# Methods of quality assurance of software development based on a systems approach

Iryna Ushakova[1], Yuri Skorin[2], Alexander Shcherbakov[3]

[1,2,3] *Simon Kuznets Kharkiv National University of Economics, Nauky Ave., 9-A, Kharkiv, 61166, Ukraine*

**Abstract**

The aim of the work is to analyze the problems and develop recommendations for quality assurance of software and testing during its creation in IT companies based on a systems approach. The object of research is the processes of testing, quality control and quality assurance. The subject of the study is the functions of quality assurance (QA) and testing (QC) within the system of development and the characteristics and models of quality assessment and software dependability. The research processes used a systematic approach, comparative analysis of quality assessment methods and approaches to the organization of testing, quality control and quality assurance of software products. The essence and main differences of the concepts "testing", "quality control" and "quality assurance" were determined. To assess the quality of the software, various aspects of quality in accordance with international standards, the relationship between them and a multi-level model of software quality were considered. To ensure the quality of the software product, it was proposed to use methods of integrated quality assessment, which allow to obtain the final integrated value of software quality as a whole, expressed in certain quantitative indicators, or its individual characteristics, and considered the most common methods based on costs and hierarchical models. A systematic approach to software quality assurance involves the creation of a QA team, which is an independent subsystem within the software development system while maintaining links with team members. To assess the differences between quality control and quality assurance, an analysis of responsibilities, work planning and documentation of relevant groups in IT companies was conducted, which made it possible to compare the functions performed and working conditions. Thus, the QC function confirms that a specific result meets standards and specifications, and QA is a broader function that covers planning and control throughout the development lifecycle. Testing is an integral part of quality control. In order for an IT company to provide management processes, QA and QC teams must work together. The scientific novelty of the work is to develop a methodological basis for assessing the quality of software, developing recommendations for improving the processes of quality assurance and testing in software development in an IT company.

**Keywords**

Software, testing, quality control, quality assurance, dependability, security, quality model, metrics, quality indicators, system approach



*III International Scientific And Practical Conference "Information Security And Information Technologies", September 13–19, 2021, Odesa, Ukraine*

EMAIL: varavina.ira@gmail.com (A. 1); skorin.yuriy@gmail.com (A. 2); oleksandr.shcherbakov.kafis@gmail.com (A. 3)

ORCID: 0000-0001-8315-0917 (A. 1); 0000-0002-4613-3154 (A. 2); 0000-0001-8315-0917 (A. 3)

## 1. Introduction

The fourth industrial revolution, of course, poses great challenges for "traditional" software development. This is due to the unpredictable behavior of software systems, lack of centralized control, cybersecurity, scalability, fault tolerance, reliability, development, definition of interfaces and communication channels and their management. However, most of these problems can also be seen as opportunities for further development of software development and testing processes [21, 24].

Quality assurance or software quality assurance is an integral part of the development process and is used in the IT industry by quality assurance professionals as well as testers. Quality assurance is associated with the concept of dependability. Dependability is, first, a guarantee of increased cybersecurity, reliability and protection against failures. In cases where the failure of a software system that belongs to the class of "high confidence" or "high integrity system" can lead to extremely negative consequences, the overall warranty of the system, which includes hardware, software and man, is the main and priority quality requirement in relation to the main functionality of the system.

Both quality assurance and software testing are designed to guarantee the quality of the software application that meets customer requirements. However, these two concepts have a fundamental difference. Testing is performed after the application has been created or for static testing after the software requirements have been defined and recorded in the relevant document [11,25]. Quality assurance involves activities that ensure the quality of the application during its creation at all stages, from the definition of requirements to the transfer of the finished application to the customer [17,28].

To understand the differences between these components of the software development process, it is necessary to give a clear definition of these concepts, to relate between their characteristics, to determine methods for assessing the quality of software.

Successful solution of software quality assurance problems is possible only with a systematic approach to software development processes, active involvement of quality assurance specialists and testers, so the work will identify differences between the responsibilities of these specialists, differences in planning tests and documentation, as well as developed recommendations for improving software development processes in terms of quality assurance.

The main purpose of the article is to analyze the problems and develop recommendations for quality assurance of software and testing during its creation in IT companies based on the principles of a systems approach.

## 2. Review of literature sources

To clarify the differences between the concepts of testing and software quality assurance, consider the related concepts of "testing", "quality control" and "quality assurance", which are widely covered both in the domestic literature and in foreign sources. [6-9, 11, 17].

Software testing according to ISO / IEC TR 19759: 2005 is a process of research, software testing, which aims to verify the correspondence between the actual behavior of the program and its expected behavior on the final set of tests selected by a particular.

Quality Control (QC) according to ISO 9000 is a part of quality management focused on compliance with the requirements for assessing the number of defects, bugs (if any) in the application. Quality control role is a set of processes (actions) aimed at assessing the developed application (draft document, development system, etc.) and compliance with customer requirements. Execution of these processes guarantees check of quality of the delivered application and defines, how well it is designed and executed. The purpose of quality control is to find defects and ensure their correction. Thus, testing is an integral part of quality control (fig. 1).

Quality Assurance (QA) is defined in ISO 9000 as a part of quality management that focuses on ensuring that defect elimination requirements are met. The purpose of quality assurance is to ensure that the application will meet customer requirements. Quality assurance consists of processes aimed at ensuring the quality of application development at each stage of the life cycle. These actions usually precede application development and continue while the process is under development. Quality assurance is responsible for the development and implementation of processes and standards to improve the development life cycle, and to ensure

that these processes are performed [1, 2]. The main purpose of quality assurance is to prevent defects at all stages of software development and its continuous improvement. While quality assurance is an activity aimed at ensuring the development of quality software, quality control is an activity that captures and evaluates the quality of an already created application. So testing is a subsystem of quality control, and quality control is a subsystem of quality assurance system.

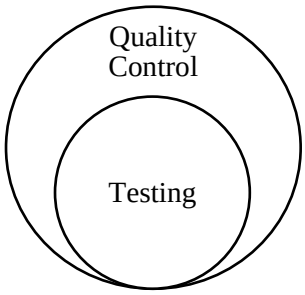


**Figure 1**: The relationship between the concepts of "testing" and "quality control"

The relationship between quality assurance, quality control and testing shows in fig. 2. Quality assurance activities include setting standards and processes, quality control, and selecting appropriate tools.

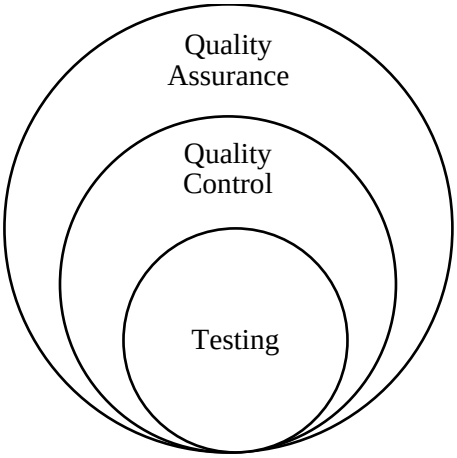


**Figure 2**: The relationship between QA, QC and Testing

The quality of software is defined in ISO 9126 as the whole set of its characteristics related to the ability to meet the stated or implied needs of all stakeholders.

There are the following aspects of software quality [6,26]:

1. The quality of technological processes of software development, which affects the creation of quality software;
2. The internal quality of the software associated with its characteristics, without taking into account the behavior of the software application;
3. External quality that characterizes the software in terms of its behavior;
4. The quality of the software when used in different contexts, that is the quality of the software application, which is manifested in its use by users in different specific scenarios.

Metrics have been created for all these aspects of quality that allow them to be evaluated

In fig. 3 shows the relationship of different aspects of software quality.

In addition, the standard describes a multi-level software quality model that can be used to describe both internal and external software quality (fig. 4). At the top level of the model there are 6 main characteristics of software quality, each of which has its own attributes:

functionality: ability to interact, functional suitability, compliance with standards and rules, security, accuracy;

reliability: completeness, ability to recover, compliance with standards, resistance to failure;

usability: intelligibility, ease of learning, ease of operation, attractiveness, compliance with standards;

productivity: time efficiency, resource efficiency, compliance with standards;

ease of maintenance: analysis, ease of making changes, stability, ease of verification, compliance with standards;

transfer: adaptability, ease of installation, ability to coexist, ease of replacement, compliance with standards.

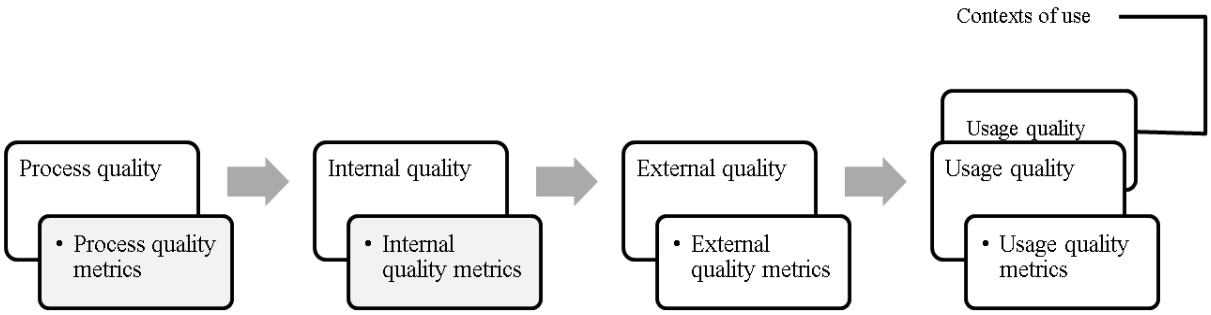


**Figure 3**: Communication of different aspects of software quality according to ISO 9126

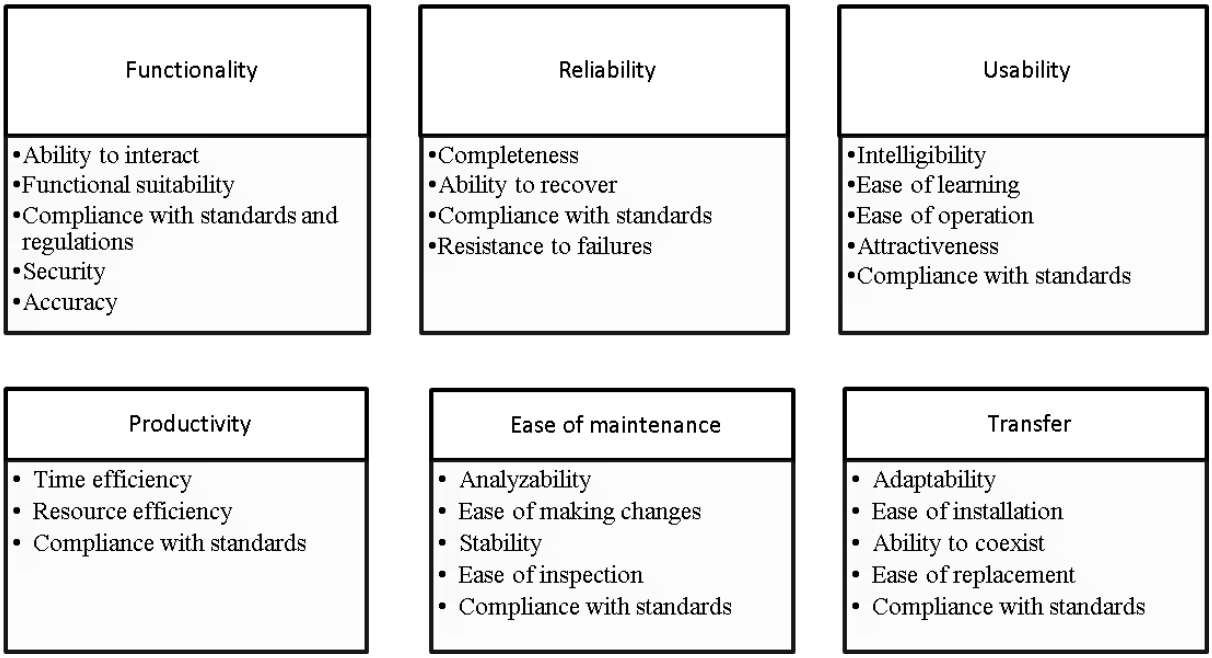


**Figure 4**: Multi-level software quality model according to ISO 9126

A set of metrics is defined for each attribute that allow it to be evaluated. Metrics must have the following properties:

1) reliability; which is associated with an accidental error; the metric is free from random error, if random changes do not affect the results of the metric;

2) recurrence; the re-use of metrics for the same application and by the same evaluators when using the same evaluation specification (including the environment), the same type of users and environment, should lead to the same results with appropriate tolerances; appropriate tolerances should take into account such components as fatigue and the result of accumulated knowledge;

3) uniformity; the application of metrics for the same application by different assessment professionals using the same assessment specification (including the environment), the same type of users and environment, should lead to the same results with appropriate tolerances;

4) possibility of application; the metric must clearly indicate the conditions (for example, the presence of certain attributes) that limit its use;

5) showiness; it is the ability of a metric to identify parts or elements of a program that need to be improved, based on a comparison of measured and expected results;

6) correctness; the metric must have the following properties:

objectivity; the results of the metric and its input should be based on facts and not be subject to the feelings or opinions of experts in assessment or testing (excluding metrics of satisfaction or attractiveness, which measure the feelings and opinions of the user);

impartiality; the measurement should not be aimed at obtaining any specific result;

adequacy of accuracy; accuracy is determined when designing metrics and especially when choosing descriptions of facts that are used as a basis for metrics; the metric developer must describe the accuracy and sensitivity of the metric;

7) significance; the measurement must give significant results concerning the behavior of the program or the quality characteristics.

Metrics must also be cost-effective. This means that more expensive metrics should provide better evaluation results [1, 2].

The developer of the metric must prove its validity. The metric must meet at least one of the following criteria for the validity of the metric:

1) correlation; the change in the values of quality parameters (promptly determined by measuring the basic metrics), due to a change in the values of the metric, should be determined by a linear relationship;

2) tracing; if the metric M is directly related to the value of the quality characteristic Q, then the change in value $Q(T_1)$, available at the time $T_1$, to the value of $Q(T_2)$, obtained at time $T_2$, must be accompanied by a change in the value of the metric from $M(T_1)$ to $M(T_{12})$ in the same direction (for example, if $Q$ increases, then $M$ also increases);

3) consistency; if the values of quality characteristics (promptly obtained by measuring the main metrics) $Q_1, Q_2, \ldots, Q_n$, associated with applications or processes 1,2 ..., n, are determined by the ratio $Q_1 > Q_2 > \ldots > Q_n$, associated with applications or processes 1,2 ..., n, are determined by the ratio $M_1 > M_2 > \ldots > M_n$;

4) predictability; if the metric is used at time $T_1$ to predict the value (promptly obtained by measuring the main metrics) of the quality characteristics Q at time $T_2$, the prediction error must fall within the allowable range of prediction errors:

$$(Q_p(T_2) - Q_f(T_2)) / Q_f(T_2), \quad (1)$$

where $Q_p(T_2)$ – the forecast value of the quality characteristics at the time $T_2$,

$Q_f(T_2)$ – the actual value of the quality characteristic at the time $T_2$;

5) selectivity; the metric must be able to distinguish between high and low quality software.

Improving the quality of software development and testing allows you to create a software application that meets customer requirements [10, 12-15,27]. Attention should be paid to the thorough improvement of all software development processes, both directly related to the development of perfect software code and all processes that affect its quality: definition and management of requirements, creation of test scenarios and testing as early as possible (starting with requirements testing), organization of teamwork, division of responsibilities between participants in the process, etc.

Recently, considerable attention in the field of software quality assurance is paid to warranty. Dependability of software includes such characteristics as fault tolerance, safety of use (safety in the context of acceptable risk to human health, business, property, etc.), information security or security - protection of information from unauthorized transactions, including access to reading, as well as guaranteeing the availability of information to authorized users, in the amount of their rights), as well as convenience and ease of use (usability) [27]. Reliability is also a criterion that can be defined in terms of warranty.

Special attention is paid to creating a perfect code despite the current trends in the field of information technology and in particular testing, [3-5, 16].

Analysis of modern strategies, approaches and methods of testing, identification of their advantages and disadvantages paid attention in [22, 25].

Ways to solve the problem of improving the quality of software development and testing can be the introduction of appropriate methods in IT companies to assess the quality of software, which will contribute to its warranty.

## 3. A systematic approach to improving quality assurance and testing processes in software development

The need for software quality assurance increases with the size of the organization and the level of its quality policy. Quality assurance is a complex multifaceted process. Therefore, the system approach provides its required level in full. This approach considers quality assurance as a separate subsystem, which is part of the development system, has certain connections with it, as well as certain independence as a system. The IT Company creates a QA group (quality assurance group). It is important that the QA function remains independent of project

management and operations. But the links between the QA team and the project team are very important and should provide them with strong support.

Some organizations have a QA feature built into the project management office. Such a model also meets the criteria of independence. However, with such an organization, you need to make sure that the QA group consists of qualified quality assurance analysts.

Given the differences between the concepts of software testing, quality control and quality assurance, there are also differences between the responsibilities of the QA group and testers.

The responsibilities of testers include:

testing planning,

writing test scripts and test cases, checking tests,

performing tests,

analysis of test results,

creation and analysis of reporting on test results for different levels of tests.

As part of their quality control role, testers may make demands on:

checking samples of project documents,

activities for managing software configurations, design, code, etc.

At the same time, the QA group performs the functions:

formation of organizational policy on quality, standards and development processes;

providing assistance with quality assurance training and project quality assurance plans;

checking compliance between project processes and quality plans;

conducting regular inspections of design applications and processes;

regular presentation of the results of quality assessment analysis to management;

resolving a situation with a deviation from guidelines or standards.

As part of its quality assurance role, the QA group monitors:

independent reviews;

availability of project change management procedures;

availability of project configuration management procedures;

availability of retrospective planning and implementation of development life cycle processes;

quality assurance based on the development of the life cycle system;

carrying out continuous improvement in the process of quality control and implementation of recommendations based on previous experience.

Performing the duties of the QA group does not mean their development by the team, but only ensuring their implementation.

When planning tests, testers prepare test strategies and plans based on basic test documents, such as software application requirements and design solutions. These test planning documents are the basis for the implementation of processes at various planned test levels. For each level of testing, tests, sets of input data and expected results, detailed test schedules, environmental requirements, documents for defect management, test management and reporting are compiled. In contrast, software application quality assurance documentation or quality plans include a broader set of actions throughout all stages of development. This affects the project management methodology.

A typical draft quality plan includes customer expectations, acceptance criteria, planned quality control and process audits, configuration management plans, and change management procedures. Quality plans are based on the organization's own policies, standards, or guidelines that form the basis of quality assurance. The project quality assurance plan is monitored continuously and the planned quality indicators are updated on its basis during the project creation. There are different intersections between risk management and quality, and therefore the risk register can make a significant contribution to the preparation of quality plans.

Recommendations for improving quality assurance processes:

independence. To be successful the QA group must be dependent on the project team. This provides the QA group with the opportunity to conduct an objective evaluation of projects. Testers and QA specialists can be in the same group in small organizations. However, there is a possibility of creating a conflict of interest in monitoring the testing activities. The solution depends on the policy of the organization in the field of quality and is as follows. A separate group can be created for reporting;

relationships within the project team. Quality assurance analysts may be overly process-oriented and may insist on processes or documentation that are of little relevance to the project. This can worsen relationships with project managers. It will be much easier for the

QA group to work with project teams if they work on the principle of taking into account the project objectives. In addition, the assistance and assistance of project teams forms the basis for maintaining good relations. This is an important aspect of successful testing;

involvement of the necessary specialists. Qualitative HR policy plays a leading role in the successful operation of the QA Group. People with experience in LC development who have knowledge of ISO standards and CMMI principles for software development have the necessary competencies for the QA team;

requirements list. Standard checklists are a useful mechanism for auditing projects, especially if they are designed in accordance with the LC phases. To ensure fruitful cooperation with project managers, it is important to ensure the participation of stakeholders in the project. This makes it possible to get feedback from them in response to suggestions for changes to the lists;

communication and reporting. Regular reporting is very important to management, developing the right templates and metrics to provide management with the information it needs to ensure that these reports are given the proper attention. This is best achieved by meeting with relevant senior management representatives, providing them with reports and receiving feedback and comments from them. In addition, the QA team must continually obtain approval for changes to quality control processes and standards and ensure effective communication with stakeholders;

constant improvement. Taking into account previous experience provides the QA team with a basis for evaluating processes and recommendations for quality assurance, including continuous improvement. The QA team must be flexible, maintain good relationships with stakeholders when making improvements in management reporting. Continuous improvement may also require amendments to the methodology of development of software systems, so QA group recommended to keep development methodology IT company.

## 4. Introduction of methods of integrated quality assessment of software applications

Methods of integrated quality assessment have the advantage that they allow to obtain the final integrated value of the quality of the software as a whole or its individual characteristics, expressed in certain quantitative indicators. Cost-based and hierarchical model-based methods of integral software quality assessment are the most common.

The method of integrated software quality assessment, which is based on costs, belongs to the group of calculation methods. According to this method, a quantitative criterion of software quality $T$ is formulated, focused on its life cycle. (LC).

The costs of software development, operation and maintenance include:

$R$ – one-time software development costs;

$V$ – one-time software implementation costs;

$E$ – recurring costs $S$ for software operation for the period of operation time $t_e$ during the life cycle T:

$$E = (\mathrm{T} / t_e) * S; \quad (2)$$

$C$ – repeated at random intervals maintenance costs, which are on average $n-th$ part of the costs $R$ and $m-th$ part of the costs $V$ and are carried out during the life cycle $T$ on average over time $t_c$:

$$\mathrm{C} = (n * R + m * V) * \mathrm{T} / t_c; \quad (3)$$

B – accidental losses due to unreliability or lateness of the result:

$$\mathrm{B} = S_\mathrm{e} * \mathrm{T} / t_\mathrm{e}, \quad (4)$$

where $S_\mathrm{e}$ – the average amount of losses incurred by a single operation of the software during its LC.

Thus, the total cost $Z$ in the software life cycle of the software will be determined as follows:

$$Z = R + V + (S_\mathrm{e} + S) * \frac{\mathrm{T}}{t_\mathrm{e}} + \\ +(n * R + m * V) * \mathrm{T} / t_\mathrm{c}, \quad (5)$$

As a quality criterion, it is proposed to use the minimization of total costs for software development, operation and maintenance. The criterion for software quality is to minimize the total cost Z:

$$Z \rightarrow min. \quad (6)$$

The main disadvantage of this method is that the actual cost values included in the formula can be determined after the development of the

software application, and therefore it cannot be used as a tool in the development process to achieve a given level of quality.

The choice of the nomenclature of quality indicators according to the method of quality assessment based on a hierarchical model for a particular software application is based on its purpose and requirements for the scope depending on the affiliation of the software to a subclass determined by the software classifier:

operating systems and means of their expansion;

database management software;

tool-technological means of programming;

software applications for interface and communication management;

software applications for the organization of the computational process (planning, control);

service programs;

software applications for computer maintenance;

research applications;

design applications;

applications for control of technical devices and technological processes;

applications for solving economic problems;

other software applications.

Evaluation of software quality is the choice of nomenclature of indicators, their evaluation and comparison with the basic values. A four-level hierarchical quality model is the basis of this evaluation method. It includes:

level 1 - quality characteristics;

level 2 - quality attributes;

level 3 - metrics;

level 4 - evaluation indicators (software attributes).

For each of the selected quality characteristics, a four-level hierarchical model is developed, which reflects the relationship of characteristics, attributes, metrics and indicators. The type of this model depends on the phase of the LC.

Tables are used for practical application of the model. These tables are created for each characteristic. So to assess the characteristics of information security, you can use the indicators that are in table 1.

**Table 1**

Indicators of assessment of the characteristic of information security

| Indicator | Evaluation method | Evaluation form |
|---|---|---|
| Proportion of incidents by type, $P_t$ | Registration, calculated | $P_t = \frac{KS_t}{\sum_t KI_t}$, (7)<br>$KI_t$ – the number of $t-th$ incidents |
| Proportion of deadlines incidents, $P_s$ | Registration, calculated | $P_s = \frac{KI_s}{KI}$, (8)<br>$KI_S$ – the number of incidents closed in time<br>KI – the total number of incidents |
| Probability of trouble-free operation, P | Registration, calculated | $P = 1 - q/n$, (9)<br>$n$ – number of tests,<br>$q$ – number of registered failures |

Quality assessment is a deterministic process that consists of certain stages. Its implementation involves the main stages:

determining the purpose of evaluation,

development of quality model,

creating a model of metrics,

search for basic metrics,

determination of derived metrics,

formalization of metrics,

determination of metric limit values,

determination of actual values of metrics,

definition of integrated software quality assessment,

software quality analysis.

The first stage involves determining the purpose of evaluation:

evaluate the quality of the finished software application, for example in accordance with the quality standard;

evaluate the quality of the software application during its development.

A certain model of integrated assessment is chosen depending on the goal and then consistently performs certain steps.

To determine, for example, the proportion of incidents of a certain type, it is necessary to record all incidents for a certain period. Then the percentage of a certain incident is determined (table 2). To assess the quality of software for this indicator, the values are compared with the allowable value. These data are used for analysis (fig. 4) and subsequent integrated evaluation of application quality.

**Table 2**

Proportion of incidents "Injection of malicious code", 01-06/2021

| Month | 01 | 02 | 03 | 04 | 05 | 06 |
|---|---|---|---|---|---|---|
| Proportion of incidents,% | 20 | 22 | 18 | 16 | 20 | 14 |

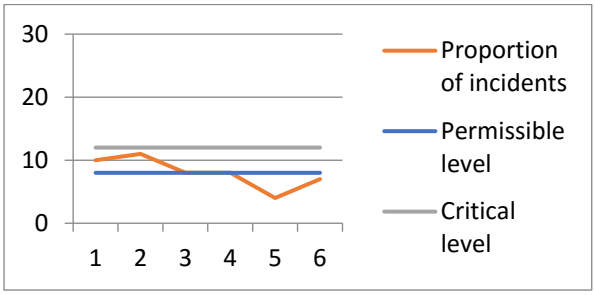


**Figure 4**: Proportion of incidents "Injection of malicious code"

To ensure quality in the process of software application development, both methods should be used:

to perform quality assessment during development to quickly ensure compliance of processes with certain standards and compliance of the software application with customer requirements,

to estimate the total cost of development, operation and maintenance of the finished software application with.

## 5. Conclusions

The paper compares the concepts of "testing", "quality control" and "quality assurance", which showed that testing is part of quality control, and quality control coincides with quality assurance in the field of quality control. Dependability, which includes fault-tolerance, safety, information security or security, as well as usability, should be provided primarily for software systems of high reliability, high availability within the quality guarantee.

Software quality assessment should take into account international standards in this field, which define various aspects of quality, such as process quality, internal quality, external quality and quality of use. To assess quality, it is recommended to use a multi-level model that includes the following characteristics:

functionality,
reliability,
usability,
productivity,
convenience of support,
transfer.

From the point of view of the systems approach, quality assurance can be defined as a separate subsystem, which is a component of the development system, has certain connections with it, as well as a certain independence as a system. To assess the differences between quality assurance and quality control processes, an analysis of the responsibilities of the relevant groups of specialists, their work planning and documentation was carry out, which made it possible to compare the functions performed and working conditions. Thus, QC functions are aimed at confirming that specific results meet standards and specifications, and QA is a broader function. It covers planning and control throughout the development lifecycle. Testing is an integral part of quality control. In order for an IT company to have effective quality management processes, the QA and QC group must work together.

A successful QA group can add significant value to an organization, namely:

improving the quality and warranty of software applications;

consistency in the delivery of software applications;

improving the organization of processes;

reduction of total delivery costs;

use applications for application support documentation.

At the same time, it should be borne in mind that QA specialists require additional costs:

firstly, in the staffing schedule for software quality analysts,

secondly, due to the complexity of processes.

At the beginning of implementation it may adversely affect the team.

Software quality assurance requires the introduction of integrated quality assessment methods and individual quality indicators. Integrated evaluation processes include:

defining the purpose of evaluation,

developing a quality model,

creating a model of metrics,

searching for basic metrics, defining derived metrics,

formalizing metrics,

defining metric limits,

determining actual metric values,

defining integrated software quality assessment,

software quality analysis.

To ensure quality, it is necessary to carry out its operational integrated assessment at all stages of LC and integrated assessment of costs for development, operation and maintenance of the finished software application.

## 6. References


[1] Dzh. Folk, Kaner, E. Nhuen, Testyrovanye prohrammnoho obespechenyia. Fundamentalnыe kontseptsyy menedzhmenta byznes–prylozhenyi, per. s anhl., Yzdatelstvo «Dya-Soft», Kyev, 2001.

[2] K. A. Kulakov, V. M. Dymytrov, Osnovы testyrovanyia prohrammnoho obespechenyia, Yzdatelstvo PetrHU, Petrozavodsk, 2018.

[3] Dzh. Makhrehor D. Saiks, Testyrovanye obъektno-oryentyrovannoho prohrammnoho obespechenyia, Dyasoft, Kyev, 2002.

[4] S. Makkonnell, Sovershennыi kod. Master-klass, Yzdatelsko-torhovыi dom «Russkaia redaktsyia», Moskva, Sankt-Peterburh, Pyter, 2005.

[5] M. A. Plaksyn, Testyrovanye y otladka prohramm dlia professyonalov budushchykh y nastoiashchykh, 2-e yzd. (эl.), BYNOM. Laboratoryia znanyi, Moskva, 2013.

[6] Prohramna inzheneriia. Yakist produktu. Chastyna 1. Model yakosti (ISO/IEC 9126-1:2001, IDT): DSTU ISO/IEC 9126-1:2013, Chynnyi vid 2014-07-01, MINEKONOMROZVYTKU Ukrainy , Kyiv, 2014.

[7] Prohramna inzheneriia. Yakist produktu. Chastyna 2. Zovnishni metryky (ISO/IEC TR 9126-2:2003, IDT): DSTU ISO/IEC TR 9126-2:2008,Chynnyi vid 2010-07-01, Derzhspozhyvstandart Ukrainy, Kyiv, 2011.

[8] Prohramna inzheneriia. Yakist produktu. Chastyna 3. Vnutrishni metryky (ISO/IEC TR 9126-3:2003, IDT): DSTU ISO/IEC TR 9126-3:2012,Chynnyi vid 2013-05-01, MINEKONOMROZVYTKU Ukrainy, Kyiv, 2013.

[9] Prohramna inzheneriia. Yakist produktu. Chastyna 4. Metryky yakosti pid chas vykorystannia (ISO/IEC TR 9126-4:2004, IDT): DSTU ISO/IEC TR 9126-4:2012, Chynnyi vid 2013-05-01, MINEKONOMROZVYTKU Ukrainy, Kyiv, 2013.

[10] S. V. Synytsыn, N. Yu. Naliutyn, Veryfykatsyia prohrammnoho obespechenyia, Yntuyt NOU, Moskva, 2016.

[11] I. O. Ushakova, Metodyka upravlinnia vymohamy v hnuchkykh metodolohiiakh, Zbirnyk naukovykh prats KhNUPS, Vyp. 2(56) (2018): 93 – 98.

[12] I. O. Ushakova, Proektuvannia informatsiinykh system: praktykum, KhNEU im. S. Kuznetsia, Kharkiv, 2015.

[13] I. O. Ushakova Roli i kliuchovi yakosti IT-spetsialista, v: Tezysы VII Mezhdunarodnoi nauchno-praktycheskoi konferentsyy “Problemы y perspektyvы razvytyia, 2015, s. 23.

[14] I. O. Ushakova, Systemnыi podkhod k upravlenyiu trebovanyiamy pry proektyrovanyy ynformatsyonnыkh system, v: Ynformatsyonnыe systemы v upravlenyy, obrazovanyy, promыshlennosty : monohrafyia. Vyd. TOV «Shchedra sadyba plius», Kharkiv :, 2014, ss. 86-91.

[15] M. Fauler, Refaktorynh. Uluchshenye sushchestvuiushcheho koda, per. s anhl., Symvol-Plius, Sankt-Peterburh, 2003.

[16] H. Foidl, M. Feldere. Integrating software quality models into risk-based testing, Software Quality Journal, V 26 (2018): 809 – 847.

[17] H. V. Gamido, M. V. Gamido, Comparative review of the features of automated software testing tools, International Journal of Electrical and Computer Engineering, Vol. 9, No. 5, (2019): 4473~4478

[18] P. M. Jacob, P. A. Mani, Framework for evaluating performance of software testing tools, Journal of Scientific and Technology Research, V. 9, Iss. 2 (2020): 2175–2180.

[19] R. S. Kenett, R. S. Swarz, A. Zonnenshain, Zonnenshain. Systems Engineering in the Fourth Industrial Revolution: Big Data,

Novel Technologies, and Modern Systems Engineering, Wiley, New York, NY, 2020.
[20] R. Pietrantuono, On the testing resource allocation problem: Research trends and perspectives, Journal of Systems and Software, V. 161 (2020): 42 p.
[21] A. A. Sawant, P. H. Bari, P. M. Chawan, Software Testing Techniques and Strategies, International Journal of Engineering Research and Applications, Vol. 2, Iss. 3 (2012): 980-986.
[22] Software Testing, Verification and Reliability: Special Issue 10th IEEE International Conference on Software Testing, Verification, and Validation (ICST 2017), Software Testing, Verification and Validation, Vol. 30, Iss. 7–8. (2020). URL: https://onlinelibrary.wiley. com/toc/10991689/2020/30/7-8:
[23] V. Garousi, A. Rainer, P. Lauvås jr, A. Software-testing education: A systematic literature mapping, Journal of Systems and Software, V. 165 (2020). URL: https://www.researchgate.net/ publication/339814384_Software-testing_ education_A_systematic_literature_ mapping.
[24] Androshchuk, A., Yevseiev, S., Melenchuk, V., Lemeshko, O., Lemeshko, V. Improvement of project risk assessment methods of implementation of automated information components of non-commercial organizational and technical systems. EUREKA, Physics and Engineeringthis link is disabled, 2020, 2020(1), pp. 48–55
[25] Oleksandr Laptiev, Savchenko Vitalii, Serhii Yevseiev, Halyna Haidur, Sergii Gakhov, Spartak Hohoniants. The new method for detecting signals of means of covert obtaining information. 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (IEEE ATIT 2020) Conference Proceedings Kyiv, Ukraine, November 25-27. pp.176 –181.
[26] Korchenko, A., Breslavskyi, V., Yevseiev, S., Sievierinov, O., Tkachuk, S. Development of a Method for Constructing Linguistic Standards for Multi-Criteria Assessment of Honeypot Efficiency.Eastern-European Journal of Enterprise Technologiesthis link is disabled, 2021, 1(2(109)), pp. 14–23
[27] Try QA. URL: http://tryqa.com/
[28] N. G. Bardis, N. Doukas, V. Kharchenko, Vl. Sklyar, S. Yaremchuk, Dependable IoT for Human and Industry: Modeling, Architecting, Implementation, in: Approaches and Techniques to Improve IoT Dependability, River Publishers, 2019, pp. 307-328.