DOI https://doi.org/10.36059/978-966-397-107-0/36-65

SOFTWARE AND METHODOLOGICAL COMPLEX OF SYSTEMS

Kyselov V. B.

1. Standardization of quality systems

Standardization refers to the activity of finding solutions to repetitive tasks in the fields of science, technology and economics, and aimed at achieving the optimum degree of ordering in a particular field.

It is known that algorithms and programming have been evolving as a kind of creative activity, poorly regulated. Industrial methods are based on strict regulation and automation of technological processes. Thus, standardization in the field of programming has become a vital necessity. The first objects of standardization have been programming languages and program documentation. Within the framework of the Unified Programming Documentation System (UPDS), about thirty standards regulating the development of program documentation are developed and standardized. Standardization is one of the most effective ways of ensuring the required level of software quality. In the software QMS of the organization-developer (enterprise) complex of enterprise standards (CES) occupies an important place. To create such a complex it is necessary to establish objects and methods of standardization.

Practice shows that the objects of standardization in the software QMS can be: programming technology, software and hardware debugging and testing programs, technological processes (design, coding, debugging, compiling, testing, documentation, support), typical algorithms and programs, quality control organization, inter-module interface, etc.

The main methods of standardization of SQMS in developer organization are systematization and classification; typing and unification; regulation. Systematization and classification are aimed at ordering control elements, establishing their rights and responsibilities, as well as the interaction between them. Typing and unification are aimed at identifying and forming similar program components and program complexes by the organization's profile, creating libraries of unified components, tools for generating applications from these

components, interface agreements. The regulation is aimed at ordering the organizational and technological procedures to ensure the required level of quality at all stages of the software life cycle.The need for enterprise standards is due to the following. State and industry standards, as a rule, contain requirements for the quality level of the final product, its consumer attributes. But to ensure this level it is necessary to specify the quality features of products in the stages of its development, the only requirements for the design of algorithmic and software modules, the only requirements for the interface between them, etc. according to the specific characteristics of products and the specifics of the enterprise. In other words, by means of enterprise standards, the requirements of state and industry standards are interpreted in terms of the conditions of a particular enterprise and are brought to attention of every contractor of the project.

When creating the regulatory and technical base of the SQMS, both the software and its development specifics should be taken into account. The work of programmers has been a highly intellectual activity. The productivity and product quality of each developer fluctuate in a wide range. The individual qualities of each developer and his/her character traits play a big role. Individualism is traditionally inherent in programming, therefore, at the initial stage of creation of the SQMS, at the stage of its testing, most regulatory and methodological documents should be given a recommendation only. Excessive regulation of all aspects of ST developers' activities in the absence of proper conditions can cause a negative effect instead of the expected positive one.

Five international ISO standards have been approved to set requirements for enterprise quality assurance systems: «Standards for quality management and quality assurance. Selection and Application Guide «(ISO 9000);» Quality System. Quality assurance models for design, development, production, installation and maintenance «(ISO 9001);» Quality system. Models of quality assurance in production and installation «(ISO 9002);» Quality system. Models of quality assurance in the process of control and testing of finished products «(ISO 9003);» Quality management and elements of the quality system. Main directions «(ISO 9004).

2. Choosing a Quality Indicators Nomenclature

The choice of a Quality Indicators Nomenclature of software products is to establish a list of names of characteristics of products attributes, which determine the quality of this type of products and provide the opportunity for a complete and reliable assessment of its quality level. The choice of a a Quality Indicators Nomenclature for a particular ST depends on the type (group) of ST, the purpose of the application and the stage of determining the indicators.

For each type (group), and sometimes specific ST, they establish their a Quality Indicators Nomenclature, which takes into account the specific purpose and conditions of use. The a Quality Indicators Nomenclature for each subclass, group and type of ST is drawn up in the form of tables of use of quality indicators. In addition to the list of recommended and mandatory quality indicators for this subclass (type, group) of ST, the coefficients (parameters) of the weights (significance) of each of the indicators should be indicated in the tables of usability. Determining the weighting of coefficients of quality indicators is the most significant and difficult task of choosing a a Quality Indicators Nomenclature. In solving this problem, one can use either the method of value-regression equations, or the method of limit nominal values. But their use is complicated by the lack of the necessary initial data. Therefore, in practice, the most common method is the expert method of determining the weighting coefficients. Usability tables are a guide or reference material for choosing a working a Quality Indicators Nomenclature for specific ST. The working nomenclature of the ST is established taking into account the purpose and conditions of ST use; results of analysis of requirements of the user (customer); quality management tasks; composition, structure and specifics of the attributes that are characterized. The goals of application of the Quality Indicators Nomenclature are set in accordance with the tasks of software quality management. Such goals may include, in particular, the following: setting up a technical specification for ST development; setting up technical specifications for the ST; filling in the technical level map; establishment of controlled indicators in ST design; establishment of controlled indicators in the experimental operation of the ST; certification of ST by quality categories. The stages of determining the quality metrics correspond to the stages of the software life cycle.

While distinguishing attributes and relevant ST quality indicators, the following basic principles must be followed: the distinguishing of groups of attributes should be performed on clear, specific features; attributes belonging to one group, as a rule, must be mutually exclusive and independent.

If the attributes are dependent on each other, then the methods for determining the quality indicators should give clear instructions to exclude multiple effects of the same attribute on the generalized evaluation of the ST quality; every initial Quality Indicators Nomenclature must be open, i.e it must allow the inclusion or exclusion of individual elements: for each of the selected attributes there must be an opportunity to express them in the scales «better – worse», «more – less»; the group should include the attributes necessary and sufficient to determine the corresponding complex (group) attribute; the formulation of the attributes must be clear; the set of attributes that characterize the quality of the evaluated ST should be ordered according to a certain rule in the form of a multilevel hierarchical structure – a tree of attributes; the attributes tree should reflect all the main features of ST usage and operation; the selected Quality Indicators should be correlated with the ST attributes respectively.

This means that a clear correspondence must be established between each of the distinguished attributes and the indicators that characterize it. Establishing such compliance allows to use the software quality indicators tree instead of the attributes tree. The quality indicators that characterize the ST attributes should help to ensure that the ST quality meets the requirements of their users and take into account the current achievements of science and technology. It is often necessary to carry out specific studies to perform this principle, since in general there may be significant contradictions between quality indicators, and the improvement of one indicator may lead to the deterioration of another. To test the performance of the selected system of quality indicators, it is necessary to establish a measure of correlation of each given indicator with the ST quality, the usefulness of the indicator, the possibility of quantitative presentation, and the automatic evaluation of the indicator[1].

[1] Feldbaum A.A., Butkovsky A.G. Methods of the theory of automatic control, Main editorial office of physical and mathematical literature "Nauka", Moscow: 1971, 744 p.

In particular, it is recommended to evaluate the usefulness of each of the selected indicators for specific ST by the following scale:

5 – it is extremely important that this indicator to be of high score;

4 – it is important that this indicator be of high score;

3 – it is good that the score of this indicator is high;

2 – to some extent it is useful to have a high score of this indicator;

1 – at a low score of this indicator there is no significant loss.

About 50% of individual indicators can be determined automatically by a computer, 25% by a comparator. Thus, 75% of indicators can be formalized. An estimate of 20% of indicators can only be performed by a qualified professional. Most indicators are set by static analysis of programs and only about 5% are set in the process of dynamic testing.

3. Quality Indicators Groups

Quality indicators nomenclatures always have a hierarchical structure. Their formation begins with the selection of groups of the upper level of the hierarchy, and then the nomenclature is detailed until single indicators are obtained.

Distinguishing the quality indicators groups is an important and complex task of forming a Quality indicators nomenclatures. Failure to complete groups can complicate the relationships between groups and individual indicators and make the Quality indicators nomenclature less constructive.

To evaluate the quality of industrial products they use the following indicators: purpose; economic use of raw materials, fuel, energy; reliability; ergonomics; aesthetics; adaptability; patent-law; unification and standardization; environmental friendliness; security.

All of these indicators can also be used to evaluate software quality. However, due to the software peculiarities, it is impractical to use some groups of indicators when evaluating its quality.

Such indicators include indicators of *aesthetics, environmental friendliness, safety.*

Aesthetic indicators are uncharacteristic for software due to the almost complete absence of organoleptic properties in the software production. At the same time, it is impossible to deny the presence of ST attributes that are close in nature to the aesthetic indicators (attributes). These are

attributes such as information expressiveness and the integrity of the ST structure depicted, for example, as a graphical scheme.

Indicators characterizing such attributes should be considered in the group of structural (constructive) indicators.

Environmental Indicators and *Safety Indicators* are also uncharacteristic for software because software products can not directly have harmful effects on the environment or on human health. Such actions are possible in cases where the ST is used as the managing elements of the objects, for example in ACS. In this case, designed computers, with a certain algorithm of the control action, can cause adverse environmental consequences, and be dangerous to humans. But this is already indirect action through regulators and enforcement mechanisms of automated technological complexes (ATC). These are taken into account as corresponding ATC Quality Indicators.

Patent-law indicators of software products cannot be used until the issues of patent-law protection of these products are resolved in the legislative (legal) aspect. The nature of the reliability of software and hardware is different.

For software products, such indicators of reliability as durability, storage, maintainability are not very meaningful. The sources of low ST reliability are mainly software bugs made at the design stage and not detected during debugging and testing. In the analysis of some software attributes, which are manifested in their functioning, we have to use

Therefore, in the quality indicators nomenclature of software it is advisable to distinguish the indicators characterizing the software attributes, which are close in their external manifestations to the equipment reliability indicators, in a separate group.

This group is called the reliability functioning proof. Thus, in the basic quality indicators nomenclature of software at the top level we distinguish the following indicators: purpose, reliability of operation, ergonomics, adaptability, unification and standardization. The quality of software is mainly formed in the process of product creation and largely depends on the effectiveness of structural (constructive) decisions. Therefore, at the same level, we distinguish structural indicators into a separate group. Indicators of purpose, reliability of operation, ergonomics and adaptability characterize the attributes of software, which are manifested in the process of their use (operation). On this basis, they can

be considered operational. Structural indicators and indicators of unification and standardization characterize the ST attributes of the structure (construction), they can be combined into one group of constructive indicators. In relation to a group of performance indicators, this group is of auxiliary character. Achieving a certain level of score of these indicators can not be an aim itself, it is only a means of providing the necessary score of one or more indicators belonging to the main group – the group of performance indicators.

4. Purpose indicators

Purpose indicators characterize the ST attributes to perform certain functions that meet their purpose in a given environment. The indicators that belong to this group answer two main questions: in what computing environment (technical, software, and information) this ST works and what functions performs.

The purpose indicators group includes the following subgroups: classification indicators, functional indicators, input area, output area, information security indicators, performance indicators.

Classification indicators characterize the ST affiliation to a particular classification group as well as the operating environment (computing environment). Belonging to a particular classification group is determined by a general classifier (class 50). Classification grouping can be refined by industry classifiers of software. Knowing the classification group to which the evaluated ST belongs, it is possible to establish special requirements common to this type of software. ST classification in the general classifier is carried out by the purpose. But when comparing the ST quality level, besides the purpose, it is necessary to consider the type of ST and the level of programs complexity. When comparing ST characteristics, when selecting basic samples for comparison, samples belonging to the same class by the corresponding feature should be used. It is recommended to divide the software complexity criteria into two broad groups: the complexity of design of the programs (software systems and subsystems) and the preparation of tasks to be solved (static complexity); the complexity of programs functioning and getting results (dynamic complexity).

The group of parameters that affect static complexity include: the size of the system, expressed by the number of commands or the number of

software modules in the system; the number of variables being processed or the amount of memory to accommodate the database; labor costs for system development; duration of development; the number of specialists involved in creating the system. Depending on the value of these parameters, we can distinguish the following levels of complexity of software systems: simple, medium complexity, complex, super complicated, unique. Dynamic complexity characterizes software systems at the stage of operation as complete functioning products. This indicator combines the following concepts: the computational complexity of the software system, the complexity of preparing data and analysis of the results of calculations. Computational complexity determines the resources of the computing system that are required to obtain a set of completed results. This group indicator may be characterized by the following indicators: the time of solving problems on the computer; the amount of memory required to accommodate the ST; data carriers' capacity used for accumulating and storing information when executing the program.

The characteristic of complexity of data preparation and performance analysis is taken into account in the group of ergonomic indicators. ST complexity indicators do not nearly reflect the consumer attributes of the ST. The ST user is somewhat indifferent to the complexity of the software he/she needs. It is important that it performs its functions reliably and is easy to operate. But the development, testing, manufacturing, implementation and maintenance of complex ST are significantly different from the same processes of simple ST.

Accordingly, requirements for indicators such as the level of infallibility, reliability, adaptability, etc., may differ. For example, for simple ST, such indicators as adaptability and supportability are of little importance. ST complexity Particularly impacts the organization of program development, including debugging and testing. The study of complexity, the assessment of the complexity of programs is also of interest for predicting the number of errors and is taken into account in the analysis of the work results in the group of ergonomic indicators.

The following factors are analyzed to predict the number of errors: logical complexity, measured by the number of logical operators; the complexity of the relationship, measured by the number of applications and system programs that are called while the program is running; the complexity of calculations, measured by the number of appropriation

operators containing arithmetic operations; the complexity of the I/O process, measured by the number of I/O operations; easiness to read, measured by the number of comments.

Functional indicators characterize the ability to perform certain functions from the potential variety of functions specific to this type of ST and useful in terms of ST users. The essence of these indicators is as follows. Two software environments of the same purpose may differ substantially from one another in functionality with other indicators being equal or similar.

When considering functional indicators, one should take into account their ambiguous dependence on other indicators. For example, the implementation of additional functions in ST usually requires additional costs of resources (labor and material, including computer resources), complicates the structure of ST, which can lead to a decrease in the ST reliability and the like. Therefore, it may sometimes be the case that an increase in the number of functions implemented in the ST will not lead to an improvement in the ST quality. This contradiction can be easily eliminated for a specific ST, if its scope is clearly defined, as well as the functions (tasks) performed and the weighting parameters of these functions.

While comparative quality assessment by these indicators, it is impossible to compare the ST belonging to different classes. It is not possible, for example, to compare SuperComputer operating systems with MicroComputer operating systems in terms of their functionalities. Coefficient of completeness of the functions implemented in the program and average arithmetic indicator of completeness of the implemented functions can be taken as the only functional indicators. The input area is characterized by a range of acceptable input rates that can be converted to the correct result. The attribute of its mass must be one of the mandatory attributes of any algorithm. This means that theoretically the rates of the variables (input data) used in the algorithm can be arbitrary. In fact, when designing a particular algorithm, and especially in its software implementation, restrictions on the permissible range of changing the rates of the variables are introduced. These restrictions are due to objective conditions (limitations on the amount of memory allocated for this program; limitation of the computer's bit rate, rules for measuring the rates of variables, etc.), as well as subjective decisions made by program

developers. Limitations lead to the fact that two programs with the same purpose may differ significantly from one another in the ranges of acceptable values of the input data.

The input area is characterized by a range of acceptable input values that can be converted to the correct result. It is natural to assume that users have a more acceptable version of the ST that has a wider range of input data changes (with other identical indicators). The range of acceptable values of the input data can be characterized by the following separate indicators: the allowed range of change of input data elements; permissible error of input data elements; valid input format; admissible speed of change of input data values; possibility of selective use of details, maximum number of simultaneously processed objects; adaptability to changing input formats, etc. Information protection can be implemented either centrally, in a scale of a particular computing environment, or autonomously in every ST that needs information protection. In this case, the ability to protect information from unauthorized access will be an attribute of specific ST. Security requirements are imposed only if the information really needs protection. Performance indicators characterize the ST's ability to perform, under the given conditions, a certain number of data processing functions (including the same type) per unit of operation time. Average performance can be taken as an elementary characteristic of productivity.

5. Performance reliability Indicators

Performance reliability indicators characterize the ST attributes which are manifested in the direct data processing on the computer and that affect the quality of the results of processing[2].

This group includes the following subgroups of indicators: *accuracy, resistance to distortion, reactivity, infallibility and reproducibility.*

Accuracy indicators characterize the closeness of data processing results to their true, specified, or theoretically correct values. The ST accuracy requirements in this interpretation should be applied to each ST, as each ST provides a certain result of data transformation, and the closeness of this result to the true values is indifferent for users. But the software is extremely diverse.

[2] Krainnikov A.V., Kurdikov V.A., Lebedev A.N. and others; Probabilistic methods in computer engineering: Textbook manual for universities on spec. Computer. Ed. A.N. Lebedeva, E.A. Chernyavsky. Moscow: Higher school, 1986. 316 p.: pic.

This diversity gives rise to a variety of unitary precision indicators (criteria). For computational programs, the following traditional indicators can be taken as unitary: an absolute error in the computational value; relative error of computation; maximum value of the relative error of computation; average value of the error of computation; mean square deviation of calculation error.

Resistance to distortion Indicators characterize the ability of ST to reduce the negative effects of a distorting actions of environment on the data conversion process.

Resistance requirements are imposed on all real-time ST of automated systems, as well as on those whose continuous operation time exceeds the average time interval between failures (uptime interval) of the computer on which this ST is implemented.

The data transformation process and the quality of the transformation results are significantly affected by various distortions from the computing environment.

In relation to the ST and the computer on which it is implemented, these actions can be both external and internal. In this case, external actions mean actions that lead to distortion of input data; internal ones lead to distortion of program codes, intermediate and final results of calculations, databases, as well as violation of functional connections between program components. The source of external actions is the external (in relation to the computer) environment. These actions are caused by failures and interruptions in the operation of information sensors, communication channels and data transfer devices; errors of computer operators, etc. The sources of internal actions are the computer equipment used in the operation of the ST. These actions are caused by interruptions, partial and complete failures of these devices. The sources of distortive actions are independent of algorithms and programs.

But the degree of suppression of the effects of these actions in automatic mode depends only on them. In the general case, software may either reduce or intensify the effects of distortive actions.

The specific actions that need to be taken in a particular situation are determined by the content of the software. Sometimes individual occasional interruptions lead to grave consequences, nullifying the results of long and difficult work. At the same time, in some cases it is possible to achieve positive results under the same conditions due to the fact that the

program provides special modules for eliminating the effects of distortive actions. Operating systems, that application programs run within, typically help solve this problem by logging crashes, interrupts, and the like. Therefore, the degree of ST resistance to distortive actions in a given operating environment is a specific characteristic of each ST. A software tool is considered to be resistant to distortion if it retains performance during the specified period of operation and provides for the transformation of any set of input signals (from a given set) into an acceptable set of output signals. In other words, a persistent program is a program that continues to remain operational, despite hardware outages and operator errors.

To quantify software counteraction to distortive actions, one can use such a criterion as the area of sustainable operation, which is understood to be an area of input and disturbance in which the functional parameter (error of the data conversion results) is not beyond the design tolerance and the ST provides a sustainable process of development output data (results).

This criterion is difficult to obtain by analytical calculations, it can be found through statistical modeling. Thus, you can set the resistance to distortion indicator.

Reactivity indicators characterize the ability of software to convert input (requests) to the desired result on time.

Reactivity indicators are of particular importance in real-time systems, in which the delay of these data leads to their depreciation and can cause complete disability of the systems.

ST reactivity indicator is not a constant. It depends on the path in which the information was transformed in this implementation, and this path is determined by the totality of the transformed data, which is generally formed randomly from data belonging to a finite set. Therefore, individual ST reactivity indicators are statistical.

The term «*reliability*» is borrowed from technology. Reliability is the ability of an object to perform the task of a function, preserving over time the values of the installed performance indicators within the necessary limits, corresponding to the specified modes and conditions of use, maintenance, repair, storage and transportation.

To quantify the reliability of the product they use indicators that take into account the specificity of a particular product. But, regardless of the specifics, at the heart of these indicators there is the assumption that at a

particular point in time, any product can be found in one of two possible states: valid or invalid. Valid condition of the product is the condition in which it is able to perform the functions assigned to it with the parameters set by the technical requirements (conditions). In the process of operation, the transition from the valid state to the invalid and vice versa is possible.

Rejection is an event that involves the lose of validity, renewal – an event that involves the transition from an invalid state to a valid one as a result of eliminating the reasons of the failure.

Recovery can be done either automatically or manually. There are persistent, self-eliminating and alternating failures. Self-correcting failures are usually called interruptions.

The reliability of equipment in technical systems and systems in general is mainly determined by the reliability of the components, as well as structural and functional features. The following are the main causes of equipment failures: design errors; production defects; deterioration of parameters due to the wearing out and aging. Design errors are difficult to predict. They are individual in nature, depend on the qualifications of the designers, the complexity of the equipment and the presence (lack of) experience of creating similar equipment. Every detail and component product can have manufacturing defects from the very beginning (poor soldering, improper wiring, errors in parts fastening, poor insulation, etc.). The causes of deterioration of the product parameters during operation are such physical phenomena as friction, overheating, oxidation, radiation, etc.

As initial we accept the following prerequisites. Reliability in technology in the traditional sense is characterized by four indicators: reliability, durability, maintainability and safety. Reliability of software products is significantly different from the reliability of the equipment. Magnetic data carriers (magnetic tapes, disks, drums, etc.) have high reliability. The records made on them can be stored for a long time without being destroyed. Program records on punch cards and punch tapes can also be stored for a long time if the necessary conditions are provided. In addition, the production of a new copy (making a copy) in advance is a simple operation that is practically accessible to every user. Therefore, the factor of destruction and aging of data carriers does not significantly affect the reliability of the ST. Some manufacturing defects (errors in data entry, punch card filling; errors in records and rewrites) are only in the original software sample and can be corrected during debugging and testing. Errors

resulting from batch production, copying of systems to magnetic and other data carriers, are relatively rare, are quickly identified and are not significant. The information part of the programs, the data itself (program codes) are neither aging nor wearable. This can only be a matter of moral aging. Thus, neither manufacturing defects nor wearing out and aging practically affect the ST's reliability. Only some similarity of durability and storage features can be detected in ST. Therefore, we exclude these attributes from further consideration. The ST reliability depends to a large extent on the number of errors made and eliminated during the development of the ST prototype. In batch production of homogeneous ST, these errors are copied along with other program text. Errors are detected and eliminated during operation. If bug fixes do not make new ones or make less than fixes, then the reliability of the software is continuously increased during operation. The more intensively the ST is used (especially in different conditions and in different organizations), the more errors are detected and the reliability of the ST is growing faster. This pattern is widely confirmed in practice. It manifests a fundamental difference between the reliability of the ST and the reliability of the equipment. Software may lose its functionality when operating or storing. This can be caused by errors that remain undetected in the program, defects in its maintenance, storage or use, or data corruption. Making defects turns out to be a quite rare and easily controlled event. Therefore, this factor will not be considered here.

ST functioning reliability is a function of the errors that remain in it after commissioning. Non-buggy ST is absolutely reliable. But for complex and large ST, absolute reliability is almost impossible. Errors that remain undetected manifest themselves under certain conditions of use (a certain set of initial data).

By the nature of the consequences we should distinguish the following two groups of errors: 1) errors, data transformations that affect accuracy but do not lead to ST failure; 2) errors that cause ST failures.

The errors of the first group can be significant and insignificant. A characteristic feature of significant errors is their negative impact on the results of the data processing, they can lead to software failure under certain unfavorable operating conditions. The signs of failure (disability) of the ST should be specified in the regulatory technical documentation for a certain type of software. All errors of the second group should be

considered as gross mistakes. In assessing their impact on the ST effectiveness they use such statistical characteristics as the probability of failure-free operation, the probability of failure, the frequency of failures, etc.

Given the decisive influence of errors that remain undetected on the reliability of the software, it is advisable to introduce an error indicator that characterizes the attribute of the ST to contain undetected errors that occur under certain conditions of operation. If the software lost its efficiency, then the user (the operator) is tasked with restoring it. In the simple case, this task is solved by overwriting the program and restarting it. But such a restart will be futile if, in the process of data conversion, there will be a need to use a defective program element, that is, a program element that contains a gross error. In this case, you need to find and fix the error to restore performance.

The operation of restoring the performance of complex software systems is a complex operation and requires some automation. Adaptation of ST to the restoration of performance is called reproducibility.

Reproducibility Indicators characterize the adaptation of ST to the rapid transition from a invalid state to a valid one in a process of its intended use.

If T = $\{m_i\}$ – set of indicators of certain accuracy; Y =$\{y_i\}$ – set of indicators of stability; P = $\{p_i\}$ – set of reactivity indicators; O = $\{O_i\}$ – set of indicators of infallibility; B = $\{b_i\}$ set of reproducibility indicators, then the group indicator of reliability can be expressed as follows:

$$NF = F (T, Y, P, O, B).$$

With some assumptions, we can assume that software failure occurs because of low levels of T, B, P, O, B indicators. Cases of manifestation of low accuracy can be attributed to the category of errors. efficiency of software functioning. Group NF indicator allows to take into account the total impact of accuracy, stability, infallibility and update on the effectiveness of the software.

6. Ergonomic indicators

Ergonomic indicators characterize the adaptability of the software to ensure optimal operating conditions for users during its operation.

This indicator also describes the convenience of controlling and maintaining (accessibility) of the ST, that is, a measure of how the

software contributes to the selected mode of use or maintenance of its components. The following indicators can be included in this group. Indicators of ease of ST preparation for work characterize the ST suitability for preparation for work, start and qualification of service personnel. This indicator is especially important when machine time is spent preparing for work and the cost of this time, especially in large computers, is still high. For the user, the most appropriate for this indicator is a software in which all operations to prepare for the job can be performed by one full-time operator, no special training of operators is required. Otherwise, large unproductive expenditures of computer resources are possible. Indicators such as the ratio of the number of events of the data conversion process displayed in a human-readable form to the total number of such events can be taken as single indicators of this subgroup; conformity of methods and means of reflection to the psychological capabilities of the person, etc.

As a comparative assessment of the quality indicators of several similar types of software is carried out, both quantitative and qualitative indicators may be useful. The indicators of the analysis characterize the adaptation of the ST to the prompt and deep analysis of the results of its work. At the end of the data conversion process, there is a need, especially in management systems, to use the results immediately, at least for preliminary analysis[3].

If the software developer has anticipated such a need in advance, then the user will be given the appropriate opportunity to quickly analyze the results. Otherwise, the user will have to spend a lot of time (including machine time) to search for the information that interests him/her.

Diagnostic indicators characterize the adaptability to ST status establishement, localization and troubleshooting, generation of failure messages. An example of a ST single indicator may be the average time of localization of the problem.

7. Indicators of manufacturability

Adaptability indicators characterize the attributes of the structure and documentation of the ST, which determines its adaptability to achieve optimal costs in the manufacturing, implementation (development),

[3] Miroshnik I.V. Automatic control theory. Linear systems. St. Petersburg: Peter, 2005. 336 p.: pic. (Training Series).

operation, modernization, adaptation to the user environment and maintenance for specified values of quality indicators, volume of supply (implementation) and conditions of performance of works. This group includes the following subgroups of indicators: manufacturability; adaptability of implementation; adaptability of support; modification; adaptation (mobility) and rational use of computing environment resources.

Manufacturing adaptability characterizes the fit of the sample-standard to the production of copies on the specified data carriers and documentation for further distribution and use at optimal use of resources.

This figure is essential for batch-produced ST. The weight of the indicator is in direct proportion to the number of software produced. It should be borne in mind that the production of new ST by making a copy from the sample-standard is the most common, but not the only way to obtain a new copy of the ST. Sample-standard copy-making operations can sometimes be preceded by a ST build operation from a specific set of custom components, or a ST build operation from some distribution system. In addition, copying can be transformed into, for example, complex technological operations such as mounting a program in a long-term storage device; making a chip that implements a program, etc. Developers should take care in advance of the adaptability of this method of ST production. The following indicators can be taken as single indicators of the adaptability of ST production: total labor costs for software production; the number of computer resources required to produce a single copy of the ST; coefficient of automation of manufacture, etc. The amount of computer resources required for the manufacture of one ST copy can be determined by the total employment of the computer or its devices in the manufacture.

Adaptability of implementation characterizes the adaptability of the software to launch at its destination (customer organization or user) at optimal cost of resources. In difficult cases, the software vendor assumes the adaptive maintenance function, which is performed to ensure that the ST can be used in a changed operating environment. The following can also be taken as indicators of this subgroup: total labor costs for implementation in machine hours; average time of ST exploration by the user support staff; the level of automation of implementation operations; availability of training courses for staff (programmed training courses are

meant), etc. The adaptability of support characterizes the adaptability of the ST to perform the support functions over it at optimal costs. Support is the most important stage in ST life. Tasks, problems, methods of support were considered earlier. The evolution of ST does not end with the creation of both a prototype and a sample – standard of this ST. Changes in the configuration of the computer system, refinement and change of requirements of customers (users), finding of previously undetected errors, changes of the task and management methods necessitate changes to the ST. Since the execution of this procedure is accessible to every user, usually after a certain time of operation, numerous versions of the same ST appear. The costs of time, labor and material resources to support the ST are significant and make up 50...70% of the total costs of securement of all stages of the ST life cycle.

These costs can be reduced by providing (at the design stage) a certain level of adaptability of the ST support. The solution of many problems that arise during the maintenance phase can be facilitated by the early (starting from the moment of the giving the TOR for development) creation of an automated software database. The database is maintained throughout the ST life cycle. It records the requirements of the customer (both satisfied and dissatisfied); general information on debugging and testing software; information about found and corrected bugs, testing tools, ST upgrades; operational quality indicators, etc.

Indicators of modified software characterize the adaptation of the ST to corrections, changes and additions both in the text of the program and in the text of the documentation. Indicators of adapted ST characterize the suitability of the software to be used in a technical, software, information and production environment of a different type than the one for which it was directly developed. This subset of indicators is essentially similar to the subset of the indicators of adaptability of implementation, but characterizes adaptability to use in an environment other than the one for which it was intended. In essence, this is about so-called re-implementation. Of course, some ST setup is required for re-implementation. The cost of this setup depends on the adaptive attributes of the ST for use in the new environment.

Indicators of rational use of resources of the computing environment characterize the ability of the software to perform the specified functions with minimal cost of resources. The main resources of the computer are the

performance of the processor and the amount of memory. Computer resources also include external devices, communication channels, media (including paper for printing devices), and the like. A software tool that has a high value of resource efficiency can reduce operating costs. This indicator is of particular importance for commonly used ST (operating systems, translators, database management systems, ACMS software, etc.).

8. Constructive indicators

Design indicators characterize the perfection of the methods of decomposition, interface tools, information expressiveness and rationality of the structure of the software. Constructive indicators, unlike all other groups of indicators, have little reflection on the consumer attributes of the ST. For a user who interacts with the software as with a «black box», to some extent, the micro- and even macrostructure of this «box» is irrelevant. But constructive indicators significantly affect almost all groups of indicators, so when evaluating the scientific and technological level and quality of ST should not be neglected. The group of constructive indicators includes the following subgroups: structured, completed, coherent, documented. The structure indicators characterize the perfection of methods used by decomposition and organization of interaction between the elements of the ST, facilitating the labor costs savings at all stages of the life cycle of the ST. A well-structured program is a program with a distinct modular structure, while encoding which structural programming methods were consistently used. As a single indicator, you can use the structure factor.

$$K_{cmp} = \frac{m_{cmp}}{m}$$

where m_{page} – the number of components of the software, the encoding of which strictly followed the methods of structural programming; m is the total number of components in the ST. Completeness indicators characterize the absence or presence of unresolved at the design stage problems of ST. In the best case, there should be no such problems in the completed and tested ST. But in practice, the Admission Commission often draw conclusions about the ST suitability for industrial exploitation, while determining the need for refinement. Such solutions may in some cases prove to be economically justified. At the same time, the presence of unresolved problems is a

disadvantage of this ST. The number of unresolved problems at the design stage is an indicator of the quality of the ST. Consideration can be given to the coefficients of significance of these problems. Consistency indicators characterize the unity of style, terminology and symbolism across all components of the software tool, including software documentation at all stages of its development. Different software design methods and tools are now developed. The methods of top-down, bottom-up design, the method of designing data structures (Jackson method), structural and modular programming, various programming technologies and forms of project representation have become widespread. Each of these methods has certain advantages and disadvantages. It is very important when designing large software complexes as a whole and each component separately to strictly and consistently adhere to pre-selected design methods.

If in the development of each component we use its methods, its symbolism, its terminology, then the difficulties of integrating the components into the software complex, maintenance and support of the complex increases excessively, and its accessibility decreases. Documentation indicators characterize the availability, accessibility for understanding in the program documentation of all information required for the production, implementation, operation and support of ST, as well as compliance with the requirements of standards and other regulatory documents, including standards in programming languages. Documentation plays a large role in all stages of the ST life cycle. Complete and accurate, understandable documentation provides management, control, and support for workflows. With good documentation, programs are written and debugged faster. Such programs are easier to learn, upgrade and adapt to different conditions of use. Therefore, all documentation throughout the software lifecycle from the beginning of development to the time of termination of use should be kept in full order and effectively monitored.

In the programming firms specialists are working who with the knowing the subtleties in programming, are able to quickly, professionally and clearly prepare the entire text part of program documentation. At the same time, they produce and reproduce not only the final reports, manuals and instructions on time, but also all general working materials: plans, terms of reference, algorithms, accepted coding tables, functional schemes, memory allocation schemes, accepted restrictions on the use of programming languages, etc. The amount of justified labor costs for

documentation is 20... 25% of all costs, so for every 5 programmers, it is sometimes advisable to keep one technical designer. Indicators of documentation should include indicators such as completeness of documentation, compliance with the requirements of standards and regulatory documents, clarity of documentation, availability of documentation (availability of tools that facilitate the search for necessary information), availability of automation tools for document correction, etc. All these indicators are qualitative.

9. Unification indicators

The unification indicators characterize the saturation of the ST with standard, unified and original components, as well as the level of unification with other software.

Unification of ST and their components avoids duplication of development, facilitates the process of integration of software systems, their assimilation and use.

Compilation of complex software complexes from unified components is relatively easy to automate. Thus, the unification of ST contributes to a significant reduction in the cost of labor and material resources for the development and use of software. To determine the level of unification, the ST and their components belong to one of the following types: standard, unified, original.

Unified ST are thoroughly tested and examined. As components or independently, they can be used in different conditions. Their use is twice advantageous. First, using ready-made ST, the user or developer saves their resources because it is no longer necessary to create this component. Secondly, the unified ST has already been thoroughly tested and is therefore of high quality. In addition, unified ST is easier to build into software complexes. When comparing two identical ST, all other things being equal, preference should be given to a ST with a higher proportion of unified components.

10. Multilevel hierarchy of structure of properties and quality indicators

The considered quality indicators nomenclature of computer ST is multilevel, hierarchical. Its structure is defined by two levels of hierarchy of indicators. The first level consists of groups of quality indicators; the

second is subgroups. The tree of attributes and quality indicators of ST is generally unbalanced in height. This means that at the same level, complex and single indicators or complex indicators relating to different levels may be found near different groups of indicators. Thus, the heights of components of a tree of attributes and indicators of ST quality do not depend on each other.

The quality indicators nomenclature of ST is common to all types of software. The working nomenclature of quality indicators for a particular type of ST is selected on the basis of a preliminary study of the attributed of the ST of this type and determination of the significance of specific quality indicators. The proposed nomenclature is open. This means that some new groups and subgroups quality indicators can be added to its membership.

11. Quality and efficiency of software. Quality Economy

The considered nomenclature of quality indicators allows to characterize the attributes of the evaluated ST and to conclude on the degree of suitability of its use for its intended purpose. But the positive features of the ST are not yet a guarantee of high efficiency. The use of ST should have some economic or socio-economic effect. The social effect is in many cases obvious but difficult to quantify and will not be considered. Cost-effectiveness indicators should constitute a mandatory stand-alone group of indicators and complement the assessments of the scientific and technical level of software. The concepts of Software Quality and Efficiency should not be confused[4].

The concepts of *efficiency* refer to such an operation by which any agreed set of actions combined by a common purpose. In a specific ST operation, as a measure of the relevance of the actual result of the use of the ST to the desired (expected) the efficiency of use should be understood. To obtain the effectiveness of a ST operation, it is required to establish a dynamic relationship between the attributes of all objects (entities) involved in the operation, the methods and conditions of the operation and the purpose of the operation itself. Therefore, the effectiveness of this operation depends not only on the quality of the ST, but also on other factors that affect the course and outcome of the operation. Generally,

[4] Popovich M.G., Kovalchuk M.G. Automatic control theory: a textbook. 2nd edition, revised. and suppl. Kyiv: Libid, 2007. 656 p.

efficiency is characterized by the following three components: goal output, resource costs, and time. At different stages of the ST life cycle, preliminary, potential, guaranteed and actual effects of the use of the evaluated software can be calculated. The preliminary economic effect is calculated before the start of development based on the TOR, technical proposals and usage forecast data. The preliminary economic effect is an element of the feasibility study of the need for software development and is used in the planning of development and implementation. The potential economic effect is calculated after completion of the development based on an assessment of the actual achieved technical and economic characteristics and the forecast of data on the maximum volumes of use of this software in the national economy. The potential effect is used in assessing existing organizations – ST developers. The guaranteed economic effect is calculated from one particular implementation, and from the implementation of several objects (guaranteed general economic effect). Guaranteed economic effect from a single implementation is calculated on the basis of data on the developer's guaranteed specific effect of the use of the ST and the terms guaranteed by the user, as well as the annual volume of its use. The guaranteed effect of a single software implementation is calculated when the contractual relationship between the developing organization and the user organization is made. The guaranteed total economic effect is calculated when setting up the ST for production on the basis of generalization of the estimated indicators of software use (by several sites of implementation), as well as data on the volumes of software implementation, corresponding to the possibilities of production, supply and maintenance. The guaranteed overall effect is the basis for the development and approval of economically justified prices for software products, production planning, delivery and implementation of software.

The actual economic effect is calculated based on the accounting data and the comparison of actual costs and results in the specific applications of the ST. The actual effect is calculated from both the single implementation of a particular ST at a particular site and the overall economic effect of using that ST at all implementation sites during the billing period. The actual effect is used to evaluate the activities of organizations that develop, implement and use ST to determine the amount of contributions to economic incentives, as well as to analyze the effectiveness of ST operation and to make proposals for ST improvement and conditions for its use.

The user applies any software product in conjunction with the computer on which it is implemented as a tool (means of production) to solve organizational, managerial, industrial, scientific and other tasks in their daily activities. Therefore, in assessing the cost-effectiveness of software, one can use a methodology for evaluating the cost-effectiveness of industrial products. In the first place, you should establish the sources of savings when using computer ST.

The main sources of cost savings for organizations (enterprises) using the software are: improving the performance of their kernel business; improvement of technical level, quality of production, and volume of work performed; shortening the time of information processing and increasing the speed of decision making; increasing the utilization rate of computing resources, means of preparation, processing and transmission of information; decrease in the number of personnel employed in data processing systems (DPS); reducing labor costs when performing certain types of work; optimizing decision making. Indicators of economic efficiency of ST are determined: for applied software – the impact of ST on the end result of their use; for ST organization of computing process and expansion of functions of operating systems – influence on technological processes of preparation, transfer and processing of data in DPS; for ST creation and program transformation – an action on the technological process of creating new ST, the productivity of programmers and the quality of programs.

In determining the economic efficiency of the ST included in the ACS, CAD, automated technological complexes, etc., the share impact of the software on the efficiency of automated systems are taken into account.

12. Assessment and methods for determining the quality level of software

An assessment of the quality level of any product is a set of operations that involves the selection of a nomenclature of quality indicators for the products being evaluated, the definition of these indicators and their comparison with baseline values. After defining the Quality indicators nomenclature, you must select the methods for determining the values of the indicators. Methods for determining the values of the quality of the evaluated products are classified as follows: by methods of obtaining information on these products (measurement, registration, organoleptic,

calculated), by sources of information (traditional, expert, sociological). The measurement method is based on obtaining information on the attributes and characteristics of software tools using measuring hardware and software. This method determines, for example, the volume of ST – the number of lines (machine commands, elementary structures, etc.) of the source text of the program and the number of lines-comments, the number of operators, operands, executed operators, branches in the program, the time of execution of branches of the program, reactivity indicators. To measure such characteristics, both technical, such as an electronic clock-timer, and software means, such as a path analyzer, a program for calculating elementary structures, etc., are used. The registration method is based on the receipt of information during the test or when running ST, when certain events are recorded and counted, such as time and number of failures, moments of time and reasons for interruptions in work, moments of transfer of control from module to module, moments of start time and end of work.

When registering such events, they also use both technical and special ST. The organoleptic method is based on the use of information obtained from the analysis of the perception of sensory organs, mainly the organs of vision and hearing. Because the software tools are poorly susceptible to organoleptic perception, the possibilities of this method are very limited. At the same time, this method can be used to determine such indicators as demonstrability, analysis capability, completeness, consistency, etc. Software and hardware are also required to implement this method. Visual perception is widely used, for example, display screens, in auditory – reproducers, etc. The calculation method is based on the use of theoretical and empirical dependencies in the early stages of development, as well as the use of statistics accumulated in the testing, operation and maintenance of ST. When designing ST, the calculation method predicts the accuracy, reliability, reactivity, etc. This method is also used to determine the actual values of the results of the testing and operation of the ST. When determining the values of some quality indicators often have to use not one, but a combination of several methods. For example, when determining the performance of a modified ST, the number and qualifications of the specialists involved in the ST modification are first recorded and then the time spent on the modification is measured and recorded. The coefficient of modification is calculated on the basis of empirical dependence.

The determination of ST quality indicators by the traditional method is carried out by employees of specialized experimental and (or) calculation units. Testing units include laboratories, landfills, departments, ST testing centers, etc., and design departments, software centers, computing centers, quality control services, etc. competent in this subject area. Determination of values of quality indicators by the expert method is carried out by a group of experts-specialists competent in this subject area. The decision is based on the experience and intuition of experts, not the direct results of calculations or experiments.

The organization and carrying out of expert evaluation of product quality is regulated by state standards of Ukraine. The expert method of software quality assessment is applied in the following cases: 1) the problem of quality assessment cannot be solved by any other existing method; 2) other methods are unacceptable due to extremely high labor costs. Sociological methods are to distribute special questionnaires with questions; conducting conferences and exhibitions to gather information on user satisfaction with the quality of the evaluated ST; elucidation of unsolved problems, peculiarities of usage and functioning of ST, directions of ST modernization, etc. In preparing for sociological research, particular attention should be paid to the preparation of questionnaires. There have been cases where the results of a major work were almost zero due to poor preparation. In order to avoid this, you need to conduct a pre-survey and data processing. The value of many ST quality indicators are random variables. Such indicators, in particular, include indicators of accuracy, reliability, reactivity, diagnoses, reproducibility. Therefore, there is a need to use statistical methods of obtaining and processing data to determine the value of these indicators. Initial data for statistical processing are either accumulated during the real-time operation of the ST, or obtained during testing when modeling the operating environment. The peculiarities of such tests will be considered further. Indicators that are evaluated on metric scales are called quantitative, and ordinal and nominal tests are qualitative. The accuracy of the rating depends on the choice of rating scales. Metric scales are the most versatile, and therefore generally more acceptable. But they are often unacceptable either because of the lack of technical capacity to measure the parameters or due to the unjustified complexity and cost of measurement. The selected scales should match the technical capabilities of their use and the tasks to be solved. Methods for

determining the values of quality indicators depend on the stage of the ST life cycle. For example, measurement and registration methods for obtaining information can only be fully applied after the development of a draft copy of the ST.

13. Selection of basic samples of quality indicators

The quality level of the evaluated product is determined by comparing its quality indices with those of an existing or hypothetical product, similar to the one evaluated, taken as the basic sample. The basic sample is the really achievable set of values of product quality indicators taken for comparison. Quality indicators of a basic sample are called baseline values of indicators. The set of baseline values of indicators should characterize the optimum level of quality of this type of production for some specified period of time.

Thus, before starting to evaluate the quality level of the software, it is necessary to select a basic sample for comparison and to set the values of quality indicators of the basic sample. By having this data and the Quality indicators values of the basic ST sample, you can set the quality level of that sample. If the evaluation of the ST exceeds the baseline values of the quality indicators in all its indicators, then the developer of this software can be considered to have achieved the goal that is set for him.

The following requirements are required for the basic values of ST quality indicators: these indicators must meet: 1) the values of the quality indicators of the best domestic and foreign software from the number of analogues; 2) the predicted value of the quality indicators of the best foreign and domestic samples-analogues until the completion of development; 3) the normative values of the indicators, which are set by individual types of ST.

Analogs-samples include real existing domestic and foreign ST of the same kind as comparable ones, having similarity of functional purpose, basic parameters, structure and conditions of use. Thus, the baseline values of the quality indicators should not only exceed the values of the best real domestic and foreign samples, analogues of the ST, but also the predicted values of the best of these samples, which can be achieved by the time of the end of the development of the evaluated sample ST. Only such an approach can ensure that the speed of software quality growth and the actual conformity of the scientific and technical level of the used products

with the best analogues are the most appropriate. The choice of the basic sample and the basic values of the quality indicators largely depends on the reliability of the results of the assessment of the quality of products and the correctness of the decisions taken. The use of outdated and imperfect samples leads to an unreasonably overestimated assessment of the quality of products. The choice of the basic sample and the baseline values of the quality indicators should be scientifically substantiated, and the decision-makers should be personally responsible for the correctness of the decisions taken. The choice of baseline samples and baseline values of quality indicators for software products is associated with great difficulties, which are due, albeit to temporary, but objective reasons: the lack of generally accepted quality indicators, suitable for comparative software evaluation; lack of data on the value of quality indicators of most foreign and domestic ST; low level of unification, limited information on the properties and characteristics of the ST, which impedes the choice of samples analogues of the ST; lack of a unified classification system that includes all hierarchical software levels (subclasses, groups, subgroups, species, subspecies of ST); weak development of methods for determining optimal values of software quality indicators. However, without defining the baseline values of quality indicators, it is impossible to establish the level of product quality, so the assessment of the quality level of specific types of software should begin with eliminating the reasons that impede the choice of baseline values of indicators. However, due to the lack of samples-analogues or their characteristics, it is often necessary to justify the optimum values of the baseline indicators, which must be carefully evaluated beforehand, which eliminates the arbitrary choice of the baseline values of the quality indicators. The selection of basic samples is carried out at the stage of development of the TOR

14. Methods for assessing the quality level of software

Differential, complex and mixed methods are used to evaluate the quality of software.

Differential method is the method of estimating the level of product quality, which is based on the use of single quality indicators. At the same time they determine the following: to achieve level of the basic sample as a whole, by what indicators it is reached and by which it is not reached. When using the differential method, the quality level of the products being

evaluated is considered to be above or equal to the level of the basic sample if all values of the relative indicators are greater than or equal to one. Otherwise, the level of quality of the evaluated products is lower than the level of the basic sample. Differential method allows to take into account the value of each indicator (among the selected) when assessing the quality level of the software. Then with poor quality, customers and developers see what software properties need improvement. This is the main advantage of this method. But this method requires careful justification of completeness and selection of quality indicators, uniformity of methods for determining the values of quality indicators of the evaluated ST and the basic sample. A comprehensive method of assessing the level of product quality is based on the use of a single generic indicator, which is a function of several main unit (group) indicators. The generalized indicator can be expressed as the main indicator reflecting the main purpose of the software product; an integral indicator of economic importance; a weighted average (geometric or arithmetic) indicator of quality. To use the main indicator, you need to set its dependence on the original indicators. This indicator is focused on accounting for the direct effect of using the ST for its intended purpose, but does not take into account the cost of achieving this effect.

The integral indicator is used when the total useful effect of the use of SP, the total cost of its creation (acquisition) and operation, as well as the acquisition (depreciation) of computer equipment (including the required system programs) and their operation are established. Weighted average indicators are used in cases where it is necessary to determine the main indicator and to establish its functional dependence on the initial performance indicators of the software product. The values of the parameters are determined when drawing up the terms of reference (specifications) for the ST being developed or the ST quality improvement plan, and are reviewed only when these documents are corrected. The advantage of a comprehensive method of assessing the quality of products is that it allows you to immediately obtain a generalized value of the quality indicator and, in the presence of an appropriate baseline value of the quality indicator to conclude on the quality level of ST. However, if the result is unsatisfactory, this method does not provide information about what ST parameters should be affected to improve its quality. It does not give information about the specific attributes of the evaluated ST of

interest to the user (for example, the properties of the modified ST, flexibility, accuracy, reactivity, etc.).

The mixed method of assessing the level of product quality is based on the joint application of single and complex (group) indicators. When using the mixed method, some of the individual indicators are grouped together. After that, the relative values of the group and some individual indicators are calculated by the formulas. The comparison of the quality of the evaluated ST with the basic sample is carried out in the same way as in the differential method. The mixed method is applicable in the following cases: the set of single quality indicators is too large and complicates the generalization of conclusions; a generalized indicator of quality in the complex method allows to draw conclusions about important groups of attributes. The mixed method compensates for the disadvantages of differential and complex methods. But its use is associated with the difficulty of finding (allocating) group and single indicators that determine the quality of the evaluated ST.

REFERENCES

1. Feldbaum A.A., Butkovsky A.G. Methods of the theory of automatic control, Main editorial office of physical and mathematical literature "Nauka", Moscow: 1971, 744 p.
2. Krainnikov A.V., Kurdikov V.A., Lebedev A.N. and others; Probabilistic methods in computer engineering: Textbook manual for universities on spec. Computer. Ed. A.N. Lebedeva, E.A. Chernyavsky. Moscow: Higher school, 1986. 316 p.: pic.
3. Miroshnik I.V. Automatic control theory. Linear systems. St. Petersburg: Peter, 2005. 336 p.: pic. (Training Series).
4. Popovich M.G., Kovalchuk M.G. Automatic control theory: a textbook. 2nd edition, revised. and suppl. Kyiv: Libid, 2007. 656 p.

Information about the author:
Kyselov V. B.
Doctor of Technical Sciences, Professor,
Director of the Institute of Municipal Administration
and Urban Economics
of the V. I. Vernadsky Taurida National University