V.I. VERNADSKY TAURIDA NATIONAL UNIVERSITY

SOFTWARE PRODUCTION AND GAME MODELING METHODS

Collective monograph

Lviv-Toruń
Liha-Pres
2019

Reviewers:

Dr inż. ***Michał Sójka****, Dean of the Faculty of Mechanical Engineering of Cuiavian University in Wloclawek (Republic of Poland);*

Dr ***Zbigniew Brenda****, Director of Logistics and Technology Institute of Cuiavian University in Wloclawek (Republic of Poland);*

Prof. dr hab. ***Ryszard Strzelecki****, Politechnika Gdańska / Gdansk University of Technology (Republic of Poland).*

Software production and game modeling methods : collective monograph / V. B. Kyselov, V. I. Domnich, M. H. Medvediev, O. M. Muliava. – Lviv-Toruń : Liha-Pres, 2019. – 180 s.

ISBN 978-966-397-107-0

Liha-Pres is an international publishing house which belongs to the category „C" according to the classification of Research School for Socio-Economic and Natural Sciences of the Environment (SENSE) [isn: 3943, 1705, 1704, 1703, 1702, 1701; prefixMetCode: 978966397]. Official website – www.sense.nl.

The issues of creation of software production that meets modern requirements and standards, indicators and methods of assessment of its quality level and its life cycle are considered. Separately, the game methods of mathematical modeling, analysis, and forecasting of decisions in the economy and business for support and decision-making are considered.

ISBN 978-966-397-107-0

CONTENTS

INTRODUCTION

The use of high-tech technology and software environments has made it possible to accumulate a database, create a knowledge base, including through the use of artificial neural networks, which in uncertainty makes it possible to make decisions that minimize energy costs, take into account the state of commodity markets and, ultimately, maximize profits.

The theoretical basis and practical tools for analyzing and forecasting decisions in economics and business are the economic-mathematical models and the calculations that follow them. Moreover, the main difficulties, as a rule, lie not in the execution of calculations, but in the construction of the models themselves, adequate to the real situation.

The paper deals with issues related to modeling and decision making in conflict situations. The mathematical theory of conflict is game theory. Classic examples of conflict situations are buyer-seller, arbitration disputes, auctions, elections, etc. The parties of the conflict tend to pursue different goals, and the outcome of any decision of each of them depends on the decisions made by the other participants. More difficult situations arise when there are associations or coalitions of participants.

The simplified formalized model of the conflict is called *a game.* Interested parties are called *players.* The main task of the game is to determine *the optimal strategies* of players to achieve their goals. *A strategy* is a set of rules (or a program) that determine which action (move) one has to take for each game implementation.

In game theory, it is assumed that each player knows his/her own winning function and the set of strategies at his disposal, as well as the other players' winning functions and strategies, and according to this information he/she organizes his/her behavior.

The theory of games was first systematically outlined by J. von Neumann and O. Morgenstern in 1944 in the monograph “Theory of Games and Economic Behavior», although some results were published in the 1920s. Since that time, considerable interest in game theory begins to emerge, first in the military field and then in other areas of practice.

DOI https://doi.org/10.36059/978-966-397-107-0/5-35

SOFTWARE TOOLS

Kyselov V. B.

1. Engineering interpretation of programming

Engineering (industrial) methods, although are not universal, are now widespread in programming. They require a revision of the traditional concepts and the development of new ones.

Let us consider, first of all, the evolution of the concept «Program». Any computer while solving a specific problem works on a specific program. In general, a computer program (P) is a record of an algorithm for solving a problem in the form of a sequence of commands or operators in a strictly formalized language accessible for the computer. This definition of the program is to some extent abstract and not sufficiently specific. It is used when we are not interested in the specific form of the program, the language of presentation, the degree of completion[1].

The program can be recorded on ordinary sheets of paper, on special forms, on data carriers. It may be debugged or undebugged, ready for its intended use, or may require some conversion before use. To make the idea of the program more specific, epithets are added to the name of the program, such as debugged (undebugged), object, controlling, etc. The abstractness of the term «Program» creates great difficulty in communicating of professionals, and in some cases is unacceptable. Therefore, there is a need for additional terms and definitions.

Under, *the software tool* (ST) one understands a program or a set of programs on data carriers with program documentation developed in accordance with standards and other regulatory documents and suitable for its intended purpose. The definition of the software emphasizes the completeness of the product (the availability of software documentation) and the readiness to use it directly for its intended purpose to solve a specific problem on the computer (recording on a data carrier). But the software may not be a commodity product, especially a product for production and technical purpose. It may only be intended for use by the

[1] Antipensky V.E., Bilousko V.S., Chujdan T.I. Computing Machines and Programming: Workshop. Kyiv: Higher School. Main issue, 1987. 245 p.

developer himself and is not used by others. In this regard, there is a need to allocate a subset of software tools which have features of products that are developed and manufactured to meet the needs of the national economy, population and programs on data carriers with program documentation developed and manufactured in accordance with standards and other regulatory documents, having undergone state, inter-departmental or departmental testing and technical control of the manufacturer, provided with guarantees. A software product is a unit of software production for technical purposes. It follows from the definition that not every software tool is a software product.

The relation between programs, software tools and software products can be established using the concept of sets. Let us suppose A = {P}; B = {ST}; C = {SP}. The following relations are established between the sets A, B, C: And ⊃ B ⊃ C. So, any software product is a software tool, but not the other way around. Accordingly, any software tool is a program, but the reverse statement is not valid. The B\C difference is a subset of software tools which are not software products; the A\B difference is a subset of programs that are not software tools.

To get a software tool from a written program, it is necessary to insert this program into the computer memory, compile, debug and compile program documentation for it. This requires some labor costs, generally exceeding the labor costs for initial writing the program text. In order to obtain a software product from the software tool, it is necessary to more fully anticipate possible application and requirements of potential users, as well as to ensure that the previously stated requirements for SP are fulfilled. Labor costs, which three times exceed labor costs to create ST, may be needed for all that.

2. Software production specificity

Software production has features that should be taken into account at all stages of the software tools' lifecycle as well as in quality management. The software production has a high scientific capacity and intellectual content, it is created on the basis of intensive use of scientific knowledge and promotes the dissemination and use of knowledge by creating banks of this knowledge, information and expert systems, etc. Software tools development almost always requires high mental stress, deep and accurate knowledge. In ST development, besides programming specialists, highly

qualified specialists in a wide variety of subject areas (chemistry, physics, control systems, technological processes, and the like) have to be involved The narrow specialization and qualification of these specialists makes them unique. The complications in the control of progress and quality of development emerge. These difficulties are often subjective. High scientific capacity causes the need for increased costs for research and development work in the process of creating programs. This feature complicates the use of engineering methods in the design and quality management of ST

Software production is not expended and does not consume its resources when used. It is known that industrial production is divided into two main classes. The first are products expended when used (fuel, raw materials, substances, etc.); the second are products that consume their resources when used (machines, appliances, worktables, etc.).

The software production cannot be assigned to any of these classes of industrial production on these grounds. It is not expended when used and does not consume its resource. Moreover, with a well-established support service, ST is improved by detecting and correcting errors, as well as upgrading of methods, structure, and parameters. During the period of storage and usage the data carrier, on which the program is recorded, loses its features and may eventually become unusable. But by removing the copies from the ST in advance, the influence of this factor can be eliminated. The question of whether a copy of ST may be compared with the original and whether a user has the legal right to make copies of ST slightly impacts the nature and character of the features of the ST because it does not cause major complications for a user.

This feature significantly affects the methods of assessing the reliability of the ST and the possibility of extending the traditional interpretation of the reliability of technical means to software products. This is especially true of reliability indicators such as durability and maintainability. The nature of the main indicator of reliability, infallibility, is also changing.

2.1 Easy to manufacture

In most cases, manufacturing a software product on magnetic or paper media involves a relatively simple operation of removing copies from the product's sample-standart (original). However, no qualitative changes

occur. The identity of the copy with the original is easily controlled. It is somewhat more complicated to make copies of a software product in a permanent storage device. But this way of storing information is, firstly, not widespread, and secondly, it is also easily controlled and automated. This feature significantly affects the organization of quality control of software production. The main difficulty of this control lies not in the process of manufacturing the product, but the processes of development and testing of the prototype. The high quality of the prototype with strict adherence to the technology of rewriting guarantees the quality of the copies made from it – new copies of software products. It should be noted that the ease of manufacturing SP complicates the control over their distribution.

2.2 Easy to make changes

Upgrading a software product requires knowledge of the structure of the product being changed, a thorough analysis of the impact of the changes. But the process itself is simple. All you need is a good editor program. This feature, when skillfully used, is a significant advantage of software. This advantage is vital in dynamic spheres of applications, such as in automated control systems, where a constant search for the most optimal control modes goes on, which leads to the need for continuous improvement of the software. But the same feature easily becomes a drawback if the flow of change becomes poorly managed and unbalanced.

The ease (sometimes it seems to exist) of modernization generates a large number of relevant proposals, wishes, and sometimes insufficiently justified orders.

Attempts to implement all these changes are often unbalanced (uncoordinated) with real needs and opportunities. Under these conditions, the text of the program and the interrelations between its elements are confused; the program loses its consistency and accessibility for review; the difficulties in maintaining program documentation emerge. After all, it can accelerate the degradation of the ST to complete ineligibility. The ST upgrade process should be subject to careful control and planning.

2.3 The abstract materiality of the software

By its formal content, any ST is an information object. But the information contained in the software is very specific. In general,

information reflects the object of knowledge. The information contained in the command (operator) part of the program itself is obtained on the basis of the study of a certain object of knowledge (such as knowledge of a controlled object or process) and contains an order for the sequence of transformation (processing) of data that reflects the state of the cognition, to the required result. Information (data) is the object of processing in human-machine systems. The application code is not subject to processing. It itself contains data processing rules. This is the fundamental difference between the program as an information object and information (data) in general.

As part of the computing system, the ST manages the data conversion process. Naturally, the question arises whether ST can be considered a material object. The materiality of the ST, its components, the internal mathematical support of computers or implemented in long-term storage devices, is not questioned as they are susceptible to organoleptic perception. The materiality of the ST implemented on magnetic data carriers is questionable, since these ST carriers are not susceptible to organoleptic perception. Physical embodiment, the materiality of the software in this case are somewhat abstract. *The absence of concepts of tolerances and landings.* Each element of the program has its size in bytes. With hardware, program elements are easily moved within memory, which greatly facilitates the build process of the programs. Therefore, in programming there are practically no restrictions on the maximum tolerances for the required dimensions when designing program elements.

3. Software production life cycle

3.1 The stage of research

Software production is science-intensive, so its life cycle begins with the stage of research, which is carried out in the framework of research work on this issue. The main result of the research is the draft Terms of Reference (TOR) for development. TOR is the document that should be guided by the team of developers when creating the software. It is developed by the customer organization and agreed with the developer organization. In some cases, on behalf of the customer, the project of TOR is developed by the developer organization.

In the development of complicated software complexes on separate components of the complex one creates private TOR[2].

*The TOR should include the following sections:*name and scope; basis for development; aim and purpose of development, scientific and technical requirements; economic indicators; phases and stages of development; the procedure of control and acceptance. It follows that the main attention in the study should be given to the conditions of use and purpose of SP, the justification of scientific and technical requirements and economic (socio-economic) efficiency, which determine the level of quality and performance characteristics of the product. In TOR for multifunctional ST, besides the general requirements for ST in general, requirements for the quality of implementation of each function, as well as the priority (significance) of functions and information interaction between them, should be defined. Such differentiation will allow more purposefully influence the quality of the software tool being developed. A software quality assurance plan should be attached to the TOR, which defines the measures to ensure the required quality of the ST being developed, the sequence of their implementation, the responsibility for carrying it out, the objects and methods of control, the forms of recording data on quality and reporting. The value of a thoroughly grounded and compiled TOR cannot be underestimated. Such underestimation, especially on the part of the customer and the contractor, leads to a delay in the development and release of defective products. When implementing complex software complexes, up to 70% of all emerging problems are directly related to the imperfections of the requirements in the TOR and only 30% are the result of errors in the development process.

The imperfection of the TOR for the development of SP is caused not only by the misunderstanding of the TOR value, but also by such objective difficulties as novelty of problems, lack of relevant experience, lack or insufficient reliability of the initial data for the design, etc. In these cases, the customer and the developer want to have good TOR, but cannot develop it. In such a situation, preliminary (at the stage of research) prototype of automated systems, data processing systems and processes give good results. The essence of the prototype is as follows. The developer, having received from the customer the most general information

[2] Antipensky V.E., Bilousko V.S., Chujdan T.I.. Computing Machines and Programming: Workshop. Kyiv: Higher School. Main issue, 1987. 245 p.

about the purpose of SP, creates a prototype (simplified preliminary sample) of SP on the basis of those computing and software tools that he possesses. In doing so, he makes extensive use of unified software modules-components and modules of software products-analogs.

At the same time, in the requirements implementation restrictions when using the hardware interface and the equipment itself should be specified. When conducting research it is necessary to predict the life cycle of SP, trying to properly account possible changes in the conditions of use, tasks performed, the direction of upgrades. Particular attention should be paid to substantiating the requirements of SP resistance to various distortions (failure of information sensors, operators' errors, errors in communication channels and computing devices).

Many guidelines for the development of Software Requirements Specifications (SRS) include the IEEE Guide to Software Requirements Specifications standard.

The *first section* of the standard provides information about the SRS environment, the characteristics of the «correct» SRS, and aspects regarding the evolution of the SRS. The characteristics of the «correct» SRS are of particular interest.

Such characteristics are: uniqueness of interpretations, completeness, verification, possibility of citation, consistency, modification, clarity (possibility of tracing), usefulness at the stages of operation and maintenance.

It is considered that the SRS has the property of uniqueness of interpretation only when each requirement contained therein permits a single interpretation. SRS is complete if it has the following properties: includes all the essential requirements related to the operation, method of display, restrictions, equipment, attributes and external interfaces; determines ST responses to various (correct and incorrect) types of input information in different situations; meets some standard (individual discrepancies must be specified); all pictures, charts and tables in it are accompanied by detailed signatures and definitions of all terms and scales of measurement. The SRS is true if its every claim is true. The SRS is considered consistent if none of the requirements contained therein are in conflict with each other. The SRS is modified if it is easy to make any necessary changes in it without making any contradictions. The SRS has a track record if the reasons for any requirements are obvious and if it

facilitates the process of justifying the requirements arising from the development or improvement of the documentation. The SRS will be useful at the operational and maintenance stage if it facilitates the use and upgrade of the ST at this stage.

The second section of the standard addresses the basic ways of expressing requirements (using I/O specifications, multiple examples, and model specification); annotation (explanation and ranking) of requirements, as well as the most common mistakes in the description of requirements. It should be noted that none of the methods considered in the standard is universal. The method of expressing requirements using I/O specifications is only suitable if possible inputs and expected results are available for review; the way of expressing using examples – if possible system situations are available for review; the method of expressing using model specification imposes restrictions on the construction of software that contradicts the purpose of the SRS.

The third section of the standard sets out the overall structure of the SRS. In particular, in the «General Questions» section, it is recommended to display the product purpose, functions, user characteristic, general constraints, assumptions, and dependencies (factors that influenced the choice of the SRS requirements). Obviously, these issues should be reflected in one way or another in the TOR for the development of the SP.

3.2 Development stage

This stage begins with the development (consistency and approval) of the TOR and ends with the test of the prototype SP. In the general case, the stage of development of SP consists of the following stages: the development of SP, the development of technical proposals, sketchy design, technical engineering, functional engineering, testing. The results of the works for each stage respectively are: TOR, technical suggestions, sketch project, technical project (algorithms for solving problems), functional project (text of the program), test prototype. Let us consider the content of the works at these stages.

3.3 Technical Suggestions and Developments

SP should include a justification for the feasibility of the proposed variant of the structure selected on the basis of the analysis of the TOR and the various options for possible solutions.

The technical proposals list all the fundamental issues to be addressed in the engineering process with a preliminary assessment of their feasibility. For example, technical proposals for the development of ACS software should reflect the following issues: the degree of system automation; the composition of the general algorithm; previous structure and scope of the algorithm; determination of the structure and scheme of information of flows between computers, information sources and managed objects; preliminary estimation of the temporal diagram of data exchange between the computer and the objects of the system being automated; development of quality assessment criteria and methodological bases for their verification (control) during development; identification of problems that require preliminary modeling; determining the scope, methods and tools of modeling; preliminary selection and evaluation of methods for solving major problems; preliminary elaboration of principles and methods of ensuring the stability and reliability of management; working out the issues of development organization, as well as providing developers with general-purpose hardware and software.

The technical suggestions are the starting point for the development of the sketch project. In justified cases, both of these stages can be combined.

The sketchy design should include fundamental solutions that give a general idea of the structure of the SP, the designation of its components, the organization of relationships between these parts, data exchange and dynamic distribution of computer resources, as well as programming technologies. As a result of sketchy design, a preliminary estimate of the computer system (CS) resources required for the development and operation of SP is given. While developing of sketchy design, the problem of choosing the optimal structure to be released, the manufacturability of design, debugging and testing programs and the construction of a common algorithm for solving the problem are solved.

If necessary, one develops structural diagrams of the general algorithm at the level of its components, including databases; pre-connect the components of SP on the time of execution, use of external computers and information; sets the acceptable range of characteristics of the input and output values for each component; make private TOR for the development of the main components; simulates the operation of kernel components in order to test the basic principles of data processing and control; establishes basic principles of quality management of SP; identify

the critical ways and paths of the SP calculation; solves organizational issues of work at the stage of technical design. At the stage of the sketch project all the fundamental issues of technology creation of software complex should be developed. The sketchy design is approved by the customer organization. It serves as a guidance document for the development of a technical project. In justified cases, it is decided to carry out the task of sketch design in the framework of a technical project. The reasons for this may be the experience of developing similar products or their simplicity.

The SP technical project is a set of design documents that give a complete picture of the algorithmic and information structure of the product under development and contain all the source data for programming. The language used to describe the algorithms for solving problems in a technical project depends on the set programming technology. In the traditional approach, the so-called linguistic-formula descriptions and graphical schemes are used.

When describing algorithms, the developer uses any terms, concepts, and designations that are understood by him (but not necessarily understood by other developers). Mutual understanding between developers of complex software systems is difficult, which leads to unproductive spending of time and other resources. Therefore, in modern programming technologies, much attention is paid to the strict regulation of both the linguistic means of description and the design procedures themselves. After the design process is completed, the functional design stage begins. Functional design (FD) consists of three main stages: the development of the program, the development of program documentation and the testing of the SP prototype. The main content of the works on the stage of FD of program complexes is programming and debugging of components of the program complex, autonomous component testing, assembling of the program complex, development of program documents, development (alignment and approval) of the program and testing methods, conducting of all types of tests, adjustment of programs and program documents according to the results.

Production of SP. Production is a set of works to ensure the production of the required amount of SP in a set period; it includes the following types of work: studying the demand for this type of SP; production planning and production management; organization of

technological preparation and maintenance of production, logistics; storage and delivery. The stage of production of products for one-time orders has a hidden (implicit) character. It lies in making the required number of copies of the SP, including the program documentation.

Thus, in a single production, the SP developer combines the functions of the SP manufacturer and supplier.

3.4 Maintenance of SP

This stage consists of collecting information about the quality of SP during operation, modifying the product and notifying users of changes made. Maintenance functions are usually performed by the SP vendor. Practice shows that the initial stage of SP operation the developer's involvement in SP maintenance is very useful, and sometimes necessary. As the user and the supplier master the SP, this need is gradually eliminated. The maintenance stage is conditioned by the need to perform such tasks as the inclusion of new features in the SP, change of functions, modification and replacement of equipment in data processing systems, error detection and correction. The stages of operation and maintenance proceed in time paralleled. At the same time, the production of new SP can be carried out.

The reasons for the end of the SP life cycle may be different: no need for further use; replacement by new, more advanced SP; incompatibility with new equipment; dissatisfaction with the results of usage, etc. Due to the mentioned specificity of software production, the problems of evaluating its non-compliance with its purpose and utilization differ significantly from the corresponding problems of evaluating technical production. These differences are basically the following:

1. The difficulty of identifying non-compliant products as a result of uncontrolled upgrading of product units by users. In these circumstances, individual non-compliant units of products can be brought to compliance by users and, vice versa, compliant units to non-compliant ones. In general, regarding the appearance of software production, the assessment of its conformity becomes ambiguous;
2. The complexity of identifying inappropriate types of products causes the complexity of their isolation, i.e. separation from products that meet the requirements;

3. Non-compliant software products are generally not suitable for any use without further refinement and processing, so they should not be disposed (used for any other purpose). Their use should be completely excluded.

Separation and clear delineation of the phases and stages of the SP lifecycle, defining the necessary relations between the stages contribute to a clearer organization of certain types of work – ways to create appropriate technologies and technological tools, including methods and means of quality control both within and after the stage. When forming stages, it is very important to define clear links between them, to identify control points and decision making, which should facilitate a more accurate transition of information from one stage to another and ultimately reduce the development timeframe and improve the quality of the developed SP. The SP life cycle is not strictly consistent. It is iterative. The terms of reference, the sketch and the technical projects of the SP after their approval shall not remain unchanged. In the development of sufficiently complex SP it is impossible to achieve the invariability of the life cycle in practice. Sometimes, some design decisions made at previous stages of the life cycle have to be modified or refined and revisited. The reasons for this are different. Basically, they exist because the customer at an early stage of development does not quite clearly imagine and formulate system requirements, and the developer does not always immediately find the best solutions.

4. Software Tools Classification

The penetration of computer technology in all spheres of human activity, the desire to solve with the help of this technique a set of completely different problems extremely diversify software products by purpose, application, nature of production and maintenance, level of complexity, etc. Each type of ST has its characteristics that can significantly affect the methods of their development and quality management. Ignoring these features leads to problems of interaction, different kinds of misunderstandings and contradictions. Requirements for quality indicators depend on the type of ST: high requirements for one or another indicator for one ST may not be necessary for another[3].

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

Depending on the purpose, five subclasses of software are identified: *system ST*, applications *for scientific research*, applications for *designing*, applications for *control of technical devices and technological processes*, applications for the solution of *economic tasks*.

The following breakdown into types of software can be considered universally accepted: application, system, and tool software. We give the following informal definition of these types. For a computer to do your job, you need to create application software. For the computer to cope effectively with many applications and to be well-adapted to the environment, you need to create system software. To make it easier to develop software, you need to create and use tool software. Application ST are developed by experts who are well versed in the processes they automate. System ST are usually more complex than application and tool ones. They are developed by experts who know all the intricacies of programming and operation of computer systems.

Examples of system software are operating systems, database management systems, and the like. Instrumental ST are used at the stages of program development and maintenance, including debugging and testing. Typical examples of software tools are compilers, text editors, data archivers, change analyzers, and the like. Instrumental ST are developed by experts who are knowledgeable in programming technology as a whole or in specific aspects (transmission, editing, debugging, testing, etc.).

All considered types of ST are classified by one attribute – purpose. But when planning development, development management, quality management of software products, it is necessary to consider not only the purpose of the software, but also their other characteristic features. Such features include, for example, the number of users. Of course, the ST that a specialist has developed for himself and which he will use for himself, do not have the requirements that apply to ST developed for thousands of users.

In the first case, the requirements for the ST are determined by the developer at his discretion, and in the second they must be determined by the customer, taking into account the possibility of using the ST in a wide range of conditions. Even when developing software for your own use, the frequency of use (one-time, daily, weekly, annually, etc.) is essential. Problems that are of great importance for specific (consisting of components) ST may be irrelevant to unspecified ST (that is, the components themselves). The requirements for real-time ST differ

significantly from the requirements for ST that implement, for example, basic computational tasks. The problem of comprehensive detailed classification of software is extremely complex and has not yet been resolved. Let us consider some classification methods that are important for understanding software quality management issues. By the nature of the manufacture one should distinguish between single and mass production. The developer himself performs the functions of the manufacturer and the supplier. He provides training for the customer support staff, assists the user with the commissioning of the ST into industrial operation, accompanies the ST.

The single nature of the manufacturing does not preclude the re-production of ST for implementation at another enterprise (re-introduction). In this case, the developer usually has to refine the software, taking into account the specifics of use in the new conditions. Depending on the scope of the revision, it may happen that the revised ST should be considered as new ST. The batch type is characterized by periodic production of batches of homogeneous software that is in high demand.

By nature of supply and use, software may be characterized by the autonomy of supply and use, or the supply and use of PCS or an automated technological complex (ATC). A distinctive feature of autonomous ST is that it can be developed, manufactured, tested and delivered (sold) autonomously An example of autonomous ST can be almost any ST related to system ST. Distinctive features of the ST supplied as part of PCS or ATC are the joint development, manufacture and testing of ST and system being automated in which it is delivered to the customer (user). Examples of such systems are the PCS. In terms of the number of functions performed, all specified SP except the software modules are multifunctional. For example, in the Logistics Supply Subsystem (LSS) of ACS considered as SP, the following functions (tasks) can be implemented: determining the need for materials, determining the need for equipment, determining the need for spare parts, developing a schedule for the supply of units, etc. Each of these features, with the exception of the latter, has a specific purpose and therefore has autonomy of use, i.e can be used independently. The LSS itself can be considered as a multi-purpose system, although it has a general (global) purpose. But multifunctional SP can be one and the same. These SP include, for example, those SP that bring two managed objects,

moving in space, together. The goal here is one (approximation for a given distance), but for approximation it is necessary to solve the following problems: determine the position of each controlled object in space at time (t+1); by extrapolation calculate the position of objects at time (t+1); calculate the optimal approximation trajectories for each object at time (t+1); to calculate the control actions corresponding to these trajectories. It is clear that the successful solution of the general convergence problem is possible with the correct solution of all partial problems. Depending on the nature of the implementation process, the following types of ST are distinguished: implemented as part of the developed ACS; implemented in existing PSC (ATK); autonomous implemented and self-relevant ones; software components that are built into the software system. To determine the level of unification, ST and their components belong to one of the following types: *standard; unified; original.*

To *standard* one includes components of a specific ST that meet the requirements of state, industry or national standards, which are referenced in the design specifications. *Unified ST* include components of a specific ST that can be used to solve the same problems in several software tools. Unified ST are purchased, borrowed ST, as well as developed according to the standards of given enterprise and used in various ST. Components of a specific ST that are not made in the organization but purchased are called acquired. Borrowed items include components of a specific ST that were previously designed as original to other ST, used in given ST, and which have developed program documentation.

Original ST include components of a specific ST developed for the first time for this ST and used only in this particular ST. In practice, original ST are often refined (unified) and thus transformed into unified ST. ST is a complex product. The average software package contains 40–50 thousand source text operators.

Taking into account the existing semantic relationship of operators, which needs to be known and recorded, the average ST is compared to the locomotive, which has about 25 thousand details. Many materials basing on ST quality and programming technology are difficult to use because it is unclear to which class the ST they belong, and unreasonable generalizations often make it difficult to create effective methods and tools for evaluating ST quality.

5. Quality features of the ST

While identifying the terms «Software tool» and «program», sometimes the measure of its relevance to the original algorithm is understood under the quality of ST. However, the following two mistakes are made: the concepts of «program» and «software tool» are not identical; errors in the original algorithm are no less likely than programming (encoding) errors. Errors may also be contained in the program documentation. Therefore, the full compliance of the program text with the algorithm text cannot guarantee the suitability of the ST for its intended purpose. The compliance of the ST algorithm is characterized not by the quality of the ST as a product, but the quality of one of the main technological processes in the creation of the ST- the quality of programming. It is more common to define the quality of the ST as a measure of compliance of the real characteristics of the ST with the characteristics given in the TOR. Such interpretation of the ST quality is acceptable only if the TOR has fully and uniquely defined all the consumer properties that the developed ST must have. But this condition is not yet feasible due to the lack of a common nomenclature of quality indicators, methods of setting scientific and technical requirements for ST and lack of experience in solving these problems.

Software quality one should understand as a set of ST features that determine their suitability to meet specific needs according to their intended purpose. It is based on three key concepts: ST feature, need, ability to meet needs. Let us consider these concepts[4].

The existence of ST as a product of labor is an objective reality. As a product, ST has many attributes – objective features that determine its difference or similarity with other objects and are manifested in its creation and operation. These attributes may be common to a given product class, and specific to a particular type of software or specific ST. According to the impact on the ST quality one should distinguish between essential attributes and insignificant. Only the essential attributes are of interest. But the measure of the impact of these attributes on the quality of specific ST is also different. These differences need to be taken into account. ST attributes are quantitatively and qualitatively characterized by quality indicators.

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

The following main aspects of ST needs can be distinguished: scientific, technical, economic social. The need for ST is established at the beginning of its life cycle, usually at the stage of research (marketing). In doing so, the following basic questions should be answered: field of usage; solved tasks; expected effect from use (scientific, economic, social). In the narrower sense, the need for accuracy of data conversion results, trouble-free operation, ease of maintenance, ease of exploration, and the like are established. In order to meet the needs, ST as a product of labor must possess certain useful (consumer) attributes that collectively determine the public usefulness of ST. Usually, each product has several useful features, each of them satisfies one or more needs. ST quality is both a technical and a socio-economic category. On the one hand, it is closely linked to features that satisfy certain needs. But the attributes of the subject are not economic categories, so a technical approach is promising from these positions on the ST quality.

On the other hand, the ST quality is a concrete expression of the public consumption value, so an economic approach should be applied to it.

Two characteristic features should be noted in the formation of ST quality.

1. The quality of any industrial product on batch or mass production depends largely on the manufacturing process. It cannot exceed the technical level achieved in design. But it may be well below this level due to non-compliance with manufacturing technology requirements. Production of SP is often a simple technological operation of making a copy from the sample-standard by rewriting from one data carrier to another. However, no qualitative changes occur. The identity of the records is easily controlled. Therefore, it is necessary to control and manage the ST quality, not mainly while its production, but in the development, that is, when the ST quality is formed.

2. SP during operation and maintenance, as a rule, are constantly changing, modernized. Therefore, the maintenance process itself can be called, for example, an extended development process. But any change in the structure of the software production leads to a certain qualitative change, so the quality control and management of the ST must occur not only during their development, but also at the stage of operation (maintenance).

Factors affecting the quality of the ST can be constant and variable, direct, indirect and inverse. Quality factors should not be confused with quality indicators. Quality factors characterize the conditions and elements that influence the formation of quality.

Quality indicators, more precisely, denotation of quality indicators, directly characterize the quality itself. Any management is a purposeful action on a managed object to achieve specific, predetermined results. Quality, as a set of consumer attributes of products and services, is a specific management object and has significant features. The product quality management itself differs from the management of the product creation process in that not the organization of production, but the regulation of the properties of the products produced is the object of management here. These attributes are formed at the stages of the life cycle under the influence of various conditions and factors. Quality management (QM) is the process of acting on those conditions, factors and socio-economic relationships that influence the formation and change of consumer attributes of products.

To accomplish this process, a system of governance is created – a set of interacting bodies, tools and methods of management. Software quality management has organizational, methodological and socio-economic aspects. The organizational and methodological side of SQM is expressed in the development and application of advanced programming technologies, consolidation of scientific and technical achievements in the relevant standards and methodological documents, equipping the developers with advanced technology, etc.

The socio-economic side of SQM is expressed in the creation of such a system of socio-economic relations between all participants in the development, which will ensure the creation and production of PP of the required and guaranteed quality. This system shall cover: a) the relationship of the administration of the developer organization with the customer organizations; b) the relationship of the administration of the organization with the staff of its units; c) the relationship between the development units; d) the relationship of the managers of all units with its individual executors.

At the same time, personal interest and responsibility of each manager and contractor for the quality of the software under development should be ensured at all levels. Users are most interested in the quality of the

software, but they are little interested in the ways in which the developer achieves a certain level of ST quality. It is only important that this level meets the actual needs. Actions aimed at assuring the user (customer) with the confidence in the proper level of purchased products constitute the External Quality Assurance. Such actions include, in particular, marketing, drafting and mutual harmonization of specifications of requirements, testing and maintenance of software. Elements of internal and external quality assurance can generally overlap. Having a specification of requirements contributes, for example, to confidence in achieving the required level of quality both by the developer and the customer.

For a better understanding of the nature of software quality management in the process of its development, let us consider the conceptual model of management.

6. Conceptual model of software quality management

In general, *management* is an integral part of the functioning of systems of organization of various nature: biological, technical, socio-economic. In each of them there are objects that subordinate to others, and therefore, and control them, forcing them to move in a certain direction, perform the specified actions, organize their activities as a whole[5].

Management of the project (object-system), its components and processes, with the purpose of increasing the efficiency of the systems functioning occurs at the stage of system design, creation, formation, development, formation, functioning of the system. The effectiveness of the management is determined by the adequacy of the control actions to the object of the management.With regard to computer ST, the quality management scheme is as follows. The object of management (action) is the quality of the object of labor. The subject of labor, depending on the stage of development, respectively, is the TOR, technical design, functional design (program text), ST prototype. ST quality is mainly formed at these stages, so quality management should start from the very beginning of the software development process and be continuously implemented throughout the process. In general, management actions can affect not only object labor directly, but also labor and technological processes, if they do not contribute to the management goal, as well as factors affecting ST quality.

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

Programming labor tools include compilers, downloaders, program builders, documenters, automated debuggers, test data generators, automated programmer jobs, including computers, etc. Technological processes consist of certain technological operations for ST creation, performed by means of labor under the control of programmers (operators) or directly by programmers. In any case, the role of the person in the technological process, its impact on the quality of the product is crucial. The purpose of management is to provide the necessary level of ST quality, which guarantees the expected socio-economic effect of the use of this ST for its intended purpose. The main role in software quality management is performed by software development, operation and maintenance managers (depending on the life cycle stage), direct developers or ST maintenance specialists, together with management tools.

The category of specialists involved in software quality management will be referred to as *developers*. Developers have an effect on the state of the ST either directly or through appropriate technological means (TM) and technological processes (TP). These actions can be both positive (coinciding with the purpose of management) and negative (not predictable perturbations). Negative actions result in program errors. Sensors of information about the status of the managed process and the quality of the software depending on the stage of the software life cycle are either the developers themselves (at the design and debugging stage), or experts of the quality control groups (at the design stage), or testers (at the test stages), or users (at stage of operation). Naturally, certain categories (concepts) should be used to describe quality. First of all, it is necessary to determine the consumer properties of the ST being developed, that is, those attributes that the software must possess to be able to be used effectively for its intended purpose. Each attribute or group of attributes is quantitatively characterized by quality indicators. In order to manage quality, it is necessary to know the acceptable rates of the quality indicators as well as the criteria for quality assessment at each stage of the ST life cycle. This information should be contained in the terms of reference and specifications for specific ST or groups of homogeneous ST.

The actual rates of the Quality Score can only be set when the ST development is completed. In the process of development, you can only make predictions about the quality of the ST, controlling the presence or absence of certain features in the project documentation and programs

when debugging. Monitoring the current state of the quality of software being developed usually relies on special quality quality management groups (QMG) that are independent from the developers. In order for the QMG to perform its functions successfully, it must have a clear understanding of the quality of the controlled ST analyzed at different stages of the life cycle, of the methods for determining quality indicators and quality criteria.

The conclusion about the current state of quality of the controlled software is made by the ST QMG on the basis of examination of the project documentation (if the program has not yet been written or has not acquired the performance capability), or by analyzing the correctness of the initial data (results) by comparing the actual data with the expected ones (in the working program). In the first case, it should have a methodology for conducting the examination, and in the second case there should be clear signs of identification of correct (incorrect) results. All ST quality information should be submitted to developers or maintenance professionals who, basing on the analysis of the information, make decisions about how to influence the management object. Primary information about the quality of software in the development stages is often symptomatic. Only external signs of design errors, deviations of the data processing process (ST operation) from normal mode, or lack of the required attribute in the ST are recorded. In order to make a decision on the impact on a management object in order to improve its quality, it is necessary to establish the reason for deviation from the required quality level and the way to eliminate this cause. When designing an impact, developers should consider the requirements and capabilities of the programming technology used, the requirements for ST from customers (users), the structure of the ST, the available resources, as well as the relationship between the signs of errors, their causes and ways to eliminate them.

The essence of managers' influence is to change the structure of the program and program documentation (error correction, introduction of new functions and procedures, improvement of methods of solving problems, etc.) in the direction of its optimization according to the criteria specified in the TOR. If necessary, labor tools and technological processes change. Thus, ST quality management tasks are a variety of optimization tasks and have the following components: defining the goal of quality management

(QM); knowledge of ST quality assessment criteria; knowledge of the current position in relation to the purpose; knowledge of the microstructure of ST and factors that affect the quality of ST; knowledge of limiting conditions in terms of execution and resources; determining the best ways to reach your goal. Under the products quality management system (QMS) one understands a set of organizational, scientific, technical and economic interrelated measures to establish, provide and maintain the required level of ST quality in its development, production and operation. The ST QM system is multilevel.

Previously, a quality management scheme for a single ST was considered. But organizations that specialize in the development of complex software solutions can simultaneously develop or prepare for the development several ST. Taken together, these ST constitute *software* produced by this organization. The governing bodies in this scheme form the administrative and technological units of the organization. Direct management objects are not software, but teams, software developers, *technological lines* (TL) *development* and *technological processes* (TP). The ultimate goal of management is the required quality level of software. Information on the state of the software development process goes to the Software Quality Control Service (SQCS) along with the QCG of the ST being developed. To make decision about management actions it is necessary to have an annual and perspective plans of software development (thematic plan of research and development works); a list of requirements for the quality of software by potential users; data on the current state of quality of the developed software; data on available labor, material and time resources; a list of organizational and economic mechanisms for regulating the activities (OEMRA) of developers, including the rationing of labor and resource costs, the promotion of high productivity and quality of software; methods and means of technological preparation of development (TPD), including formation of technological lines and technological processes; data on the availability and characteristics of technological programming modules, etc.

Having this data at its disposal, the governing body influences the quality of the software created in the organization by beforehand and purposeful technological preparation of the development of specific ST, setting and correction of the ST QM goals for separate periods of time depending basing the state of the developed ST, regulating the team of

developers, stimulating the creation of high quality software. It is important to emphasize the special role of the latter factor, since an individual in the system of SQM certainly plays a decisive role.

The quality of software is formed at all stages of its creation, therefore, operational quality control is necessary for the operational impact on quality. During the operation of the SQM system, there is a need to collect, store and process large amounts of information. Naturally, the SQM system should be as automated as possible. Like any automated control system, it consists of the following elements: organizational, methodological, technical, software and information providing. The model under consideration contains the basic elements of the SQM system at the level of the developer organization and the relationship between them. On its basis, by further detailing, it is possible to determine the composition of the necessary regulatory, methodological, information and software tools for supporting the SQM systems, as well as the tasks and overall structure of the quality system of the developer organization.

Three quality objectives, that the organization faces, have been identified. These tasks can be interpreted as follows: the organization must achieve and, in the case of support, maintain the quality of the software at a level that ensures continuous satisfaction of the user set or offered requirements; the organization must assure its management that the required quality is achieved and maintained at the required level; the organization should provide the user with the assurance that the required quality of the delivered software is achieved or will be achieved. If necessary, the user may require appropriate evidence to be provided. Solving these tasks requires the introduction and definition of key terms and definitions. Quality policy is the main directions, goals and objectives of a quality organization, formally formulated by its senior management. Overall quality management is an aspect of the overall management function that defines and implements quality policy. General management includes quality planning, resource allocation, evaluation and other systematic quality actions.

A quality system is a set of organizational structure, responsibilities, procedures, processes and resources that ensure the overall quality management. As a condition of the contract, the customer may require clear evidence of the use of certain elements of the system. Methods and activities of an operational nature are used to meet the quality

requirements. In order to avoid confusion, it is advisable to add specific references to narrower, specific concepts, such as «quality management in the design process». Quality assurance is a set of planned and systematic activities needed to create confidence that a product meets certain quality requirements. The quality of software is formed at all stages and stages of its life cycle. Therefore, the quality system functions simultaneously with all other activities affecting quality. The quality loop of software has some differences from the quality loop of other industrial products. These differences are mainly due to the decisive role of the software prototype in shaping the quality of the development stages rather than the production stage, as is the case in industry.

7. Factors affecting the quality of software

The software quality depends on many factors. Let us consider the main of them.The responsibility of the management in quality assurance is determined by the presence in the organization of the quality system of the following elements: documented policy in the field of quality, goals and obligations; responsibility and interaction of the staff which affects quality; means of inspection and specially trained personnel; representative of management bearing personal responsibility for meeting product quality requirements; periodic analysis of the effectiveness of the quality system which runs in the organization, the quality of regulatory documents of the software being developed, in the part of optimality and completeness of the claims set in it.

Preparation of Terms of Reference (TOR) for ST development and defining the main list of requirements therein is the first stage of ST design. TOR must be composed both on software, supplies (software complexes), which are standalone objects, and on program components. When developing complex ST that have no analogues at the time of design, direct assembly of the TOR is usually preceded by research work, the purpose of which is to determine the purpose of the ST, areas and features of its application, as well as to analyze the requirements of potential users.

Efficiency of programming technologies. Technological preparation of software development. The process of creating a PP is costly and time consuming. Programming technology, management of the software creation process should provide the maximum beneficial effect at

certain costs. Naturally, such an effect can only be achieved by using the most advanced methods and tools to develop software. The technological preparation of software development should be complete and timely.

Regularity and effectiveness of quality control of development. The process of creating software should be under constant and careful control. The technology for detecting and eliminating errors, as well as temporary material resources for the implementation of this technology, should be installed in advance. Practice shows that for the production of high quality products, it is necessary to plan up to 60% of labor costs in advance to ensure proper control, debugging and testing of programs, to establish control procedures in advance, to create software and technical means for debugging and testing. Regular use of inspection methods prevents up to 60% errors in advance.

Developer Qualification. The quality of the created ST is determined by the following properties of developers: the level of knowledge (knowledge of problems, programming languages and computers, engineering techniques, data processing principles), the availability of practical skills (experience in creating similar programs and software systems);) level of initiative (understanding of the tasks being solved and their relationships, efficiency of working time use, the desire to bring each task to a complete completion, maintaining working contacts with the co-workers); level of responsibility (focus on the work being performed, constant desire for self-improvement, healthy self-esteem).

Content and quality of software (instrumental) tools used in development. The development of sophisticated software is associated with the need to use various computer hardware and system software. These tools serve as a kind of technological equipment for programmers. Naturally, the quality of ST created depends on the reliability of this equipment and the stability of technological operations. Also timely and fully meeting the developers' need for these tools is important.

Stimulating the creation of high quality software. Despite significant achievements in the field of programming industrialization, the nature of programmers' work is also individual and largely dependent on the personal abilities of the performers. The performance and quality of programmers working under the same conditions can vary several times. Therefore, an effective system of stimulating the creation of high quality software must be introduced when creating programs, which

involves the remuneration programmers depending on the quantity and quality of results. Development managers should always remember that, with stimulation for quality work, developers will find effective ways to achieve a set goal. Conversely, with the absence of stimulation, many useful start-ups will be unfulfilled. This is one of the manifestations of the human factor.

Formation and adherence to uniform principles of software development. Based on the results of the study and analysis of the factors affecting the quality of the software, taking into account the specificity and experience of creating these products in each development organization, the basic principles of software development should be formulated: development management with the help of a project plan broken down into stages, quality control throughout the development period, from the early stages, ensuring strict control of compliance of the features of the original software product with the requirements set out in the specifications; use of advanced methods and programming tools; supporting a high sense of responsibility for the quality of the programs being developed in each project partner; use of the minimum number of highly qualified employees; continuous improvement of methods, means and software development organization. Another quality system is based on the following principle: all stages of development are clearly distinguish At each stage, the outputs and quantitative and qualitative criteria for their evaluation are determined. Quality processes and output are standardized according to quality. Outputs are monitored according to previously established criteria; special attention is paid to the organization and quality control of the work of autonomously working groups of programmers. Various methods of software quality checking are considered, which are considered not an optional occupation, but one of the most important elements of design[6].

Marketing. The quality of a particular ST depends on the effectiveness of the system of market research measures and the consumer features of that ST (marketing effectiveness) throughout its life cycle under different conditions of application. Marketing units should work closely with the software support units, as the support team usually receives information not only about ST errors found during their operation, but also suggestions on ways to improve the software.

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

The clarity of the results of the quality control. For each software at the earliest stages of development quite simple and clear criteria (signs) of high quality and lack of design should be set. Information on the progress of software development and the results of its monitoring should be clear and publicly available. Software developers should always be prepared not only to guarantee high quality ST, but also to demonstrate it convincingly.

Existence of a comprehensive quality assurance plan for the software developed. The plan includes a set of measures to ensure and maintain the required level of quality of software, distributed by contractors, in time and by material resources. It is based on the specification of the requirements for the software, the knowledge of the quality factors, the specifics of the ST being developed, and the necessary resources for implementation. The plan is developed at the same time with the development of the TOR as an appendix to it. The listed quality factors (first order factors) are common to all types of software products and to all the attributes of these products. In addition to these factors, it is possible to distinguish into separate groups such factors (second-order factors), which most significantly influence the formation of a specific attribute or group of software attributes. The specific attribute of the ST in this case can be considered as a consequence of the actions of the selected factors.

8. Errors in software and ways to prevent them

Errors in the programs of automated process control systems lead to the violation of technological regimes and the production of defective products. Errors in automated organizational management systems lead to irrational use of material resources and labor costs. In some cases, bugs in the programs can have catastrophic consequences. In addition, bugs in the software, poor quality, or lack of quality assurance for individual software are reasons for poor implementation rates. With the implementation of software containing gross errors, in tens or even hundreds of enterprises, the negative effect will increase an appropriate number of times. This effect is exacerbated by the need to involve in the search and eliminate the mistakes of many of the most qualified professionals who are doing the job with the detriment of their kernel business. It is an admitted pattern that the earlier a project error is detected, the easier it is to correct it. The dependence of the relative cost of bug fixing on the time of its detection is shown in table 1.

Table 1

The dependence of the relative cost of correcting the error on the time of detection

Stage of the life cycle	The relative cost of bug fixing
Development of the TOR	0,1…0,2
Sketch design	0,3
Technical engineering	0,5
Programming	1,0
Combined testing	2,0
Preliminary tests	3,0
Experimental operation	4,0
Acceptance Tests	5,0
Operation	20…30

Therefore, ST bug prevention measures in the early stages of design should take a special place in software quality management systems in development organizations (enterprises). In order to develop effective measures to prevent software bugs, it is first of all necessary to establish their nature, causes and symptoms. To understand the nature of bugs, it is needed to consider the following characteristics: *nature of the external manifestation, physical essence*, *stages of introduction*, *nature of bugs*, their *types* and *classification*. Any program, after all, is a set of instructions, the execution of which provides the conversion of the varied initial data to the desired result. An error (a set of errors) in the program leads to an incorrect result. This is the essence of the external manifestation of bugs in the program. The physical essence of the software product is a record of the program on a data carrier. Therefore, the physical expression of the error is the incorrect entry of any element of the program (commands, macros, elementary construction, operator, data set, etc.). The error correction process in this case is a replacement the incorrect entry with the correct one. Thus, an error in the software product from the end-user perspective is the entry of a program element on a data carrier or in the software documentation, which results in the wrong result being sought. Note that this definition allows the correct result to be obtained in the presence of errors in the program. This is possible indeed in cases where program elements containing bugs are not used in specific implementation conditions. The elements of the program can be not only prescriptions for the order of conversion of the initial data into the desired result, but also records of quantities, descriptions of variables, etc.

Therefore, the definition indicates the use, not execution of the program element. Bugs in ST can be made at different stages and phases of their life cycle. Accordingly, there are *errors in the statement of the task*, in *designing*, in *programming* and *in recording on the data carrier*.

Errors in formulating a ST development task. The formulating a ST development task is formulated in the form of terms of reference and technical conditions. These documents define the consumer attributes of the ST, which must take into account all requirements of potential users. In turn, user requirements should be based on knowledge of the purpose and conditions of use of the ST. Thus, to understand the tasks of development means, first of all, to set the aim and purpose of development, conditions of application, expected ranges of input data and results. Misunderstanding of the problem being solved, inaccurate knowledge of the initial data, conditions of operation and expected results lead to errors in the formulation of the task, resource planning, which may eventually make all further work of the designers unnecessary. The requirements for the quality (specifications of quality) of the ST should be an integral part of the general technical requirements. Moreover, they must be comprehensive and well-grounded. Otherwise, the ST will be disabled. There are situations in which the TOR for development did not have the requirements for the stability of ST ACS in the presence of distortive effects. Such ST had to be radically modified immediately after experimental operation. If the ST has the ability to be modified, then the problem of improving the stability of the software will be solved. Otherwise, the design process must be started from the beginning.

Design errors. Design errors include: errors in the choice of methods for solving problems and parameters; inconsistency in the use of data in time (in real time systems); neglect of correlation between individual components, etc. All these errors can be qualified as the inadequacy of mathematical models to real processes occurring in the system, to researched processes. Design errors are sometimes referred to as algorithmic errors because they are formally contained in problem solving algorithms. All the errors that are not detected at the stage of algorithm development are subsequently transformed into programming errors.

Programming errors. Modern programming languages and translators contain some set of tools for debugging and checking programs. However, these tools are not enough to guarantee error-free programming. Therefore, programming (encoding) is also a source of ST errors.

Software errors include errors in the choice of numerical methods of implementation of algorithms for solving problems, schemes and calculations; interpretation of algorithmic constructions (semantic errors); coding (syntax errors); in conjunction of program modules and programs; in the implementation of logical conditions; in the data description; in the documentation.

Errors while recording on data carrier. Compiled program text must be recorded to a specific data carrier before entering the computer. This work is mostly done manually and can cause new errors. The percentages of these errors are small because they are easily controlled and eliminated. ST errors can also be introduced during operation and maintenance. Such errors are the result of unqualified correction of predicted errors, unqualified ST modification, negligent treatment of data carriers, etc.

The classification of errors considered is a priori. It is based on the types of ST creation and operation work (at the stages of the ST life cycle). This classification is useful for forecasting errors at different stages, assessing the quality of work of teams specializing in the performance of particular types of work, and making the necessary decisions. For example, data input/output errors are symptomatic because they have external characteristics (symptoms), which, however, do not allow to explicitly identify the causes of these errors. Computational errors usually directly indicate the true cause (error in sign, index, etc.), but have no characteristic features.

Table 2 shows the distribution of errors by ordering the signs of causes by frequency of occurrence. An attempt to establish the interdependence of causes and signs of manifestation of errors was made. The general pattern was not established, but it was possible to identify the signs of errors that are most common in these projects. These include: -bit grid overflow – 30.4%; incorrect management transfer – 16.4%; incompatibility of programs with databases – 14.5%; incompatibility of programs by the types of data being forwarded – 9%; failure to perform additional functions by the program – 4,9%, incompatibility of programs – 7%.

Table 2

Error distribution by frequency of occurrence

Type of an error	Error distribution, % from total quantity	Type of an error	Error distribution, % from total quantity
Calculations	7	Of interface	10
Logical	22	Database initialization	6
I/O	10	In the documentation	8
Data manipulation	15	Other	22

Collecting, processing error data, classifying errors, establishing their causes and probabilities make it possible to do purposeful work on error prevention and thus affect the quality of ST.

REFERENCES

1. Antipensky V.E., Bilousko V.S., Chujdan T.I. Computing Machines and Programming: Workshop. Kyiv: Higher School. Main issue, 1987. 245 p.
2. Ivashchenko N.N. Automatic regulation. Theory and elements of systems. Textbook for universities. Ed. 4th, rework. and ext. Moscow: Mechanical Engineering, 1978. 236 p.
3. Miroshnik I.V. Automatic control theory. Linear systems. St. Petersburg: Peter, 2005. 336 p.: pic. (Training Series).
4. Popovich M.G., Kovalchuk M.G. Automatic control theory: a textbook. 2nd edition, revised. and suppl. Kyiv: Libid, 2007. 656 p.
5. Tsypkin Ya.Z. Fundamentals of automatic systems. Main Editing Physical and Mathematical Literature Publishing "Science", Moscow, 1977, 56 p.

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

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

SOFTWARE AND METHODOLOGICAL COMPLEX OF SYSTEMS

Kyselov V. B.

1. Standardization of quality systems

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

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

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

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

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

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

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

2. Choosing a Quality Indicators Nomenclature

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

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

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

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

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

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

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

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

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

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

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

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

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

3. Quality Indicators Groups

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4. Purpose indicators

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5. Performance reliability Indicators

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6. Ergonomic indicators

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

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

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

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

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

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

7. Indicators of manufacturability

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

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

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

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

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

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

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

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

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

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

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

8. Constructive indicators

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

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

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

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

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

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

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

9. Unification indicators

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

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

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

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

10. Multilevel hierarchy of structure of properties and quality indicators

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

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

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

11. Quality and efficiency of software. Quality Economy

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

13. Selection of basic samples of quality indicators

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

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

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

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

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

14. Methods for assessing the quality level of software

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

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

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

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

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

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

REFERENCES

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

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

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

DOI https://doi.org/10.36059/978-966-397-107-0/95-123

RESOLUTION METHODS AND APPLIED PROBLEMS OF GAME THEORY

Medvediev M. H.

1. Methods for Solving Matrix Games

Let the game involve two parties *A* and *B.* The playing field is given by the payoff matrix (payment matrix – table 1):

Table 1

A		B				
		B_1	...	B_j	...	B_n
		x_1	...	x_j	...	x_n
A_1	y_1	a_{11}	...	a_{ij}	...	a_{1n}
...	...	...	...	...	...	...
A_i	y_i	a_{i1}	...	a_{ij}	...	a_{in}
...	...	...	...	...	...	...
A_m	y_m	a_{m1}	...	a_{mj}	...	a_{mn}

The strategy chosen by the party *A,* will be denoted as $A_1,A_2,...,A_t$; and side *B* strategy will be given as $B_1,B_2,...,\boldsymbol{B_n}$;$\boldsymbol{y_i}$ – probability of strategy use by the first party; x_j – the probability of using the *j* trategy by the second party *B. A vector is the first (second) player's mixed strategy*

$$\overline{\boldsymbol{y}} = (\boldsymbol{y_1}, \dots, \boldsymbol{y_m}), \overline{\boldsymbol{x}} = (\boldsymbol{x_1}, \dots, \boldsymbol{x_n}),$$

for which

$$\sum_{i=1}^{m} y_1 = 1;\ \sum_{j=1}^{n} x_i = 1;\ y_i \geq 0 (i = \overline{1,m});\ x_j \geq 0 (j = \overline{1,n}).$$

Elements of the payoff matrix can be positive, negative, or equal to zero. If the element of the matrix is positive, then party *B* in a certain situation pays the party *A* a sum of money equal to the element of the matrix.

If the element of the matrix is negative, then party *A* pays party *B* a um of money equal to the absolute value of the element. If the element is zero, no payment is made.

We will consider *zero-sum paired games*[1].

These are games whose payment amount is zero, that is, the loss of one player is equal to the win of another. In this case, the average gain (loss) – a mathematical expectation is a function of mixed strategies $\overline{\boldsymbol{x}}, \overline{\boldsymbol{y}}$:

Function *S (x*, y) is called *a payment function* of the game with matrix $\left[\boldsymbol{a_{ij}}\right]_{\boldsymbol{mxn}}$.

Strategies $\overline{\boldsymbol{y}^*} = (\boldsymbol{y_1^*}, \dots, \boldsymbol{y_m^*}), \overline{\boldsymbol{x}^*} = (\boldsymbol{x_1^*}, \dots, \boldsymbol{x_n^*})$ are called *optimal*, if for the random strategies $\overline{\boldsymbol{y}} = (\boldsymbol{y_1}, \dots, \boldsymbol{y_m}), \overline{\boldsymbol{x}} = (\boldsymbol{x_1}, \dots, \boldsymbol{x_n})$ these requirements are satisfied

$$S(\bar{y}, \overline{x^*}) \leq S(\overline{y^*}, \overline{x^*}) \leq S(\overline{y^*}, \bar{x}). \qquad (1)$$

Using the optimal mixed strategies $\overline{\boldsymbol{y}^*}, \overline{\boldsymbol{x}^*}$in game gives the first player a win no less than while using any other strategy $\overline{\boldsymbol{y}}$;and gives the second player a loss no bigger than while using any other strategy $\overline{\boldsymbol{x}}$.

The value of the payment function with optimal strategies determines *the price of the game C,* i.e $\boldsymbol{C = S(\overline{y}^*, \overline{x}^*)}$

The combination of optimal strategies and the price of the game is the solution of the game.

It is proved that in order for the number C to be the price of the game, and $\overline{\boldsymbol{y}}^*$ and $\overline{\boldsymbol{x}}^*$ to be optimal strategies, it is necessary and sufficient the inequalities to work

$$\sum_{i=1}^{m} a_{ij}\, y_i^* \geq C(j = \overline{1,n});\ \sum_{j=1}^{n} a_{ij} x_j^* \leq C(i = \overline{1,m}). \qquad (2)$$

In the future, for certainty, assume that $\mathrm{C} > \mathbf{0}$. This can always be achieved by that the adding to all elements of the payoff matrix the same constant number *d* does not change the optimal strategies, but only increases the price of the game for *d.*

[1] Neumann D., Morgenstern O. Theory of Games and Economic behavior. Moskow: Science, 1970, 708 p.

1.1 Reduction of problems of theory of games to problems of linear programming By dividing both parts of the first of inequalities (15) by C, we get the system in the expanded form[2]:

$$\begin{cases} a_{11}y'_1 + a_{21}y'_2 + \cdots + a_{m1}y'_m \geq 1, \\ a_{12}y'_1 + a_{22}y'_2 + \cdots + a_{m2}y'_m \geq 1, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{1n}y'_1 + a_{2n}y'_2 + \cdots + a_{m2}y'_m \geq 1, \end{cases} \quad (3)$$

where $y'_1 = \frac{y_i^*}{C} (i = \overline{1, m})$; $y'_1 \geq 0$.

Using the last notation, condition $\sum_{i=1}^{m} y_i^* = 1$ can be written as $\sum_{i=1}^{m} y_i'^* = \frac{1}{C}$

As the first player tries to get the maximum win, he must provide a minimum value of 1/ *C.* With this in mind, determining the optimal strategy for the first player comes down to finding the minimum value of the function

$$\mathbf{f} = \sum_{i=1}^{m} y'_i = y'_1 + y'_2 + \cdots + y'_m \quad (4)$$

under conditions (16).

Similar considerations show that determining the optimal second player's strategy comes down to finding the maximum value of the function

$$F = \sum_{j=1}^{n} x'_i = x'_1 + x'_2 + \cdots + x'_n \quad (5)$$

under conditions

$$\begin{cases} a_{11}x'_1 + a_{12}x'_2 + \cdots + a_{1n}x'_n \leq 1, \\ a_{21}x'_1 + a_{22}x'_2 + \cdots + a_{2n}x'_n \leq 1, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x'_1 + a_{m2}x'_2 + \cdots + a_{2n}x'_n \leq 1, \\ x'_j \geq 0 \ (j = \overline{1, n}) \end{cases}$$

where $x'_j = x_j^* / C$.

Thus, in order to find the solution of the game given by this payment matrix (see table. 1), it is necessary to make dual (conjugated) linear programming problems and solve them.

[2] Akulich I.L. Mathematical programming in problem examples. Moskow: Higher school, 1986, 318 p.

The straightforward problem is to find the maximum value of the function *F*, given by expression (5) under conditions (6).

Dual (conjugate) problem is find the minimum value of function $\boldsymbol{f}$ given by expression (4) under condition (3).

Using a solution of a pair of dual problems

$$\overline{\boldsymbol{y'^*}} = (\boldsymbol{y'^*_1}, \dots, \boldsymbol{y'^*_m}), \overline{\boldsymbol{x'^*_1}} = (\boldsymbol{x'_1}, \dots, \boldsymbol{x'^*_n}), \quad (6)$$

we get formulas for determining strategies and the price of the game:

$$\boldsymbol{y^*_i} = \frac{\boldsymbol{y'^*_i}}{\sum_{i=1}^{m} \boldsymbol{y'^*_i}} = \boldsymbol{C}\boldsymbol{y'^*_i}; \boldsymbol{x'^*_j} = \frac{\boldsymbol{x^*_j}}{\sum_{j=1}^{n} \boldsymbol{x'^*_j}} = \boldsymbol{C}\boldsymbol{x'^*_j}, \quad (7)$$

$$\boldsymbol{C} = \frac{\boldsymbol{1}}{\sum_{i=1}^{m} \boldsymbol{y'^*_i}} = \frac{\boldsymbol{1}}{\sum_{j=1}^{n} \boldsymbol{x'^*_j}}. \quad (8)$$

So, the process of finding a solution to the game using linear programming methods involves the following steps:

1. Assembling of a pair of dual (conjugate) linear programming problems that are equivalent to such a matrix game.
2. Determining optimal plans for dual problems.
3. Finding a solution to the game, using the relationship between dual problems' plans, optimal strategies and the price of the game.

According to these steps, we will solve the above-mentioned problem of supply of raw materials by linear programming methods. In this problem (game) the payment matrix is given in Table 2. In order for the price of game C to be greater than zero, we add the number d = 400 to all elements of this matrix. This, as mentioned above, will not change the optimal strategies, but will only increase the price of the game by d = 400. After that adding a payment matrix will look like

$$A = \begin{pmatrix} 300 & 0 \\ 250 & 100 \\ 210 & 150 \\ 70 & 200 \end{pmatrix}.$$

According to the first stage, we make a pair of dual (conjugate) linear programming problems that are equivalent to a given matrix game.

Direct problem (relations (5), (6)) is to find the maximum value of the function

$$F = \sum_{j=1}^{2} x'_i = x'_1 + x'_2 (n = 2; m = 4) \quad (9)$$

with restrictions

$$\begin{cases} \mathbf{300}x_1' \le \mathbf{1}, \\ \mathbf{250}x_1' + \mathbf{100}x_2' \le \mathbf{1}, \\ \mathbf{210}x_1' + \mathbf{150}x_2' \le \mathbf{1}, \\ \mathbf{70}x_1' + \mathbf{200}x_2' \le \mathbf{1}, \\ x_1', x_2' \ge \mathbf{0}. \end{cases} \quad (10)$$

Dual (conjugate) problem (relations (16) and (17)) is to find the minimum value of the function

$$f = \sum_{i=1}^{4} y_i' = y_1' + y_2' + y_3' + y_4' \quad (11)$$

with restrictions

Having solved the problems of linear programming (9) – (12) by the simplex method, we obtain

$$\boldsymbol{x_1'^* = 5/3150;\ x_2'^* = 14/3150;\ F_{max} = 19/3150;}$$

$$\boldsymbol{y_1'^* = 0;\ y_2'^* = 0; y_3'^* = 13/3150;\ y_4'^* = 6/3150; f_{min} = 19/3150.}$$

Substituting these solutions into relations (20) and (21), we obtain the optimal strategies of the firm *A:*

$$\boldsymbol{y_1^* = \frac{y_1'^*}{\sum_{i=1}^{4} y_i'^*} = \frac{0}{\frac{19}{3150}} = 0;\ y_2^* = \frac{y_2'^*}{\sum_{i=1}^{4} y_i'^*} = \frac{0}{\frac{19}{3150}} = 0;}$$

$$\boldsymbol{y_3^* = \frac{y_3'^*}{\sum_{i=1}^{4} y_i'^*} = \frac{\frac{13}{3150}}{\frac{19}{3150}} = 0,685;\ y_4^* = \frac{y_4'^*}{\sum_{i=1}^{4} y_i'^*} = \frac{\frac{6}{3150}}{\frac{19}{3150}} = 0,315,}$$

optimal strategies of the supplier company ***B:***

$$\boldsymbol{x_1^* = \frac{x_1'^*}{\sum_{j=1}^{2} x_j'^*} = \frac{\frac{5}{3150}}{\frac{19}{3150}} = 0,263;\ x_2^* = \frac{x_2'}{\sum_{j=1}^{2} x_j'^*} = \frac{\frac{14}{3150}}{\frac{19}{3150}} = 0,737}$$

and the price of the game

$$\boldsymbol{C = \frac{1}{\sum_{i=1}^{4} y_i'^*} = \frac{1}{\sum_{j=1}^{2} x_j'^x} = \frac{1}{\frac{19}{3150}} \approx 165,8.}$$

Since adding to all elements of the payment matrix the number d = 400 has increased the price of the game by 400, the true price of the game of the initial problem (expected losses of the firm *A)* will be

165.8 – 400 = -234.2 $

As it is easy to check, the optimal strategies and the price of the game found by linear programming methods are exactly the same as those found above using the graphical method.

Unlike the graphical method that can be applied when either $\boldsymbol{n \leq 2}$ or $\boldsymbol{m \leq 2}$, the linear programming method can be applied to arbitrary finite values *mi n*

1.2 An iterative (approximate) method for solving the problems of game theory Two approaches to solving the problems of game theory have been considered above: graphic and reduction to linear programming problems. In both cases there is an exact solution to the problems of game theory – the price and optimal mixed strategies of players A and B.

Let us now consider an approximate method for solving the problems of game theory, which reflects to some extent the real situation of the players' gradual accumulation of experience in adopting rational strategies as a result of many repetitions of conflict situations (games)[3].

This method allows you to simulate the process of training (behavior) of players during the repetition of the game, when each of them evaluates the behavior of the opponent and responds to it in the best way for themselves. Each time at the beginning of the game, they choose the most advantageous strategies for themselves, basing on the previous choices of the opponent.

Let us solve, using this method, the previous problem with firms *A* and *B,* for which the payment matrix is given in Table 2 in the case when the game is antagonistic.

On the first day after the conclusion of the contract, firms *A* and *B* accept random strategies, for example: firm *A* uses strategy A_3 *(*–190, –250), firm *B* uses strategy B_2 (–400, –300, –250, –200).

Let us build a model that describes the rules for choosing the next «moves» by firms *A* and *B.*

[3] Kudryavtsev E.M. Research of operations in problems, algorithms and programs. Moskow: Radio Communication, 1984, 184 p.

On the second day, the firm *A* chooses its strategy so that its win with the strategy B_2 of the company *B* was the maximum, i.e the losses, taking into account the signs of payment, were minimal (–200). Obviously, this will be the strategy A_4 (–330, –200).

Firm *B,* taking into account the previous day, chooses the strategy B_2 again to inflict the firm *A* with the greatest losses (-250) when its strategy is A_3.

On the third day, the firm *A* chooses its strategy so that its accumulated (total) losses for the previous two days with the strategies B_2 of the firm *B*

$$\left(\boldsymbol{S_{A_1}}, \boldsymbol{S_{A_2}}, \boldsymbol{S_{A_3}}, \boldsymbol{S_{A_4}}\right) = (\boldsymbol{-400}, \boldsymbol{-300}, \boldsymbol{-250}, \boldsymbol{-200}) + \\ +(\boldsymbol{-400}, \boldsymbol{-300}, \boldsymbol{-250}, \boldsymbol{-200}) = (\boldsymbol{-800}, \boldsymbol{-600}, \boldsymbol{-500}, \boldsymbol{-400})$$

were minimal (they are highlighted). Obviously, this will be the strategy A_4. Firm*B* selects its strategy on the same day, based on information on the strategies of the firm $\boldsymbol{A}$ for the previous two days, so that the total losses of the firm $\boldsymbol{A}$ with its strategies $\boldsymbol{A_3}$ $\boldsymbol{iA_4}$,

$$\left(\boldsymbol{S_{B_1}}, \boldsymbol{S_{B_2}}\right) = (\boldsymbol{-190}, \boldsymbol{-250}) + (\boldsymbol{-330}, \boldsymbol{-200}) = (\boldsymbol{-520}, \boldsymbol{-450}),$$

were maximal (they are highlighted). This is $\boldsymbol{B_1}$ strategy

On the fourth day, the situation is repeated. Firm *A,* Basing on the previous actions of the firm *B,* in three days chooses its strategy so that its total losses for these days with the strategies B_2, B_2, B_1 of the firm *B,*

$$\left(S_{A_1}, S_{A_2}, S_{A_3}, S_{A_4}\right) = (-800, -600, -500, -400) + \\ +(-100, -150, -192, -330) = (-900, -750, -690, -730),$$

were minimal. This is strategy A_3.

Firm *B,* whose purpose is to maximize the losses of the firm *A* with its strategies A_3, A_a,A_4,

$$\left(\boldsymbol{S_{B_1}}, \boldsymbol{S_{B_2}}\right) = (\boldsymbol{-520}, \boldsymbol{-450}) + (\boldsymbol{-330}, \boldsymbol{-200}) = (\boldsymbol{-850}, \boldsymbol{-650}),$$

chooses the strategy B_1.

In the following days, the situation is repeated, the behavior of the choice of strategies by firms *A* and *B* does not change, its results are shown in table 2:

Table 2

n	l	S'_{B_1}	S'_{B_2}	S'_n	j	S'_{A_1}	S'_{A_2}	S'_{A_3}	S'_{A_4}	S''_n	$\bar{S}_n$	r_n	R_n	$\bar{r}_n$
1	3	—190	—250	—250	2	—400	—300	—250	—200	—200	—225	—250	—250	—250
2	4	—520	—450	—260	2	—800	—600	—500	—400	—200	—230	—200	—450	—225
3	4	—850	—650	—283	1	—900	—750	—690	—730	—230	—257	—330	—780	—260
4	3	—1040	—900	—260	1	—1000	—900	—880	—1060	—220	—240	—190	—970	—243
5	3	—1230	—1150	—246	1	—1100	—1050	—1070	—1390	—210	—228	—190	—1160	—232
6	2	—1380	—1450	—242	1	—1200	—1200	—1260	—1720	—200	—221	—150	—1310	—218
7	1	—1480	—1850	—264	2	—1600	—1500	—1510	—1920	—214	—239	—400	—1710	—244
8	2	—1630	—2150	—269	2	—2000	—1800	—1760	—2120	—220	—244	—300	—2010	—251
9	3	—1820	—2400	—267	2	—2400	—2100	—2010	—2320	—223	—245	—250	—2260	—251
10	3	—2010	—2650	—265	2	—2800	—2400	—2260	—2520	—226	—246	—250	—2510	—251
11	3	—2200	—2900	—264	2	—3200	—2700	—2510	—2720	—228	—246	—250	—2760	—251
12	3	—2390	—3150	—263	2	—3600	—3000	—2760	—2920	—230	—246	—250	—3010	—251
13	3	—2580	—3400	—262	2	—4000	—3300	—3010	—3120	—232	—247	—250	—3260	—251
14	3	—2770	—2650	—261	2	—4400	—3600	—3260	—3320	—233	—247	—250	—3510	—251
15	3	—2960	—3900	—260	2	—4800	—3900	—3510	—3520	—234	—247	—250	—3760	—251
16	3	—3150	—4150	—259	2	—5200	—4200	—3760	—3720	—233	—246	—250	—4010	—251
17	4	—3480	—4350	—256	2	—5600	—4500	—4010	—3920	—231	—243	—200	—4210	—248
18	4	—3810	—4550	—253	2	—6000	—4800	—4260	—4120	—229	—241	—200	—4410	—245
19	4	—4140	—4750	—250	2	—6400	—5100	—4510	—4320	—227	—239	—200	—4610	—243
..	..	...	...	...	...	...	...	...	...	...	...	...	...	...

where *n* denote the number of days elapsed after the conclusion of the contract, or a pair of successive strategies («moves») of the firms *A* and *B;*

i denotes the strategy number selected by the company A;

$\boldsymbol{S'_{B_1}S'_{B_2}}$*- denote accumulated (common) losses of the firm A for the first n*

days using the strategies B_1, B_2 of the company *B;*

$\boldsymbol{S'_n}$- maximum average losses of the firm *A*, which are equal to the maximum accumulated losses for the first *n* days divided by the number of these days;

j – denote the strategy number selected by B.

$\boldsymbol{S'_{A_1}, S'_{A_2}, S'_{A_3}, S'_{A_4}}$are accumulated (general) losses of the firm *A* for the first days according to its strategies respectively A_1, A_2, A_3,$\boldsymbol{A_4}$;

$\boldsymbol{S'_n}$is the minimum average losses of the firm *A*, equal to the minimum accumulated losses for the first *n* days divided by the number of these days;

$\boldsymbol{\overline{S_n}}$denotes an average value of maximum $(\boldsymbol{S'_n})$ and minimum $(\boldsymbol{S''_n})$average losses of firm *A;*

$\boldsymbol{r_n}$*- denotes real company A losses for each day;*

$\boldsymbol{R_n}$*- denotes actual accumulated losses of the firm A for n days;*

$\boldsymbol{\overline{r_n}}$is the real average losses of the firm *A* in one day, which are added with the accumulated real losses for the first *n* days divided by the number of these days.

Table 2 shows that with increasing *n* all three values:

$\boldsymbol{S'_n, S''_n}$ i $\boldsymbol{\overline{S_n}}$ approach the exact value of losses (price of the game) of the company *A*, which equals to $234,2. and were previously found by the graphical method (§1.2), but the average $\boldsymbol{\overline{S_n}}$ coincides relatively faster since $\boldsymbol{S'_n < -234,2 < S''_n}$.

The mixed strategies of the firms *A* and *B* also increase with their exact values as they increase *n* (see §1.2, 1.4), respectively

$\boldsymbol{\overline{y^*} = (0; 0; 0,685; 0,315), \overline{x^*} = (0,263; 0,737)}$, but slowlier.

For example, after *n*=19 repetitions of the game (days), the approximate values of losses of the firm *A*(the price of game) $\boldsymbol{\overline{S_{19}} = -293\ y.o.}$, and the approximate values of mixed strategies of firms *A* i *B*are often determined by their clean strategies:

$$\boldsymbol{y_{19} = \left(\frac{1}{19}, \frac{2}{19}, \frac{11}{19}, \frac{5}{19}\right) \approx (0,053; 0,105; 0,579; 0,263);}$$

$$\boldsymbol{x_{19} = \left(\frac{4}{19}, \frac{5}{19}\right) \approx (0,211; 0,789).}$$

For comparison, the last three columns of table 13 provide real information about the course of the game (each game implementation), which shows that the model (algorithm) adequately reflects the behavior of the players (firms *A* and *B)* during the repetition of the game and allows them to determine their optimal strategies and the price of the game (losses of the company *A).*

It can be seen from the above that the iterative method is practical and universal at the same time. Using it, you can easily find an approximate solution to any matrix game. The volume and complexity of calculations increase relatively slowly as the matrix game size increases.

1.3 Direct Solution of Matrix Games

In principle, any matrix game can be solved by inequalities (15). But it requires a lot of calculations, which increases with the increment of number of players. Therefore, if possible, reduce the number of clearplayer strategies using the «dominance principle» that is as follows[4].

If the elements of some row of the payoff matrix are smaller than the corresponding elements of some other row of the same matrix, then the last row dominate the first. The first row is removed from the matrix. The case with columns is similar, only the column with larger elements is removed.

Further we have to check the inequalities (15). If inequation (15) is fulfilled, then players have pure optimal strategies (the player has the pure maximin strategy and the player the pure minimax). And if not, at least one player's optimal strategies will be mixed.

Let us consider the principle of dominance on the example of the problem of planning the production of by-products (antagonistic case).

1.4 The problem of planning the production of by-products (antagonistic case)

Let it be: in some city there are two enterprises, which in addition to their main products may produce some by-products of the same purpose for the population, but it may be different in design and convenience, etc. Let us suppose that enterprise *A* A_1, A_2, A_3, A_4, A_5, and enterprise *B* produces byproducts of type B_1, B_2, B_3, B_4, B_5. The cost and sales price of all products are the same. Demand forecasting sociologists have determined that N=1000 units will be sold; moreover, if the first enterprise *A* (player I) will

[4] Dyubin G. N., Suzdal V. G. Introduction to Applied Game Theory. Moskow: Science, 1981, 336 p.

produce products of type $\boldsymbol{A_i}$, and the second enterprise *B* (player II) – products of type $\boldsymbol{B_j}$, then the city will find sales $\boldsymbol{p_{ij}N}$ of goods of type$\boldsymbol{A_i}$ and $\boldsymbol{(1-p_{ij})\,N}$ of goods of type $\boldsymbol{B_j}, \boldsymbol{0 \le p_{ij} \le 1}$.. The capacity of the enterprises is such that each of them can provide the city. Taking the profit from the sale of a unit of goods equal to one, and the usefulness of the player I equals its profit, the payoff matrix *H* of player I can be written as follows:

$$\boldsymbol{H = \left(p_{ij}\,N\right)_{i=1,\dots,m,\ j=1,\dots,m.}}$$

Similarly, the payoff matrix of player II is written, whose element (i, j) is $\boldsymbol{(1-p_{ij})N}$. Since in any situation the sum of profits of players I and II is equal to the same number $\boldsymbol{= p_{ij}N + (1-p_{ij})N}$, an increase of player I winnings is equivalent to a decrease of player II winnings, i.e the interests of players are opposite. Therefore, player II, minimizing sales $\boldsymbol{p_{ij}N}$of goods $\boldsymbol{A_i}$ of player I, maximizes $\boldsymbol{(1-p_{ij})N}$ sales of his goods B_j Therefore, the game given by the matrix *H*, simulates an antagonistic game.

The solution of the game determines the optimal strategies $\overline{\boldsymbol{x}}^*, \overline{\boldsymbol{y}}^*$for players I and II, respectively, as well as the mathematical expectation of winning of player I is equal to $\boldsymbol{S(\overline{y}, \overline{x})}$ In this game, the mathematical expectation of winning of player II is equal to $\boldsymbol{-S(\overline{y}, \overline{x})}$. Since the sum of goods sold equals to*N*, the mathematical expectation of goods sold by the enterprise *B* equals to $\boldsymbol{N - S(\overline{y}, \overline{x})}$.

Let us consider the solution of the game on a specific numerical example. Suppose that the estimated share of sales of enterprise *A* products is given in Table 3.

Table 3

Підприємство *A*	**Підприємство *B***				
	B_1	**B_2**	**B_3**	**B_4**	**B_5**
A_1	0,5	0,5	0,4	0,5	0,2
A_2	0,5	0,4	0,7	0,1	0,6
A_3	0,2	0,3	0,4	0,1	0,7
A_4	0,3	0,6	0,7	0,3	0,2
A_5	0,4	0,4	0,3	0,0	0,2

It is necessary to determine the types of products produced by each enterprise. In this case, the player's I payoff matrix will look like this

$$\boldsymbol{H} = \begin{pmatrix} \mathbf{500} & \mathbf{500} & \mathbf{400} & \mathbf{500} & \mathbf{200} \\ \mathbf{500} & \mathbf{400} & \mathbf{700} & \mathbf{100} & \mathbf{600} \\ \mathbf{200} & \mathbf{300} & \mathbf{400} & \mathbf{100} & \mathbf{700} \\ \mathbf{300} & \mathbf{600} & \mathbf{700} & \mathbf{300} & \mathbf{200} \\ \mathbf{400} & \mathbf{400} & \mathbf{300} & \mathbf{0} & \mathbf{200} \end{pmatrix}.$$

Noting that it is enough to solve the game with a matrix of winnings $\boldsymbol{H^1} = \frac{\mathbf{1}}{\mathbf{100}}\boldsymbol{H}$, i.e

$$\boldsymbol{H^1} = \begin{pmatrix} \mathbf{5} & \mathbf{5} & \mathbf{4} & \mathbf{5} & \mathbf{2} \\ \mathbf{5} & \mathbf{4} & \mathbf{7} & \mathbf{1} & \mathbf{6} \\ \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{1} & \mathbf{7} \\ \mathbf{3} & \mathbf{6} & \mathbf{7} & \mathbf{3} & \mathbf{2} \\ \mathbf{4} & \mathbf{4} & \mathbf{3} & \mathbf{0} & \mathbf{2} \end{pmatrix}.$$

The game with the payoff matrix H^1 is called the *subgame* of the game with the matrix *H*. The set of pure strategies of each of the players in the game is contained in the set of its pure strategies in the game itself, from which it follows that the set of mixed strategies of each player in the subgame is contained in the set of the mixed strategies of the game.

We apply the principle of dominance. It is easy to determine that the elements of the fifth row of the matrix H^1 are not greater than the corresponding elements of the first row, and therefore the first strategy of player I dominates the fifth. In addition, the elements of the first and second columns are not less than the corresponding elements of the fourth column. Therefore, player's fourth strategy dominates his first and second strategy. According to the principle of dominance, we remove the fifth row and the first and second columns. Obtain a subgame of the game with the payoff matrix H^1, which in the matrix form is given by the matrix

$$\boldsymbol{H^2} = \begin{pmatrix} \mathbf{4} & \mathbf{5} & \mathbf{2} \\ \mathbf{7} & \mathbf{1} & \mathbf{6} \\ \mathbf{4} & \mathbf{1} & \mathbf{7} \\ \mathbf{7} & \mathbf{3} & \mathbf{2} \end{pmatrix}.$$

Note that the *i*th row of the matrix H^2 is corresponded by *i*th strategy, and*j*th column – (*j* + 2)-th strategy of the game $\boldsymbol{H^1}$. Analysis of the matrix $\boldsymbol{H^2}$ shows that the third strategy of player II is dominated by a

mixed strategy that uses fourth and fifth strategies with the probabilities 3/5 and 2/5 respectively. According to the principle of dominance, we remove the first column of the matrix $\boldsymbol{H^2}$ and get a subgame with a matrix

$$\boldsymbol{H^3} = \begin{pmatrix} \mathbf{5} & \mathbf{2} \\ \mathbf{1} & \mathbf{6} \\ \mathbf{1} & \mathbf{7} \\ \mathbf{3} & \mathbf{2} \end{pmatrix}.$$

any solution of which is the solution of the game H^2, and game $\boldsymbol{H^1}$ i H.

From the analysis of the matrix H^3 it is easy to determine that the elements of the second row are not larger than the corresponding elements of the third row, and the elements of the fourth row are not greater than the corresponding elements of the first row. Therefore, the first and third strategies of player I dominate respectively the fourth and second strategies of player I.

Again, using the dominance principle, we obtain a subgame with a matrix

$$\boldsymbol{H^4} = \begin{pmatrix} \mathbf{5} & \mathbf{2} \\ \mathbf{1} & \mathbf{7} \end{pmatrix}.$$

Let us see if the game has a solution in pure strategies, with optimal strategies of players I and II respectively being a pure maximin strategy and a pure minimax strategy. However, if the game with a payoff matrix $\boldsymbol{H^4}$ is not solved in pure strategies, then both players have only optimal strategies that use all their pure strategies with positive probabilities.

The matrix $\boldsymbol{H^4}$ does not have *saddle point*, because the equation of elements is not satisfied

$$\max_i \min_j \boldsymbol{h_{ij}} = \min_j \max_i \boldsymbol{h_{ij}}$$

matrix $\boldsymbol{H^4}$, i.e the optimal strategies of the players are mixed.

Let $\overline{\boldsymbol{x}}$ – be a random mixed strategy of player I. If $\boldsymbol{x_1}$ is the probability of a player's choice of his first strategy in terms of $\overline{\boldsymbol{x}}$, then the probability of him choosing a second strategy is $\mathbf{1} - \boldsymbol{x_1}$. Similarly, if $\overline{\boldsymbol{y}}$ is a random mixed strategy of player II, then it looks like $(\boldsymbol{y_1},\ \mathbf{1} - \boldsymbol{y_1})$. It is easy to prove that the optimal strategies of players I and II

$$\overline{\boldsymbol{x^*}} = (\boldsymbol{x_1^*}, \mathbf{1} - \boldsymbol{x_1^*}), \overline{\boldsymbol{y^*}} = (\boldsymbol{y_1^*}, \mathbf{1} - \boldsymbol{y_1^*})$$

are calculated by the formulas

$$x_1^* = \frac{h_{22} - h_{21}}{h_{11} - h_{12} - h_{21} + h_{22}}, y_1^* = \frac{h_{22} - h_{12}}{h_{11} - h_{12} - h_{21} + h_{22}},$$

and the payment function of the game is equal to

$$S(\overline{y^*}, \overline{x^*}) = \frac{h_{11}h_{22} - h_{12}h_{21}}{h_{11} - h_{12} - h_{21} + h_{22}}.$$

As a result of calculations we get

$$x_1^* = 2/3\,, y_1^* = 5/9\,, S(y, x) = 11/3.$$

Strategies $\overline{x^*} = (2/3\,, 1/3)$ and $\overline{y^*} = (5/9\,, 4/9)$are consistent to strategies $\overline{x^*} = (2/3\,, 0, 1/3, 0, 0)$ and $\overline{y^*} = (0, 0, 0, 5/9, 4/9)$ of the initial game. The value of the game with the payoff matrix *H* is equal to 1100/3.

The result means that the enterprise *A* selects the production A_1 i A_3 with probabilities that are equal to 2/3 and 1/3 respectively, and the enterprise *B* – production B_4 *and* $B_{and\ 5}$ with probabilities of 5/9 and 4/9 respectively. Thus the mathematical expectation of the number of goods sold by enterprises *A* and *B* will be equal to 1100/3 and 1900/3 respectively.

2. Non-zero-sum bi-matrix games

Above, the zero-sum paired games, which are entirely determined by one payment matrix, were considered (Table 12). *The optimal strategies* are the following *strategies* $\overline{y^*}$ *and* $\overline{x^*}$ respectively for the parties *A* and *B*, which satisfy the conditions (15), under which it is not advantageous to deviate from these strategies for any player. This is called the *equilibrium situation.* It proves that zero-sum games always have at least one optimal solution $(\overline{y^*}, \overline{x^*})$, i.e at least one equilibrium point with the price of the game $C = S(\overline{y^*}, \overline{x^*})$. As a rule, such a solution is unique[5].

But, even when there are no such points of equilibrium, the price of the game is always the same and is equal to $C = S(\overline{y_i^*}, \overline{x_i^*})(i = 1, 2, \ldots)$. Therefore, such equilibrium points are considered equivalent and in the general case one can assume that zero-sum games always have the only optimal solution.

[5] Zamkov OO, Tolstenko AV, Cheremnykh Yu.N. Mathematical Methods in Economics. Moskow: DIS, 1997, 368 p.

Unlike zero-sum games, there are non-zero-sum games where it is not necessary for one player to win and the other to lose; they can both win and lose at the same time.

As the interests of players in such games are not completely opposite, their behavior becomes more diverse. For example, if a zero-sum game made it unprofitable for each player to tell his or her strategy to the other (this could reduce his or her winnings), then in a non-zero-sum game, it becomes desirable to coordinate with or influence a partner in some way.

Non-zero-sum games are also called *bimatrix*, as they are defined either by two matrices indicating the payments (winnings) of each party *A* and *B:*

$A \setminus B$	B_1	...	B_n		$A \setminus B$	B_1	...	B_n
A_1	a_{11}	...	a_{1n}		A_1	b_{11}	...	b_{1n}
....	...	...	...		...	...	...	...
A_m	a_{m1}	...	a_{mn}		A_m	b_{m1}	...	b_{mn}

or by one block matrix whose elements are pairs or blocks (a_{ij}, b_{ij}),

A/B	B_1	...	B_n
A_1	(a_{11}, b_{11})	...	(a_{1n}, b_{1n})
...	...	...	...
A_m	(a_{m1}, b_{m1})	...	(a_{mn}, b_{mn})

There are two types of bimatrix games – *non-cooperative games*, that prohibit any co-operation of the parties, and *cooperative games*, that allow such cooperation. It is obvious that cooperative games are a more complex object of study (at least because forms of cooperation can be diverse).

3. Non-cooperative games

In most economic, industrial, military, political, environmental, and **adaptive maintenance**administrative-legal conflicts, the purpose of each participant is to obtain as much individual gain as possible. All participants in such conflicts, for example, can win at the same time. Therefore, the non-compliant interests of participants are not quite the opposite, which makes the conflict non-antagonistic. Such a conflict may be modeled *by a non-cooperative game* if it fulfills such conditions.

1. Conflict is determined by the non-antagonistic interaction of the participants.

2. The parties of the conflict cannot (or have no right) to make mutually binding agreements.

3. The parties' own actions are performed independently of each other, that is, each of them has no information about the actions taken by the other party; the results of these actions are estimated by the real numbers that determine the usefulness of the situation for each of the parties.

4. Each of the parties of the conflict knows, both for themselves and for others, the usefulness of any possible situation that may result from their interaction.

3.1 Situations (points) of equilibrium

Let us take a closer look at non-cooperative games. In this case, an important role is played by situations of equilibrium, characterized by the fact that it is disadvantageous for none of the parties to violate them. and earlier, through $\mathbf{y} = (\mathbf{y_1} \ldots, \mathbf{y_m})$, $\mathbf{x} = (\mathbf{x_1}, \ldots, \mathbf{x_n})$mixed strategies of players A and B.

Then their average winnings will be accordingly equal to

$$\mathbf{S_A}(\bar{\mathbf{y}}, \bar{\mathbf{x}}) = \sum_{\mathbf{i=1}}^{\mathbf{m}} \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{a_{ij}}\, \mathbf{y_i x_j};\ \mathbf{S_B}(\bar{\mathbf{y}}, \bar{\mathbf{x}}) = \sum_{\mathbf{i=1}}^{\mathbf{m}} \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{b_{ij}} \mathbf{y_i x_j}. \qquad (12)$$

If among the common strategies there are$\overline{\mathbf{y}^*} = (\mathbf{y}_1^*, \ldots, \mathbf{y}_\mathbf{m}^*)$ and $\bar{\mathbf{x}}^* = (\mathbf{x}_1^*, \ldots, \mathbf{x}_\mathbf{n}^*)$that satisfy the conditions

$$\mathrm{S_A}(\bar{\mathrm{y}}, \bar{\mathrm{x}}^*) \le \mathrm{S_A}(\bar{\mathrm{y}}^*, \bar{\mathrm{x}}^*);\ \mathrm{S_B}(\bar{\mathrm{y}}^*, \bar{\mathrm{x}}) \le \mathrm{S_B}(\bar{\mathrm{y}}^*, \bar{\mathrm{x}}^*) \qquad (13)$$

then using $\overline{\overline{\mathbf{y}^*}^*}$ and $\bar{\mathbf{x}}^*$ creates an equilibrium situation.

The theory holds that every non-cooperative bimatrix game has at least one equilibrium situation (point) determined by inequations (13). When such a point (pair) $(\bar{\boldsymbol{y}}, \bar{\boldsymbol{x}})$ is unique, it can be considered as the optimal strategies $\overline{\boldsymbol{y}^*}$and $\overline{\boldsymbol{x}^*}$ of the sides *A* and *B.*

Uncertainty arises when there is more than one equilibrium point that satisfies conditions (27). And, unlike zero-sum games, the winnings of the parties *A* and *B* in these points differ – they are not equivalent.

Consider this situation using a simple example.

Let the block payment matrix (Table 4) look like this

Table 4

A \ B	B_1	B_2
A_1	(7,3)	(0,0)
A_2	(0,0)	(3,7)

By a straightforward substitution of formula (12), it is easy to check that pure strategies are $\overline{\boldsymbol{y_1}}^* = (\mathbf{1}, \mathbf{0})$, $\overline{\boldsymbol{x_1}}^* = (\mathbf{1}, \mathbf{0})$ and

$\overline{\boldsymbol{y_2}}^* = (\mathbf{1}, \mathbf{0}), \overline{\boldsymbol{x_2}}^* = (\mathbf{0}, \mathbf{1})$satisfy the equilibrium conditions. The winnings of the parties *A and B* at these points of equilibrium are respectively equal to

$$\boldsymbol{S_A}(\overline{\boldsymbol{y_1}}^*, \overline{\boldsymbol{x_1}}^*) = \mathbf{7};\ \boldsymbol{S_B}(\overline{\boldsymbol{y_1}}^*, \overline{\boldsymbol{x_1}}^*) = \mathbf{3};$$

$$\boldsymbol{S_A}(\overline{\boldsymbol{y_2}}^*, \overline{\boldsymbol{x_2}}^*) = \mathbf{3};\ \boldsymbol{S_B}(\overline{\boldsymbol{y_2}}^*, \overline{\boldsymbol{x_2}}^*) = \mathbf{7};$$

Now let us check whether there are points of equilibrium among the mixed strategies of the parties *A* and *B.*

Since

$$\boldsymbol{y_1} + \boldsymbol{y_2} = \mathbf{1}; \boldsymbol{x_1} + \boldsymbol{x_2} = \mathbf{1}; \boldsymbol{m} = \boldsymbol{n} = \mathbf{2},$$

then from relations (13) and Table 15 it implies that the average winnings of the parties *A* and *B* are respectively equal to

$$\boldsymbol{S_A}(\overline{\boldsymbol{y}}, \overline{\boldsymbol{x}}) = \mathbf{7}\boldsymbol{y_1}\boldsymbol{x_1} + \mathbf{3}\boldsymbol{y_2}\boldsymbol{x_2} = \mathbf{7}\boldsymbol{y_1}\boldsymbol{x_1} + \mathbf{3}(\mathbf{1} - \boldsymbol{y_1})(\mathbf{1} - \boldsymbol{x_1}), \quad (14)$$

$$\boldsymbol{S_B}(\overline{\boldsymbol{y}}, \overline{\boldsymbol{x}}) = \mathbf{3}\boldsymbol{y_1}\boldsymbol{x_1} + \mathbf{7}\boldsymbol{y_2}\boldsymbol{x_2} = \mathbf{3}\boldsymbol{y_1}\boldsymbol{x_1} + \mathbf{7}(\mathbf{1} - \boldsymbol{y_1})(\mathbf{1} - \boldsymbol{x_1}),$$

that is, S_A and S_B are functions from two variables $\boldsymbol{y_1}$ and$\boldsymbol{x_1}$:

$$\boldsymbol{S_A}(\boldsymbol{y_1}, \boldsymbol{x_1}) = \mathbf{7}\boldsymbol{y_1}\boldsymbol{x_1} + \mathbf{3}(\mathbf{1} - \boldsymbol{y_1})(\mathbf{1} - \boldsymbol{x_1});$$

$$\boldsymbol{S_B}(\boldsymbol{y_1}, \boldsymbol{x_1}) = \mathbf{3}\boldsymbol{y_1}\boldsymbol{x_1} + \mathbf{7}(\mathbf{1} - \boldsymbol{y_1})(\mathbf{1} - \boldsymbol{x_1}).$$

The equilibrium situation is characterized by the fact that it is not profitable for the side *A* to change its strategy $\boldsymbol{y_1}$, and for the side *B* – its strategy $\boldsymbol{x_1}$, because this will reduce their average winnings. It follows that the equilibrium conditions in this case have the form

$$\begin{cases} \dfrac{\boldsymbol{\delta S_A}}{\boldsymbol{\delta y_1}} = \mathbf{7}\boldsymbol{x_1} - \mathbf{3}(\mathbf{1} - \boldsymbol{x_1}) = \mathbf{0}, \\ \dfrac{\boldsymbol{\delta S_B}}{\boldsymbol{\delta x_1}} = \mathbf{3}\boldsymbol{y_1} - \mathbf{7}(\mathbf{1} - \boldsymbol{y_1}) = \mathbf{0}. \end{cases}$$

Solving this system of equations, we find the third equilibrium point among the mixed strategies for the sides *A* and *B:*

$$y_1^* = 0,7; y_2^* = 1 - 0,7 = 0,3; x_1^* = 0,3;\ x_2^* = 1 - 0,3 = 0,7,$$

that is

$$\bar{y}_3^* = (0,7; 0,3);\ \bar{x}_3^* = (0,3; 0,7)$$

with the winnings calculated by the formulas (28):

$$S_A(\bar{y}_3^*, \bar{x}_3^*) = 2,1;\ S_B(\bar{y}_3^*, \bar{x}_3^*) = 2,1.$$

It is easy to check that the equilibrium conditions (27) are satisfied at this point:

$$S_A(\bar{y}, \bar{x}_3^*) = 7y_1 \cdot 0,3 + 3(1 - y_1) \cdot 0,7 = 2,1 = S_A(\bar{y}_3^*, \bar{x}_3^*),$$

$$S_B(\bar{y}_3^*, \bar{x}) = 3x_1 \cdot 0,7 + 7(1 - x_1) \cdot 0,3 = 2,1 = S_B(\bar{y}_3^*, \bar{x}_3^*).$$

Obviously, the first situation (point) of equilibrium is more favorable for the side *A*, the second – for the side *B.* In the third equilibrium point, the parties' gains are the same, but they are smaller than in the first and second points. In the end, it is difficult to understand what the outcome of the parties *A* та *B* may be and how they should behave.

Thus, if there is more than one point (situation) of equilibrium, unambiguous recommendations for the choice of optimal strategies for the parties *A* and *B* cannot be given. In many cases, mutual contacts and agreements between the parties *A* and *B* make it possible.

In general, non-cooperative games are examined on a case-by-case basis.

3.2 The problem of planning the production of the by-product (non-antagonistic case)

Let us consider the problem of planning the production of the by-product (non-antagonistic case).

Suppose that two enterprises can produce by-products in the same production conditions as in the antagonistic case, but the possibility of selling these products has changed.

Now, according to sociologists, if the first enterprise (player I) will produce products of type $A_i (1 \le i \le m)$, and the second (player II) – products of type $B_j (1 \le j \le n)$, then the city will find sales $a_{ij}$of goods of type A_i and sales b_{ij} of goods of type B_j.

Since the sale of products of any enterprise depends on what products the other enterprise produces, and each enterprise tries to maximize the volume of sales, we have a production-trade conflict. This conflict is modeled by the game of the same players I and II with the same respectively *m* and *n* strategies as in the antagonistic game.

But this game is non-antagonistic, since the amount of products sold will now depend on the situation.

Taking the profit from the sale of units of goods equal to one, and the utility of players I and II equal their income, we model this conflict by a bi-matrix game given by a pair of matrices

$$\boldsymbol{A} = (\boldsymbol{a_{ij}})_{\substack{i=1,\dots,m \\ j=1,\dots,n}} \text{ і } \boldsymbol{B} = (\boldsymbol{b_{ij}})_{\substack{i=1,\dots,m \\ j=1,\dots,n}}$$

where $\boldsymbol{a_{ij}}$and $\boldsymbol{b_{ij}}$ – wins of the players I and II respectively in the situation*(i, j).*

Consider the solution of this game on a specific numerical example, assuming that companies I and II plan to produce by-products of types $\boldsymbol{A_i}\,(\boldsymbol{i = 1, 2})$ and$\boldsymbol{B_j}\,(\boldsymbol{j = 1, 2})$, respectively, and the expected profits from the sale of these products are given by the matrices:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{600} & \boldsymbol{300} \\ \boldsymbol{300} & \boldsymbol{900} \end{pmatrix} \text{ і } \boldsymbol{B} = \begin{pmatrix} \boldsymbol{500} & \boldsymbol{1500} \\ \boldsymbol{2000} & \boldsymbol{500} \end{pmatrix}.$$

It is necessary to determine the type of products that make sense for each enterprise.

Let us denote

$$\boldsymbol{A^* = a_{11} - a_{12} - a_{21} + a_{22}, a = a_{22} - a_{12};}$$

$$\boldsymbol{B^* = b_{11} - b_{12} - b_{21} + b_{22};, b = b_{22} - b_{21}.}$$

If$\boldsymbol{A^* \neq 0}$ and$\boldsymbol{B^* \neq 0}$, then the game has a balance of mixed strategies, namely

$$\boldsymbol{\overline{x}^* = (x_1^*, 1 - x_1^*), \overline{y}^* = (y_1^*, 1 - y_1^*)}$$

where

$$\boldsymbol{x_1^* = \frac{b}{B^*}, y_1^* = \frac{a}{A^*}.}$$

As a result of calculations we get

$$\boldsymbol{A^* = 900, a = 600;\ B^* = -2500, b = -1500.}$$

Therefore, the equilibrium situation is formed by vectors

$$\overline{\boldsymbol{x}}^* = (\mathbf{3/5}, \mathbf{2/5}), \overline{\boldsymbol{y}}^* = (\mathbf{2/3}, \mathbf{1/3})$$

and the mathematical expectation of the winnings of players I and II in the equilibrium situation will accordingly be

$$\boldsymbol{S_A}(\overline{\boldsymbol{y}}^*, \overline{\boldsymbol{x}}^*) = (\boldsymbol{a_{11} - a_{12} - a_{21} + a_{22}})\boldsymbol{x_1^* y_1^*} + (\boldsymbol{a_{12} - a_{22}})\boldsymbol{x_1^*} + \\ +(\boldsymbol{a_{21} - a_{22}})\boldsymbol{y_1^*} + \boldsymbol{a_{22}} = \mathbf{500}$$

$$\boldsymbol{S_B}(\overline{\boldsymbol{y}}^*, \overline{\boldsymbol{x}}^*) = (\boldsymbol{b_{11} - b_{12} - b_{21} + b_{22}})\boldsymbol{x_1^* y_1^*} + (\boldsymbol{b_{12} - b_{22}})\boldsymbol{x_1^*} + \\ +(\boldsymbol{b_{21} - b_{22}})\boldsymbol{y_1^*} + \boldsymbol{b_{22}} = \mathbf{1100}.$$

The result means that the enterprise *A* selects the production of type A_1 and A_2 with probabilities that are equal to 3/5 and 2/5 respectively, and the enterprise *B* – production of type B_1 and B_2 with probabilities of 2/3 and 1/3 respectively. Thus the mathematical expectation of the number of goods sold by enterprises *A* and *B* will be equal to $500 and $1100 respectively.

4. Cooperative games

4.1 Problem Statement

Most non-antagonistic conflicts in the economy and related industries are characterized by the fact that their participants can join forces through cooperation. Cooperation between players results in a qualitatively new conflict compared to a non-cooperative case.

As we have seen, in non-cooperative games, deviating one of the participants from the equilibrium situation does not give him any advantage. But if several players deviate, they can earn more than in the equilibrium situation. Therefore, in conditions where cooperation between players is possible, the principle of equilibrium does not come true.

For example, let a non-antagonistic game be given by the following matrices:

$$\boldsymbol{A} = \begin{pmatrix} \mathbf{5} & \mathbf{0} \\ \mathbf{10} & \mathbf{1} \end{pmatrix}, \boldsymbol{B} = \begin{pmatrix} \mathbf{5} & \mathbf{10} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}.$$

Here, the only equilibrium situation will be a situation (0,0) in which each player chooses his or her second pure strategy and wins a unit.

However, it is obvious that if players agree and choose their first pure strategies, then in the situation (1,1), each of them will win five units.

However, it is clear that this situation, which may arise in the case of cooperation, is rather unstable, since each player, randomly changing his strategy, increases his winnings.

4.2 By-Product Production Planning Problem (Cooperative Case)

Let two enterprises produce by-products under production conditions adopted as in antagonistic case, but taking into account sales opportunities, as in a non-cooperative case. Then, as it was established, such a conflict is modeled by a finite game of two persons with a non-zero sum given by a pair of $\boldsymbol{m} \times \boldsymbol{n}$ matrices $\boldsymbol{A} = (\boldsymbol{a_{ij}})$ and $\boldsymbol{B} = (\boldsymbol{b_{ij}})$ elements of which are the winnings (in units of utility) of players I and II respectively, if they are chosen respectively by their *i*-th and *j*-th pure strategies.

Now, in this game, given the nature of the conflict, it is allowed to cooperate without transferring utility from one player to another, that is, players can make agreements and choose a compatible strategy$\bar{\boldsymbol{z}}$.

Obviously,

$$\bar{z} = (z_{11}, \dots, z_{ij}, \dots, z_{mn}) z_{ij} \geq 0, \sum_{i,j} z_{ij} = 1,$$

where $\boldsymbol{z_{ij}}$ – denotes the probability of choosing respectively compatible strategies (*i, j*) by players I and II.

The mathematical expectation of winning, respectively, players I and II under the conditions of their strategy is naturally determined by the formulas

$$S_A(\bar{z}) = \sum_{i,j} a_{ij}, z_{ij},$$

$$S_B(\bar{z}) = \sum_{i,j} b_{ij}, z_{ij}.$$

The points $(\boldsymbol{S_A}(\bar{\boldsymbol{z}}), \boldsymbol{S_B}(\bar{\boldsymbol{z}}))$ form *the valid set R.*

By agreement, players can get as a win a random vector of this set $(\overline{\boldsymbol{S}}_{\boldsymbol{A}}(\bar{\boldsymbol{z}}), \overline{\boldsymbol{S}}_{\boldsymbol{B}}(\bar{\boldsymbol{z}}))$.

Obviously, with compatible actions, players I and II must win no less than the values as if playing the antagonistic game $\boldsymbol{S_A(\bar{y}^*, \bar{x}^*)}$ and $\boldsymbol{S_B(\bar{y}^*, \bar{x}^*)}$, calculated by formula (26), which are players' winnings when they fail to reach an agreement.

To find $\left(\bar{S}_A(\bar{z}), \bar{S}_B(\bar{z})\right)$ use the following*arbitration scheme.*

1. The beginning of coordinates is transferred to a point with coordinates $S_A(\bar{y}^*, \bar{x}^*)$ and $S_B(\bar{y}^*, \bar{x}^*)$, that is, this point is transferred to a point (0,0), where the set *P* becomes the setP'.

2.There is a single point with the coordinates $\bar{S'}_A(\bar{y}^*, \bar{x}^*)$ and $\bar{S'}_B(\bar{y}^*, \bar{x}^*)$ with P' where $\bar{S'}_A(\bar{y}^*, \bar{x}^*) > 0$ and $\bar{S'}_B(\bar{y}^*, \bar{x}^*) > 0$, and

$\bar{S'}_A(\bar{y}^*, \bar{x}^*)\bar{S'}_B(\bar{y}^*, \bar{x}^*)$is the maximum of all earnings

$$S'_A(\bar{y}^*, \bar{x}^*)S'_B(\bar{y}^*, \bar{x}^*).$$

3. We find the arbitration solution by inverse transformation of utility relative to $\bar{S'}_A(\bar{y}^*, \bar{x}^*)$ and $\bar{S'}_B(\bar{y}^*, \bar{x}^*)$.

Let us find an arbitration solution for specific data of the problem of planning the production of by-products in a non-cooperative case, that is, let a cooperative game without side payments be given by the following matrices:

$$A = \begin{pmatrix} 600 & 300 \\ 300 & 900 \end{pmatrix} \text{ and } B = \begin{pmatrix} 500 & 1500 \\ 2000 & 500 \end{pmatrix}.$$

In the non-cooperative case, the equilibrium vectors were vectors $\boldsymbol{\bar{x}^*}$= (3/5, 2/5), $\boldsymbol{\bar{y}^*}$ = (2/3, 1/3). As it has been explored, in a non-cooperative bimatrix game, where cooperation is neglected and players choose their strategies independently, the mathematical expectation of winning of the player I is equal to $\boldsymbol{S_A(\bar{y}^*, \bar{x}^*) = 500}$ and player II -$\boldsymbol{S_B(\bar{y}^*, \bar{x}^*) = 1100}$.

Now suppose that players can cooperate and choose a compatible mixed strategy without passing on utility to one another.

We transform the coordinates by moving the origin to the point (500, 1100) by the formulas

$$S_A(\bar{y}^*, \bar{x}^*) = S_A(\bar{z}) - 500,$$

$$S_B(\bar{y}^*, \bar{x}^*) = S_B(\bar{z}) - 1100,$$

thus constructing the areaP'.

Let us find the point with the coordinates $\bar{S'}_A(\bar{y}^*, \bar{x}^*)$ and $\bar{S'}_B(\bar{y}^*, \bar{x}^*)$that maximizes the function

$$S'_B = \bar{S}'_A(\bar{y}^*, \bar{x}^*)\bar{S}'_B(\bar{y}^*, \bar{x}^*)$$

на множині P' при $\bar{S}'_A(\bar{y}^*, \bar{x}^*) > 0$ і $\bar{S}'_B(\bar{y}^*, \bar{x}^*) > 0$.

The equation of the line passing through the points (-200, 900) and (400, -600) has the form

$$\boldsymbol{S'_B(\bar{y}^*, \bar{x}^*) = -\frac{5}{2}S'_A(\bar{y}^*, \bar{x}^*) + 400}.$$

Substituting this into function $\boldsymbol{S'_{AB}}$, we differentiate the result expression, equate the derivative to zero, solve the obtained equation with respect to $\boldsymbol{\bar{S}'_A(\bar{y}^*, \bar{x}^*)}$, and find

$$\boldsymbol{\bar{S}'_A(\bar{y}^*, \bar{x}^*) = 80}, \text{a } \boldsymbol{\bar{S}'_B(\bar{y}^*, \bar{x}^*) = 200}.$$

Next, by inverse transformation, we find the arbitration solution for the original cooperative game:

$$(\boldsymbol{\bar{S}_A(\bar{z}) = 580, \bar{S}_B(\bar{z}) = 1300}).$$

The arbitration award can be implemented by applying a compatible mixed strategy $\boldsymbol{\bar{z} = (0, 0, z_{21}, z_{22})}$ The strategy j components are found from the formulas for calculations $\boldsymbol{\bar{S}_A(\bar{z}), \bar{S}_B(\bar{z})}$, substituting $\boldsymbol{S_A(\bar{z}) = \bar{S}_A(\bar{z}), S_B(\bar{z}) = \bar{S}_B(\bar{z})}$.

In particular, we find $\boldsymbol{z_{21} = 8/15}$, $\boldsymbol{z_{22} = 7/15}$, according to which player I uses only the second strategy, and player II applies the first and second accordingly with probabilities 8/15 and 7/15. In this case, the agreement between the players leads to the fact that the mathematical expectation of winning players I and II will accordingly equal $580. ($500 in non-cooperative case) and $1300 ($1100 in the non-cooperative case).

Thus, cooperating in a non-antagonistic conflict increases the mathematical expectation of winning (in the sense of utility) of each player.

5. Optimizing product quality control

Let us consider, for example, using an example of the optimization of product quality control, the non-cooperative case and the case of players' cooperation[6].

[6] Ivanilov Yu. P., Lotov A. V. Mathematical models in economics, Moskow.: Science, 1979, 304 p.

5.1 Problem statement

Let some products, manufactured by the supplier company *A* (raw materials for light industry, primary agricultural production, etc.), be supplied to the enterprise *B* for the recycling and manufacturing of finished products (clothing, shoes, food, etc.). Each enterprise is interested in increasing its profits. In this regard, the enterprise *B* controls the quality of the products of the enterprise *A*, and the enterprise *A* is not always interested in improving its quality.

As the control frequency decreases, impunity for product suppliers increases, which in pursuit of quantitative indicators weaken attention to product quality.

As the control frequency increases, the quality of the products of company *B* improves, but the cost of control increases. It is necessary to determine the optimal frequency of control over the quality of products of the enterprise *A* by enterprise *B*,, as well as the optimal enterprise *A* strategy to increase their profits.

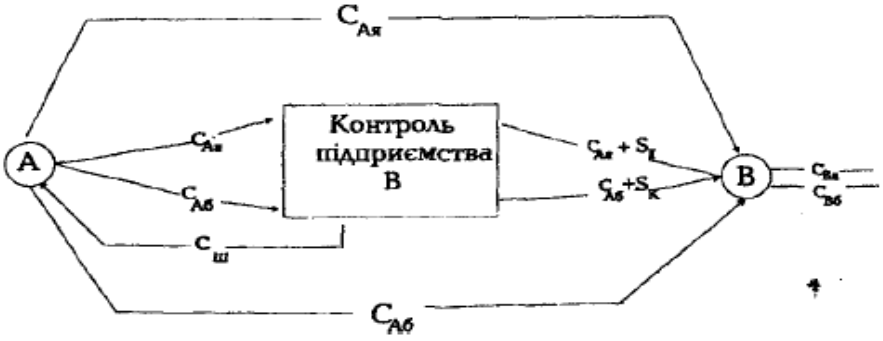

Fig. 1

Let us enter the symbols:

$C_{Aя}$ і $S_{Aя}$- respectively the price and cost of quality products of the enterprise *A*;

$\boldsymbol{C_{Aб}}$ і $\boldsymbol{S_{Aб}}$ – the corresponding price and cost of the defective products of the enterprise *A;*

$\boldsymbol{C_{Вб}}$ і $C_{Вя}$ ~ respectively the prices of defective and quality products of the enterprise *B;*

$\boldsymbol{S_{В}}$ – cost of manufacturing of products by the enterprise *B;*

$\boldsymbol{S_{к}}$- cost of control for the enterprise *B*;

$C_{ш}$ – the cost of the fine paid to the State bythe enterprise *A* in the case of finding a defect.

We present graphically the movement of products from the enterprise *A* to the enterprise *B* (Fig. 4).

5.2 Non-cooperative case

We use the theory of non-cooperative games to solve this problem. Let us denote by $\boldsymbol{y}_{\text{я}}$ the probability of producing quality products by the company *A* (strategy$\mathbf{A_1}$), and by $\boldsymbol{y}_{\text{б}}$ – defective ones (strategy *A2)*, while$\boldsymbol{y}_{\text{я}} + \boldsymbol{y}_{\text{б}} = \mathbf{1}$. Let us denote by $x_к$ the probability of production control of the enterprise *B* (strategy $\boldsymbol{B_1}$), and by $x_в$ – the probability of lack of control (strategy *B2), $x_к + x_в = 1$*. Let us draw up the matrix of wins (profits) for enterprises *A* and *B* respectively (Tables 5 and 6).

Table 5

A\B	$x_к$	$x_в$
$y_я$	$C_{Aя}–S_{Aя}$	$C_{Aя}–S_{Aя}$
$y_б$	$C_{Aб}–S_{Aб}–C_ш$	$C_{Aя}–S_{Aб}$

Table 6

A\B	$x_к$	$x_в$
$y_я$	$C_{Bя}–C_{Aя}–S_к–S_B$	$C_{Bя}–C_{Aя}–S_B$
$y_б$	$C_{Bб}–C_{Aб}–S_к–S_B$	$C_{Bб}–C_{Aя}–S_B$

Then their average profits (winnings) according to formulas (26) will be equal to

$$S_A = (C_{Aя} - S_{Aя})y_я x_к + (C_{Aя} - S_{Aя})y_я x_в + \\ +(C_{Aб} - S_{Aб} - C_Ш)y_б x_к + (C_{Aя} - S_{Aб})y_б x_в;$$

$$S_В = (C_{Bя} - C_{Aя} - S_к - S_в)y_я x_к + (C_{Bя} - C_{Aя} - S_в)y_я x_в + \\ +(C_{Bб} - C_{Aб} - S_к - S_в)y_я x_в + (C_{Bб} - C_{Aя} - S_в)y_б x_в.$$

Using the notation

$$\boldsymbol{y}_{\text{я}} = \boldsymbol{y}, \boldsymbol{y}_{\text{б}} = \mathbf{1} - \boldsymbol{y}, \boldsymbol{x}_{\text{к}} = \boldsymbol{x}, \boldsymbol{x}_{\text{в}} = \mathbf{1} - \boldsymbol{x},$$

we get

$$\boldsymbol{S}_{\text{А}} = \boldsymbol{y}(\boldsymbol{C}_{\text{Ая}} - \boldsymbol{S}_{\text{Ая}}) + (\mathbf{1} - \boldsymbol{y})[\boldsymbol{x}(\boldsymbol{C}_{\text{Аб}} - \boldsymbol{C}_{\text{ш}} - \boldsymbol{C}_{\text{Ая}}) + \boldsymbol{C}_{\text{Ая}} - \boldsymbol{S}_{\text{Аб}}]; \quad (15)$$

$$\boldsymbol{S}_{\text{В}} = \boldsymbol{y}(\boldsymbol{C}_{\text{Вя}} - \boldsymbol{C}_{\text{Ая}} - \boldsymbol{S}_{\text{В}} - \boldsymbol{S}_{\text{к}}\boldsymbol{x}) + \\ +(\mathbf{1} - \boldsymbol{y})[\boldsymbol{x}(\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}} - \boldsymbol{S}_{\text{к}}) + \boldsymbol{C}_{\text{Вб}} - \boldsymbol{C}_{\text{Ая}} - \boldsymbol{S}_{\text{В}}].$$

The equilibrium situation in this problem is characterized by such an optimal pair (point) (y*, x*) – the optimal frequency (probability) of control x* of the enterprise *A* by the enterprise *B* and the optimal frequency (probability)*y** of production of quality products by the enterprise *A,*, in

which it is unprofitable for the side *B* to change its strategy x*, and for the side *A* to change its strategy *y**, as it will decrease the average profits (winnings). The equilibrium conditions are:

$$\frac{\delta S_{\text{А}}}{\delta y} = C_{\text{Ая}} - S_{\text{Ая}} - [x(C_{\text{Аб}} - C_{\text{ш}} - C_{\text{Ая}}) + C_{\text{Ая}} - S_{\text{Аб}}] = 0;$$

$$\frac{\delta S_{\text{В}}}{\delta x} = -yS_{\text{к}} + (1-x)(C_{\text{Ая}} - C_{\text{Аб}} - S_{\text{к}}) = 0.$$

Solving this system of equations we obtain

$$y^* = y_{\text{я}}^* = \frac{C_{\text{Ая}} - C_{\text{Аб}} - S_{\text{к}}}{C_{\text{Ая}} - C_{\text{Аб}}};$$

$$y_{\text{б}}^* = 1 - y_{\text{я}}^* = \frac{S_{\text{к}}}{C_{\text{Ая}} - C_{\text{Аб}}};$$

$$x^* = x_{\text{к}}^* = \frac{S_{\text{Ая}} - S_{\text{Аб}}}{C_{\text{Ая}} - C_{\text{Аб}} + C_{\text{ш}}}; \qquad (16)$$

$$x_{\text{в}}^* = 1 - x_{\text{к}}^* = 1 - \frac{S_{\text{Ая}} - S_{\text{Аб}}}{C_{\text{Ая}} - C_{\text{Аб}} + C_{\text{ш}}}$$

It follows that for any non-zero control value $\boldsymbol{S}_{\text{к}}$ for enterprise *B* there is some optimum defective part for the enterprise *A*, which is equal to $\boldsymbol{y}_{\text{б}}^*$. In order to reduce the critical control frequency $\boldsymbol{x}_{\text{к}}^*$ of the enterprise *B*, it is necessary to increase the value of the fine $\boldsymbol{C}_{\text{ш}}$.

Substituting the obtained values $\boldsymbol{y}^*$ and $\boldsymbol{x}^*$, calculated by the formulas (30), into the relation (29), we obtain the expected optimal profits (wins) of the enterprise *A* and *B*.

$$\boldsymbol{S}_{\boldsymbol{A}}^* = \frac{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}} - \boldsymbol{S}_{\text{к}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}}(\boldsymbol{C}_{\text{Ая}} - \boldsymbol{S}_{\text{Ая}}) + \frac{\boldsymbol{S}_{\text{к}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}} \cdot$$
$$\cdot \left[\frac{\boldsymbol{S}_{\text{Ая}} - \boldsymbol{S}_{\text{Аб}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}} + \boldsymbol{C}_{\text{ш}}}(\boldsymbol{C}_{\text{Аб}} - \boldsymbol{C}_{\text{ш}} - \boldsymbol{C}_{\text{Ая}}) + \boldsymbol{C}_{\text{Ая}} - \boldsymbol{S}_{\text{Аб}}\right];$$

$$\boldsymbol{S}_{\boldsymbol{B}}^* = \frac{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}} - \boldsymbol{S}_{\text{к}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}}\begin{pmatrix} \boldsymbol{C}_{\text{Вя}} - \boldsymbol{C}_{\text{Ая}} - \boldsymbol{S}_{\text{в}} - \\ -\boldsymbol{S}_{\text{к}}\dfrac{\boldsymbol{S}_{\text{Ая}} - \boldsymbol{S}_{\text{Аб}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}} + \boldsymbol{C}_{\text{ш}}} \end{pmatrix} + \frac{\boldsymbol{S}_{\text{к}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}} \cdot$$
$$\cdot \left[\frac{\boldsymbol{S}_{\text{Ая}} - \boldsymbol{S}_{\text{Аб}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}} + \boldsymbol{C}_{\text{ш}}}(\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}} - \boldsymbol{S}_{\text{к}}) + \boldsymbol{C}_{\text{Вб}} - \boldsymbol{C}_{\text{Ая}} - \boldsymbol{S}_{\text{к}}\right];$$

after simplification we have

$$\boldsymbol{S_A^*} = \boldsymbol{C}_{\text{Ая}} - \boldsymbol{S}_{\text{Ая}};\ \boldsymbol{S_B^*} = \boldsymbol{C}_{\text{Вя}} - \boldsymbol{C}_{\text{Ая}} - \boldsymbol{S}_{\text{в}} - \frac{\boldsymbol{C}_{\text{Вя}} - \boldsymbol{C}_{\text{Вб}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}}\boldsymbol{S}_{\text{к}}. \quad (17)$$

5.3 Cooperative Case

The theory of non-cooperative games was used above to solve the problem, that is, the situation was considered when the enterprises *A* і *B* did not have any agreements (cooperation) about increasing own profits – each company operates at its own discretion. In this case, the total profit at their optimal strategies is equal to

$$S_A{}^* + S_B{}^* = C_{\text{Вя}} - S_{\text{Ая}} - S_{\text{в}} - \frac{C_{\text{Вя}} - C_{\text{Вб}}}{C_{\text{Ая}} - C_{\text{Аб}}}S_{\text{к}}.$$

Now let us suppose that between the enterprises *A* and *B* there is an agreement to join their efforts in order to increase the total profit. In particular, this may be the case when an enterprise *B* absorbs an enterprise *A.* In this case, they have one goal – to increase the total profit – which corresponds with one payoff matrix (profit) equal to the sum of the payoff matrices separately for enterprises *A* and *B* (tables 5 and 6):

Table 7

A\B	$x_к$	$x_в$
$y_я$	$C_{Вя}–S_{Ая}–S_к–S_В$	$C_{Вя}–S_{Ая}–S_В$
$y_б$	$C_{Вб}–S_{Аб}–S_к–S_В–C_ш$	$C_{Вб}–S_{Аб}–S_В$

Since the elements of the second column of this matrix (Table 7) are larger than the corresponding elements of the first column, then for arbitrary strategies of the enterprise *A* the second strategy of the enterprise *B,*, which is characterized by the lack of control over the products of the enterprise *A* ($x_к$=0; $x_в$=1), is optimal for increasing the overall profit of the enterprises *A* and *B,* which average (expected) value in this case is

$$S^*{}_{A+B} = (C_{\text{Вя}} - S_{\text{Ая}} - S_{\text{В}})y_{\text{я}} + (C_{\text{Вб}} - S_{\text{Аб}} - S_{\text{В}})y_{\text{б}} =$$
$$= (C_{\text{Вя}} - S_{\text{Ая}} - S_{\text{В}})(1 - y_{\text{б}}) + (C_{\text{Вб}} - S_{\text{Аб}} - S_{\text{В}})y_{\text{б}}. \quad (18)$$

Due to the fact that the profit from the sale of quality products is higher than from the defective ones,

$$\boldsymbol{C}_{\text{Вя}} - \boldsymbol{S}_{\text{Ая}} - \boldsymbol{S}_{\text{В}} > \boldsymbol{C}_{\text{Вб}} - \boldsymbol{S}_{\text{Аб}} - \boldsymbol{S}_{\text{В}},$$

and unlike the first case, when an enterprise *A* works only for its own profit and it is profitable for it to produce some defective products $y_б$*, in order to increase the total profit 5 $*_{A+B}$ it wants (is interested) to reduce this proportion. When$_b$= 0, the total profit equals to

$$S^*_{A+B} = C_{\text{Вя}} - S_{\text{Ая}} - S_{\text{В}}.$$

We calculate how much greater the total profit of enterprises *A* and *B* are, when they work together, from the total profit when they work separately, each for its own result (see (31), (32)):

$$\Delta_{\boldsymbol{AB}} = \boldsymbol{S}^*_{\boldsymbol{A+B}} - (\boldsymbol{S}^*_{\boldsymbol{A}} + \boldsymbol{S}^*_{\boldsymbol{B}}) = (\boldsymbol{C}_{\text{Вя}} - \boldsymbol{S}_{\text{Ая}} - \boldsymbol{S}_{\text{В}})(\boldsymbol{1} - \boldsymbol{y}_{\text{б}}) +$$
$$+(\boldsymbol{C}_{\text{Вб}} - \boldsymbol{S}_{\text{Аб}} - \boldsymbol{S}_{\text{В}})\boldsymbol{y}_{\text{б}} - \boldsymbol{C}_{\text{Вя}} + \boldsymbol{S}_{\text{Ая}} + \boldsymbol{S}_{\text{В}} + \frac{\boldsymbol{C}_{\text{Вя}} - \boldsymbol{C}_{\text{Вб}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}}\boldsymbol{S}_{\text{к}};$$

after simplifications

$$\Delta_{\boldsymbol{AB}} = -(\boldsymbol{C}_{\text{Вя}} - \boldsymbol{C}_{\text{Вб}} - \boldsymbol{S}_{\text{Ая}} + \boldsymbol{S}_{\text{Аб}})\boldsymbol{y}_{\text{б}} + \frac{\boldsymbol{C}_{\text{Вя}} - \boldsymbol{C}_{\text{Вб}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}}\boldsymbol{S}_{\text{к}}. \qquad (19)$$

Since the value of the expression in parentheses is always positive, the difference $\Delta_{\boldsymbol{AB}}$ is a linear descending function relative to $\boldsymbol{y}_{\text{б}}$(the share of defective products of the enterprise *A).* Therefore, the maximum difference value looks like

$$\Delta^{(\boldsymbol{max})}_{\boldsymbol{AB}} = \frac{\boldsymbol{C}_{\text{Вя}} - \boldsymbol{C}_{\text{Вб}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}}\boldsymbol{S}_{\text{к}}.$$

when $\boldsymbol{y}_{\text{б}} = \boldsymbol{0}$

Let enterprise A, working with company *B,* produce the same proportion of defective products $y_б$* when it works independently.

Substituting $\boldsymbol{y}_{\text{б}}{}^*$, which is determined by relations (16), into expression (33), we obtain

$$\Delta^*_{\text{АВ}} = -(\boldsymbol{C}_{\text{Вя}} - \boldsymbol{C}_{\text{Вб}} - \boldsymbol{S}_{\text{Ая}} + \boldsymbol{S}_{\text{Аб}})\frac{\boldsymbol{S}_{\text{к}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}} + \frac{\boldsymbol{C}_{\text{Вя}} - \boldsymbol{C}_{\text{Вб}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}}\boldsymbol{S}_{\text{к}}.$$

after simplifications

$$\Delta^*_{\text{АВ}} = \frac{\boldsymbol{S}_{\text{Ая}} - \boldsymbol{S}_{\text{Аб}}}{\boldsymbol{C}_{\text{Ая}} - \boldsymbol{C}_{\text{Аб}}}\boldsymbol{S}_{\text{к}}$$

which is obviously less than $\Delta_{AB}^{(max)}$-

Finally, it is possible to calculate the share of the enterprise *A* defective products, at which.$\Delta_{AB} = 0$From relation (33) we obtain

$$y_б^0 = \frac{c_{Вя} - c_{Вб}}{(c_{Ая} - c_{Аб})(c_{Вя} - c_{Вб} - s_{Ая} - s_{Аб})} S_к, \quad (20)$$

that is, if the enterprise *A* works together with the enterprise *B* with this share of the defective products, then the total profit of the enterprises *A* and *B* does not increase, compared to the total, when they work separately, and the share of the enterprise *A* defective products is equal to $y_б^*$. Obviously, $y_б^0 > y_б^*$.

REFERENCES

1. Neumann D., Morgenstern O. Theory of Games and Economic behavior. Moskow: Science, 1970, 708 p.
2. Akulich I.L. Mathematical programming in problem examples. Moskow: Higher school, 1986, 318 p.
3. Kudryavtsev E.M. Research of operations in problems, algorithms and programs. Moskow: Radio Communication, 1984, 184 p.
4. Dyubin G.N., Suzdal V.G. Introduction to Applied Game Theory. Moskow: Science, 1981, 336 p.
5. Zamkov O.O, Tolstenko A.V, Cheremnykh Yu.N. Mathematical Methods in Economics Moskow: DIS, 1997, 368 p.
6. Ivanilov Yu.P., Lotov A.V. Mathematical models in economics, Moskow.: Science, 1979, 304 p.

Information about the author:
Medvediev M. H.
Technical Sciences, Professor,
Head at the General Engineering
and Thermal Power Engineering Department
of the V. I. Vernadsky Taurida National University

DOI https://doi.org/10.36059/978-966-397-107-0/124-152

MODELING OF ECONOMIC SYSTEMS. GAME APPROACH

Medvediev M. H.

1. General Model

Let as suppose that a number of players participate in a game where they follow certain rules. The win that everyone gets as a result depends on their own actions as well as the actions of other players. If we consider this game in terms of its logical characteristics, abstracting from its social content, we will notice a clear similarity with the situations we discussed. The players are our participants in the economic process, the rules of the game – our setting or physical or institutional constraints, the winnings – our usefulness or income. That is why the general concept of game theory is well applicable to the study of economic sphere[1].

We denote each player or participant by the index r or s (r, s = 1, 2,..., n). Actions r can be represented in an adequate mathematical way, which in the general case is a vector a_r in some space. Rules or restrictions require that a_r should belong to some predetermined set A_r.

$$a_r \epsilon A_r, r = 1, 2, \dots, n \quad (1)$$

The player who wins the prize r, is a numerical function of the actions made between all participants:

$$W_r(a_1, a_2, \dots, a_n). \quad (2)$$

This presentation of the game is rather conditional. But it does not suggest that the game consists of one move and all players act at the same time. In fact, a_r should be interpreted as a strategy that determines the actions of player r at each move in all situations in which he/she may find him/herself as a the result of other players' actions. Let us suppose, for example, that a two-player game (A and B) consists of three moves, with the first one making the first and third moves and the second making the second move. Let us suppose that B has only two possible moves, denote them respectively by 1 and 2; player A knows in

[1] Malenvo E. Lectures on Microeconomic Analysis. Moskow: Science, 1985, 392 p.

the third move which choice is made by player B. Actions a_r of player A will thus have three components: what A does in the first move; what he does in the third move if B chose 1, and what he does on the third move if B chose 2. In games, even if not very difficult, a_r a component has obviously a very large number: presenting a game with A_r and W_r can be very complicated. But this is not a barrier to abstract and general exploration. In setting such a logical structure, the problem of game theory is to determine what actions are taken or should be taken by players if each of them knows not only their own multitude and their own function but also the multitude A_s and the win functions W_s, of other players.

It should be noted that the knowledge of A_s and W_s envisaged by all participants may prove to be very limiting for the application of game theory to the study of economic phenomena. It contains the natural assumption that the number of participants is small and each of them can effortlessly learn about the conditions of activity of each other participant. However, it is clear that this assumption makes game theory inadequate for the consideration of all the issues that arise from the need to organize information sharing in communities with large numbers of participants.

Game theory, if it were able to provide a general solution to the problem, could form the basis of a broad field of microeconomic theory. In all game theory, the difference between the presence and absence of cooperation between participants is essential both for formalization and for exploring the applicability of one or another of its variants.

In the formal examination, the above mentioned difficulties relate to the choice of general concepts, which allows to describe the result of cooperation between the participants. This choice is not easy. But it does not cause difficulties if the cooperation is removed. The concept of *non-cooperative equilibrium*, which is also called *the Nash equilibrium*, is natural and can be applied to quite a variety of situations. Such an equilibrium E_0 is a possible state, that is, a set of certain values of $a_1{}^0, a_2{}^0, \dots, a_n{}^0$vectors$a_1, a_2, \dots, a_n$ belonging to a set A_r that

$$W_r(a^0{}_1, \dots, a^0{}_{r-1}, a_r, a^0{}_{r+1}, \dots, a^0{}_r, a^0{}_{r+1}, \dots, a^0{}_n) \leq$$
$$\leq W_r(a_1{}^0, \dots, a^0{}_{r-1}, a^0{}_r, a^0{}_{r+1}, \dots, a^0{}_r, a^0{}_{r+1}, \dots, a^0{}_n) \qquad (3)$$

for all a_r is A_r and for all r. In other words, $E_0$is a non-cooperative equilibrium if neither participant is interested in changing its actions and if he/she considers the actions of others as set ones.

As we can see in the two examples, the non-cooperative equilibrium is not very plausible for the large number of cases in which the number of participants is small, because each of them is aware that his/her decisions affect the decisions of his/her partners. On the contrary, the case where there are many participants and each of them is insignificant and poorly aware of the other's capabilities, is more in line with the non-cooperative equilibrium in which the participants' awareness requirements are low.

Therefore, the structure of the participant community is essential when choosing between these two basic assumptions, but it is not just that.

The nature of the relationship between participants (partners and adversaries, suppliers and clients, managers and their employees, etc.) also influences the degree of cooperation that is established between them. and duopoly. To begin the consideration of the application of game theory having imperfect competition, let us first consider a bilateral monopoly and a duopoly.

Let us note that most of the models studied using economic theory are complications of the general game theory model: the set of A_r possible actions of a participant r is initially not completely specified, but partly depends on the actions of other participants, i.e

$$a_r \in A_r(a_1, \dots, a_{r-1}, a_{r+1}, \dots, a_n), r = 1,2, \dots, n. \qquad (4)$$

However, this complication does not essentially relate to the definition of basic concepts, such as the Nash equilibrium. (Of course, this implies that n conditions (38) are not mutually contradictory).

2. Bilateral Monopoly

Bilateral monopoly is a situation where one consumer and one supplier act on the market of some goods.

We believe that the first is such a good, and in the markets of other goods there is perfect competition. We also believe that both the consumer and the supplier are enterprises, and the good 1 is the intermediate, that is, the products of the first enterprise and the resource of the second. For both the supplier and the consumer the prices of other goods are set. Both partners must agree on the price of $p_1$and the amount of good 1 that is exchanged.

Let us suppose that $C_1(y_1)$is the cost of production of the supplier enterprise, $R_2(y_1) - p_1 y_1$ is the profit received by the consumer enterprise as a result of the use of y_1.

The profits of both participants are equal to

$$W_1 = p_1 y_1 - C_1(y_1), W_2 = R_2(y_1) - p_1 y_1. \quad (5)$$

Let us suppose that $C_1$and $R_2$are twice differentiated functions, $C''_1 > 0, R''_2 < 0$.

To determine the payoff functions, as it is customary in game theory, we need to clarify the actions a_1 and a_2 of both entrepreneurs and the respective areas A_1 and A_2. It is possible to make various models, which are different variants of a bilateral monopoly and contain a specific definition of a pair (p_1, p_2) as a function of performed actions(a_1, a_2). We believe that the first enterprise A determines the pricep_1, and the second enterprise B – the amount that it will buy, i. ey_1. AreasA_1 and A_2 are thus defined for $p_1 \geq 0$ and $y_1 \geq 0$ respectively.

We find out what the non-cooperative equilibrium is. Enterprise B, if it considers the price p_1 as a given value, behaves as if the market for this good was competitive. It selects y_1 that

$$R'_2(y_1) = p_1 \text{or leaves } y_1 = 0 \text{if } R'_2(0) < p_1 \quad (6)$$

The first enterprise, if it considers y_1 as a given value, is interested in setting perhaps a higher price p_1 (infinitely large if the area is A_1 unlimited), except when $y_1 = 0$, i.e when p_1 can be selected by anyone. Strictly speaking, the only possible non-cooperative equilibrium is $y_1 = 0$ and $p_1 \geq R_2'(0)$, which results in zero output of the good under consideration. Obviously, Enterprise A, when choosingp_1, cannot ignore the impact that this choice will have on enterprise B. It should not set a very high price that would lead to the disappearance of demand, but could maximize its profit, given that its partner sets the y_1 and according to (40).

In this case, it will act as a monopolist, the demand for products is determined by this equation Simple calculations show that in this case it will produce pure products in the quantity y_1^*, which is the solution of the equation

$$\boldsymbol{C_1'(y_1) - y_1 R_2''(y_1) = R_2'(y_1)}$$

and sell it for$\boldsymbol{p_1^* = R_2'(y_1^*)}$. But the company *B* can not satisfy equation (40) because it knows that *A* is the only partner. It may, for example, refuse to purchase for the price $\boldsymbol{p_1^*}$ the whole quantity of

products $\boldsymbol{y_1}^*$, having the right to believe that such a position will force *A* to agree to a price reduction. Before defining its actions, every enterprise is interested in discovering a rule of behavior that another enterprise will follow. It can do this by putting itself in the place of a partner and determining the most appropriate rule for him.

Thus, both enterprises should understand immediately or after a mutual «probe» that it is advantageous for them to reach an explicit or implicit agreement that would be acceptable to both of them. It is indifferent that the first sets $\boldsymbol{p_1}$, and the second -$\boldsymbol{y_1}$, as they thus act together to determine the acceptable combination $(\boldsymbol{p_1}^0, \boldsymbol{y_1}^0)$

This combination must satisfy the following conditions:

1) the profit $\boldsymbol{W_1}$ is at least equal to$\boldsymbol{C_1(0)}$, otherwise case *A* is not interested in exchange with B;

2) $\boldsymbol{W_2}$ is at least equal to $\boldsymbol{R_2(0)}$;

3) the combination maximizes $\boldsymbol{W_1}$ provided that $\boldsymbol{W_2}$ preserves the value $\boldsymbol{W_2}^0$, otherwise *A* could offer *B* a more acceptable combination for itself, which would also remain good for B;

4) the combination maximizes $\boldsymbol{W_2}$ provided that $\boldsymbol{W_1}$preserves the value $\boldsymbol{W_1}^0$.

To clarify the above mentioned we find out what follows from condition 3). If we put $\boldsymbol{y_1} \neq \boldsymbol{0}$, then from 3) it follows that such a number $\boldsymbol{\lambda}$ exists that the derivatives of the expression

$[\boldsymbol{p_1 y_1} - \boldsymbol{C_1(y_1)}] + \boldsymbol{\lambda}[\boldsymbol{R_2(y_1)} - \boldsymbol{p_1 y_1}]$of $\boldsymbol{p_1}$ and $\boldsymbol{y_1}$ simultaneously turn into zero. A derivative of $\boldsymbol{p_1}$ equals to zero when $\boldsymbol{\lambda} = \boldsymbol{0}$. Since the derivative of is zero, we have an equation

$$\boldsymbol{C_1}'(\boldsymbol{y_1}^0) = \boldsymbol{R_2}'(\boldsymbol{y_1}^0), \tag{7}$$

which defines $\boldsymbol{y_1}^0$ in a unique way as $\boldsymbol{C_1}'$ increases and R_2' decreases.

The study of condition 4) leads, obviously, to the same result. Conditions 1) and 2) thus determine the interval to which the price should belong $\boldsymbol{p_1}^0$ *:*

$$C_1(y_1{}^0) - C_1(0)/y_1{}^0 \leq p_1{}^0 \leq R_2(y_1{}^0) - R_2(0)/y_1{}^0 \tag{8}$$

This means that all combinations *(*$\boldsymbol{p_1}^0$, $\boldsymbol{y_1}^0$) that allow both parties to come to an agreement contain the same amount of products, and the price must be in the interval (42). Thus, there are many similar combinations. We will assume that this set is the kernel of bilateral monopoly.

Let us show the set in the graph, on the abscissa axis of which $\boldsymbol{y_1}$, is laid, and on the y-axis – $\boldsymbol{p_1}$ (Fig. 1). The dashed curve corresponds to combinations for which $\boldsymbol{W_1}$or $\boldsymbol{W_2}$ takes the same set value. Curves $\boldsymbol{W_1 = const}$and $\boldsymbol{W_2 = const}$touch each other at the points of vertices with abscissa $\boldsymbol{y_1^0}$. The kernel is represented by the interval *RS* of this vertical, which is located between the two curves passing through the origin.

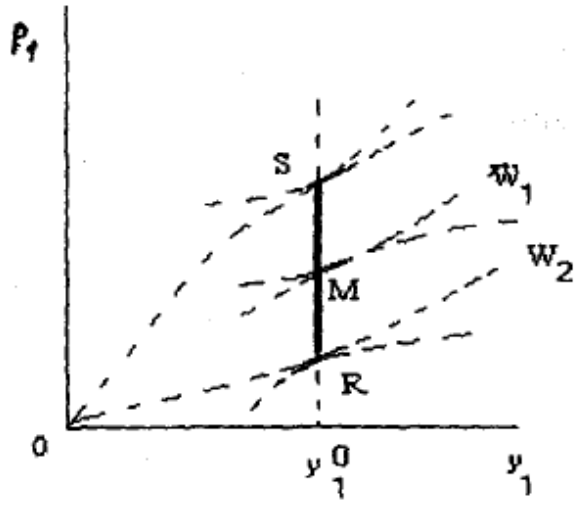

Fig. 1

How can p_1 be determined inside the interval (42)? Enterprise *A* is interested in choosing the largest price, and enterprise *B* interested in choosing the lowest price. Inside the core, the interests of both partners are completely opposite. Therefore, they believe that the final combination chosen depends on the relative power of both partners. Each may be threatened with refusal to comply with the agreement and thus persuade the other to fulfill their requirements. However, none of the partners can substantiate their threats by being able to make a big profit alone by refusing to cooperate altogether. Threats are only effective if an agreement is eventually obtained.

In view of the above mentioned, we can draw the following conclusions.

1. The non-cooperative equilibrium is not a productive competition of bilateral monopoly.

2. Partners are interested in negotiating with each other and executing one of the core-owned combinations.

3. Using threats as a means of achieving a particularly advantageous combination has the risk of breaking the agreement, which will eventually lead to an out-of-core combination.

3. Duopoly

Let us consider the theory of *duopoly*,, which is a market maintained by two manufacturers, in which demand is determined by numerous but small-size consumers. Economic theory gives an idea of this situation, assuming that each unit of good under consideration is exchanged at the same price and demand is competitive in the sense that the total quantity of sold products depends only on its price (and therefore it makes no sense to include for the consideration the individual consumer strategies). For convenience, we consider that this is a good 1 market and that the demand function is decreasing and can be written down

$$p_1 = \pi(y_1), \tag{9}$$

as for monopoly. The total number of pure products $\boldsymbol{y_1}$is produced by enterprises 1 and 2, each of which produces respectively pure products in the quantities $\boldsymbol{y_{11}}$ and $\boldsymbol{y_{21}}$.

For the study of the duopoly, let us suppose that the prices $\boldsymbol{p_2, p_3, \dots, p_L}$ of other goods are determined, for example in competitive markets, and do not depend on $\boldsymbol{p_1}$and $\boldsymbol{y_1}$. Strictly speaking, this is possible only when the good 1 is relatively insignificant and thus the demand of enterprises 1 and 2 in the markets of other goods can be neglected. The function $\boldsymbol{\pi}$, obviously, depends on the values $\boldsymbol{p_2, p_3, \dots, p_L}$as parameters. Let us denote the cost functions of enterprises 1 and 2 by $\boldsymbol{C_1(y_{11})}$ and $\boldsymbol{C_2(y_{21})}$. The corresponding profits will be

$$\boldsymbol{W_1(y_{11}, y_{12}) = y_{11}\pi(y_{11} + y_{21}) - C_1(y_{11})} \tag{10}$$
$$\boldsymbol{W_2(y_{11}, y_{21}) = y_{22}\pi(y_{11} + y_{21}) - C_2(y_{21})}$$

Since the quantities of pure products $\boldsymbol{y_{11}}$ and $\boldsymbol{y_{21}}$are variable, they reflect the behavior of both enterprises, $\boldsymbol{W_1}$and $\boldsymbol{W_2}$ are their profit functions respectively.

A. Cournot, who first investigated the theory of duopoly, proposed as a solution the non-cooperative equilibrium, which, when applied to the duopoly, is called *the Cournot equilibrium*. This solution assumes that each enterprise passively observes the other enterprise and accepts its choice as

a given one, and then makes its own choice so as to maximize its profit. The equilibrium in this case is determined by the pair (y^0_{11}, y^0_{21}) that y_{11} maximizes $W_1(y^0_{11}, y^0_{21})$, which is considered as a function y_{11}, and y_{21} maximizes $W_2(y^0_{11}, y^0_{21})$, which is considered as a function y_{21}.

However, in this situation, it is even less obvious than under a bilateral monopoly that enterprises occupy a similar passive position (Fig. 2).

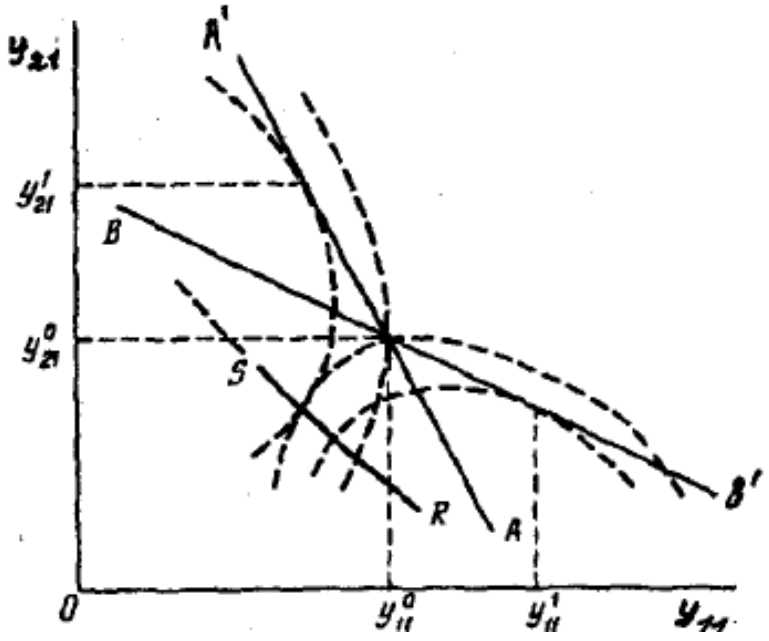

Fig. 2

The curves bent down are lines of level $W_1 = const$; curves, curved to the left are the lines W_2 = const. Curve *AA '* is the geometric location of the points of the lines of level $W_1 = const$, which have the largest ordinate. It determines for each y_{21} the choice of enterprise 1 if it occupies a passive position. In fact, the profit W_1 obviously increases when moving down along the vertical and, thus, on the horizontal (y_{21} set) enterprise 1 is interested in choosing the coordinate of the point at which this horizontal touches the lines of the level $W_1 = const$.

Similarly, the curve BB', that connects the most right points of the lines of the level $W_2 = const$, determines the behavior of enterprise 2 when it takes a passive position.

Thus, the Cournot equilibrium is the point of intersection of curves AA' and BB', let it be (y^0_{11}, y^0_{21}). However, it is assumed that enterprise

1 knows not only its function W_1 but also the function W_2 of its competitor. It can then determine the curve $\boldsymbol{BB'}$, that characterizes the behavior of enterprise 2 if it takes a passive position. In this case, enterprise 1 is interested in choosing the point on the curve $\boldsymbol{BB'}$ at which it touches the curve $\boldsymbol{W_1 = const}$, that is, in the production of quantity $(\boldsymbol{y_{11}}$, which is significantly higher in our case $\boldsymbol{y^0{}_{11}}$. It is likely that enterprise 1 is aware that it can make more profit than with the Cournot equilibrium. It will then select, for example, production $\boldsymbol{y'{}_{11}}$. But the same considerations are applied to enterprise 2, which is interested in choosing production $\boldsymbol{y'{}_{21}}$ if it states the passive position of its competitor. At the same time, choosing a pair ($\boldsymbol{y'{}_{11}}$, $\boldsymbol{y'{}_{21}}$) means a profit for both enterprises that is much less than provided by the Cournotes equilibrium.

As with a bilateral monopoly, each participant, while accepting the situation of the other, must sooner or later come to an explicit or implicit agreement with him, since only in this case one can avoid a struggle that harms both competitors, on the assumption that neither of them believes that it can oust another from the market. An agreement is possible in such pairs ($\boldsymbol{y'{}_{11}}$, $\boldsymbol{y'{}_{21}}$ *)* when, on the one hand, each enterprise makes a profit at least equal to what it would gain by withdrawing from the market and which on the other hand, maximizes the profit of one enterprise at a given value of the other enterprise's profits. These pairs are depicted in Fig. 7 by the points of the curvilinear segment *RS*,belonging to the curve connecting the points of contact of the lines of levels $\boldsymbol{W_1 = const}$and $\boldsymbol{W_2 = const}$, where the point *R*is located on the curve $\boldsymbol{W_1 = -C_1(0)}$and the point *S*- on the curve $\boldsymbol{W_2 = -C_2(0)}$

Як and in a bilateral monopoly, the set of pairs depicted by the points *RS*,can be called *a core.* Inside the core, the position of the pair $\boldsymbol{(y_{11}, y_{12})}$ seems uncertain at first. Each of the two enterprise can try to achieve a particularly advantageous combination for itself, threatening to refuse to fulfill the agreement. But this position is only beneficial if the threat is not fulfilled. The implementation of the combination within the kernel is specified by the agreement between the two enterprises, which, of course, will not behave as the monopolist would have done in their place. The monopoly is trying to increase the total profit $\boldsymbol{W_1 + W_2}$,, which would usually lead to an unambiguous determination of the pair ($\boldsymbol{y^*{}_{11}}$, $\boldsymbol{y^*{}_{21}}$) inside the core.

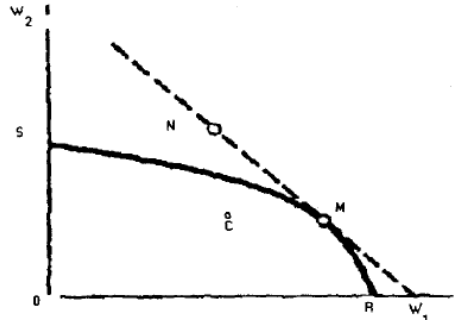

Fig. 3

The difference in their behavior can be traced in Fig. 3, in which the values of profits W_1 *and* W_2 are laid along the abscissa and the ordinate. The kernel is depicted by the curve *RS,*which limits the top and right sets of combinations (W_1W_2)*,* which follow from all choices of values $\boldsymbol{y_{11}}$ and $\boldsymbol{y_{21}}$. (The Cournot equilibrium is represented by a point C inside *RS.)*The sum $W_1 + W_2$ is maximum for the combination *M,* at which tangent to the curve *RS*is parallel to the bisectric line. Point *M* does not necessarily satisfy the two enterprises equally. The enterprise may not agree with choosing this point, which hopes to achieve a more profitable point for it on *RS.*However, we must remember that if there is a complete agreement between the two enterprises, then they can implement any point on tangent to *RS*at point *M,* e.g *N.*To do this, it is sufficient for them to agree to a direct payment of one enterprise to another. In our case, shown in Fig. 3, one enterprise must pay another the sum equal to the projection length of the segment *NM*on the corresponding coordinate axis. In the case of a full agreement, both enterprises behave as one monopolist, and the only disputable issue between them is the division of total profit,, that is, a decision on *a side payment,* which one party must provide to the other. In the process of discussing this, everyone can obviously exploit threats, at the risk of breaking the deal.

The two cases considered give us the right to draw several conclusions.

1. The implementation of non-cooperative equilibrium is, as we see, impossible.

2. If there are hidden or explicit agreements, then it is possible to make judgments based on them and without paying attention to the actions of the participants themselves (only combinations of winnings that are possible as decisions of the game are important).

4. Trade when concluding transactions

Let as suppose that a certain number of players participate in a game where they follow certain rules. The gain that each of them will receive as a result depends on his or her own actions and the actions of the other players. Before determining their actions, each enterprise is naturally interested in discovering a rule of behavior that will be imitated by another enterprise. It can do this by putting itself in the place of a partner and determining the most appropriate rule for him. That is, as noted, a kernel is formed, within which a compromise solution for both parties must be chosen if they are genuinely interested in reaching an agreement. However, the kernel contains many elements, and there are doubts as to which of them should ultimately be selected. It is quite appropriate to try to find a deterministic solution in the absence of additional circumstances, such as a bilateral monopoly (a situation where one consumer and one supply stand in the market of some goods, a duopoly (the market is maintained by two producers, whose demand is determined by numerous but insignificant consumers). and some others. In fact, any logical analysis of the complications that occur in each case can lead to the same problem – the multiplicity of possible outcomes. It is advisable to try to deduce the principles for finding such a solution. It is about principles, that is, finding a general rule for a category of situations.

The problem of so-called trade defines the scope of this study. They are easy to define. The vector $\boldsymbol{\omega}$ of winnings $\boldsymbol{\omega_1}$ and $\boldsymbol{\omega_2}$ of both participants must belong to the set *P*. It is known that it will take the value $\boldsymbol{v}$ (which belongs, obviously, to *P),* unless the parties reach an agreement. What vector $\boldsymbol{\omega}^*$ of *P* should be agreed with? The general answer is to find out how $\boldsymbol{\omega}^*$ depends on *P* and $\boldsymbol{v}$; this (solvable) function allows you to get a solution

$$\omega^* = \mu(v, P) \tag{11}$$

the value of which is defined on the set *P.*

In order to investigate the properties that the function $\boldsymbol{\mu(v, P)}$should possess and to consider its capabilities arising from these features, we have to accept several common axioms.

A comprehensive answer to this question was given by J. Nesh, who found it necessary to accept the following four axioms.

A1. The solution must be *the Pareto optimal*,in other words $\boldsymbol{\mu(v, P)}$should be located on the boundary of *P* on the upper right.

A2. The solution must be individually rational in the sense that each participant should receive a win no less than that which he or she would have received in the absence of the agreement, i.e p $\boldsymbol{\mu}(\boldsymbol{v}, \boldsymbol{P}) \geq \boldsymbol{v}$.

A3. The decision should not be changed if P is replaced by a subset Qcontained in P and containing $\boldsymbol{\mu}(\boldsymbol{v}, \boldsymbol{P})$

A4. If there are two linear increasing functions $\boldsymbol{\varphi_1}$ *and* $\boldsymbol{\varphi_2}$ *that the conditions*

$$\boldsymbol{v_i^2 = \varphi_i(v_i^1), i = 1, 2,} \tag{12}$$

and $\in \boldsymbol{P^1}$, when and only when

$$\boldsymbol{\varphi(\omega) \in P^2,} \tag{13}$$

are carried out, then the decisions $(\boldsymbol{v^1}, \boldsymbol{P^1})$ and $(\boldsymbol{v^2}, \boldsymbol{P^2})$ must be the same in the sense that $\boldsymbol{\mu(v^2, P^2) = \varphi[\mu(v^1, P^1)]}$. Using some low-boundary conditions for ($\boldsymbol{v}$, *P),*Nash showed that there is the only one function that satisfies the axioms *A1 – A4.* More precisely, $\boldsymbol{\mu}(\boldsymbol{v}, \boldsymbol{P})$ is a vector that maximizes in *P* the product $(\boldsymbol{\mu_1 - v_1}) \bullet (\boldsymbol{\mu_2 - v_2})$ of additional wins that both participants receive from their collaboration.

5. Coalition and Decisions

The distribution is the n-dimensional vector $(\boldsymbol{\omega_1, \omega_2, \dots, \omega_r, \dots, \omega_n})$*, the components of which are players' winnings prior to the end of the game. Distribution is possible if there is a multiple of possible distributions n of players, which allows to make winnings corresponding to such distribution. Most often, for the participant r there is a minimum value* $\boldsymbol{v_r}$ *of winnings, which he can provide for himself regardless of the actions of other players. For example, in an exchange economy, it will be the usefulness* $\boldsymbol{S_r(\omega_r)}$*that he/she will receive if abandons other exchanges.*

The distribution $(\boldsymbol{\omega_1, \omega_2, \dots, \omega_r, \dots, \omega_n})$ is considered to be individually rational if $\boldsymbol{\omega_r \geq v_r}$ for all *r*. In fact, it is a priori possible to remove from consideration a result in which some participant does not receive the minimal win that he can provide himself with. It is also believed that the distribution $\boldsymbol{\omega}$ *is rejected by* or *is blocked* by a player *i if* $\boldsymbol{\omega_i < v_i}$.. Therefore, individually rational distribution is not blocked by any participant.

By definition *the coalition* a subset C of the set *I* is meant, consisting of players: $\boldsymbol{I = \{1, 2, \dots, n\}}$. In a theoretical study, it is convenient to

preserve the term «coalition» to denote both the entire set *I*and a subset consisting of one player *r*, for example $\{r\}$. The possibility of coalitions influences the outcome of the game, since only one coalition can achieve some result, or a particular coalition may block the implementation of another result. To investigate this issue, we introduce a simple formalization.

Distribution $(\boldsymbol{\omega_1}, \boldsymbol{\omega_2}, \dots, \boldsymbol{\omega_r}, \dots, \boldsymbol{\omega_n})$ is called *possible* for coalition *C*, if *C* can provide its members with winnings $\boldsymbol{\omega_r}$ (for $\boldsymbol{r \in C}$), whatever actions are made by players which are not in *C*. Coalition can*block* making some distribution if it can provide its members with more winnings than that distribution. Therefore, a formal definition can be given. Coalition *C blocks* distribution $(\boldsymbol{\omega_1}^{\boldsymbol{0}}, \boldsymbol{\omega_2}^{\boldsymbol{0}}, \dots, \boldsymbol{\omega_r}^{\boldsymbol{0}}, \dots, \boldsymbol{\omega_n}^{\boldsymbol{0}})$ if there is possible a distribution $(\boldsymbol{\omega_1}', \boldsymbol{\omega_2}', \dots, \boldsymbol{\omega_r}', \dots, \boldsymbol{\omega_n}')$ that $\boldsymbol{\omega_r}' \geq \boldsymbol{\omega_r}^{\boldsymbol{0}}$for each player *r* from *C* and $\boldsymbol{\omega_r}^{\boldsymbol{1}} \geq \boldsymbol{\omega_r}^{\boldsymbol{0}}$ for at least one player *r* from *C*. As an example, let us consider a bilateral monopoly. Let the enterprise *A* be a player 1, and the enterprise *B* be a player 2. Coalition {1}, consisting of single player 1, blocks any distribution corresponding to player 1 winning less than $\boldsymbol{C_1(0)}$; coalition {2} blocks any distribution that matches player 2 winnings less than $\boldsymbol{R_2(0)}$ coalition {1,2} of two enterprises blocks any distribution that maximizes $\boldsymbol{W_1}$ at a given value of $\boldsymbol{W_2}$ or does not maximize $\boldsymbol{W_2}$ at given value $\boldsymbol{W_1}$. We state that the kernel thus consists of all possible combinations $(\boldsymbol{p_1}, \boldsymbol{y_1})$, corresponding to distributions that are not blocked by any coalition. Similar considerations can be made for the duopoly. This is the explanation of the following statement.

The kernel consists of the set of possible distributions that are not blocked by any of the coalitions.

The value of this statement lies in the idea that the game naturally leads to some kernel-owned distribution.

There are three situations where this is not the case.

1. The use of threats by some players can break agreements and lead to outcomes adverse for all participants.

2. When the number of players is large enough, the information of each of them about the position of the other becomes often incomplete and the making of agreements, which a priori seems to be fruitful, may require long, costly negotiations. To reflect this, they talk about the costs of information and communication that make the participants sometimes content with non-core distributions.

3. There are situations where the kernel is empty. This means that for every possible division, you can find a coalition that can block it.

This is explained by the fact that when considering cooperation and the clash of interests of many participants, game theory is not limited to a single concept of the kernel, which, however, is most commonly used in economic theory. The purpose of conceptual research in game theory is to find a good description of the likely outcome of the game. To do this, it would be enough to have a solution concept that satisfies three conditions: it gives an intuitively correct view; applicable to all or most cases; usually leads to a single solution of the problem. Three conditions cannot be satisfied at the same time. Thus, various existing theories are theoretical compromises.

We see that the kernel does not fully meet the last two conditions. It seems to fit well with the former. However, in some cases, the emergence of the blocking coalitions that are needed is doubtful as they involve reaching an agreement between the parties, the communication between which is difficult. This means that all blocking coalitions must be treated equally, regardless of their origin. in introducing some options to negotiate for players who obviously depend on the outcome of the game. In order to avoid the extreme consequences of this circumstance, we introduce the principle of finding solutions, which offers us to simultaneously consider all the coalitions in which each player can participate, and to introduce some opportunities to negotiate for players, on whom obviously the outcome of the game depends. This principle was introduced by Shapley and developed by him with M. Shubik. Regarding this principle, the chosen decision is thought to have a Shapley price, or just a price. Let us consider the contribution $\boldsymbol{g_r(C)}$ that an individual *r* contributes to a winning of a coalition *C* if it becomes a part of it. For any Coalition C that does not include *r,*this contribution is equal to the payoff that the considered coalition $\boldsymbol{C \cup \{r\}}$ can receive, minus the payout that C can get. The definition of this contribution is simple when the winnings are *transferable*, i.e they can be transferred from one person to another so that the overall winnings retain value when using considerations close to those made for the trade problem).

The determination of the contribution *r* in C can also be made in the case where the winnings are non-transferable. The Shapley price is defined as a distribution whose components are, accordingly, an average $\boldsymbol{\bar{g}_r}$ of

values $\boldsymbol{g_r C}$ on the set of all coalitions *C,* that do not contain *r.* In the game each average determines the natural measure for the ability of the individual *r* to reach an agreement – a measure that must be considered by others in such a way that, as a result of a general agreement, he can get a win equal to $\overline{\boldsymbol{g}}_{\boldsymbol{r}}$ at the end of the game, thar determines the final division (the Shapley price). This concept is considered acceptable when considering some economic problems, and it is often an interesting alternative to the concept of the kernel, when the solution involves cooperation between participants. In each case, the question remains whether the most non-cooperative equilibrium is appropriate here. The larger the number of participants, the more complex the links between them are; the more problematic the possibility of a coalition is, the more plausible the realization of a non-cooperative equilibrium is. Conversely, a small number of participants, naturally interacting for a long time in recurring situations, are naturally cooperative.

6. Arbitration and exchange between the parties

After examining some special situations, let us return to the general economic models. We look for states that can be realized if the exchanges are made not under the laws of *perfect competition.* It is assumed that all forms of imperfect competition are a priori possible. Let us find out what states can be achieved.

Let us begin to study this problem without any preconceived idea, as Englishman Edgeworth did at the end of the XIX century. This consideration will help us better understand some aspects of equilibrium. We use the terminology adopted by M. Ale in exploring the same issues.

Let two consumer individuals *i* and $\boldsymbol{\alpha}$ have the goods $\boldsymbol{x_{ih}}$ and $\boldsymbol{x_{\alpha h}}$ $\boldsymbol{h = (1, 2, .., l)}$ respectively. These are the numbers they originally owned ($\boldsymbol{\omega_{ih}}$ and $\boldsymbol{\omega_{\alpha h}}$) or as a result of exchanges. Let us suppose that the operation, which involves the exchange of goods, is beneficial for both. Denote by $\boldsymbol{z_h}$ the amount of good *h,* which *i* inferior to $\boldsymbol{\alpha}$ in such an operation, or by $(-\boldsymbol{z_h})$ the amount of this good, which $\boldsymbol{\alpha}$ inferior to *i.*Since the operation is beneficial for both, then $\boldsymbol{S_1(x_i - z) > S_i(x_i)}$, $\boldsymbol{S_\alpha(x_\alpha + z) > S_\alpha x_\alpha}$, where *S* denotes utility. The possibility of such an operation may be unknown either *i* or$\boldsymbol{\alpha}$. In doing so, any third party that becomes an intermediary in carrying out this operation will be able to derive some benefit for themselves. In fact, because of the continuity $\boldsymbol{S_i}$, there is a

non-zero vector ш with indefeasible components such as $\boldsymbol{S_1(x_i - z - \omega) > > S_i(x_i)}$. All three participants will find a exchange profitable. in which the quantities of benefits *h* will change to $\boldsymbol{-(z_h + \omega_h)}$ for *i*, to $\boldsymbol{z_h}$ for $\boldsymbol{\alpha}$ and to $\boldsymbol{\omega_h}$ for the mediator. Such an operation is called *an arbitration.*

In the previous example, the possibility of exchange is of interest to two consumers – mediation is two-way. Multilateral mediation is also possible in cases where multiple consumers are involved in the exchange. A mediator that facilitates the transaction will be able to benefit from this. In the future, we assume that either the mediator is himself a participant in the economic process, or the charge $\boldsymbol{\omega_h}$ for the mediation is small enough and can be neglected.

We will call a state, in which both bilateral and multilateral mediation are impossible, all operations have already been completed, no exchange can take place, *a stable distribution* Obviously, there are no reasons why this state must coincide with the competitive equilibrium.

The stable distribution $\boldsymbol{E^0}$, defined in this way is obviously the optimum of distribution. Otherwise, there would be another possible condition E^1, selected by a random consumer that others consider to be no worse than $\boldsymbol{E^0}$. The statement that E^1 is possible is tantamount to the statement that the transition from $\boldsymbol{E^0}$toE^1 is an exchange.

Therefore, there is a possibility of mediation (which may cover all consumers), which is contrary to the stability of distribution $\boldsymbol{E^0}$. The concept of mediation can also be used to describe the exchange process. If the initial position in which each consumer owns $\boldsymbol{\omega_{ih}}$, is not a stable distribution, some exchanges and mediation may occur. The amounts of goods owned by different participants change as many times as necessary for a stable distribution. The benefits of $\boldsymbol{S_i}$ cannot be diminished during these exchanges. Assuming that no advantageous opportunity remains missed (that is, the information is fairly well disseminated to the mediator, or that no participant refuses the advantageous for him operation if he could behave having formulated the requirements acceptable to others), then such a process converges.

The disadvantage of such a theory is that there may be different ways to achieve a stable distribution.

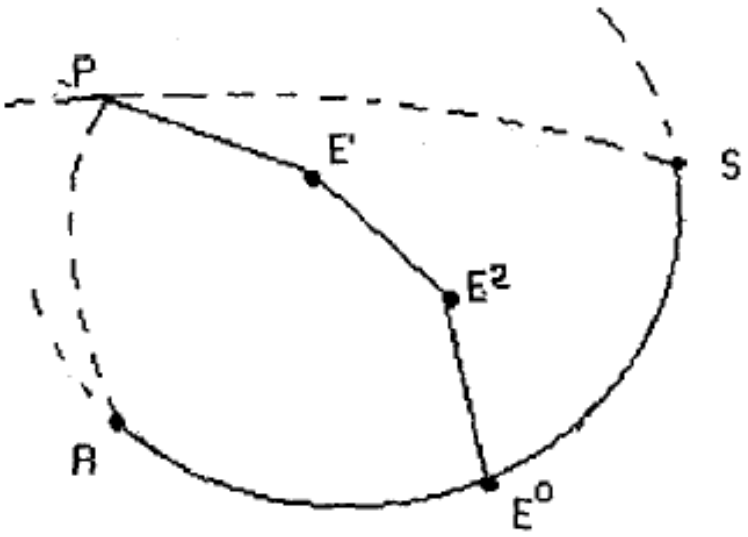

Fig. 4

Let us explain this in the example of the Edgeworth diagram for obtaining the two goods and two participants shown in Fig. 4. Curves *PR* and *RS* are indifference curves passing through the point *P* of the initial stock. The *RS curve* is the geometric location of the Pareto optimum. The path consisting of three exchanges (from*P* and E^1, fromE^1 and E^2, from E_2 to E^0, depicted by a polygonal chain $\boldsymbol{PE^1E^2E^0}$. Each exchange increases the satisfaction of both consumers. However, you can imagine many different paths ending at any point of the curved line *RS*.

7. The Kernel in the Economy of Exchange

the Economy of Exchange is inherently a game because, under some constraints, participants choose their own strategies, the combined action of which ultimately leads them to reach some utility levels $\boldsymbol{S_i}$. It is difficult to describe the primary actions of the exchangers: worries, offers, counterproposals, etc. In the Economy of Exchange, the distributions are determined by the levels of utility corresponding to the consumption vectors. Now we can think directly on the basis of the set consisting of *t* vectors $\boldsymbol{x_i}$, -. The general definitions that were given earlier can be easily transferred to this case.

The coalition is a subset of *C* of the set *t* of consumers.

State $\boldsymbol{E^0}$ is *possible* for coalition *C* if

$$\boldsymbol{x_i^0 \in X_i, i \in C} \qquad (14)$$

$$\sum(\boldsymbol{x_{ih}}^{\mathbf{0}} - \boldsymbol{\omega_{ih}}) = \mathbf{0}, \boldsymbol{h} = \mathbf{1}, \mathbf{2}, \ldots, \boldsymbol{l}. \quad (15)$$

Conditions (48) and (49) ensure that the achievement of $\boldsymbol{x_i^0}$ is possible for Coalition C members acting jointly and independently of other non-coalition members. State $\boldsymbol{E^0}$ is possible if it is possible for a coalition of all members. $\boldsymbol{E^0}$ *is blocked* by Coalition C, if there is a state E^1, possible for C such as

$$\boldsymbol{S_i}(\boldsymbol{x_i^1}) \geq \boldsymbol{S_i}(\boldsymbol{x_i^0}), \boldsymbol{i} \in \boldsymbol{C}, \quad (16)$$

with strict inequation for at least one member of *C.*Condition (50) guarantees that $\boldsymbol{x_i}^{\mathbf{1}}$ is preferable to $\boldsymbol{x_i}^{\mathbf{0}}$ for members of C. The kernel of the economy of exchange is obviously a set of possible states *E,* not blocked by any coalition. It is contained in the set of all the optimum distribution, but contains all the competitive equilibria. Let us graphically represent a kernel for the case where there are two goods and two consumers (Fig. 5), constructed on the basis of the Edgeworth diagram.

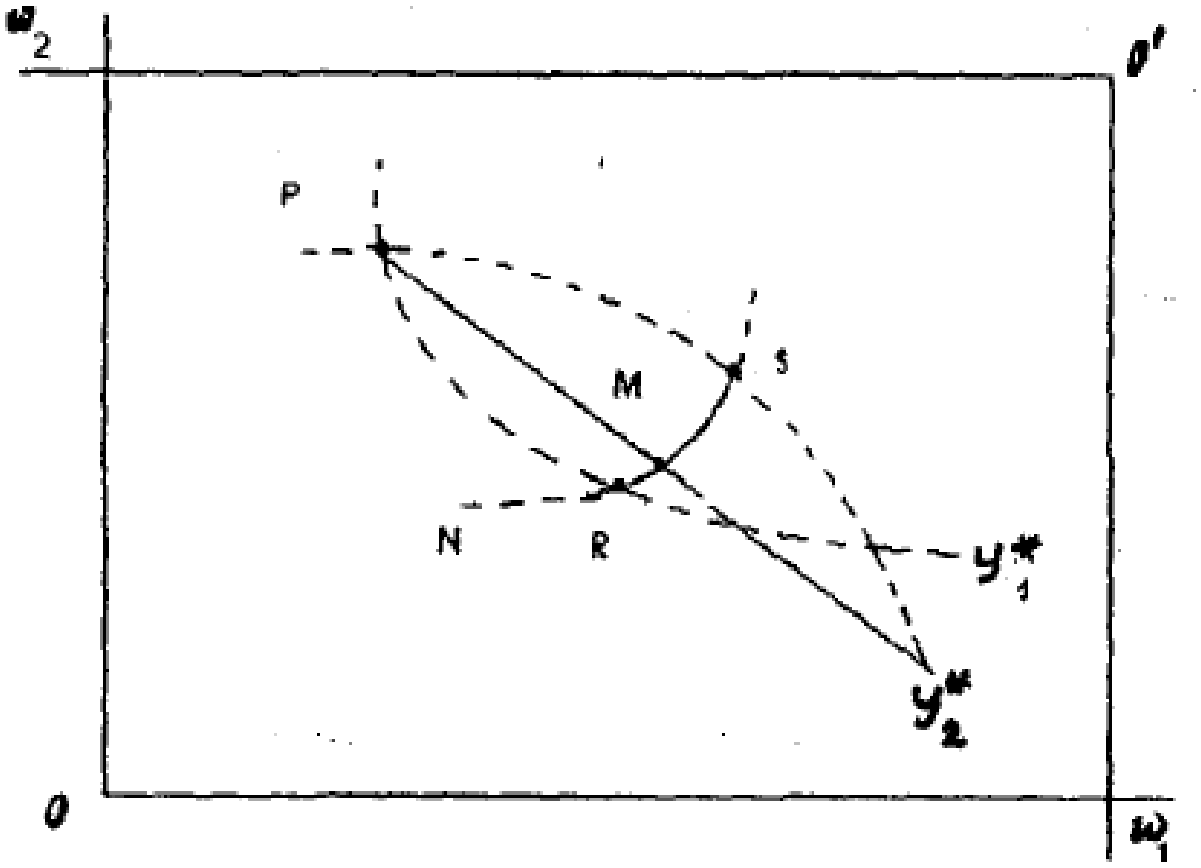

Fig. 5

It is known that the kernel is represented by a portion of the *MN* curve, which is the geometric location of the distribution optimum, that is, the points at which the indifference curves of two consumers touch each other. The states depicted by external points with respect to the *MN* points blocked by the coalition {1, 2}. In addition, states blocked by Coalition {1} are the points located to the left of the indifference curve $\boldsymbol{y_1}^*$passing through the point *P*, which is the initial distribution of resources between consumers. The states blocked by coalition {2} are those points that are located to the right of the indifference curve $\boldsymbol{y_2}^*$, passing through *P*. Thus, the kernel is a part of the curve *MN*, extending from the intersection point from $\boldsymbol{y_1}^*$ to *point* of intersection with $\boldsymbol{y_2}^*$. We see that the competitive equilibrium is *M,* where the common tangent to both indifference curves passes through *P,* belongs to the kernel. In the graph (see Fig. 5), the set of states of stable distribution coincides with the kernel everywhere except the boundary points *R*and *S*.. An arbitrary non-kernel distribution determines the state in which mediation is possible. Conversely, an arbitrary kernel-owned state $\boldsymbol{E^0}$(except *R*and *S*) is a stable distribution for the specified economy, since the transition from the initial state *P* to the state $\boldsymbol{E^0}$ is made through favorable mediation and no mediation is possible, after $\boldsymbol{E^0}$ is reached. This attribute does not take place if there are more than two participants. The reason for this is the difference of opinion on the equilibrium establishment process.

Let us suppose there are two goods and three participants who initially own the resources in quantities

$$\boldsymbol{\omega_1} = \binom{\mathbf{0}}{\mathbf{2}}, \boldsymbol{\omega_2} = \binom{\mathbf{1}}{\mathbf{1}}, \boldsymbol{\omega_3} = \binom{\mathbf{1}}{\mathbf{1}}. \quad (17)$$

We believe that the benefits of these consumers are the same and are described by the following utility functions:

$$\boldsymbol{S_i(x_i) = x_{i1}x_{i2}, x_{ih} \geq 0}. \quad (18)$$

The following two exchanges determine the possible path that ends in a sustainable distribution. Consumers 1 and 2 enter into an agreement that the former gives the latter 3/2 of good 2 in exchange for 1/4 of good 1. The pleasure of the first increases from 0 to 1/8, the pleasure of the second – from 1 to 15/8. After sharing, everyone has the following goods:

$$x_1 = \binom{1/4}{1/2}, x_2 = \binom{3/4}{5/2}, x_3 = \binom{1}{1} \quad (19)$$

Then the second and third parties enter into an agreement whereby the third party gives the second the 1/4 of the good 1 in exchange for the 1/2 of the good 2. The pleasure of the second increases from 15/8 to 2, the pleasure of the third – from 1 to 9/8. Ultimately, consumers will own the goods

$$x_1^0 = \binom{1/4}{1/2}, x_2^0 = \binom{1}{2}, x_3^0 = \binom{3/4}{3/2} \quad (20)$$

It is easy to check that the obtained state $\boldsymbol{E^0}$is a stable distribution; this is the optimal distribution to which prices p_1 = 2, p_2= 1 can be linked.

According to the definitions we adhere to, the state does not belong to the kernel because it is blocked by a coalition consisting of the first and third party.

By pooling their initial resources defined by vectors (51), they could distribute

$$x_1^* = \begin{pmatrix} 1/4 \\ 1 \end{pmatrix}, x_3^* = \begin{pmatrix} 3/4 \\ 2 \end{pmatrix}$$

which, for them, is obviously better than the distribution shown by vectors (54). As this example shows, the difference between the kernel and the set of stable distributions does not lie in the difference of approaches which use the central concepts of «arbitration» and «coalition», respectively.

Arbitration can be defined as an operation whereby a coalition moves from one division to another, which is best for its members. The difference lies in the description of the exchange implementation process.

The idea that the final distribution must belong to the kernel does in fact implicitly imply non-kernel agreements, which could lead to the non-kernel results, or similar agreements that have already been concluded and may be terminated for the sake of others. To clarify this idea, Edgeworth hypothesized that parties could freely renegotiate agreements, that is, concluded contracts could always be canceled later, if a better contract is possible.

The hypothesis that contracts are not considered conclusive before the state inside the kernel is reached is very unrealistic. In addition, it should not be taken literally. Rather, it means that the participants do not make the final decisions before they evaluate the outcome of the various possible contracts.

The ability to renegotiate contracts accepted by Edgeworth is essentially similar to *the Walras hypothesis*according to which the contracts are not concluded until the equilibrium prices are established. It implies that there is a great deal of opportunity for contracts between the parties and leads to a fairly accurate theory.

Rejecting this possibility, the stable distributions obtained from a given initial situation are very uncertain, especially in economies with a large number of participants. Of course, we know that this distribution will be optimum and that it is more preferable than the initial situation for all participants. But with the help of general logical analysis nothing more definite can be said. We have to choose between two theories: a less restrictive but less accurate theory of stable distribution, and a more restrictive but more accurate theory of kernel.

Again, if the number of participants in the exchange is large, then the costs of information and communication can significantly complicate finding the distribution belonging to the kernel To accept that the end result lies within the kernel means to assume that the optimality problem, which is the subject of a great part of microeconomic theory, is solved.

8. Closed Bid Simulation

In closed tenders, tenderers tend to announce their bids, usually once, without informing each other. The lowest or highest bid is accepted depending on the type of bidding.

An example of the first case may be a competition for the cheapest project of an administrative building, an example of the second case is the rent offer for the right to use the parking lot by the firms.

Before deciding on bidding and setting a bid, it is necessary to estimate the costs associated with the object of the auction. Typically, a rate that exceeds these costs is set, and if accepted, the difference is the amount of profit.

In the case of closed bidding, only the winning bid is often announced. For certainty, we will consider the case when the lowest bid is accepted. Then we can have cost estimates and minimum rates for cases when contracts are not concluded. Let us suppose that based on the accumulated statistics of the ratio

$$x = \text{Lowest Bid} / \text{Cost Estimate}$$

has a normal distribution with mathematical expectation c and a variance of $\boldsymbol{\delta^2}$. The task of bidders is to set a bid that maximizes expected profit.

Let own costs according the certain contract be equal to *c* and the tenderer has set a price of *p*. Then its profit is *p-c*, when this price was the lowest, and equals to 0 otherwise.

The probability that this participant has set the lowest price is equal to the probability that the ratio *p/c* will be less than a random variable that has a normal distribution with a mathematical expectation ц and a variance $\boldsymbol{\delta^2}$. This probability is equal to $\boldsymbol{(p/c)}$, where

$$\boldsymbol{f(z) = \frac{1}{\sqrt{2\pi\delta}} \int_z^{\infty} e^{-\frac{(x-\mu)^2}{2\delta^2}} dx.}$$

Therefore, the expected profit is

$$\boldsymbol{P = (p - c) f\left(\frac{p}{c}\right).}$$

We need to maximize *P* by, If we find the first derivative of the last expression by *p* and equate it to zero, we obtain

$$\boldsymbol{\frac{dP}{dp} = f\left(\frac{p}{c}\right) + (p - c) f'\left(\frac{p}{c}\right)\frac{1}{c} = 0.}$$

We denote $\boldsymbol{\frac{p}{c} = z}$, then

$$\boldsymbol{f(z) + (z - 1) f'(z) = 0.}$$

If we put $\boldsymbol{= \mu + t\delta}$, then

$$\boldsymbol{f(z) = \frac{1}{\sqrt{2\pi}} \int_t^{\infty} e^{-\frac{x^2}{2}} dx, f'(z) = -\frac{1}{\sqrt{2\pi\delta}} e^{-\frac{t^2}{2}},}$$

and to find *t* it is necessary to solve the equation

$$\boldsymbol{\frac{1}{\sqrt{2\pi}} \int_t^{\infty} e^{\frac{x^2}{2}} dx = (\mu + t\delta - 1) \frac{1}{\sqrt{2\pi\delta}} e^{\frac{t^2}{2}}.}$$

The values of the integral and the exponents can be found in the tables of normal distribution and the equation can be solved by the approximate (graphical) method (Fig. 6):

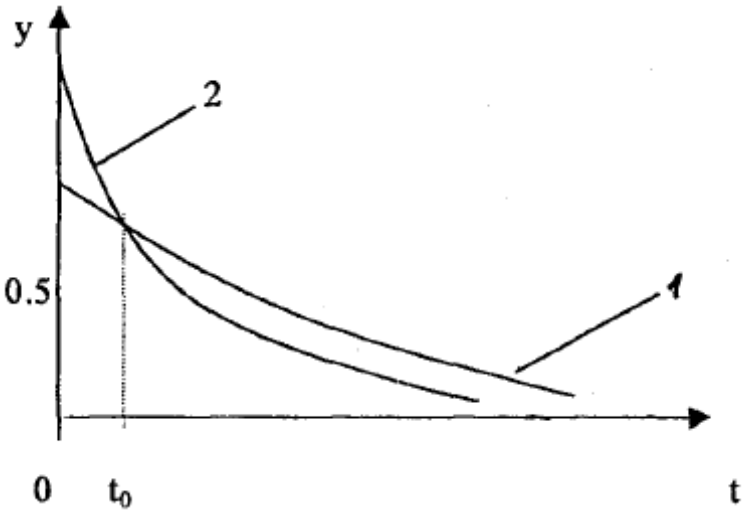

Fig. 6

where the curves *1* and *2* are the graphs of the functions $y_1(t)$ and $y_2(t)$,which are respectively in the left and right parts of the equation. If the solution of the last equation, then the optimal rate that maximizes expected profit,

$$\boldsymbol{p_0 = c(\mu + t_0 \delta)}.$$

The foregoing calculations implicitly assumed that the winning bid distributions did not change with any change in the behavior of one of the bidders. Such an assumption holds for a short period of time when a large number of independently operating firms are involved in the auction.

9. Modeling of auction bidding

Let u suppose that two objects are offered for sale one by one at the auction, and there are two buyers *A* and *B*, own \$100 and \$140 respectively. It is assumed that the known prices at which the purchased objects can then be sold: the first object is priced at \$75, the second is \$125. Consider the problem of defining buyer strategies to maximize their profits[2].

If an item of value of \$75 is initially offered for sale, then the buyer *A* will raise the bid as long as its profit in the case of acquisition of the item for \$75 is not equal to its profit in the case of purchase of the object for \$125. If *B* is able to buy an object worth \$75 for \$*x*, then he will have

[2] Akof R., Sasiena M. Fundamentals of Operations Research. Moskow: World, 2007, 2010 p.

$140 – $*x* and *A*should purchase an object worth $125 for an amount that is a bit higher than the 140 – x provided $\mathbf{140 - x \leq 100}$.

If *A* acquires the first object for $x, then his profit will be $75 – x, and if the first object at the same price is acquired by *B,* then the profit *A* will be equal to 125-(140-x). Therefore *A* will raise the price until the condition is fulfilled

$$\boldsymbol{75 - x = 125 - (140 - x) = x - 15,}$$

whence $x = 45$.

The buyer *B* understands that he could purchase both objects. If he can buy the first object for less than $40, then he will for sure buy the second for a price of just over $100. Thus, if *B* is able to get the first object for the sum of $y < 40$, then its total profit will be

$$\mathbf{75 - y + 125 - 100 = 100 - y}.$$

However, if the first object is for *A* the sum *y*, then *B* will receive the second one for the sum $100 – *y,* and therefore, its profit will be equal

$$125 - (100 - y) = 25 + y.$$

As a result, *B* will raise the price for the first item until the condition is fulfilled

$$100 - y = 25 + y,$$

whence $y = 37.5$.

Assuming that both participants of the auction made these calculations, then the buyer *B* will reach the following conclusions: the maximum price he can offer for the first object, when he intends to pay for both, is $37.5; the buyer *A* will never allow him (*B*) to acquire the first object for less than $45, as otherwise the profit for *A* will be less than he can afford if he raises the bid to this price; if *A* acquires the first object, the more he pays for it, the cheaper the second item will be for *B* buy it.

It follows that the buyer *B* will force *A* to pay for the first object of $45, and the second one will be sold to *B* for the price of $100 – $45 = $55. The profit of *A* will be $75-$45=30 $, and the profit of*B* will be $125-$55=$70.

Note that when the number of items being auctioned exceeds two, the approach of finding the best strategies for buyers *A* and *B* becomes very cumbersome and practically impossible to put into practice. In this case, it

is appropriate to consider such a problem as a dynamic programming problem and solve it in stages.

Let us suppose that only one object is put up for auction and *A*owns the sum $\boldsymbol{\alpha}$, a *B* has the sum $\boldsymbol{\beta}$. Both participants of the auction consider that the value of the object is $\boldsymbol{c_1}$.

Let us denote by $\boldsymbol{u_1(\alpha,\beta)}$ the profit of *A*, and by $\boldsymbol{v_1(\alpha,\beta)}$ – the profit of *B.*

If *B* has set the price *x*, then *A*, having increased it to the value $+\Delta$, will purchase the object and receive a profit $\boldsymbol{c_1 - x} - \Delta$. If the object is purchased by *B*, then *A* will not receive any profit. Therefore *A* will increase the price provided that $\boldsymbol{x < c_1}$. In addition, since *A* has only the sum a, condition $\boldsymbol{x < \alpha}$ must be satisfied. Obviously, *B* will think similarly. The following conclusions can be drawn from here:

1) if $\boldsymbol{\alpha > \beta}$ and $\boldsymbol{\beta > c_1}$, then the object will be purchased by *A* at a price that slightly exceeds $\boldsymbol{\beta}$ and $\boldsymbol{u_1(\alpha,\beta) = c_1 - \beta}$, and $\boldsymbol{v_1(\alpha,\beta) = 0}$;

2) if $\boldsymbol{\alpha \geq c_1}$and $\boldsymbol{\beta \geq c_1}$, then the object will be purchased at the price of $\boldsymbol{c_1}$and $\boldsymbol{u_1(\alpha,\beta) = v_1(\alpha,\beta) = 0}$;

3) if $\boldsymbol{\alpha < \beta}$and $\boldsymbol{\alpha} < \mathrm{c_1}$,, then the object will be purchased by *A* at a price that slightly exceeds $\boldsymbol{\alpha}$ and $\boldsymbol{u_1(\alpha,\beta) = 0}$, and $\boldsymbol{v_1(\alpha,\beta) = \mathrm{c_1} - \alpha}$.

Now let us suppose that the second object is being auctioned with value c_2*, and it is being offered first. We denote by* $\boldsymbol{u_2(\alpha,\beta)}$*and* $\boldsymbol{v_2(\alpha,\beta)}$*the total profits of A and B, when two objects are auctioned.*

If *B* has set the price x for $\mathrm{c_2}$, then *A* may give him a chance to purchase this object and the total profit of *A* will be $\boldsymbol{u_1(\alpha,\beta - x)}$. However, *A* may raise the price a little more than *x*, in this case when *B* gives way to it, the profit *A* will be equal to $\boldsymbol{c_2 - x + u_1(\alpha - x,\beta)}$.

If *A* has enough resources, then he will continue to raise the price until conditions are met

$$\boldsymbol{x \leq \alpha, c_2 - x + u_1(\alpha - x,\beta) \geq u_1(\alpha,\beta - x)} \qquad (21)$$

Similarly *B* will raise its rates as long as the conditions are met

$$\boldsymbol{x \leq \beta, c_2 - x + v_1(\alpha,\beta - x) \geq v_1(\alpha - x,\beta)} \qquad (22)$$

If you tabulate the values of the functions $\boldsymbol{u_1(\alpha,\beta)}$ and $\boldsymbol{v_1(\alpha,\beta)}$ at different values of $\boldsymbol{\alpha,\beta}$,, it is easy to find the smallest values *x*under which these conditions begin to be violated. Let them respectively be equal to $\boldsymbol{x_1^{(2)}}$ and $\boldsymbol{x_2^{(2)}}$. Then we conclude that with respect to object c the

participant *A* of the auction will increase bids up to $\boldsymbol{min}\left(\boldsymbol{x_1}^{(2)}, \boldsymbol{\alpha}\right)$, and *B* – up to $\boldsymbol{min}\left(\boldsymbol{x_2}^{(2)}, \boldsymbol{\beta}\right)$. An object c_2 will be bought by *A* provided

$$\boldsymbol{min}\left(\boldsymbol{x_1}^{(2)}, \boldsymbol{\alpha}\right) > \boldsymbol{min}\left(\boldsymbol{x_2}^{(2)}, \boldsymbol{\beta}\right) \quad (23)$$

When we know who acquired the object c_2, we define (tabulate) the functions $\boldsymbol{u_2}(\boldsymbol{\alpha}, \boldsymbol{\beta})$and $\boldsymbol{v_2}(\boldsymbol{\alpha}, \boldsymbol{\beta})$.

Now let us suppose that the other object is being auctioned with value $\boldsymbol{c_i}$, and it is being offered first. Then *A* and *B* will raise prices as long as the conditions are met

$$\boldsymbol{x \le \alpha, c_i - x + u_{i-1}(\alpha - x, \beta) \ge u_{i-1}(\alpha, \beta - x)}$$

for *A* and

$$\boldsymbol{x \le \beta, c_i - x + v_{i-1}(\alpha, \beta - x) \ge v_{i-1}(\alpha - x, \beta)}$$

for *B.*

If

$$\boldsymbol{min}\left(\boldsymbol{x_1}^{(i)}, \boldsymbol{\alpha}\right) > \boldsymbol{min}\left(\boldsymbol{x_2}^{(i)}, \boldsymbol{\beta}\right),$$

then the object $\boldsymbol{c_i}$ will be acquired by *A,* otherwise it will go to *B.* Determining who purchases the object $\boldsymbol{c_i}$, we find (tabulate) the functions $\boldsymbol{u_i}(\boldsymbol{\alpha}, \boldsymbol{\beta})$and $\boldsymbol{v_i}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ – the total profits of *A* and *B*, when *i* objects were auctioned.

Let *n* items be auctioned. Then sequentially giving to i the values 2, 3,..., *n*, we get to the object c_n, which is put up for auction first. Analyzing the ratio

$$\boldsymbol{x \le \alpha, c_n - x + u_{n-1}(\alpha - x, \beta) \ge u_{n-1}(\alpha, \beta - x)};$$

$$\boldsymbol{x \le \beta, c_n - x + v_{n-1}(\alpha, \beta - x) \ge v_{n-1}(\alpha - x, \beta)},$$

as noted above, depending on whichever is greater $\boldsymbol{min}\left(\boldsymbol{x_1}^{(n)}, \boldsymbol{\alpha}\right)$ or $\boldsymbol{min}\left(\boldsymbol{x_2}^{(n)}, \boldsymbol{\beta}\right)$, we determine who buys the object c_n,, and the total profits $\boldsymbol{u_n}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ and $\boldsymbol{v_n}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of the buyers *A* and *B*.. Putting $\boldsymbol{i = n-1, n-2, \dots, 2}$, we find who buys $\boldsymbol{c_{n-1}, c_{n-2}, \dots, c_1}$objects and profits from each.

Let's solve the above-mentioned problem with the method of dynamic programming, when three objects worth $\boldsymbol{c_1 = 125\ y.o.}$, $\boldsymbol{c_2 = 75\ y.o.}$, $\boldsymbol{c_3 = 3 - 0\ y.o.}$ м are auctioned and buyers *A* and *B* have $\boldsymbol{\alpha = 100\ y.o.}$ and $\boldsymbol{\beta = 140\ y.o.}$ at their disposal

If one object $\boldsymbol{c_1}$ is being auctioned, then it will be purchased by *B* at a price slightly higher than $\boldsymbol{u_1(\alpha,\beta)=0, v_1(\alpha,\beta)=125-100=}$ $\boldsymbol{=25\ y.o.}$

Now let us suppose that two objects $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ are being put up for auction, and c_2 is being offered first. Let us determine who acquires object c_2 *this time*, and who – $\boldsymbol{c_1}$ at different values of $\boldsymbol{\alpha \le 100}$ and $\boldsymbol{\beta \le 140}$, using the relation (21) – (23).

Let us consider the case when $\boldsymbol{\alpha < \beta}$ ($\boldsymbol{\beta > \alpha}$ is symmetrical, buyers *A* and *B* change places).

Then

$$u_1(\alpha - x, \beta) = 0, u_1(\alpha, \beta - x) = \begin{cases} 0, \text{коли } x < \beta - \alpha, \\ 125 - (\alpha - x), \text{коли } x > \beta - \alpha; \end{cases}$$

$$v_1(\alpha, \beta - x) = \begin{cases} 125 - \alpha, \text{коли } x < \beta - \alpha, \\ 0, \text{коли } x > \beta - \alpha; \end{cases}$$

$$v_1(\alpha - x, \beta) = 125 - (\alpha - x)$$

and conditions (55), (56) have the form

$$\begin{cases} 75 \ge x \\ \alpha \ge x, \end{cases} \begin{cases} 37{,}5 \ge x, \\ \beta \ge x, \end{cases} \text{коли } x < \beta - \alpha, \tag{24}$$

$$\begin{cases} \frac{\beta - 50}{2} \ge x, \\ \alpha \ge x, \end{cases} \begin{cases} \frac{\alpha - 50}{2} \ge x, \\ \beta \ge x, \text{коли} x > \beta - \alpha. \end{cases} \tag{25}$$

Put in them α=100,β=140. Then from (58) it follows

$$\begin{cases} 45 \ge x \\ 100 \ge x, \end{cases} \begin{cases} 25 \ge x \\ 140 \ge x, \text{коли } x > 40, \end{cases}$$

and $x_1^{(2)} = 45.$,

Since

$$\boldsymbol{min\left(x_1^{(2)}, \alpha\right) = min(45, 100) = 45,}$$

$$\boldsymbol{min\left(x_1^{(1)}, \beta\right) = min(37, 5, 140) = 37, 5,}$$

then, under condition (57) (45> 37.5), object c_2 is acquired by the buyer A at a price of $45. and his profit is $u_2(100{,}140) = 75 - 45 = 30\ y.o.$

Buyer B will purchase the object c_1 at a price of \$100-\$45=\$55. and his profit will be $v_2(100,140) = 125 - 55 = 70\ y.o.$

This solution is the same as the two-object solution obtained above by the second method.

Using conditions (58), (59), similar to the above, we determine who acquires object c_2 at $\alpha \leq 100, \beta \leq 140$. Knowing this, we calculate (tabulate) functions (profits) $u_2(\alpha, \beta)$, $v_2(\alpha, \beta)$ (Table 1).

Table 1

β	α			
	100	**90**	**86**	**85**
140	30	25	21	20
	70	85	93	95
130	35	35	31	30
	65	75	83	85
126	38	37	35	25
	62	73	79	105
125	37,5	37,5	36	35
	62,5	72,5	78	80

In the table, the top row is the value of function $\boldsymbol{u_2(\alpha, \beta)}$, the bottom row is the value of function $\boldsymbol{v_2(\alpha, \beta)}$.

Now let us suppose that the third object is being auctioned with value $\boldsymbol{c_3 = 30\ y.o.}$, and it is being offered first. Then A and B will raise prices as long as the conditions are met

$$\boldsymbol{x \leq 100, 30 - x + u_2(100 - x, 140) \geq u_2(100, 140 - x)};$$

$$\boldsymbol{x \leq 140, 30 - x + v_2(100, 140 - x) \geq v_2(100 - x, 140)}.$$

Using the table 19, we easily find that when$x$= 10

30 – 10 + 25> 35;

30 – 10 + 62 = 85,

that is, $\boldsymbol{x_2^{(3)} = 10}$, and when$x$ = 14

30 – 14 + 21 ≈ 38;

30 – 14 + 62 <93,

that is $\boldsymbol{x_1^{(3)} \approx 14}$ Since $\boldsymbol{x_1^{(3)} > x_2^{(3)}}$, the object $\boldsymbol{c_3}$ will be bought by the buyer *A* at the price of \$14. and his profit will be 30-14=\$16.

After that, he has \$100-\$14=\$86, and buyer *B* has \$140. Using the table 19 for these values we find the profits from the other two objects $\boldsymbol{c_2}$ and $\boldsymbol{c_1}$. Accordingly, they are \$21 for *A* and \$93 for *B*, and *A* acquires object $\boldsymbol{c_2}$ at the price of \$54. (profit \$75-\$54=\$21), and *B* will get the object c_x at a price of \$32.

(profit \$125-\$32=\$93).

Therefore, when three objects are auctioned at a cost of $\boldsymbol{c_1 = 125\ y.o.}$, $\boldsymbol{c_2 = 75\ y.o.c_3 = 30\ y.o.}$, the buyer *A*, in order to maximize his profit, will buy the objects $\boldsymbol{c_3}$ and $\boldsymbol{c_2}$ and his total profit will be \$16+\$21=\$37, and the buyer *B* will buy the object $\boldsymbol{c_1}$ (the profit is \$93).

REFERENCES

1. Malenvo E. Lectures on Microeconomic Analysis. Moskow: Science, 1985, 392 p.

2. Akof R., Sasiena M. Fundamentals of Operations Research. Moskow: World, 2007, 2010 p.

Information about the author:
Medvediev M. H.
Doctor of Technical Sciences, Professor,
Head at the General Engineering
and Thermal Power Engineering Department
of the V. I. Vernadsky Taurida National University

DOI https://doi.org/10.36059/978-966-397-107-0/153-176

MATRIX GAMES AND STATISTIC CRITERIA

Muliava O. M.

INTRODUCTION

Let us consider game situations in modeling of various aspects of work of a really existing enterprise. This is a very difficult situation, as not only the rules and customs that govern the contracting process, but also some situations that give the individual or company the opportunity to enter into an agreement on particularly favorable terms, play a role.

A common feature of different situations is that when making his/her own decision, each participant must have an idea of the decisions made by the other participant.

However, unfortunately, game theory has proven to be inadequate to address all of the problems posed by the need for organizations with many participants. However, its application has made it much easier to investigate some simple cases.

1. Problem statement

Consider the problem of supply of raw materials.

Suppose that a firm *A* entered into an agreement with another company *B* for the supply of perishable raw materials, valued at $100 a day.

If raw materials are not available during the day, the firm *A* incurs losses of $400. from the downtime of the workers.

It can use her own transportation (an additional cost of $50), but experience shows that in half of the cases, the transport returns empty.

It is possible to increase the likelihood of receiving raw materials up to 80% if you first send your representative to the company *B*, but this requires an additional cost of $40.

It is possible to order a daily rate of raw material from another company at a price up to 50% higher, but in addition to transportation costs ($50), there may be additional costs of $30 associated with the overtime of the teams that sell unnecessary raw materials if a centralized supply arrives on the same day.

What strategy should firm *A* follow if it is not known in advance whether a centralized supply of raw materials will occur or not?

To solve this problem, first of all, we will list the possible strategies of the supplier (firm *B*):

B_1 – delivery on time;

B_2 – no delivery.

The company *A*, according to the condition of the problem, has four strategies:

A_1 – Take no further action;

A_g – send to company *B* own transport;

A_3 – send to company *B* own representative and transport;

A_4 – order additional raw materials from another company.

In the general case, if the first player (firm *A) m* has possible strategies, and the second one – *n*, then always *m*, *n* possible situations are created, each of which corresponds to a certain payment of one player to the other.

There are a total of 8 situations that describe all combinations of four firm strategies *A* and two company *B* strategies

These situations and their associated losses and costs are presented in Table. 1.

Table 1

Ситуація	Денні витрати фірми *A*					
	Вартість сировини	Збитки від простою	Транспортні витрати	Витрати на відрядження	Витрати на понад - нормовану роботу	Всього в день
$A_1 - B_1$	100	0	0	0	0	100
$A_1 - B_2$	0	400	0	0	0	400
$A_2 - B_1$	100	0	50	0	0	150
$A_2 - B_2$	50	200	50	0	0	300
$A_3 - B_1$	100	0	50	40	0	190
$A_3 - B_2$	80	80	50	40	0	250
$A_4 - B_1$	250	0	50	0	30	330
$A_4 - B_2$	150	0	50	0	0	200

In many situations, table. 1 becomes cumbersome and incomprehensible, it is more convenient to move from it to an additional payment matrix A. It is a rectangular matrix and has t rows (by the number of first player strategies) and n columns (by the number of second player strategies).At the intersection of the *i*rows and the *j*columns, the second player's payment is placed first in the situation where the *i* strategy is applied by the first player and *j* strategy by the second. If the second player wins, the payment will have minus.

The payment matrix in the problem (game) under consideration has a dimension of 4 x 2 and is shown in Table. 2. All payments have a negative sign because in this task they determine the costs of the firm *A.*

Table 2

Стратегії фірми *A*	Стратегії фірми *B*	
	B_1	B_2
A_1	– 100	– 400
A_2	– 150	– 300
A_3	– 190	– 250
A_4	– 330	– 200

Payment Matrix *A* looks like

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{-100} & \boldsymbol{-400} \\ \boldsymbol{-150} & \boldsymbol{-300} \\ \boldsymbol{-190} & \boldsymbol{-250} \\ \boldsymbol{-330} & \boldsymbol{-200} \end{pmatrix}.$$

The task of firm A is to find the optimal strategy that ensures the minimum of expected losses in the conditions of uncertainty of the supplier's behavior (firm *B).* Choosing a company behavior strategy *A* under the conditions described in table.2, depends on the reliability of the supplier, which is quantified in terms of probability. For example, let it be 40% (meaning that delivery is timely with a probability of 0.4). Then the expected losses (negative gain) of the company *A* when applying the first pure strategy *A*, are

$$S_1(0{,}4) = -100 \cdot 0{,}4 - 400 \cdot 0{,}6 = -280\ y.o.,$$

and when applying the fourth one,

$$S_4(0{,}4) = -330 \cdot 0{,}4 - 200 \cdot 0{,}6 = -252\ y.o.,$$

We see that costs have decreased. If you calculate losses when applying other strategies, then the best strategy will be A_3. In fact, using the second strategy, the firm *And* will bear the losses,

$$S_2(0{,}4) = -150 \cdot 0{,}4 - 300 \cdot 0{,}6 = -240\ y.o.,$$

and when using the third strategy, only

$$S_3(0{,}4) = -190 \cdot 0{,}4 - 250 \cdot 0{,}6 = -226\ y.o.$$

2. Geometric and economic interpretations of game theory problem solving

Let us give a geometric interpretation of the game (problem) under consideration[1].

To do this, we draw the horizontal axis of the reliability of the supplier (firm *V),* which is measured by probabilities in the range 0 – 1 and denote it$\boldsymbol{x_1}$. The value of$\boldsymbol{x_2 = 1 - x_1}$is thus the magnitude of unreliability of the supplier.

The numbers $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$, which are equal to one, indicate the probability that the supplier of pure strategies $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$are used in each party. The set of strategies $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$, which have a probability estimate of $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ their implementation is called *mixed strategy.*

The points $x_1 = 0$ and $x_1 = 1$in Fig. 1 correspond to the second and first pure strategies of the firm *B,*, and all points $0 < x_1 < 1$ on the segment – to the mixed strategies. It is clear that there is an infinite number of mixed strategies for each player.

Let us plot the graphics of the firm's *A* costs when applying its pure strategies against the company's mixed strategy *B.* Let's start with the first strategy. If the supplier is absolutely reliable (that is, always applies the strategy B_1 and means $x_1 = 1, x_2 = 0$), the costs of the firm *A* are equal in accordance with the payment matrix – 100 USD.

Let us set a point with coordinates (1; – 100).

If the supplier is completely unreliable (that is, always applies the strategy B_2*;* $x_1 = 0, x_2 = 1$), then the cost of the company *A* equals – 400 $. and it is necessary to set the point with coordinates (0; -400).

[1] Kofman A. Methods and models of operations research. Moscow: World, 1977, 432p.

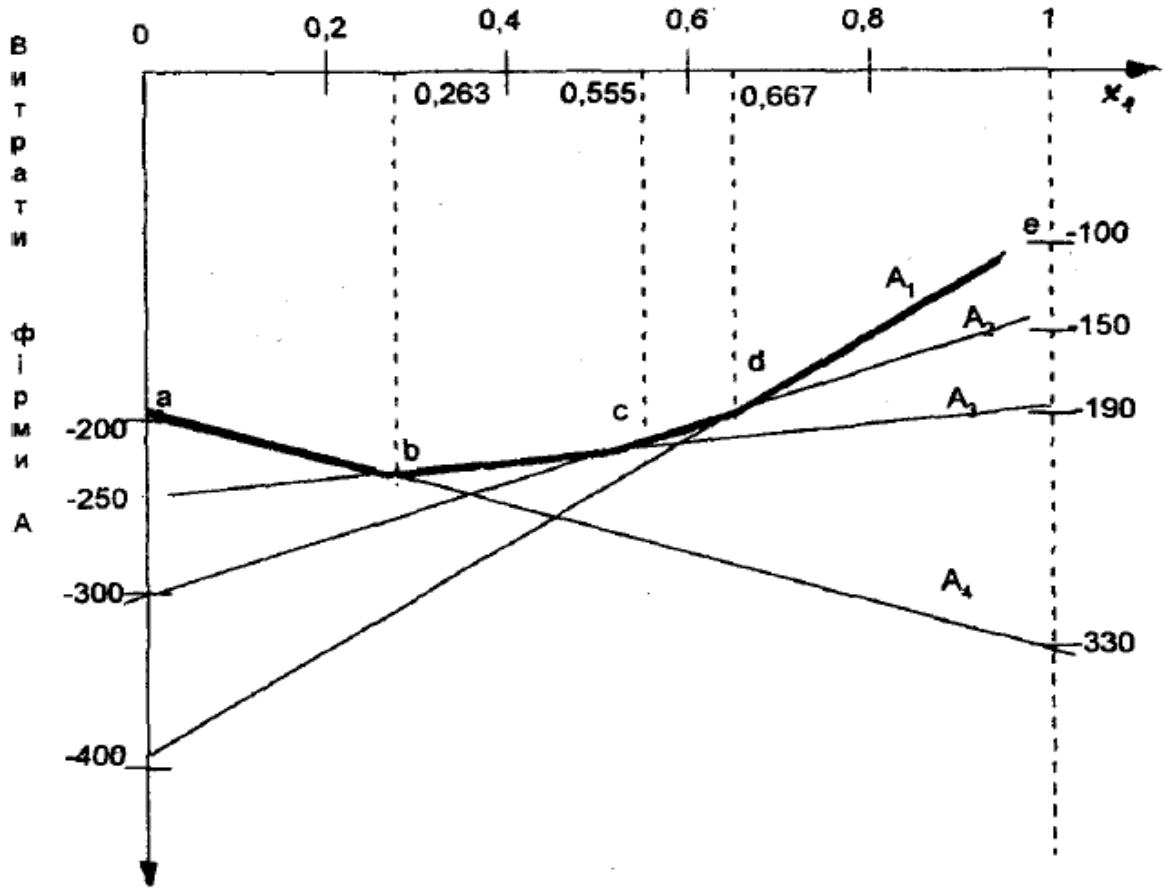

Fig. 1

If the reliability of the firm *B* $0 < x_1 < 1$, then the daily expenses of the firm *A*, which applies the first strategy against the mixed strategy of the supplier, depend on the probability x_1, and equal

$$S_1(x_1) = -100x_1 - 400x_2 = -100x_1 - 400(1 - x_1) = 300x_1 - 400. \quad (1)$$

The graph of this function is a straight line, which is shown in Fig. 1 $\mathbf{A_1}$.

Similarly, the graphs of the functions of the expected costs of the firm *A* when applying each pure strategy against the mixed strategies of the supplier company *B:*

$$S_2(x_1) = -150x_1 - 300x_2 = -150x_1 - 300(1 - x_1) = 150x_1 - 300; \quad (2)$$

$$S_3(x_1) = -190x_1 - 250x_2 = -190x_1 - 250(1 - x_1) = 60x_1 - 250; \quad (3)$$

$$S_4(x_1) = -330x_1 - 220x_2 = -330x_1 - 200(1 - x_1) = 130x_1 - 200, \quad (4)$$

which are respectively indicated in pic. 1 as A_2, A_3, A_4.

With the reliability of the supplier $\boldsymbol{x_1 = 0,4}$ to the intersection with the lines of functions of the expected costs of the company *A* we find out that the strategy A_3, that will provide the minimum cost –226 USD will be optimal.

If the reliability of the provider is $\boldsymbol{x_1 \leq 0,263}$, it is better to use the fourth strategy; with the reliability of the supplier $\boldsymbol{0,263 \leq x_1 \leq 0,555}$ the optimal strategy will be A_3, at $\boldsymbol{0,555 \leq x_1 \leq 0,677 - A_2}$, at $\boldsymbol{0,667 \leq x_1 \leq 1 - A_1}$, (see Fig. 1).

These critical reliability values are derived from the overall solution of equations (1) – (4), which are in pairs: (3) and (4) – Point *b*, (2) and (3) – dot *c,* (1) and (2) – dot *d.*

The following is a broken line *abccde* shows how the expenses of the firm *A* when the supplier's reliability changes to 0 to 1 are changed.

As you can see from the graph, the increase of the supplier's reliability does not automatically reduce the cost of the firm *A.* In fact, when the provider's reliability grows from 0 to 0.263, the company costs *A* increase from –200 to

$$\boldsymbol{S_4(0,263) = -200 - 130 \cdot 0,263 = -243,2 y.o.}$$

The increase in costs is due to the fact that the raw material is purchased from the second supplier, and the irregular deliveries of the main supplier (with a probability of 0.263) lead to additional costs.

With the reliability of the supplier $\boldsymbol{x_1 = 0,263}$ cost of the firm *A* maximum of all possible at a reasonable choice of the firm *A* Their strategies (this maximum depends on the values of the conditionally selected costs (see table1)).

If the game was antagonistic, that is, the supplier wanted to inflict maximum damage to the company *A*, its optimal reliability would have to be equal to$\boldsymbol{x_1 = 0,263}$. At the same time, the company's *A* costs would have been 234.2 and the optimal one would be the strategy A_3 and $\boldsymbol{A_4}$, (point *b*is at the intersection of linesA_3 and $\boldsymbol{A_4}$*)*. In fact, substituting $\boldsymbol{x_1 = 0,263}$ into the equation (3), (4), we get

$$S_3(0{,}236) = S_4(0{,}236) = -234{,}2\ y.o.$$

Due to the fact that the company-supplier seeks to inflict firm *A* maximum damage, the latter can not choose any one of the net strategies A_3 or A_4, for in this case, if the firm *B* will change the reliability of deliveries in the lesser side of $\boldsymbol{x_1 = 0,263}$ (in the case of the Strategy $\boldsymbol{A_3}$)

or in a bigger side (in the case of a strategy A_4), losses will increase and be greater than – 234,2$.

As in the antagonistic game the first and second strategies of the firm *A*are ineffective, consider the possibility of finding a mixed strategy A_3 i $\boldsymbol{A_4}$, with such probabilities of application, in which the losses of firm *A* would not be greater than $234.2 under any strategies of firm *B.* We will construct the chart of expenses of the firm *A,* which applies its mixed strategy, consisting of pure strategies A_3 i $\boldsymbol{A_4}$ against each clear strategy $\boldsymbol{B_1}$ i B_2 of firm *B* (fig. 2).

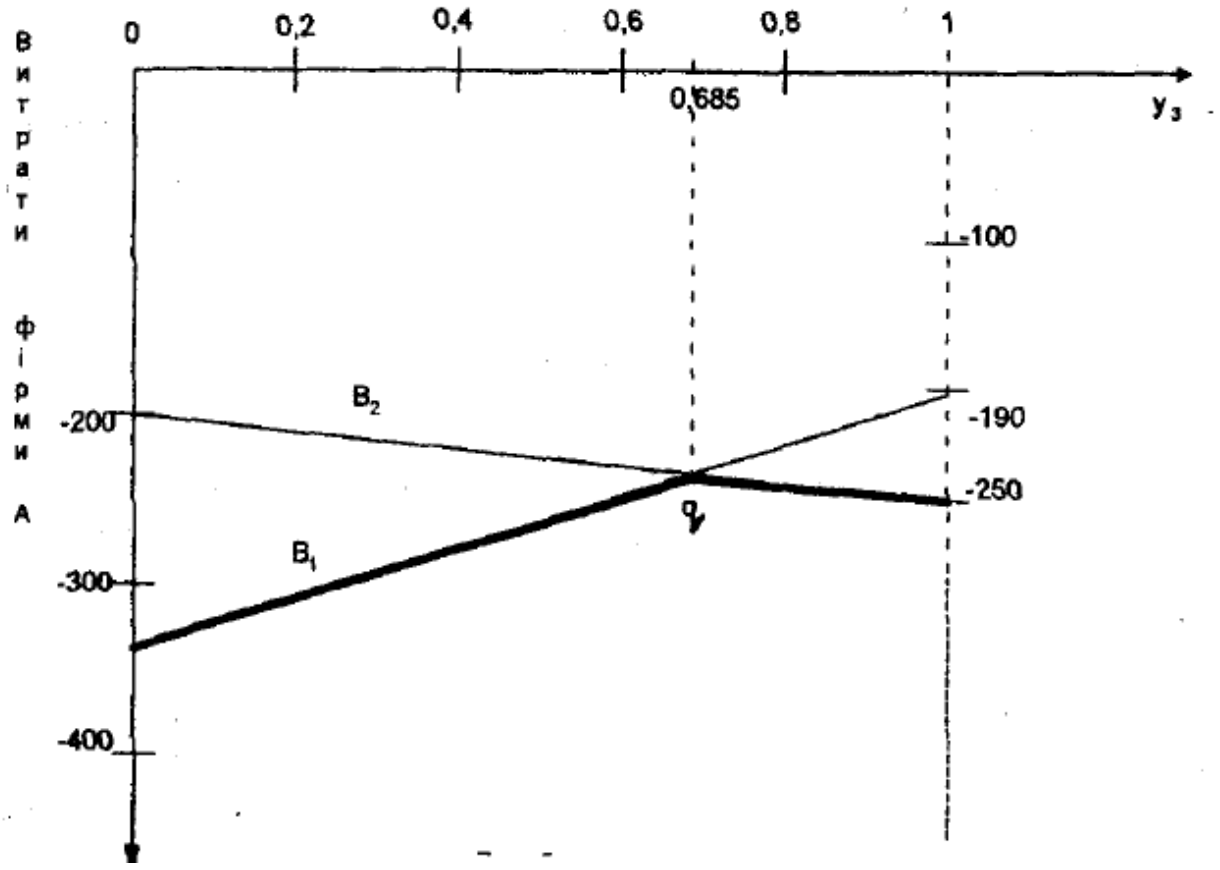

Fig. 2

Using y_3 let's denote the probability of application of strategy A_3, and using y_4 – strategy $\boldsymbol{A_4}(\boldsymbol{y_3 + y_4 = 1})$. From a graph constructed similarly to the graph in Fig.1, it is seen that the optimal mixed strategy of the firm *A* includes the strategies A_3 and A_4, which are applied with probabilitie $s_3 = 0{,}685$ and $y_4 = 0{,}315$.

The optimal costs of firm *A* (called in the case of an antagonistic game *the price of the game)* equal to the ordinates of the intersection point q. Substituting y_3 = 0,685 into any of the equations of the straight $S_1(y_3) = 140y_3 - 330$; $S_2(y_3) = -50y_3 - 200$,, we get the same cost value – $234.2, which was previously calculated. Figure 2 shows that in an antagonistic game firm D should not deviate from its optimal mixed strategy $y_1 = y_2 = 0$; y_3 = 0.685; y_4 = 0.315 as costs increase (in the direction of the lines shown). When y_3< 0,685 firm *B* will start to apply a pure strategy B_1,when y_3> 0,685 – a pure stategy B_2 and will cause losses to firm*A* greater than – 234,2 $.

Thus, if the game was antagonistic (i.e each player inflicts maximum damage to the opponent), players should be recommend the following optimal strategies:

to the firm $\boldsymbol{A - y_1 = y_2 = 0; y_3 = 0,685, y_4 = 0,315}$;

to the firm $\mathbf{B - x_1 = 0,263, x_2 = 0,737}$.

The cost of the game (i.e expected losses of the firm *A)* equals – 234.2$.

3. Statistical games and criteria for decision making

Production processes are managed by implementing a sequence of solutions. In the absence of sufficiently complete information on the state of the management object, uncertainty arises in decision making[2].

The reasons for this may be different: the inability to obtain information before the decision is made; very high information costs; inability to eliminate uncertainty due to objective nature. For example, the random nature of the demand for products makes it impossible to accurately predict the volume of its output.

In order to mitigate the adverse effects in each case, the degree of risk and the information available must be taken into account. In this case, the decision-maker enters a game relationship with some abstract person who can be conditionally called «nature».

In other words, the decision-maker must be able to find management decisions when nature does not consciously choose its optimal strategies. Any economic activity can be considered as a game with nature, about whose conditions there are some probability characteristics.

Nature's indifference to the game (win) and the opportunity for the decision-maker to obtain additional information about its condition

[2] Churchman W., Acof R., Arnoff L. Introduction to Operations Research. Moscow: Science, 1968, 488 p.

distinguish the game from nature from a regular matrix game involving two conscious players.

Statistical games are the main model of decision theory in the context of partial uncertainty.

Let's return to the problem again – games with firms *A* and *B.* Since such a game is usually not antagonistic, its solution cannot be considered optimal. In fact, the supplier company *B* does not want to cause the firm *A*maximum damage and therefore its reliability may be any, not necessarily the worst from the point of view of the company *A* (the worst for the firm *A* – supplier reliability 0.263).

If, for example, the reliability of firm B$\boldsymbol{x_1 = 0,4}$, and firm *A* continue to apply the optimal mixed strategy for the antagonistic game, then the expected costs of firm *A* do not decrease. Indeed,

$$\mathbf{S(0,4) = 0,685 \cdot S_3(0,4) + 0,3 \cdot S_4(0,4) =}$$
$$\mathbf{= 0,685(60 \cdot 0,4 - 250) + 0,315(-130 \cdot 0,4 - 200) = -234,2 y.o.}$$

where $\boldsymbol{S_3(x_1), S_4(x_1)}$are determined by relations (3), (4).

In order to reduce costs for such reliability of the supplier, it is necessary to abandon the optimal strategy and use, as shown above, the pure third strategy A_3 (see Fig. 1).

The costs are reduced to -226 USD.

Thus, the peculiarity of the solution of games with nature in the conditions of certainty is that a mixed strategy of nature is given, that is, all the probabilities of states are known:

$$\boldsymbol{x_j, j = 1,2,\dots,n;\ \sum_{j=1}^{n} x_j = 1}$$

This allows for each *i* pure strategy of the active player to calculate the mathematical expectation of his/her win against the known mixed strategy of nature by the formula

$$\boldsymbol{S_i(x_1,\dots,x_n) = \sum_{j=1}^{n} a_{ij}x_j, i = 1,2,\dots,m,}$$

where a_{ij} – element of the payment matrix, located at the intersection of the i-th row and the j-th column:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{pmatrix}.$$

The maximum element in the calculated column $S_i\left(i = \overline{1, m}\right)$ of mathematical expectations of winnings,

$$S_{max} = \max_i S_i(x_1, \dots, x_n),$$

determines the most profitable strategy of the active player and quantifies the maximum possible winnings.

If there are two or more maximum elements in this column, then the respective strategies can be used, either purely or in any combination.

This approach to the solution of games against nature takes place only when the probabilities of the states of nature are given. Often decisions are made in the absence of information about such probabilities. Then, knowing the possible list of states of nature, consider them equally probable.

At the same time, the maximum mathematical expectation of winning (Laplace criterion) can be used to select the optimal strategy, but this criterion can only be used for an even distribution of probabilities $x_i = 1/n\ (i = 1, 2, \dots, n)$.

Let us consider the other criteria that are applied to solve the games with nature under uncertainty conditions[3].

4. Wald's Maximin Criterion.

In this case, such a solution is chosen that guarantees a win at least

$$S_B = \max_i \min_j a_{ij}.$$

With respect to the game under consideration, under any behavior of the supplier firm *B* the firm *A* may choose any of its pure strategies. There can be two consequences for each strategy. For guarantee, the company *And* takes into account the one that gives the smallest winnings. Write it down in the column of minimums of rows (tab. 3).

[3] Churchman W., Acof R., Arnoff L. Introduction to Operations Research. Moscow: Science, 2007, 2010 p.

Table 3

Стратегія фірми *A*	Стратегія фірми *B*		Мінімум рядків
	B_1	B_2	
A_1	— 100	— 400	— 400
A_2	— 150	— 300	— 300
A_3	— 190	— 250	— 250 (максимін)
A_4	— 330	— 200	— 330
Максимум стовпчиків	— 100	— 200(мінімакс)	

From these lines, you can choose the one with which this minimum win will be the maximum (-250). This is the optimal strategy of firm A, chosen in accordance with the Wald's criterion.

Table 3 also defines the minimax strategy of the firm *B*, for which the maximum payout is selected from each column and such strategy is accepted that gives the firm *A* the minimal of these maximal payoffs.

In this case, the strategy of the firm *B.* is the second one. Thus, the maximin strategy $A_3$of the firm *A* neutralizes the minimax strategy B_2 of the firm *B.*

Obviously, the Wald's criterion can be seen as extreme pessimism in the assessment of circumstances. According to it, it is recommended to choose one of the alternative strategies, the pessimistic assessment of which is the best.

5. Maximin criterionCriterion

This criterion assumes that the state of nature will be most favorable for us, so we must choose a solution that provides the maximax gain among the maximum possible, i.e

$$S_m = \max_i \max_j a_{ij}.$$

Using the maximax criterion in the problem under consideration, we obtain S_m= -$ 100. This is in line with strategy $\boldsymbol{A_1}$, that is, firm *A* should not take any actions, assuming that the firm *B* will apply the most favorable for itself strategy B_1.

In contrast to the Wald's criterion, the maximax criterion can be considered as extreme optimism in the assessment of the circumstances while making the decision.

6. Hurwitz's (pessimism – optimism) Criterion

When choosing a solution instead of two extremes in the assessment of the situation (optimism – pessimism) it becomes logical to adhere to some intermediate position, which takes into account the possibility of both the worst and the best behavior of nature. Such a compromise criterion was suggested by Hurwitz.

In his opinion, we must determine the linear combination of the minimum and maximum payoffs for each decision and choose the strategy for which this value will be greatest:

$$S_r = \max_i \left[\alpha \max_j a_{ij} + (1 - \alpha) \min_j a_{ij}, \right]$$

where $\alpha (0 \leq \alpha \leq 1)$is the degree of optimism. When $\alpha = 0$, the Hurwitz criterion goes to the maximum Waldo criterion; at $\alpha = 1$ – matches the maximax criterion. The choice of the degree of optimism is influenced by the measure of responsibility: the greater the consequences of wrong decisions, the greater the desire to insure, the closer$\alpha$to zero.

The influence of the degree of optimism on the choice of the solution in the problem under consideration is given in Table 4.

Table 4

Стратегія	Ступінь оптимізму								
	1/9	2/9	3/9	4/9	5/9	6/9	7/9	8/9	1
A1	370	-340	-310	-280	-250	-220	-190*	-160*	-130*
A2	-285	-270	-255	-240	-225	-210*	-195	-180	-165
A3	-244*	-238*	-232*	-226*	-220*	-214	-208	-202	-196
A4	-317	-304	-281	-278	-265	-252	-239	-226	-213

Note. ***The value*** $\mathbf{S_r}$ ***for each value*** **a** ***is marked with a *.***

When $\alpha \leq 5/9$ Hurwitz's criterion recommends to the firm *A* using strategy A_3, when $5/9 \leq \alpha 2/3$– strategy A_2, in other cases – $\boldsymbol{A_1}$.

7. Savage's minimax regret criterion

The essence of this criterion is to choose such a decision so as not to allow excessively large losses, to which the wrong decision can lead.

For this purpose, a «risk matrix» is built, the elements of which show how much damage we will bear if we do not choose the best solution for each state of nature.

The risk of a player when choosing a solution (strategy) $\boldsymbol{A_i}$ under the conditions $\boldsymbol{B_j}$is the difference between the maximum win, available in these conditions, and the win that the player will receive in the same conditions using the strategy$\boldsymbol{A_i}$, Let us denote this value by $r_{ij.}$

If the player knew in advance the future state of nature $\Pi_{\boldsymbol{j}}$, he would choose a strategy that would correspond to the maximum element in the specified column: max $a_{ij.}$ Then, by definition, the risk is equal to

$$\boldsymbol{r_{ij} = \max_i a_{ij} - a_{ij}}.$$

The risk matrix is constructed as follows:

1) the largest element is determined for each state of nature (column);
2) the risk matrix element is obtained by subtracting the corresponding element of the payment matrix from the maximum element of this column.

Savage's criterion recommends that in uncertainty conditions, one should choose a solution that provides minimal rate of the maximum risk:

$$\boldsymbol{S_c = \min_i \max_j r_{ij} = \min_i \max_j \left(\max_i a_{ij} - a_{ij} \right)}.$$

The risk matrix for the problem under consideration is given in Table. 5.

Table 5

Стратегія	B_1	B_2	Максимум ризиків
A_1	0	200	200
A_2	50	100	100
A_3	90	50	90*
A_4	130	0	130

To the right of the risk matrix there is the maximum risk column for each strategy $\boldsymbol{A_i}$,. Minimax risk is reached when choosing $\boldsymbol{A_3}$: $\boldsymbol{S_c = 90}$.

8. Bayes-Laplace criterion

Applying this criterion, they depart from the conditions of complete uncertainty (lack of information about the state of nature), believing that a certain probability *of their* occurrence can be used for the probable states of nature.

In this case, determining the mathematical expectation of winning for each decision, they choose the one that provides the highest value of the win:

$$S_{БЛ} = \max_i \sum_{j=1}^{n} a_{ij} X_j \ . \qquad (5)$$

The Bayes–Laplace principle can be applied if the states of nature under study and the decisions made are many times repeated.

Then, for example, statistical methods, based on the frequency of occurrence of certain states of nature in the past, can estimate the likelihood of their occurrence in the future.

For single solutions that do not repeat, the Bayes–Laplace principle cannot be applied even when the states of nature are repeated.

This is because such solutions violate the stationarity of the probability distribution of nature.

Let us suppose that firm *A*, before making a decision, has analyzed how accurately the supplier firm *B* had previously followed the delivery deadlines, and has determined that in 25 cases out of 100 raw materials were delayed. It follows that the state $\boldsymbol{B_1}$, can be assigned with the probability X_1 = 0,75, and the state B_2 – with the probability X_2 = 0,25. Then, according to the Bayes-Laplace criterion, the solution (strategy)A_1 is optimal (Table 6).

Table 6

Стратегія	$\sum_{j=1}^{n} a_{ij} X_j$
A_1	–175,0*
A_2	–187,5
A_3	–205,0
A_4	–297,5

The criteria listed do not draw out the full variety of decision selection criteria under uncertainty, including the criteria for selecting the best mixed strategies.

The solutions recommended by the considered criteria for the studied task are given in Table. 7.

Table 7

Стратегії	Критерій						Кількість прийнятих рішень за різними критеріями
	Вальда	Максимаксний	Гурвиця	Севіджа	Лапласа	Байєса – Лапласа	
A_1		+	+			+	3
A_2			+			+	2
A_3	+		+	+	+	+	5
A_4							0

The table shows that the optimal behavior depends largely on the accepted optimization criterion. Therefore, the selection of the criterion is the most important question in the study of operations.

Each choice of criterion leads to the approval of a decision, which may differ from the decision made in accordance with another criterion. However, the situation is never so uncertain that it is impossible to obtain at least partial information on the probability of the distribution of the states of nature in the situation being analyzed.

In this case, estimating the probability distribution of the states of nature, they apply the Bayes-Laplace criterion or conduct an experiment to clarify the behavior of nature.

9. Modeling Effectiveness of Information Retention Costs under Uncertainty Conditions

In games with nature, making one or the other decision, we can not find in advance in what state it is at the time of implementation of the decision, even when we know the probability distribution of its states. Therefore, the solution, that is the best for such a probability distribution of the states of nature, will not be better than the state which nature will truly take[4].

[4] Akof R., Sasiena M. Fundamentals of Operations Research. Moscow: World, 2007, 2010 p.

It follows from the Bayes-Laplace criterion that the solution is the player's strategy A_i, which provides the maximum average win. This strategy is best in a situation where nature «chooses» its states by chance, but with a known law of distribution.

However, the maximum average win is not the maximum attainable win. Indeed, let us imagine a perfect case when we know the future state of nature precisely before deciding. Then we can apply that pure strategy that allows against this state of nature $\Pi_j$to get a maximum gain of$\boldsymbol{v_j = \max_i a_{ij}}$.

With a sufficiently large number of repetitions of the game in terms of full prediction, the average maximum win will be equal to

$$S = \sum_{J=1}^{n} v_j X_j \tag{6}$$

Since $\boldsymbol{a_{ij} \le v_j}$, the value (5) always does not exceed the value (6). Then, the difference

$$\boldsymbol{d = \sum_{j=1}^{n} v_j X_j - \max_i \sum_{j=1}^{n} a_{ij} X_j} \tag{7}$$

is the magnitude of the additional average gain due to an accurate knowledge of the future state of nature at the time of making the decision.

Suppose that there is reliable information (an ideal experiment) that accurately predicts the future state of nature. We estimate the expediency of acquiring such information (conducting such an experiment) at the cost of obtaining it *p*. Since the additional gain (excluding the cost of information) is *(1 – c,* purchase is expedient when *ŭ – c*> 0, i.e taking into account (7) when

$$\boldsymbol{\sum_{j=1}^{n} v_j X_j - \max_i \sum_{j=1}^{n} a_{ij} X_j > c.} \tag{8}$$

And if not, the acquiring of Information (experiment) has to be abandoned and the strategy $A_{i,}$ which provides the maximum average win.

Let us now return to the aforecited problem, when the reliability of the supplier company *B* is, for example, 0,6, and the cost of reliable information (ideal experiment) is $ 40 per day.

The table 8 shows the costs (benefits) of the company *A.*

Table 8

Рішення фірми А	Стратегія фірми В		Середній виграш фірми А при $X_1 = 0{,}6$
	B_1	B_2	
A_1	–100	–400	–220
A_2	–150	–300	–210
A_3	–190	–250	–214
A_4	–330	–200	–278
Максимальний виграш фірми А	–100	–200	–140

Obviously, if the delivery is timely (state $\boldsymbol{B_1}$), the best solution is A_1 and the costs are equal to \$100, if there is no supply (state B_2), is the best solution D, and the costs are \$200.

The average maximum payout (minimum cost) of the firm *A* will be $\boldsymbol{S = -100 \cdot 0,6 - 200 \cdot 0,4 = -140\ у.о.}$,which is greater than the maximum average payout, which equals to \$210, for$\boldsymbol{-140 - 210 = 70\ у.о.}$

As the information on timely delivery costs the company *A* the amount of *c* = \$40, it receives an additional gain of c = \$70 – \$40 = \$30 a day. Obviously, the acquisition of such information is appropriate.

The condition for the expediency of acquiring information (experiment) (8) can be written as

$$\boldsymbol{\min_i \sum_{j=1}^{n} (v_j - a_{ij}) X_j > c}, \qquad (9)$$

where $\boldsymbol{v_j - a_{ij} = r_{ij}}$is nothing but risk (see § 3.4) and the sum on the left (9) is the average risk. Therefore, an acquisition (experiment) is appropriate when the cost of obtaining it is less than the minimum average risk:

$$\min_i \sum_{j=1}^{n} X_j r_{ij} > c.$$

Above, the situation where we can accurately determine the future state of nature has been considered. However, more often, additional information can only clarify the a priori probability distribution of these states. In this case, they are talking about clarifying information (a non-ideal experiment). Now instead of a clear answer «Tomorrow the delivery

will happen» or «Tomorrow the delivery will not happen» we will most likely receive answers that can be formulated as follows:

C_1 – the supplier is more reliable than we think;

C_2 – the supplier reliability is almost the same as we think;

C_3– the supplier is less reliable than we think.

Each of these results comes with some probability, the distribution of such probabilities depends on the states of nature, that is, on the conditions in which information is obtained (an experiment is conducted). Let us suppose that the multiple experiments that were conducted before, allowed us to collect data on the conditional probabilities of the results of experiments $\boldsymbol{P\left(C_i/B_j\right)}$, which are given in Table 9.

Table 9

Результат експерименту	Стан постачальника		Повна ймовірність результату при $X_1 = 0{,}6$
	B_1	B_2	
C_1	0,5	0,2	0,38
C_2	0,4	0,3	0,36
C_3	0,1	0,5	0,26

Knowing the conditional probabilities of the results $\boldsymbol{P\left(C_i/B_j\right)}$ and the probabilities of the states of nature $\boldsymbol{X_j}$, one can calculate the full probabilities of the results $\boldsymbol{P(C_i)}$ by the formulas

$$\boldsymbol{P(C_i) = \sum_{j=1}^{n} P\left(C_i/B_j\right)X_j}.$$

For probability $\boldsymbol{X_1 = 0,6}$, they are:

$$\boldsymbol{P(C_1) = \sum_{j=1}^{2} P\left(C_1/B_j\right)X_j = 0,5 \cdot 0,6 + 0,2 \cdot 0,4 = 0,38};$$

$$\boldsymbol{P(C_2) = \sum_{j=1}^{2} P\left(C_2/B_j\right)X_j = 0,4 \cdot 0,6 + 0,3 \cdot 0,4 = 0,36};$$

$$\mathbf{P(C_3) = \sum_{j=1}^{2} P(C_3/B_j)X_j = 0,1 \cdot 0,6 + 0,5 \cdot 0,4 = 0,26}$$

and are given in the last column of the table 9. Knowing them, you can determine the specified probabilities of the state of nature after the experiment.

Bayes formulas serve to calculate the posterior (post-experimental) the distribution of probabilities of the state of nature.

$$X_j(C_i) = \frac{X_j P(C_i/B_j)}{P(C_i)}$$

We calculate this distribution if the result of the experiment was $\boldsymbol{C_1}$:

$$\boldsymbol{X_1(C_1) = \frac{X_1 P(C_1/B_1)}{P(C_1)} = \frac{0,6 \cdot 0,5}{0,38} = 0,79;}$$

$$\boldsymbol{X_2(C_1) = \frac{X_2 P(C_1/B_2)}{P(C_1)} = \frac{0,4 \cdot 0,2}{0,38} = 0,21;}$$

Now (after receiving the information, conducting the experiment) the reliability of the supplier instead of 0.6 is estimated at 0.79. The best strategy of the company *A* according to Bayes-Laplace criterion instead of A_2 will become A_1 (table 10), which provides the minimum of costs – \$163.

Table 10

Рішення фірми А	Стан фірми В		Середні витрати фірми А при надійності фірми В X_j			
	B_1	B_2	0,6	0,79	0,667	0,231
A_1	–100	–400	–220	–163	–200	–331,1
A_2	–150	–300	–210	–181,5	–200	–265,7
A_3	–190	–250	–214	–202,5	–210	–236
A_4	–330	–200	–278	–303	–286,7	–230,1

Table 10 shows that without additional information (conducting the experiment), the costs were in the amount of – \$210, after that they became – \$163. The extra payoff is –163 – (–210) = \$47. The cost of obtaining information (conducting the experiment) is \$40. But these

calculations are still not enough to conclude that it is advisable to acquire information, since the result could have been different (C_2 or C_3, see Table 9), and additional payoffs having these results are still unknown. Let us calculate them.

If the result of the experiment is C_2, the posterior probabilities of the states of the firm *B* are

$$\boldsymbol{X_1(C_2) = \frac{X_1 P(C_2/B_1)}{P(C_2)} = \frac{0,6 \cdot 0,4}{0,36} = 0,667;}$$

$$\boldsymbol{X_2(C_2) = \frac{X_2 P(C_2/B_2)}{P(C_2)} = \frac{0,4 \cdot 0,3}{0,36} = 0,333;}$$

For reliability of the supplier company *B* average costs of the company *A* are shown in table 10. It is seen that the solutions *A*, or A_2 will be optimal or any combination of these strategies, the win is $200

If the result of the experiment is C_3, then

$$\boldsymbol{X_1(C_3) = \frac{X_1 P(C_3/B_1)}{P(C_3)} = \frac{0,6 \cdot 0,1}{0,26} = 0,231;}$$

$$\boldsymbol{X_2(C_3) = \frac{X_2 P(C_3/B_2)}{P(C_3)} = \frac{0,4 \cdot 0,5}{0,26} = 0,769;}$$

With the reliability of the company *B* 0,231 the minimum average cost of the company *A*, which is equal to $230,1, is achieved when applying the strategy $\boldsymbol{A_4}$.

Using the full probabilities of information obtaining results, we calculate the average cost (gain) of the firm *A:*

-163 0.38 – 200 0.36 – 230.01 0.26 = -193.8 $

In the absence of additional information, they are -210u.o. The extra payoff is –163 – (–210) = $47. Therefore, at the cost of additional information $40 its acquisition is inappropriate.

10. Determining the optimal inventory of commercial companies

Let us by *x* denote the market demand for the products of a trading firm for some fixed period of time (day, week, month, etc.), which is unknown in advance. Units of products for sale can be both physical (kilograms, liters, etc.) or monetary. Let us suppose that unrealized in this

period products lose their consumer qualities during storage and can not be sold in the next period[5].

Let us hereafter using C_1 denote the sum of the prime cost and additional costs of storage of a unit of production which was not realized in the mentioned period of time due to the fact that the demand for it was less than projected, and using C_2 we denote the loss of profit per unit of production, which is caused by its absence, when the demand for it exceeds its quantity d, which is available in the firm.

In view of the above designations, the loss of the firm is determined by the function

$$\boldsymbol{V(x,s)} = \begin{cases} \boldsymbol{C_1(s-x)}, \text{коли } \boldsymbol{s \geq x}. \\ \boldsymbol{C_2(x-s)}, \text{коли } \boldsymbol{s < x}. \end{cases} \tag{10}$$

We will consider the demand for products *x* as a random variable with a distribution function *F (x),* which can be determined on the basis of statistical observations or other information. Then the losses of firm *V (x, s*), determined by the ratio (10), are a function of the random value *x* (demand) and the value of the product stock *s*, and the task of defining the optimal stock of products of a trading company can be considered as a statistical game with «nature». Player *A* is a trading company, player *B* is a certain conditional customer (market) with a known distribution function *F (x).*

The purpose of the company is to find such a value of the inventory *s*, that would minimize the mathematical expectation (average)

$$[\mathbf{V(x,s)}] = \int_{-\infty}^{\infty} \mathbf{V(x,s)dF(x)} \tag{11}$$

of its costs.

Substituting in (11) the function of losses (10), we obtain

$$\boldsymbol{M[V(x,s)] = C_1 \int_{-\infty}^{s} (s-x)dF(x) + C_2 \int_{s}^{\infty} (x-s)dF\,(x) =}$$

$$\boldsymbol{= C_1 \left[s \int_{-\infty}^{s} dF(x) - \int_{-\infty}^{s} xdF(x) \right] + C_2 \left[\int_{s}^{\infty} xdF(x) - s \int_{s}^{\infty} dF(x) \right] =}$$

$$\boldsymbol{= C_1 \left[sF(s) - \int_{-\infty}^{s} xdF(x) \right]}$$

[5] Degtyarev Yu.I. Operations Research. Moscow: Higher chool, 1986, 320 p.

$$+\boldsymbol{C_2}\left[\int_{s}^{\infty} \boldsymbol{xdF(x)} + \int_{-\infty}^{s} \boldsymbol{xdF(x)} - \int_{-\infty}^{s} \boldsymbol{xdF(x)} - \boldsymbol{s}\big(\boldsymbol{1} - \boldsymbol{F(s)}\big)\right] =$$

$$= \boldsymbol{C_1}\left[\boldsymbol{sF(s)} - \int_{-\infty}^{s} \boldsymbol{xdF(x)}\right] + \boldsymbol{C_2}\left[\boldsymbol{M[x]} - \int_{-\infty}^{s} \boldsymbol{xdF(x)} - \boldsymbol{s}\big(\boldsymbol{1} - \boldsymbol{F(s)}\big)\right] =$$

$$= (\boldsymbol{C_1} + \boldsymbol{C_2})\boldsymbol{sF(s)} - \boldsymbol{C_2 s} - (\boldsymbol{C_1} + \boldsymbol{C_2}) \int_{-\infty}^{s} \boldsymbol{xdF(x)} + \boldsymbol{C_2 M[x]},$$

where M[x] denotes the mathematical expectation of the random variable x.

To find the minimum value of the mathematical expectation $\boldsymbol{M[V(x,s)]}$, which is a function of the inventory t, we equate the first derivative of this function to zero on the variable *s:*

$$\frac{dM[V(x,s)]}{ds} = (C_1 + C_2)[F(s) + sf(s)] - C_2 - (C_1 + C_2)sf(s) =$$
$$= (C_1 + C_2)F(s) - C_2 = 0, \qquad (12)$$

where $\boldsymbol{f(s)}\left(\boldsymbol{f(s)} = \frac{\boldsymbol{dF}}{\boldsymbol{ds}}\right)$denotes the probability density of demand distribution at point s.

From relation (12), which is the equations, it implies that the optimal value of the inventory of trading company 5_0, which minimizes its losses, satisfies the condition

$$F(s_0) = \frac{C_2}{C_1 + C_2}. \qquad (13)$$

By definition $\boldsymbol{F(s_0) = P|(x < s_0)}$, i.e equality (13) means that the optimal value of the inventory s_0 should meet a requirement that the probability that demand is less than s_0 is equal to $\frac{\boldsymbol{C_2}}{\boldsymbol{C_1 + C_2}}$.

A simple algorithm for determining s_0 follows from the latter. On the basis of statistical observations a graph of the distribution function (cumulative) is constructed. Graphically or from the analytic expression of the distribution function F(x) we find the following value s_0, for which equation (13) works. If the distribution is close to known, for example,

normal, the value s_0 can be determined from the tables of normal distribution.

Let us consider how this is done in practice.

Suppose that the optimal value of inventory should be determined when $C_1 = 0,6$, $C_2 = 0,4$ and we have the statistical observations of daily demand for products over 31 days, which are shown as income (Table 11).

Table 11

Дохід, у. о.	0–2	2–4	4–6	6–8	8–10	10–12	12–14	14–16	16–18	18–20
Частота	0	1	2	4	6	7	5	3	2	1
Відносні частоти	0	0,03	0,06	0,13	0,20	0,23	0,16	0,10	0,06	0,03
Накопичені частоти	0	0,03	0,09	0,22	0,42	0,65	0,81	0,91	0,97	1,00

Based on these data, we calculate frequencies, relative frequencies, cumulative frequencies (cumulative line) and build a distribution graph (Fig. 3) using known formulas from statistics.

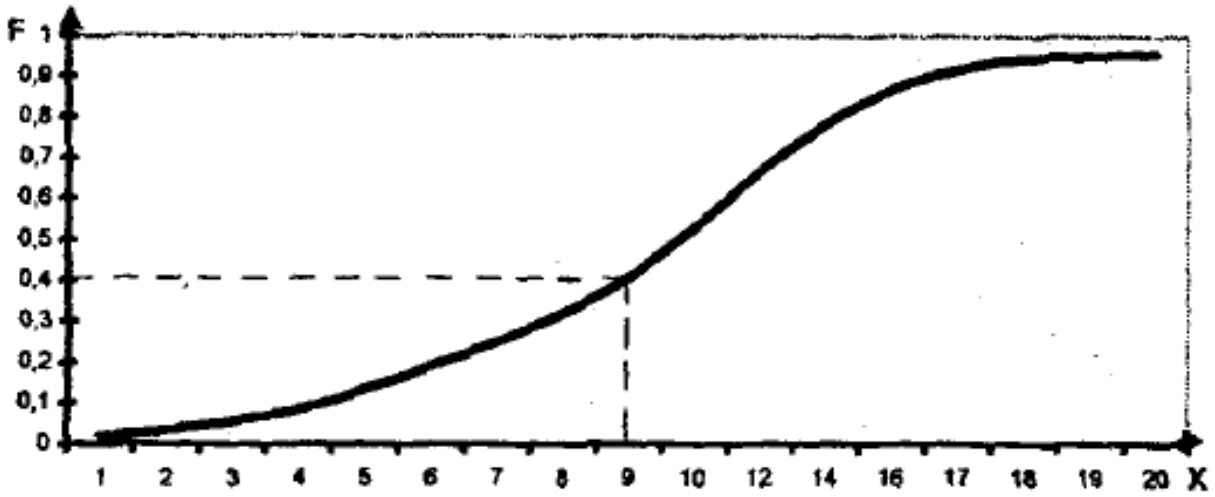

Fig. 3

Next, by the formula (13) we calculate

$$F(s_0) = \frac{C_2}{C_2 + C_1} = \frac{0,4}{0,6 + 0,4} = 0,4$$

and graphically determine $\$\boldsymbol{s_0} \approx \mathbf{9}$, where the needed optimal value of the daily stock of products of a trading firm will be given in a value that minimizes the mathematical expectation (average value) of its losses per one day.

Similar to the above mentioned, calculations can be made to predict the optimal stock of production of firms for any period of time.

REFERENCES

1. Kofman A. Methods and models of operations research. Moscow: World, 1977, 432p.
2. Churchman W., Acof R., Arnoff L. Introduction to Operations Research. Moscow: Science, 1968, 488 p.
3. Churchman W., Acof R., Arnoff L. Introduction to Operations Research. Moscow: Science, 2007, 2010 p.
4. Akof R., Sasiena M. Fundamentals of Operations Research. Moscow: World, 2007, 2010 p.
5. Degtyarev Yu. I. Operations Research. Moscow: Higher chool, 1986, 320 p.

Information about the author:

Muliava O. M.

Candidate of Physical and Mathematical Sciences, Associate Professor, Deputy Dean of the National University of Food Technology

NOTES

NOTES

NOTES

Publishing house “Liha-Pres”
9 Kastelivka str., Lviv, 79012, Ukraine
44 Lubicka str., Toruń, 87-100, Poland

Printed by the publishing house “Liha-Pres”
Passed for printing: August 30, 2019.
A run of 150 copies.