of achieving the main objectives during the use of software. He can also have his own vision of the functionality of the software and certain expectations about its quality.

Typically, the customer concentrates on the functionality of the software for the first and does not think about its quality, and about the cost associated with quality assurance. Therefore, at the initial stage of the project implementation, the subject of discussion may be the question about a full understanding by the customer the benefits of using the software and the cost of its development related to the achievement of a particular level of software quality, as well as its involvement in the process of making appropriate decisions. Ideally, most of these decisions need to be discussed at the stage of defining custom software requirements [11, 27, 60], but these questions can (and should) be taken into account during all stages of its development. There are no "standard" rules on how to make such decisions. However, business analysts must clearly demonstrate different alternative ways to achieve a certain level of software quality and the appropriate cost of development so that you can predict the total cost of implementing a software project.

Fig. 1. Information security system quality evaluation process peculiarities

To demonstrate dependency between the cost of software implementation and the quality of software, let's consider the peculiarities of the information security system development, namely its functional model (Fig. 1) [22, 29, 30]. This model does not show the price of information – an object of confidentiality (for example, bank deposit accounts or access codes to them, since this information does not lose its value over time). Consequently, in this figure the following notation is introduced: \( P \) – level (probability) of information security protection (practically, \( 0.6 \leq P < 1.0 \); \( Z(P) \) – allowable expenses for the information protection as a function from the required level of protection. These costs increase with requirements increasing to a certain level of information security protection.

The willing to achieve a very high level of information security usually results in a sharp increase in costs that may exceed the information price that needs to be protected. Possible losses of the information holder \( U(P) \) due to improper level of protection is a function of the existing level of protection \( P \). It is clear from the figure that the sum of \( Z(P) + U(P) \) determines the price of \( V(Z, U) \) to provide information security. At the same time, the optimal level of protection of \( V_{opt}(Z, U) \) will correspond to the minimum value of protection costs \( Z(P) \) plus possible losses \( U(P) \) due to loss of information because of incompleteness of its protection. The desire to exceed it will lead to a sharp increase in the cost of \( Z(P) \) to provide information security; the decrease of the protection level will lead to an increase in possible losses \( U(P) \) due to imperfect functioning of the information security system.

Consequently, the quality of software is a relative concept, the meaning of which customers understand only when taking into account the real conditions of its application, so the requirements to the software quality should correlate with the conditions of its use and the specific area of its application [57]. The quality of software is characterized by the following components: the quality of software development processes, the quality of product, and the quality of maintenance or software deployment. The component related to the software development processes determines the degree of formalization, the reliability of the processes themselves at each stage of its development, and is closely linked with the verification and validation (approval) (briefly – V & V) of the intermediate results of these processes. Finding and elimination of errors in the finished software is made by using testing methods [18, 25], which reduce their quantity and increase the quality of the future software product.

The quality of software is achieved through the use of procedures for controlling of intermediate project results during all stages of software development, testing them on compliance with the required quality criteria, as well as through the use of modern methods and tools for software maintenance. The effect from particular software deployment to a large extent depends on the knowledge of the staff, the functionality of the software product.

The software quality model has four levels of representation [45]. The first level corresponds to the definition of the characteristics (indicators) of the software quality, each of which reflects a separate point of view of the user to its quality. According to available standards [36], the quality model includes six characteristics / indicators of software quality: functionality, reliability, usability, maintainability, efficiency, portability. At the second level, the attributes of the software quality are determined for each specific characteristic that specify its various features. A set of these attributes is then used in the metric analysis of software quality.

The third level is intended to measure quality with using metrics, each of which, according to standard [36], is defined as a combination of methods for measuring the attribute and the scale for setting its values. To determine the quality attributes of software at all stages of its development (when viewing the documentation and project development, as well as the results of their testing), metrics with their given weight are used to level out the results of metric analysis of a particular criterion aggregate attributes or quality of software indicator. The quality attribute is determined with using
one or more software quality techniques at various stages of its development and at the final stage of its testing [67].

At the fourth level, for evaluating the quantitative or qualitative value of the individual software quality attributes, the estimating element of the metric is used – its priority. Depending on the purpose, features of operation and terms of maintenance of the software the most important characteristics and attributes of its quality are chosen. Selected attributes and their priorities are reflected in the requirements for the software development process, or use the relevant priorities of a benchmark of the software class to which it belongs.

**Use metric analysis to determine the quality of the software.** The evaluation of software quality according to ISO 25010:2011 [34] follows the next procedure: based on the quality attributes specified in ISO 25023: 2016 [35], the Subcharacteristics and characteristics of software quality are evaluated which, in turn, provide a comprehensive evaluation. According to the ISO 25010: 2011 [34] standard, the characteristic is defined as a set of software properties that help you to describe and evaluate its quality. Subcharacteristics of software quality are expressed in terms of weighted arithmetic indicators, taking into account the values of attributes that evaluate this sub-characterization, and their coefficients of importance [54].

There are number of models that make it possible to calculate the quality of software, but the multiplicity of their characteristics interpretation complicates such calculations. Models are based on the use of different software metrics [57]. The introduction of quantitative software quality metrics helps to solve some practical problems [49]: 1) predicting the errors number in the software from the beginning of the design; 2) predicting the level of software complexity and its maintenance on the basis of design results analysis; 3) predicting the complexity of the testing processes and the number of undetected errors based on the analysis of the program code; 4) forecasting the final size of the program code based on the analysis of the software architecture design complexity; 5) determining the impact of individual characteristics of the software code on the quality of the finished software; 6) monitoring the stages of program implementation; 7) analysis of found and hidden defects in the finished software; 8) identifying the best methods and technologies for software development based on their comparison.

By the definition of ISO/IEC 9126-2 [38], the software quality metric is “... a model for measuring the attribute associated with the quality of the software. This is a combination of the specific measurement method (the method which is used to obtain values) of the entity attribute and the measurement scale” [7]. According to the standard ISO 24765: 2010 [41], the metric can be defined as a degree of possession of a property that has a numerical value. In general, the software metric is a measure that allows to calculate the numerical value of some software property as a weighted arithmetic, taking into account the metrics values that evaluate this metric, and their weight factors [48].

The evaluation of software quality based on the results of metric analysis is as follows: based on the software quality indicators the values of quality metrics are calculated, which, in turn, provide a comprehensive assessment of the overall software quality. Despite numerous studies of software metrics [53, 54], there still are many unresolved issues. For example, the lack of common standards for metrics (more than a thousand of metrics [7, 68] are created) leads to the fact that each supplier of the measuring system offers its own methods for determining the quality of software and corresponding metrics. It is also a difficult task to interpret metric values, since for most users both metrics and their meanings are not completely clear and informative.

In [56] various software quality metrics were analyzed from the point of their application possibility at the early stages of the project implementation with the obtaining of the exact or predicted value. According to the results of this investigation, several metrics which had the exact meaning at the software architecture designing stage were found, namely: 1) Chapin metric – analyzes the nature of the variables usage from the list of input and processing information; 2) cohesion metrics – internal characteristic of the software module, which depends on the type of module or project; 3) coupling metrics – an external characteristic of the module, which is desirable and worth to reduce. The degree of modules data interdependence; 4) Gilb metric (component of the metric) – the complexity of the program, through which it is possible to calculate the number of intermediate relationships at the design stage; 5) Kafura metric – based on data flows; 7) Metric of reference to global variables – is characterized by a pair (p, r) where p is a module that has access to the global variable r; 8) duration of models modification – the metric of the software development process; 9) the number of found errors during models and prototypes of subsystems, modules, functions, requirements inspecting and error rates.

Analysis of software quality metrics from the point of view of the possibility of their application at the design stage of its architecture to obtain the exact or predicted value allows us to select a set of metrics that have a predictive value at the design stage [57]: 1) expected LOC-assessment (by expert); 2) Halstead's metrics – is calculated based on the analysis of the line number and syntax elements of the program source code; 3) McCabe’s metrics – cyclo-metric complexity; 4) Gilb metric – relative logical complexity of the program; 5) estimated number of program operators; 6) predicted estimation of software components interfaces complexity; 7) estimated total duration of the software development process – the metrics of the software development process; 8) the duration of the designing software stage – the metrics of the software development process; 9) expected cost of software development process; 10) estimated cost of quality assurance – the metrics of the software development process; 11) predicted productivity of the software development process; 12) forecasted expenses for implementation of the code – the metric of the software development process; 13) predicted functional size FP – measures the essence of the possibilities of the future program; 14) estimated labor costs and the duration of the project implementation – based on the Boehm model.

So, for constructing of the intellectual method of software architecture designing results evaluation and forecasting the characteristics of its quality, we selected 9 metrics of the design stage with exact values and 15 metrics of the software design stage with predicted values. Other metrics are derived from the selected basic metrics [7, 53]. The most used sources of information on software complexity are metrics [54, 58]: Ned Chapin, Dave Mcclure, Dennis Kafura, Maurice H. Halstead, Thomas J. McCabe, and Tom Gilb. Regarding the software quality, the metrics of connec-
hensive expected cost of development, predicted cost of quality check, estimated cost of development, expected total duration of software development process, predicted duration of design stage are often used for its estimation. For example, let's consider the Chapin metric and the Kafura metric.

**Chapin metric** analyzes the usage of variables from the list of input-output, that is, the corresponding processed information. There are several Chapin metric modifications. The essence of the simplest in terms of its practical use as a sufficiently effective method is to evaluate the information complexity of a separate program module through the variables functionality from the list of input-output. Since each variable can perform several functions simultaneously, it must be taken into account in each corresponding functional group. Chapin metric can be calculated by the formula [56]:

\[ Is = Qm \cdot (P + 2 \cdot M + 3 \cdot C + 0.5 \cdot T), \]

where: \( Qm \) – number of program modules; \( P \) – variables for calculations and output; \( M \) – modified or created variables in the program; \( C \) – control variables; \( T \) – not used in the program ("parasitic") variables.

**The Kafura metric** is based on the data flow. According to the information given in [49, 53], the complexity of the software can be calculated using the following formula:

\[ I = Qm \cdot (W \cdot R + WrRd \cdot (W + R + WrRd + 1)), \]

where: \( Qm \) – number of program modules; \( W \) – the average number of module procedures updating the data structure; \( R \) – the average number of module procedures that read information from the data structure; \( WrRd \) is the average number of module procedures that read and update the data structure.

The number of the program modules, based on the given formulas of the Chapin and Kafura metric, can be defined as with using of Chapin metric

\[ Qm = \frac{Is}{P + 2 \cdot M + 3 \cdot C + 0.5 \cdot T}, \]

as with using of Kafura metric

\[ Qm = \frac{I}{W \cdot R + WrRd \cdot (W + R + WrRd + 1)}. \]

Comparing them with each other, we get a formula for determining the software information complexity

\[ I = \frac{Is \cdot (W \cdot R + WrRd \cdot (W + R + WrRd + 1))}{P + 2 \cdot M + 3 \cdot C + 0.5 \cdot T}. \]

So Kafura metric can be calculated using Chapin metric (and vice versa), indicating their mutual correlation.

Different metrics reflect specific peculiarities of software quality [26]. For the thorough consideration of these aspects, when evaluating the software quality, not single metric, but their composition is used. If the value of several metrics has been obtained, then each such value should be multiplied by the corresponding weighting factor established by the experts taking into account the dominant quality criteria of the software according to the tasks, features, its functional purpose and properties. After that you need to add all the indicators to get a comprehensive assessment of the quality level of software or the quality of its architecture design process. It is most important to apply quite large sets of quality metrics to software at the design stage of its architecture, and in the subsequent stages of its development, the meaning of these metrics can only be specified.

Thus, the peculiarities of the use of the metric analysis for determining the quality of the software, according to which there is a lack of uniform standards for metric, so each supplier “metering” system offers its own methods of evaluating the quality of software and related metrics. There is a challenging interpretation of the values of the metrics, since most users as the metrics and their values are not informative. Revealed that the main parameters in selecting embodiment of the software is its cost and duration of the development process and the company's reputation, the designer, but the decisions taken on the basis of these parameters, not always guarantee proper quality software.

**Software for the software quality determination.** To determine the software quality by using of metric analysis methods with using of the information obtained from the requirements specification, it is necessary to develop an appropriate software tool that will enable the use of quality indicators to calculate relevant metrics and determine the complex quality indicator of a software product. The corresponding software tool (Figure 2) is intended to determine the quality of software on the basis of metric analysis, that is, using quality metrics with exact and predicted values [8, 28]. The advantages of this software over the known is the ability to perform software analysis based on the received metric values with prediction of further success of the development process. Also, as a result of the corresponding calculations on the output, it is possible to form an appropriate set of data that allows to approximate the results of the metrics, provide a quantitative assessment of the software product quality and the value of the forecast quality characteristics of the developing software.

The software is developed in the Microsoft Visual Studio .NET 2017 development environment. Internet connection is not required for the correct operation of the tool. Execution of the task began with a detailed prototype of the user interface with a gradual increase in the functionality of the software. The interface of the software is shown in Fig. 2.a, the results of the corresponding calculation are shown in Fig. 2.b,c.

After making all the values of the input parameters and performance metrics of their calculation software, collecting the necessary information and build forecast about the quality of the investigational software. Software tool table displays total metric values, provides a graphical representation of the results using bar graphs and pie charts, and provides an overall assessment of the quality of software.

Therefore, a software tool for determining the quality of software by means of metric analysis has been developed, which provides the possibility of forecasting the further effectiveness of its development process, and allows to form the appropriate set of data for determining the value of a complex quality index of a software.

**An example of the software.** To better understand the principle of the software will conduct research to determine the quality and the total projected assessment of success of its development. Analyzing studied software received this initial input values to determine the metrics contained in the Table 1. After making all the values of the input parameters and performance metrics of their calculation software, collecting the necessary information and build forecast about the quality of the investigational software. Software tool table displays total metric values (Fig. 3), provides a graphical representation of the results using bar graphs and pie chart (Fig. 4), and provides an overall assessment of the quality of software.
Thus, developed a software tool for determining the quality metric analysis software methods that provides the ability to predict the further development of its process efficiency and create the appropriate set of data to determine the value of complex quality software. An example of the software and the research that determine the quality of a projected total software and evaluate the success of the process of its development.
Tab. 1. The input for the software

<table>
<thead>
<tr>
<th>No</th>
<th>Parameter name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How many times a module really will have access to a global variable</td>
<td>265</td>
</tr>
<tr>
<td>2</td>
<td>How many times module could access the global variable</td>
<td>348</td>
</tr>
<tr>
<td>3</td>
<td>Number of lines of code</td>
<td>4670</td>
</tr>
<tr>
<td>4</td>
<td>The duration of the project software</td>
<td>126</td>
</tr>
<tr>
<td>5</td>
<td>Some software architecture design phase in the process of development</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Errors Module</td>
<td>108</td>
</tr>
<tr>
<td>7</td>
<td>Number of modules</td>
<td>345</td>
</tr>
<tr>
<td>8</td>
<td>The expected number of lines of source code functions</td>
<td>54, 34, 28, 58, 6</td>
</tr>
<tr>
<td>9</td>
<td>The estimated cost of development of the line features</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Part step verification, validation and testing software during its development</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Some stages of quality checks on product design stages of verification, validation and testing</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>The expected number of lines of source code in a similar function</td>
<td>45, 30, 25, 50, 5</td>
</tr>
<tr>
<td>13</td>
<td>Performance development process similar functions</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>Estimated productivity software development process</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>The number of external inputs functions that have different effects on Duties</td>
<td>5, 11, 6, 5, 34</td>
</tr>
<tr>
<td>16</td>
<td>Number of external output function to significantly different nontrivial algorithms and functionality</td>
<td>8, 56, 7, 12</td>
</tr>
<tr>
<td>17</td>
<td>The number of external requests</td>
<td>3, 3, 10, 2, 4</td>
</tr>
<tr>
<td>18</td>
<td>Internal logical file number or unique logical group of user data</td>
<td>1, 1, 53, 5, 7</td>
</tr>
<tr>
<td>19</td>
<td>The number of external logical file or logical groups of unique user data</td>
<td>4, 1, 1, 8, 2</td>
</tr>
<tr>
<td>20</td>
<td>The level of connectivity</td>
<td>a functional</td>
</tr>
<tr>
<td>21</td>
<td>Clutch type</td>
<td>the content</td>
</tr>
<tr>
<td>22</td>
<td>Number of functions</td>
<td>5</td>
</tr>
</tbody>
</table>

Fig. 3. The results metrics

Fig. 4. Graphical representation of results as a histogram

Conclusions. A modern software tool for determining the quality of software by means of metric analysis is developed, which allows to calculate the corresponding metrics with using of the quality indicators and to determine the value of the software product quality complex index. According to the research results, the following conclusions can be done:

1) The peculiarities of the software quality evaluation process were determined, which gave an opportunity to detect the possibility of its increase by forming appropriate requirements to the quality evaluation criteria, improvement of quality metric analysis models and methods of its quantitative measurement at all stages of the program implementation.

2) The peculiarities of the metric analysis use for determining the software quality, according to which there is a lack of common standards for metrics, is established, therefore, each supplier of the measuring system offers its own methods for assessing the software quality and the corresponding metrics. It is also a difficult task to interpret metric values, since for most users both metrics and their meanings are not completely clear and informative.

3) It is revealed that the main parameters in choosing a software implementation option are its cost and duration of development process and reputation of the designer company, but the decisions made on the basis of these parameters do not always guarantee the proper quality of the software.

4) A software tool for determining the quality of software by means of metric analysis is developed, which provides the possibility of forecasting the further effectiveness of the
process of its development and enables to form a corresponding set of data for determining the value of a software quality complex indicator.

References


onal Electrotechnical Commission, 136 p. (International Standard)
pages/0132582201.pdf
formatika/analiz_ya_kosti_programmnoho_zabepchenyny/42.
cepublishinggroup.com/pdf/10.11648/j.ajasea.20130202.11.pdf.

lity characteristics of software. Bulletin of the Khmelnytsky Natio-
nal University, 2, 165–174. [In Ukrainian].
60. Rahman, W. N. W. Ah., Kamal, A. B., Talha, H., Josiah, B., Adam,
i, Lining, W., & Rosli, N. S. M. (2015, January). Software Quality Assurance for E-commerce Customers Satisfaction in Require-
SOFTWARE-A-CASE-Scarpino-Chicone/c837aaeeaced295baabe6594aadaec6155ecc174f52
ra.com/papers/Vol2_Issue1/BS021452454.pdf
demia.edu/3713846/Different_Software_Quality_Model
65. Srivastava, P. R., Singh, A. P., & Vageesh, K. V. (2010). As-
ved from: http://ijict.ijesce.com/Volumes/Volume-5(3)/ijict-
epaper/ijict20140540177.pdf
plks/194.pdf
Ю. І. Грицюк, Т. О. Муха  
Національний університет "Львівська політехніка", м. Львів, Україна

МЕТОДИ ВИЗНАЧЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розроблено сучасний програмний засіб для визначення якості програмного забезпечення (ПЗ) методами метричного аналізу, що дає змогу за допомогою показників якості розрахувати відповідні метрики і визначити значення комплексного показника якості програмного продукту. З'ясовано особливості процесу оцінювання якості ПЗ, тобто проаналізовано поняття якість програмного продукту як предмет стандартизації, а також рівні подання моделі якості ПЗ, що дало змогу встановити можливість її підвищення шляхом формування відповідних вимог до критеріїв оцінювання якості, вдосконалення моделей метричного аналізу його якості та методів кількісного її вимірювання на всіх етапах реалізації програмного проекту. Встановлено особливості використання метричного аналізу для визначення якості ПЗ, згідно з якими існує відсутність єдиних стандартів на метрики, тому кожен постачальник вимірювальної системи пропонує власні методи оцінювання якості ПЗ і відповідні метрики. Також є складним завданням інтерпретації значень метрик, позаяк для більшості користувачів як метрики, так і їх значення не зовсім є зрозумілими та інформативними. Виявлено, що основними параметрами при виборі варіанту реалізації ПЗ є його вартість та тривалість процесу розроблення й репутація фірми-проектанта, але рішення, прийняте на підставі цих параметрів, не завжди гарантують належну якість ПЗ. Приймаючи до уваги результати аналізу стандартів, моделей та метрик якості ПЗ, а також методів вимірювання показників його якості було розроблено інтелектуальний програмний засіб, який виконує такі основні функції: 1) обчислює за допомогою розрахункових і експертних методів точні або прогнозовані значення метрик ПЗ вже на етапі проектування його архітектури; 2) дає змогу не тільки будувати метрики, але й аналізувати й обробляти результати метричних оцінок, на підставі чого можна надати відповідні рекомендації, висновки і прогнози про якість розробленого ПЗ.

Ключові слова: інформаційні технології; програмний проект; стандарти якості; вимоги до програмного забезпечення; специфікація вимог; критерії оцінювання якості; показники якості програмного забезпечення; комплексний показник якості.