DOI https://doi.org/10.36059/978-966-397-107-0/66-94

SCIENTIFIC AND TECHNICAL LEVEL OF SOFTWARE TOOLS

Domnich V. I.

1. Evaluation of the scientific and technical level of software tools

Under the scientific and technical level (STL) of ST one understands a relative characteristic of ST quality, based on a comparison of values of indicators characterizing the scientific level and technical perfection of evaluated ST, with the corresponding baseline values of these indicators. The term STL is also used to summarize the quality of design decisions during the ST development stages.

The value of STL of ST is used in solving the following tasks: 1) conducting feasibility studies at the established stages of developing new ST; 2) determination of the best among developed homogeneous ST; 3) solving the issue of readiness to move to the next stage of ST development; 4) resolving the issue of ST readiness for transmission to the customer; 5) ST certification by quality categories; 6) addressing the need for upgrading or replacing the ST that arises during operation.

Depending on the stage of determination and sources of information, the following types of STL are distinguished: predicted; design; guaranteed; operating[1].

Under the predicted STL os ST one understands the scientific level and technical excellence of pre-design decisions based on the verification of the sufficiency and validity of the data contained in the TOR or in the replacement documents for the creation of the ST, which exceeds the known analogues or requirements of the ST in its characteristics, and the effectiveness of the preparatory action to provide high quality ST.

Under the design STL of ST one understands a characteristic of the scientific level and technical excellence of design decisions, based on the verification of their completeness and compliance with the requirements of the TOR and other regulatory and technical documents.

[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.

Guaranteed STL of ST one understandsa characteristic of ST, based on the comparison of values of indicators characterizing the scientific level, technical implementation of the evaluated ST and obtained in the tests of the ST sample, with the corresponding baseline values of these indicators.

Under operational STL of ST one understands the relative quality of ST, based on a comparison of the values of indicators characterizing the scientific level, the technical perfection of the evaluated ST and obtained during its operation, with the corresponding baseline values of these indicators. When evaluating the STL of ST, quality indicators are used to characterize the scientific level and technical excellence of the evaluated software (hereinafter referred to as the STL indicators). Accordingly, the features that make up the ST STL are included in the overall set of product quality attributes. The STL indicators are set for each of the evaluated ST or groups of homogeneous ST (types of ST). When selecting the indicators of the STL of the evaluated ST (type of ST) they use the nomenclature of indicators recommended for this type of ST. If the initial nomenclature of the quality of the evaluated ST is predefined and defined in the TOR or the document that replaces it, then the STL indicators are selected from that initial nomenclature. There are no generally accepted criteria for the selection of attributes that characterize the STL of ST. Decisions are made by people charged with assessing the STL based on their own experience and understanding of the task, or by specially appointed experts. This decision must be agreed with the officials who issued the Task Assessment and the ST developers.

At the same time with selecting STL indicators, you must select the Quality Indicators Score Estimates. If different rating scales are used in estimating STL, then in order to calculate STL, the values of the indicators should result in a single multidimensional scale. In order to order and simplify the casting procedure for each case, it is necessary to establish the scales and methods used for casting in advance.

Three variants of rating scales are the most convenient to use: 1) scales of baseline values of quality indicators; 2) a single ordinal scale; 3) scales of different orders.

*The first variant*of the rating scale is used when the baseline values of the quality and usage indicators to determine the scale are known. The values of each *I*indicator are determined on the same scale as the base

value. The values of quality indicators to a single scale is given by calculating the relative values of the indicators.

The second variant of rating scale is used mainly for expert methods of determining the values of quality indicators. Within one examination, it is recommended to use one pre-set rating scale in points (for example, a ten-point score).

The third variant of rating scale is also used for expert methods of determining the values of quality indicators.

The objects of control (evaluation) while determining the values of quality indicators depend on the stage of the ST life cycle and the type of STL that is determined. The main object of control in determining the predicted STL is the terms of reference or documents that replace it, design documentation (sketchy, technical or working software projects), ST sample, software tools (software products) that are in operation.

When estimating the projected and projected STL, the main source of information is the expertise of the project documentation, and the method of obtaining the information is estimated.

When evaluating a guaranteed and operational STL, you can use any means and sources to obtain information about ST features.

The validity of the STL estimates depends on the methods and sources for obtaining the information, as well as the quality scoreboards used. Therefore, the scales should be defined in advance in the quality assurance plan or in another document agreed with the unit or the person who issued the task for the evaluation of the STL. ST quality data at the stages of development and testing are collected by the development units, during the period of experimental operation – the developers together with the experts who carry out the experimental operation.

The composition of the registered data and the procedure for their collection are determined by the program and methodology of the experimental operation. After delivery of the ST to the customer (user), the user of the ST and the ST support service of the organization-supplier collect the data for the evaluation of the operational STL. When assessing the STL of a ST, it is necessary to use as much as possible all previously accumulated data on the quality of the ST that is credible. The average weighted arithmetic is most often used as a generalized STL. The calculation of this indicator is based on the nomenclature of indicators.

Evaluation of each of these types of STL has its own specificity, due to the specificity of control objects at different stages of the ST life cycle.

For each control object, indicators should be set to characterize its quality. The direct use of quality indicators set out in the TOR or the specification of ST requirements in the early stages of ST development is almost impossible, as these indicators are geared towards assessing the performance of the ST ready for its intended use. They reflect the consumer properties of this ST, and design decisions need to be monitored from the earliest stages of ST development. A good specification of requirements is a must, but not sufficient condition for a high projected STL. The prognosis of a software STL may not be favorable with a good specification of requirements, but the lack, for example, of the necessary resources (labor and material) to meet these requirements. Therefore, it is not by chance that the three other groups of STL indicators are estimated. The assessment of the operational STL is made on the basis of the analysis and generalization of information about the consumer attributes of the ST, which were manifested during their industrial operation. If the evaluated ST is a batch product, then the quality of all these products installed during operation under different conditions of use should be taken into account when evaluating the operational STL. In this case, it is advisable to use the sociological method of data collection as a source of information.

As a result of the STL calculation, the relationship between the quality indicators, characterizing the scientific and technical excellence of the evaluated ST, and some baseline values of the indicators, taken as the standard of excellence, should be obtained. This ratio characterizes the scientific and technical level of the evaluated ST.

When using the Quality Score Estimation Scale, this ratio is obtained automatically because the scores themselves are pre-ranked.

2. Technical software for the quality management system

2.1 The modern concept of programming technology and its connection with software quality management

Programming technology is a set of methods, ways, techniques, automation tools, technological equipment and regulated order of their application, aimed at the development, production and use of software products in the given conditions and with the specified quality indicators.

The ultimate goal of using any programming technology is to ensure high productivity at all stages of the software life cycle and the required quality level created and maintained (accompanied) by the software. This goal is achieved by improving the methods and technological methods of creation, operation and maintenance of software, their strict regulation and high automation. R-technology is the implementation of multi-circuit and multi-level software design by the method of step-by-step specification of any informal concepts of graphic structures in algebra that provide the «assembly» style of programming; writing algorithms, programs, data and processes in graphical form; friendly interface of all experts involved in the development. This technology is intended for the development of a wide range of ST, including structurally and logically sophisticated software in all areas of computing. In most cases, the values of the quality of software and other software technologies are unknown to developers or suppliers, and are often not guaranteed in the prescribed manner. Thus, from the point of view of quality assurance of the software under development, currently used programming technologies, the very concept of programming technology needs improvement. First of all, this improvement should cover all technological processes. The desire to regulate all technological processes of the ST life cycle urgently requires the introduction of new and refinement of old concepts and definitions in the technology of programming. The very name of this area is outdated. In the software life cycle, programming (coding) is only one of the steps. Other stages (system analysis, design, testing, manufacturing, operation, support) meet their goals and technological (production) processes. They also need regulatory, technical and software tools. For example, the software testing process should be provided with testing programs and techniques, as well as testing tools, measuring values of quality indicators and processing results[2].

The tasks, content, and therefore the name of a particular technology, must be considered in conjunction with the supported process and the requirements of technological readiness. The technological readiness of some SP process means the existence of complete sets of design and technological documentation, as well as the means of technological process (TP) with established technical and economic indicators. According to the

[2] Atans M. and Falb P. L. Optimal management. Translation from English. Ed. Dr. Techn. Sciences prof. Y. Topcheeva. M., "Mechanical Engineering", 1968, 764 p.

technological preparation of the SP process, they call the set of measures that ensure the technological readiness of this process. In the general case, the technological preparation of the SP process consists in the completion of the previous technological process by the development of appropriate design documentation, in its examination, as well as in the preparation of technological documentation and software (software-instrumental) tools for the execution of TP. For example, technological preparation of the stage (process) of programming consists in the following: completion of a technical project (development, algorithm for solving a problem) and registration of the relevant design documentation; examination of a technical project; preparation (selection and/or development) of all technological documents that regulate the programming process (selection and description of programming languages, setting restrictions on their use, choice of programming methods and preparation of appropriate instructions, etc.); preparation of software and hardware for programming (translators, automated tools for debugging programs, static analyzers, etc.).

Technological processes consist of sequentially or sequentially-parallelly performed operations.

The technological documents governing these operations, as well as the software (software-instrumental) tools that support them, are called technological modules (eg, programming language, translator, text editor, documentation system, software test method, text data generator, static analyzer, etc.).

The set of technological modules, mutually linked by a common scientific and technological idea, which regulates and ensures the successful execution of the technological process of SP, forms the technology of this process. Thus, there can be no technology at all, there can only be a technology for a particular production process or a set of processes.

Process technology is an integral technology. For example, the integrated technology of development (creation) of ST prototype should regulate all the processes of creation, from system analysis to testing of ST prototype.

The technologies themselves can be developed in the following ways:

- creation of integrated technologies focused on a specific application environment, type of automation tasks and the entire ST life cycle;

- creation of a bank of problem-oriented technological modules from which it is possible to synthesize technological lines of support of certain technological processes;

- creation of partially integrated technologies with their subsequent development by adding missing technological modules.

Integrated technologies that cover all stages and stages of the software life cycle have not yet been created. Work in this direction is being carried out, in particular, within the framework of the R-technology concept. Creating a bank of problem-oriented technology modules is still going on. The interface module is not unified.

The modules themselves do not have the autonomy of application. Only with the elimination of these shortcomings can one count on the effective integrated application of technological modules.

Most existing technologies are partially integrated. In many respects, this is entirely justified. A single (integrated) technology in the general case may not be of interest to any of the categories of specialists: developers need development technology; manufacturers – manufacturing technology; users – technology of operation. According to these interests, the degree of technology integration should be chosen.

According to the initial concept of programming technologies, integrated technology must meet the following requirements:

- methodologically cover the entire ST life cycle; be flexible, mobile, integrated on the basis of technological lines of different problem orientation;

- provide a significant increase in the productivity of programmers and ST development with the required quality indicators;

- to provide possibility of realization of technological processes on the existing and perspective computer systems;

- to provide automated planning, regulation, execution of works, control over the course of technological process and quality of products;

- contain a set of normative-methodological and legal documents defining as a way of describing technological modules and lines, as well as the procedure of carrying out a technological process and the forms received at each stage of documents;

- to provide, when using computers, purposeful activity of both professional programmers and their teams with a well-defined industrial

organizational structure, as well as specialists of other professional orientation (non-programmers);

- to be easy to learn, to automatically include tools for teaching and learning, as well as recommendations for its implementation, including all levels of education (schools, higher education institutions, training institutes, etc.).

Software attributes are formed at all stages of development and production, as well as stages of operation (use for the intended purpose) and support. Therefore, the SQMS should contain measures that cover all stages of the software life cycle.

The SQMS should fit into the programming technology organically. It is subject to the basic requirements and principles of the concept of programming technology. However, software quality control and assessment systems should be independent of programming technology.

Software quality management (SQM) to the formation of technological lines for the development, production and use of specific software should be started after establishing the affiliation of this ST to a particular classification group.

There are a number of complex, under-researched problems on the way to creating an effective SQMS both at the country level and at the level of the development organizations. Let us look at some of them.

Establishment of the initial nomenclature of software quality indicators. This nomenclature includes those indicators that characterize the main attributes of software as an independent class of products for industrial and technical purposes.

With the help of the initial nomenclature, the properties and indicators characteristic of homogeneous product groups are distinguished, and thus the nomenclature of quality indicators for each of these groups is formed. In this case, it may be necessary to introduce new indicators characterizing the specific attributes of a group of software.

Software products and their individual types have numerous attributes. But the effect of these attributes on the quality of the software is different: some properties are only desirable; in the absence of others, it becomes impossible to use the ST for its intended purpose. Therefore, it is very important to study the correlation of the properties of specific types of software with the quality of these products. As a result of these studies, the weighting parameters of quality indicators for all types of software should be determined.

In order to manage quality, it is necessary to investigate the factors that influence the formation of software quality at different stages of its life cycle. Particular attention should be paid to the stages of development and support of software. It is important to establish not only the factors, but also the whole mechanism of their action. Only in this case can one count on effective software quality management.

Managing any process involves controlling the states of that process. The programming process has been developing as a purely intellectual kind of activity that is weakly controlled from the outside. Methods of quality control by customers and users of software are still poorly developed. Due to the transition to industrial methods of software design and production, these methods need to be intensively developed.

The development of software should be completed with comprehensive testing, during which it establishes the actual achieved quality indicators and the suitability of the software for its intended use.

Products that do not meet the requirements that are put forward to it, does not have the full set of consumer attributes, or returned to refinement, or discontinued production. Thus, software testing is the most important technological process in the system of quality control and management of these products.

However, so far, little attention has been paid to existing processes in existing programming technologies. The test process itself is usually replaced by less efficient processes of examination, inspection, test and test of the performance of programs with the power of control examples. Improvement of methods of optimization of quality indicators, automation of these methods deserve special attention. In the long term, these methods should be brought to a level that ensures the creation of software with predefined properties.

When defining the relationship between quality problems and programming technology problems, it should be borne in mind that quality and technology are completely different categories. Quality is an aggregate property of products, technology in a materialized form is a certain set of technological modules, each of which defines a process and, when used, should contribute to the achievement of the required level of quality of the created ST (relative to a given property) and/or increase the productivity of developers.

Some properties of software are largely dependent on the technology used in programming, others are provided in a constructive, algorithmic way and depend little on the technology. Most software attributes are provided both constructively and technologically.

2.2 Automated software creation and support environment

Automated support tools (AT) are required for each stage of the software life cycle. Together, automated tools of system analysis (ATSA), design and coding (ATDC), debugging and testing (ATDT), production (ATP), and support (ATS) form an automated environment for creating and support (AECS) of software that are integral part of integrated programming technology.

Automated software creation and support environment is a set of language, software, technical, organizational and methodological tools and databases that provide support for technological processes at all stages of the software life cycle.

Realizing the modular principle of programming technology formation, it is advisable to create object-oriented complexes of technological modules (TM), intended for automation of technological processes of creation and maintenance of certain subclasses (groups, types) of software products. Some TM may be suitable for use in the creation and maintenance of any kind of software, that is, universal. Universal TM form a separate set or group of sets[3].

The Program Modernization Analyzer is designed to automate the analysis of the source text of a program on its information and logical structure and perform the following basic functions:

- syntax analysis of the source text of the program in a high-level language, given by context-free grammar;
- construction of program control graphs and module hierarchy;
- combination of two versions of the program for finding added, deleted and common fragments of program texts;
- analysis of the impact of added fragments in the upgraded program and deleted fragments from the original text of the program on the common (saved) fragments caused by the change of information relations and the control graph;

[3] Tsypkin Ya. Z. Fundamentals of automatic systems. Main Editing Physical and Mathematical Literature Publishing "Science", M., 1977, 56 p.

- formation of a combined listing of two versions of the program with the statement of the operators that need to be tested for efficiency (relative to the final results of the program);
- analysis of the control graph to identify the input, output and hanging vertices, unconditional loops;
- determining the values of structural quality indicators of the initial and upgraded versions of the program.

3. Typical Requirements for Automated Debugging and Testing Programs

Software products are extremely diverse in their nomenclature, conditions and areas of application, complexity, modes of operation, which causes a variety of debugging methods and software testing.

The software is a constituent part of the *automated debugging and testing tools* programs (ADTT), so we use the nomenclature of software quality indicators when formulating the requirements. The experience of automating debugging and testing programs allows you to formulate the following typical requirements for the ADTT programs.

Multifunctionality. Debugging and testing of software is a long and time consuming process associated with the need to simulate input, documentation (event logging), processing of results, analysis of completeness of checks, accounting of resources used, etc. In addition, ADTT is a kernel element of the automated software creation and support environment and is often used throughout the ST life cycle as a means of analyzing the results of operation, modification, training of service personnel, and the like. All these functions should be implemented in economically justifiable terms by the ADTT.

However, it should be remembered that an insufficiently substantiated desire to implement all the functions can lead to negative consequences.

Openness and Modification. Despite the desire to automate software debugging and test processes, experience in creating ADTT systems is still poor.

It is often difficult to determine in advance all those functions that need to be assigned to the ADTT, so consistent system development may be most appropriate. But this requires that the system has the properties of openness and modification.

Adequacy of the simulated environment to the real environment of functioning of the tested ST. The ADTT system should provide simulation of incoming messages of the tested ST, adequate to the incoming messages in the real operating environment. Moreover, the format of incoming messages, their time sequence and distribution on the communication channels should be equivalent to real formats, sequences and distributions, regardless of the method of modeling.

High reliability of functioning. The ADTT system is a kind of measuring tool used in ST debugging and testing. It is known that the accuracy and reliability of measuring instruments have very high requirements, which are usually no less than an order of magnitude greater than the requirements for the instruments, devices and systems controlled by these measuring instruments. A similar requirement is relevant for the distribution of ADTT. ADTT systems that do not have high reliability, or will not be used at all (due to lack of trust in them), or will be used, causing more problems than solutions.

The highest requirements must be advanced to the accuracy and infallibility of the ADTT. The reactivity and recovery rates are less significant

In the case of low ADTT resistance to distortive actions, signs of ADTT failure due to these actions should be clearly identified in order to invalidate the results of the respective experiments.

High ergonomics. Using the ADTT system as a tool for debugging and testing programs, the programmer should be able to communicate extensively with the ADTT in order to quickly analyze the results of debugging (testing) and changing experimental conditions.

The ADTT system should be easy to prepare for operation and maintenance, demonstrably sufficient and able to be analyzed. The system should be dialog.

Reproducibility of results. To localize and correct bugs in the program, it is necessary that the system of ADTT allows to repeat any experiments for an unlimited number of times with the exact observance of the same conditions.

Uniformity. Unified elements (technological modules) should be used to the fullest extent possible when creating a system of ADTT. Fulfillment of this requirement will, firstly, increase the reliability of functioning (unified components are more carefully tested and therefore

error-free); second, the assembly and development of new (flexible) technological lines.

Technology of assembly and application. When designing the components of the ADTT, one must stick to the unified interface design requirements that facilitate the assembly of process lines. The technology of application of the ADTT must be consistent with the concept of programming technology.

Information secureness. ST automated debugging and testing tools should be protected against unauthorized access because, first, with access opened, users may inadvertently corrupt information from each other and, second, certain information may need to be restricted for commercial or other reasons.

Documentation support. During the debugging and testing of the software, it is usually necessary to make numerous changes to the project documentation. To reduce labor costs, this technological operation should be automated in both graphic and textual information. When designing, there are different versions of solving problems. Practice shows that these versions should be stored at least until the project is completed. Designing and storing versions can also be a function of documentation support tools.

Analysis of changes. When making changes to the program, the task of promptly establishing the consequences of these changes. testing has to be repeated when numerous changes occur. Change analysis tools should identify program elements affected by the changes and offer an optimal test plan.

Methodological unity. The ADTT complex should be built on a single, pre-formulated, methodological basis consistent with the foreseeable programming technologies. Otherwise, there will be additional difficulties when using it.

Flexibility. The ADTT components designed to debug and test one program must be suitable for use in other similar tasks.

Easy to learn. When using ADTT, users should not have difficulties that cause them to abandon the idea of use.

4. Software Testing

The object of the test may be either the product itself or its model. Modeling of the product is performed when it is impossible to directly test it either for safety reasons or because of the excessive complexity and

costly testing caused by, for example, the unacceptable consumption of object resources.

None of these factors is a barrier to direct software testing. Therefore, the simulation of SP as a test object is unnecessary. With this in mind, the test of software products should be understood as an experimental determination of the quantitative and / or qualitative characteristics of the properties of products when operating in a real environment or modeling the environment.

The purpose of the test is to experimentally determine the actual (achieved) characteristics of the properties of the test SP.

These characteristics can be both quantitative and qualitative. It is important that they can be used to conclude that the SP is suitable for its intended use. If the conclusion is negative, then the sample SP is returned for revision.

This overrides the access of poor quality products to the user. Directly during testing, the quality of SP may not change, since bug localization is not the purpose of the test. However, some defects in programs and documentation may be eliminated during the test.

The test is the final stage of development. It is preceded by a stage of static and dynamic debugging of programs. The main method of dynamic debugging is testing. In a narrow sense, the purpose of testing is to detect errors, but the purpose of debugging is not only to detect, but also to eliminate errors. However, you cannot limit with only the debugging of the program, if you are sure that all errors in it are eliminated. The goals for debugging and testing are different.

A fully debugged program may not have certain consumer attributes and, thus, be unusable for its intended purpose. There can be no alternative to testing and checking the validity of the program in the control example, because the program, working in the conditions of the control example, may not work in other conditions of use. Attempts to cover the control example all the expected conditions of operation are reduced to the testing. Under testing programs one understands the establishment of compliance of the program with the specified requirements and program documents. This definition is based on the assumption that the terms of reference for the program development define all the requirements (characteristics) that ensure that the program is fit for its intended purpose. But such a requirement is rarely met in practice.

In some cases, especially in automated systems, the TOR on the ST is either not written at all, or has only those features that rely on the software, without specifying requirements for other consumer attributes. In the absence of TOR for ST development or a complete and reasonable list of requirements for the characteristics of the ST being developed, the task of testing the ST becomes indeterminate and non-constructive. What does it mean to set a program to meet a set requirement if it is not formally set?

What is the benefit of establishing such compliance if these requirements are deliberately «truncated» and do not reflect the kernel consumer attributes of the program? It will not be useful to the user if the program is working poorly, but it does not explicitly contradict the requirements of the TOR. In the presence of the necessary characteristics of the basic characteristics of the consumer's SP attributes, the definition of the term «test» for the purpose of testing almost coincide. However, in this case, too, the first definition is more constructive, since it formulates not only the purpose but also the main test method – the validation of a SP that functions in a real or simulated, but close to real, environment. In the literature, including software standards for software, the concept of «testing» is often equated with the concept of «examining».

For example, the following definition of examining is given: «... the process of active analysis of software to identify differences between the actual and required ST standards (i.e, errors in programs) and to evaluate the characteristics of ST elements.» This definition combines the two definitions of the term «test» with the only difference that, when adopted (see definition), the search and localization of errors are not explicitly stated objectives of the test.

Given the above considerations, the term «examining» used in foreign literature will be interpreted as testing by the method of testing.

One of the features of software is the comparative simplicity and controllability of SP replication, so the focus on quality control and evaluation should be given to prototype products. Quality control of homogeneous products in their batch production is to verify the identity of the new record on the data carrier of the reference record.

The experience of SP development shows that the process of testing it is time-consuming and expensive. Moreover, the bulk of the cost is not spent on testing, but on their preparation and processing of results.

The labor costs of creating a complex of automated debugging support and software testing are often found to be of the same order as the labor costs of creating the most tried and tested means.

At the same time, the costs of debugging and testing software justify themselves, especially when creating mass-produced software tools and automated software systems. The costs will be reduced as the specialized (by problems) banks establish unified technological debugging modules and test them and the means of their assembly into technological lines.

The duration of the test depends on the type, configuration of the software, as well as the purpose and degree of automation of the process.

When testing operating systems, it ranges from one to six months. Compound software complexes after integration can be tested for a longer time.

The purpose of the software test is usually detailed depending on the type of test. The state system of standards provides for about forty types of testing of industrial products. Types of test are classified according to the following characteristics: time, venue, departmental level, type of action, duration, immediate purpose of the test. Due to the specific nature of the software and the process of its creation, most of these species during testing or until they are not used, or devoid of practical meaning (e.g, resource, mechanical, electrical, thermal tests).

The main types of software testing are preliminary, acceptance and operational tests, including experimental operation.

Depending on the venue, there are bench and proving ground testing. The bench means a set of technical devices and mathematical models that provide an automatic simulation of the operating environment; input of input data and distortive actions; registration of information on the functioning of the ST, as well as management of the process and object of the test[4].

If the principle of bench testing is based on the principle of modeling, then the appropriate test benches are called modeling.

In the simple case, bench tests use a computer and pre-prepared tests.

The test proving ground is called a place intended for testing in conditions close to the operating conditions and provided with the necessary test facilities.

[4] Ivashchenko N.N. Automatic regulation. Theory and elements of systems. Textbook for universities. Ed. 4th, rework. and ext. M.: Mechanical Engineering, 1978. 236 p.

Proving ground testing consists of systems performed in real time. In polygon conditions, they usually combine full-scale tests using real objects, automated systems, and modeling of some objects and their processes.

Recently, in some development organizations, test polygons are a collection of test benches specialized in the profile of this organization.

Depending on the test, the developers distinguish between dependent and independent tests.

In the dependent tests, major ST operations (preparation to work, preparation and input of initial data, registration and analysis of results) are performed by program developers. The evaluation of the test results is done by the commission with the active participation of the developers.

Independent testing is conducted by special units that are not responsible for program development and are not directly subordinate to the development executives.

The advantages of independent testing are the following:

- direct developers, knowing that their work will be tested by other specialists, try to execute it better;
- work managers are more concerned about the quality of the software being developed, plan the necessary resources for debugging programs, trying to avoid complications in quality control by independent skilled professionals;
- employees of testing units accumulate experience in performing specific testing works, improve methods of carrying out these works and their qualification, which ultimately increases the reliability of results and reduces the likelihood of omission of poor quality products.

Independent test units should be created from the beginning of the development of complex ST. They must exercise control functions at all stages of their creation.

By the time the acceptance tests begin, these units must form process lines, and develop test programs and techniques. Thus, the functions of the independent testing units are the same as those of the quality control services.

Practice confirms the high efficiency of independent testing. However, when deciding on the benefits of this type of quality control, other factors, including the additional costs and complexity of organizing the interaction between developers and testers, need to be carefully considered.

5. Flow scheme of the test

To increase the efficiency of the test, its acceleration and reduction of cost, it is necessary to develop scientifically grounded methods, tools and techniques that allow to overcome the disadvantages of the approach to testing as a kind of heuristics, underestimation of its role in ensuring the required level of quality of software, replacement of tests by procedures of checking the validity on check example, etc.

This goal can only be achieved by developing a technological test scheme that provides:

- knowledge of the purpose of the tested software, conditions of its functioning and requirements for it by users;
- automation of all the most time-consuming processes and, above all, modeling of the operating environment, including distortive actions;
- a clear representation of the purpose and sequence of the test;
- purposefulness and irregularity of the test, which exclude or minimize the repetition of homogeneous procedures under the same operating conditions of the tested software;
- systematic monitoring of progress, regular maintenance of the protocol and the test log;
- clear, consistent definition and implementation of the test plan;
- a clear comparison of the available resources with the estimated test volume;
- можливість забезпечення, а також об'єктивної кількісної оцінки повноти і достовірностірезультатів випробування на усіх етапах.

Any type of test should be preceded by careful preparation. The preparation of ST testing includes the following measures:

- drawing up and approval of the test schedule;
- development, acquisition, testing and certification of software and hardware used in the tests;
- analysis of the suitability of the test facilities used during the preliminary tests for acceptance tests;
- analysis of the suitability of the accumulated data on the quality of the software for use in the final determination of the quality indicators of the tested ST;
- inspection and coordination with the representative of the Customer of the design documentation for the software presented during the tests;
- development, harmonization and approval of test programs and techniques;

- certification of specialists for admission to testing;
- acceptance of the ST test prototype on the data carrier and documentation;
- conducting measures aimed at ensuring the reliability of the tests.

Particular emphasis should be placed on the need for the early development and testing of all software that will be used in the tests. It should be borne in mind that the level of accuracy and reliability of the measuring equipment used in the testing of any object should be significantly higher than the corresponding indicators of the test object.

Therefore, the real characteristics of software and test tools must be established in advance, and their eligibility to be agreed between the developers, testers and customers of the software.

Disregard for this rule distrusts the test results and, as a consequence, extends the test time.

The complexity of software and testing tools, the requirements for their perfection, and therefore, the cost of resources for their development depend directly on the relevant indicators of the software being tested.

The volume of test software, expressed in machine commands, can reach the volume tested by their programs. Therefore, the development of software designed to test a particularly complex software, should begin simultaneously with the development of prototypes.

It is appropriate for enterprises and organizations specializing in the development or testing of software to create unified testing programs. Each test program must have a passport containing its characteristics.

Based on the above, we can determine the following five stages of testing: examination of the projected software, analysis of project documentation, determination of the most important subsystems, functions and paths of projected software to be tested, analysis of software quality indicators and methods of determining their quantitative values, development of programs and test methods, development (development) of test software, test libraries and databases (if they are required), direct testing, analysis of results, decision making.

Each stage of the technological scheme of tests depends on the previous ones. The execution of the staged work may partially intersect. Depending on the specifics, conditions of use, quality requirements of the investigated ST, tests can be carried out either by testing, or by statistical modeling of the operating environment, or on the basis of full-scale and mixed experiments.

It is often helpful to use all of these methods. The values of some quality indicators can be obtained expertly, with the available time and material resources, in an effort to provide the necessary completeness and reliability of the test results. The task is complex and controversial, so it is of particular importance to assess the completeness and reliability of test results in the reporting documents. The user, when purchasing the ST, should receive full information about the actual characteristics of the software. This will allow him to properly organize his use.

6. Planning and evaluation of test completion

The test plan should be oriented to ensure comprehensive validation of the software and maximize the reliability of the results obtained using the limited resources allocated to the test. The following approaches to solving this problem are possible: 1) analyze the entire range of input data. Based on the analysis, a set of data combinations (test datasets) is prepared in advance, covering the most characteristic subsets of the input data. The program is considered as a black box. Testing is limited to the sequential introduction of test datasets and analysis of the results obtained; 2) analyze the many situations that may arise during the operation of the ST. Choose the most typical situations. Each is expressed through a test set of input data. Further, the essence of testing and analysis of results is reduced to approach 1), using a graph model to analyze the microstructure of ST. One selects multiple paths that completely cover the ST scheme, and such a sequence of test sets of input data, the execution of which will take place on a dedicated path. Test organization is similar to approaches 1) and 2); ST is tested in a real operating environment; ST is tested in a statistically simulated operating environment adequate to the real environment.

None of these approaches are universal. Each of them has its own advantages and disadvantages, which are differently manifested depending on the specifics of the tested software. The most reliable results are obtained when tested in a real operating environment. But such tests are rarely possible. Therefore, combinations of all kinds are used in practice.

A typical example of such a combination would be a mixed method where the software environment is simulated and the reliability of the results is verified by comparing the results obtained with the real-world software operation. Tests, like any other type of quality control and evaluation, are only possible in this case if there are methods, tools or

procedures in place to determine with reasonable accuracy the accuracy of the results of the data conversion using controlled ST. Otherwise, this ST is formally uncontrolled.

Uncontrolled or partially controlled ST are considered, in particular, ST designed to find the unknown result and which generate too much data to be validated; ST that controls specification requirements other than those provided for the assignment; ST whose requirements have not been predefined in due course. When planning tests, software control issues should be analyzed with particular care. The signs by which ST performance results are classified as correct (incorrect), as well as the methods for determining them, must be agreed in advance between developers, customers, users and testers.

The analysis shows that absolute verification of the ST is not possible in any of the considered approaches. Therefore, when planning the tests it is necessary to analyze the structures of the tested programs and the input data in advance. In particular, you should set the paths of the program diagram that are most likely to be used when converting data. The technique of solving the problem of test planning includes the following steps: finding all the ways of implementation; selection of a minimum subset of paths that allow verification of all sections of the program; development of tests to check the selected paths.

It should be noted that as a result, the solutions receive not one subset of paths, but some set of such subsets. Analyzing these sets by the criteria of the minimum time of their implementation on the computer, the choice of the most probable paths, the absence of these sets of incompatible paths (the methods considered this disadvantage), choose the most acceptable set. To create the input test data for each dedicated implementation path, they make special tables. The tables represent only the conditional statements that belong to this path and the operators in which the control variables are calculated. As a result of the analysis of the prescriptions that satisfy the conditional statements, they produce these tests.

The considered method of planning at the stage of stand-alone statistical testing of SP modules can significantly reduce the material and time costs of program testing. The orientation to one or another test approach depends on the type of ST being tested.

For real-time systems and other systems whose state at some point in time depends on the prehistory and the transformed data set, approach 1 is

most appropriate, but using initial data modeling methods (approach 5). The essence is not so much the efficiency of the method, but the practical impracticability of the early preparation of test data for each clock interval of the program and many different conditions of operation.

In general, the design and organization of testing should seek a compromise solution that takes into account two conflicting requirements: ensuring maximum reliability of the generalized SP quality assessment and performing the test in a limited time using limited resources. Considering that an absolute evaluation of the ST is impossible, the task of planning the test in these conditions reduces to finding solutions that maximize the impact with limited material and time resources.

The maximum impact is the maximum attainable completeness, depth and reliability of estimates.

The highest efficiency of ST quality control is achieved when the control itself is carried out at all stages of the software life cycle, and preparation for testing begins from the moment of software development. There are three stages of the test: preparatory; direct testing; final (preparation of reporting materials).

The tasks of these stages are obvious. Let us focus more on the tasks of the preparatory stage. This stage is the longest and requires the highest labor costs. Its main tasks are: test planning, development of technological scheme of tests and test facilities; development of programs and test methods; accumulation of preliminary statistics that characterize the ST. Purposeful and accurate organization of work on the accumulation of statistical data can significantly improve the reliability of the quality assessment of software, eliminate duplicate checks and reduce the time of testing and cost of material resources.

However, in some cases, due to poor organization of work, test results at the program debugging and pre-testing stages are not recorded, so they cannot be used for the final evaluation of the program's quality. Between the selected stages of the ST test are direct and inverse relationships, similar to the links between the stages of the ST life cycle. This means that the completion of the works of the final stage may reveal the need to return to the stage of direct testing (or even to the preparatory stage) to clarify individual characteristics.

Clear planning of all test work is the basis for the success of ST evaluation and quality assurance. Testing plan preparation should be

preceded by analysis of TOR for ST development, structural and functional schemes, modes of operation, dependencies between program modules, schedules for development and debugging of software components, results of their quality control in the early stages of development. the course of the test.

As a result of this analysis, it is necessary to develop and substantiate a general testing strategy, and on its basis – a set of documents on the organization of tests, which should contain the answers to the following questions: 1) the task of testing at each phase, the sequence of development of phases; 2) the use of special testing facilities; 3) the amount of machine time required at each test phase; 4) configuration of general hardware and software; 5) evaluated properties, evaluation criteria, methods for obtaining them; 6) procedures for monitoring the registration, collection, processing and synthesis of test results; 8) conditions (criteria) of beginning and completion of each phase of testing.

The program and methodology of *acceptance* tests are developed by the customer with the participation of developers. The test program and test procedure can be formulated as a single document or as two separate but clearly agreed documents. If the test program is a separate document, it shall contain the following sections: test object, test purpose, general conditions, test volume, conditions and procedure for the test; the composition of the hardware and software needed for the test; reporting; applications.

It is difficult to overestimate the value of carefully tested programs and test methods. Without these documents, the tests turn into a formal, futile procedure that consumes considerable resources without due diligence due to the mismatch of the test attributes.

Comparing the traditional program structure and test methods with the structure of the test plan, it is easy to establish a common similarity.

Consider those sections of the test plan that are significant but not explicitly reflected in the structure of the test programs and techniques. Such sections include non-verifiable characteristics; test principles; criteria of suitability/unsuitability of test items; criteria for suspension and resumption of tests; tasks of staffing and training; risks and contingencies; documentation.

The section «Non-testable characteristics» one should identify all the features of the software and their combinations that are not tested during the test and give a justification for that. For example, in preliminary ST

tests, it may be decided not to evaluate the ergonomic parameters, given that they will be evaluated in experimental operation by a separate method. This decision is recorded in this section. In the «Test Principles» section, for each significant group of characteristics or combination of characteristics, a principle or method should be specified, the implementation of which ensures a complete and adequate verification of all these characteristics. The main types of work, techniques and tools that should be used in the verification of this group of characteristics, test completion criteria for testing / testing (test item readiness, test resources and timing) should be listed. clear criteria for determining whether or not the test has passed the element. suspension of test work, list the test work that must be repeated after the test is resumed. The section «Criteria of suitability/unsuitability of testitems» indicates clear criteria for determining whether or not the element has passed these tests. In the section «Criteria for suspension and resumption of test «, the criteria for complete or partial suspension of test works should be stated, and a list of test works must be repeated, which must be repeated after the test is resumed. In the section «Tasks of staffing and training» lists the measures for the recruitment of test teams, requirements for qualification, measures for professional development and gaining the necessary experience. The «Risks and Contingencies» section defines the greatest possible risk in the test plan (for example, the risk of untimely completion of the test or the inaccuracy of their results), and approximately estimates the unanticipated cost for some adverse test cases.

Having analyzed the contents of the selected sections, we can conclude that it is advisable to include the information contained in these sections in the ST programs and test methods. Such inclusion will help increase the information content of these documents and streamline the process of testing. It is necessary to spend considerable labor and material resources for conducting software tests. The timing of the tests is always limited. Therefore, the testers are always tasked with finding ways to minimize the costs of material, labor and time resources to achieve the purpose of the test. To accomplish this task, it is necessary to establish test completion criteria, which can serve as a basis for deciding whether to complete the test.

Test completion of technical devices (systems) is usually done on the basis of an analysis of the completeness and reliability of the verification

of all the characteristics specified in the TOR for the development of the device. If necessary, they check the conformity of the structure. When evaluating the level of completeness of software tests and the reliability of the results obtained, serious complications often arise.

Note the following: 1) most software are unique and either have no analogues to compare characteristics or have analogues whose characteristics are unknown; 2) the absence of generally accepted indicators, as well as methods of calculating the necessary and actual values, leads to the fact that in the TOR for the development of ST requirements for the characteristics of the ST are either actually absent (in quantitative terms), or are not complete.

Not every error can be quickly identified, so it is recommended that you document all non-standard events that affect the test and require further analysis, as reports. The following structure of this report is recommended: test incident report identifier, annotation, incident description, incident impact on the further course of the test. The last two sections are basic.

The description of the incident should include the following elements: input, expected and actual results, deviation from the norm, date and time of the test, step of the test procedure, operating environment, results of attempts to repeat the conditions of the experiment, testers, observers-registrars. In the section «Impact» one should indicate (if known) the possible actions of a registered incident on the course of the test, changing the conditions of the test or test procedures. The registration of deviations from the specified modes of operation of the ST (incidents) and detected errors during the tests gives a one-sided characteristic of the tested software and the test process itself. As the purpose of testing is to determine the quantitative and qualitative characteristics of the properties of the ST under test, therefore, in the presence of clearly formulated and comprehensive ST requirements, the main criterion of test completion is the fact of establishing the conformity (inconsistency) of the actual characteristics of the ST specified in the TOR. However, in some cases, the requirements for the ST are either not formally defined or cannot be considered sufficient.

At the same time, it is necessary to conclude at a certain stage about the degree of validation of the ST and the expediency of termination of tests.

7. Benches for debugging and testing programs

The idea of simulation is the basis for the creation of complex simulation test benches used for debugging and testing complex control systems in real time. processing of simulation results functionally combined on the basis of the tested software complex. Integrated simulation-testing bench (ISTB) is a set of means of the system investigated and their models, the model of the environment and the programs of processing of the simulation results, functionally integrated on the basis of the tested software complex.

Complex simulation and testing benches are used in polygon testing of complex systems. Testing the performance and performance evaluation of such systems in real-world conditions are often impossible for technical reasons or because of the high cost of experimentation. Therefore, the idea to create a model of the means of the test system and the model of the environment and to combine them on the basis of a software complex arose.

Functional integration of models and programs is achieved by reconciling simulated models of system tools and the external flow of data about the managed process with communication channels and timing diagrams of programs. The general idea behind the creation of ISTB is based on the fact that for testing (research) ST, implemented directly on the computer control, it is necessary to simulate the controlled process and simulate the entry into the computer information about this process.

The ST under test is «indifferent» to direct sources of information. It is only important that all information is distributed on real physical channels of the computer and time intervals, and also corresponds to the set (expected) range of environmental conditions. Pairing models with real system assets is necessary to evaluate the simulation results by comparing them with real data. Using ISTB directly from the ST itself, not its model, allows you to obtain more reliable results in the simulation and avoid large additional labor costs for software model development. ISTB is created on the basis of a computer system and a set of programs (software), intended to convert the input information into the output control information.

The control computer, and the software package implemented on it, constitute the controlling object of the system. The input to the controlling object can come from either the actual system elements or their models.

Such combinations are also possible when some of the information comes from the real elements of the system and some of it is simulated.

According to its purpose, ISTB should provide: simulation of the flow of applications for system maintenance; simulation of the issuance by the objects of the system of functional and one-time signals about the flow of applications and the state of the controlled process; imposing random interference on simulated signals; synchronization of simulated input (in the computer) information with the timing diagram of the functioning of the test system; required reliability of simulation results; repeated reproduction of input conditions under which errors in the output parameters of the system exceed the acceptable limits; statistical processing of simulation results; real-time work. To create ISTB, in addition to the main computer on which the tested ST is implemented, use a computer of approximately the same productivity to implement the complex The first computer (OS) is usually called technological, the second – instrumental. Instrumental computers and software form ISTB.

Such ISTB is a cross system (CROSS-ISTB). The simulated on an instrumental computer data is transmitted to the technological computer, where it is processed as real data. Automated technological complex (ATC) consists of elements of the following types: controlled technological unit (CTU), automated process control system (APCS), information sensors (IS) on the state of the controlled process. The processing object (PO) enters the input of the ATC, the output – the processing result (PR).

If we stop access to information in the computer from real physical objects of the ATC, and instead enter adequate information simulated by ISTB on the instrumental computer, then the process of software functioning of the PCS will be adequate to the real one. The CTU operator can be involved in both modes. The modeling subsystem includes: a processing request model (PRM), a processing object model (POM); models of information sensors (MIS); interference simulator (IS); model of managed technological unit (MTA). The application model simulates the flow of applications for processing, based on planned and production considerations. According to a given priority or a random act, a serviceable PO is selected from the set of POs simulated by the RM and its characteristics. Or information sensor models are information models of specific types of information sensors used in an ATC control system. They

simulate the issuance of current coordinates that characterize the state of the technological process. The model of the controlled technological unit (eg, rolling mill) simulates the controlled technological process (eg, rolling of steel) with the release of relevant information about this process. The impedance simulator, according to the given probabilistic characteristics, simulates the effect of random factors on the elements of the simulated system and the controlled process. Thus, the simulation subsystem, simulating the technological process in a controlled unit, provides the reproduction of the input information flow in the control computer, adequate to this flow in the real conditions of operation of the ATC. The simulated input information flow is to the input of the tested ACS software and initiates its operation, the result of which is the output information flow that is issued to the CTU or its model. A closed control circuit, adequate to the control circuit in a real ATC, is formed. The main components of the test result analysis subsystem are: program of sampling of results of transformation of input data, programs of formation of standard values for the analysis of correctness of results, program of comparison of actual results with standard ones and evaluation of their acceptability (correctness). The Event Logging subsystem provides documentation of the progress of the test and the recording of all those characteristics that may be useful for determining the values of the quality of the ST under test and for evaluating the efficiency and status of the test process itself. The planning and control subsystem, based on the analysis of the state of the tests, the results obtained, the tested paths of the scheme of the test ST and the tasks coming from the test programmers, plans the experiments and prepares the corresponding initial data for the simulation subsystem. The same subsystem relies on coordination (initialization) of all ISTB elements. ISTB benefits are obvious. Its use allows to carry out a complex combination of objects of the tested system and to check the principles of control long before the creation of all elements of the system (the element of the system, which is not completed, is replaced by a model).

The application of modeling allows to diversify the test conditions and save material resources. Complex test simulation stands can be used not only for testing programs, but also for investigating the interaction of all elements of the system. The combination of the actual means of the test system with their models allows to diversify the test conditions and conduct semi-natural experiments. You can, for example, test the work of a

technological unit, that is automated, by modeling the behavior of the processing object or, conversely, simulate the operation of the processing unit when working with a real processing object. Such variations allow, on the one hand, to check the adequacy of the models to their originals and thus to make sure that the results of the statistical tests are accurate and, on the other hand, use ISTB at the earliest stages of the development of the software sample to select and approve the best design decisions.

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. Atans M. and Falb P.L. Optimal management. Translation from English. Ed. Dr. Techn. Sciences prof. Y. Topcheeva. M., "Mechanical Engineering", 1968, 764 p.

3. Tsypkin Ya.Z. Fundamentals of automatic systems. Main Editing Physical and Mathematical Literature Publishing "Science", Moscow, 1977, 56 p.

4. Ivashchenko N.N. Automatic regulation. Theory and elements of systems. Textbook for universities. Ed. 4th, rework. and ext. M: Mechanical Engineering, 1978. 236 p.

Information about the author:
Domnich V. I.
Candidate of Technical Sciences, Professor,
Head at the Department of Automated Process Control
of the V. I. Vernadsky Taurida National University