Four-dimensional ACC analysis

Mustafina Nazgul Ibragimovna
HSE University
Perm, Russia
ORCID: 0000-0001-8617-659X

Plaksin Mikhail Alexandrovich
HSE University
Perm State University
Perm, Russia
ORCID: 0000-0002-6288-8610

Mikisheva Polina Alekseevna
HSE University
Perm, Russia
ORCID: 0009-0003-0246-5418

***Abstract*— The article discusses the issues of planning and resource management in the process of testing software systems. The paper presents the ACC analysis method used at Google to optimize the distribution of efforts for testing different parts of the system. Extending the method by adding a fourth characteristic - actors (classes of system users) - allows for a more flexible assessment of action requirements and user skill levels. Illustrative examples of system attributes and components help understand the principles of the method. The work proposes a new approach to risk management and process improvement in testing software systems in a multidimensional space. The effectiveness of applying the enhanced ACC analysis method using a risk-oriented approach was demonstrated using the example of a control system for technological operations in the repair of electric motors, for which attributes, components, actors were identified, opportunities at their intersection were analyzed, and testing was conducted, which helped improve the system's quality.**

Keywords*— *Testing, risk registry, test plan, test cases, system for controlling technological operations, ACC methodology, Google+, risk-oriented approach, risk identification, efficiency improvement, risk assessment, productivity enhancement, process optimization, data analysis, software engineering.

I. INTRODUCTION

Testing is one of the most important processes in software system development. The main goal of testing is to detect errors and defects in the product being tested, as well as to identify discrepancies between the product's characteristics and the requirements and expectations of users [1-15]. One of the main problems associated with testing is the lack of resources for exhaustive testing. A good test plan should be created at the beginning of a project and may change during the development of the software product. This makes tasks such as planning and management relevant: the task of best resource allocation allocated to the testing phase, and the task of assessing the progress made during testing.

At Google, a special method called ACC analysis [16] is used to address these challenges. It allows ranking different parts of the developing system using a popular risk-oriented approach today, providing recommendations on what percentage of efforts should reasonably be planned for testing specific parts of the system.

The original ACC analysis manipulates three characteristics of a software system: attributes, components, and capabilities. ACC stands for Attribute, Component, Capability. This article proposes to expand the method by including a fourth characteristic - actors (classes of system users). This increases the manageability of the testing process, allows for a different perspective on testing organization, makes the process more flexible, clarifies requirements for implementing specific actions, and also the level of user qualification.

The article describes the ACC analysis method used for software testing and its proposed enhancement. ACC analysis is a risk-oriented approach that ranks different parts of a developing system based on the percentage of efforts required for testing. The original ACC analysis considers three characteristics: attributes, components, and capabilities. The proposed enhancement adds a fourth characteristic: actors (roles of system users). It increases the manageability of the testing process, provides a different perspective on testing organization, makes the process more flexible, clarifies requirements for implementing specific actions, and considers the level of user qualification. The text also provides an example of applying the improved ACC method for analyzing a Repair Shop system. The enhancement allows for better distribution of testing resources, prioritization of testing based on risk levels, and consideration of user roles and qualifications.

II. THE ORIGINAL ACC METHOD

Since ACC analysis is not well known enough, first a description of the traditional methodology is provided, followed by our proposed enhancement. The results of applying ACC to analyze the system for repairing electric motors are then presented.

As an illustration, the application of ACC analysis for testing the system for controlling technological operations in the repair of electric motors (Repair Shop) is described. The system operates according to the following scheme:

- Users of the Maintenance and Repair Shop can be divided into roles: worker, master, workshop supervisor, director, and Maintenance and Repair Shop administrator. Several individuals can belong to each category, except for the director.
- During engine repair, various "operations" are performed. Each operation belongs to one of the "operation groups."
- A broken engine is brought to the company for repairs. The workshop supervisor registers it in the Maintenance and Repair Shop: creates a "card" for it, assigns a unique number to the engine, and determines the list of necessary operations. Subsequently, as the repair progresses, the card will be marked with the completion of each operation.
- The card is placed in the "In Progress" list. The card is visible to all users of all types.
- The workshop supervisor assigns a master responsible for the engine repair.

- The master assigns workers to perform each operation, is responsible for the start and end of the work, makes notes in the engine card about the completion, suspension, and completion of the operations as the repair progresses.
- After the completion of the last operation, the engine card is automatically moved from the "In Progress" list to the "Completed" list.
- The Maintenance and Repair Shop administrator: adds, edits, and deletes users, assigns them a category; adds, edits, and deletes operations and operation groups, assigns an operation to a group; adds, edits, and deletes customers;
- The director has the ability to generate reports that provide information on which worker performed a specific operation, and how much time was spent.

The system model created using ACC analysis significantly differs from traditional models such as functional, structural, flow, and parametric. From an ACC perspective, the system is represented as a matrix, where columns correspond to system attributes, rows to components, and cells contain the capabilities that the system provides to the user. This matrix is constructed as follows.

First, key characteristics of the system are identified, qualities that are important to the user and in which the developed software system should stand out from analogs. In the context of ACC, these are called attributes and are typically expressed as adjectives. Their number is small.

As an illustration, the following list of attributes for the social network Google+ is provided: Social (allows users to exchange information and thoughts), Expressive (users use the product's features for self-expression), Simple (users easily understand how to do what they want), Relevant (shows only information that interests the user), Expandable (integrates with other Google resources, third-party sites, and applications), Confidential (user data will not be disclosed).

For our illustrative Repair Shop system, the following attributes were identified: "Simple" (offers users only intuitive actions), "Convenient" (minimizes time for frequently performed actions), "Accessible" (allows users with different roles to connect), "Secure" (protects information from external threats).

The second step of ACC analysis involves identifying "components." The concept of components in ACC differs from the traditional understanding. Components are the structural units of the system, not in terms of program structure but from the user's perspective. Components are the key parts of code that make the program what it is.

For the social network Google+, components include Profile, People, Feed, Circles, Notifications, Interests, Posts, Comments, Photos. For our illustrative Repair Shop system, components include Search, Repair Card, In Progress, Completed, Reports, Users, Groups, Operations, Customers.

The third stage of ACC analysis involves describing the "capabilities of the system" - actions that the system can perform at the user's request. As expected for actions, they are expressed using verbs. In the ACC model, capabilities do not exist on their own. They are linked to components and attributes. It is considered that each capability is implemented by a certain component with the aim of providing a certain quality of the product (a certain attribute). For example, for the social network Google+, the component "View Page" interacts with the attribute "Accessible" in three capabilities:

- make the document accessible to employees;
- allow employees to edit the document;
- display the employee's position on the page.

This results in a matrix where columns correspond to attributes, rows to components, and capabilities are recorded in the cells. Figure 1 shows the matrix model for Google+ from [16, p.132].

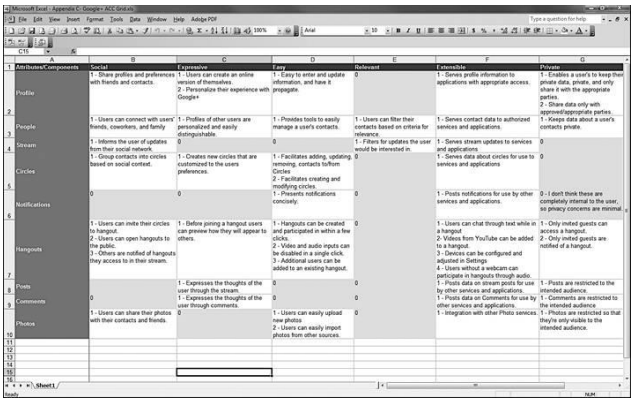

Fig. 1. ACC Table for Google+

Next, a matrix of "attributes-components-capabilities" for our illustrative Repair Shop system is shown in Tables I-II.

TABLE I. ACC TABLE FOR THE ATTRIBUTES SIMPLE AND USER-FRIENDLY FOR REPAIR SHOP

	1/ Simple: Intuitive actions		*2/ Convenient: minimizing operations for frequently performed actions*
A/ Search	**Administrator**: 1. Search for employees by name and email	**Administrator**: 2. Search for a specific order by repair number	**Administrator**: 1. On each tab, you can search (jobs, groups, operations, reports, customers) by key information

	(probability - very rarely, criticality minimal) *Risk 1*	*(probability- very rarely, criticality minimal)* *Risk 1*		*(probability- very rarely, criticality minimal)* *Risk 1*	
B/ Repair Map	**Master**: 1. Adding a description to the order *(probability- very rarely, criticality low)* *Risk 2*	**Master**: 2. Assign workers to order *(probability- often, criticality significant)* *Risk 12*		1. The order automatically moves from the completed list to the finished list *(probability- very rarely, criticality significant)* *Risk 4*	**Master**: 2. Start/finish/suspend repairs *(probability- often, criticality maximum)* *Risk 16*
C/ In progress	**Head of Department:** 1 Review cards of unfinished tasks *(probability- often, criticality maximum)* *Risk 16*			**All**: 1. Display selected number of current tasks *(probability- very rarely, criticality minimal)* *Risk 1*	
D/ Completed	**Head of Department:** 1. Review cards of completed tasks *(probability- often, criticality maximum)* *Risk 16*			**Master**: 1. Change task status from completed to incomplete *(probability- sometimes, criticality maximum)* *Risk 12*	**Director**: 2. Send email notification of task completion *(probability- rarely, criticality minimal)* *Risk 2*
E/ Reports	**Director**: 1. Download report *(probability- rarely, criticality low)* *Risk 4*			**Director**: 1. Check how much time a worker spent on performing the operation *(probability- sometimes, criticality low)* *Risk 6*	**Director**: 2. Review which operations were performed by specific workers during a specified period *(probability- sometimes, criticality low)* *Risk 6*
F/ Users	**Administrator**: 1. Add users *(probability- very rarely, criticality significant)* *Risk 3*	**Administrator**: 2. Delete users *(probability- very rarely, criticality significant)* *Risk 3*	**Administrator**: 3. Edit users *(probability- very rarely, criticality significant)* *Risk 3*	**Administrator**: 1. View selected number of user records *(probability- very rarely, criticality low)* *Risk 2*	**Administrator**: 2. Dismiss users *(probability- very rarely, criticality low)* *Risk 2*
G/ Groups	**Administrator**: 1. Add groups *(probability- very rarely, criticality significant)* *Risk 3*	**Administrator**: 2. Delete groups *(probability- very rarely, criticality significant)* *Risk 3*	**Administrator**: 3. Edit groups *(probability- very rarely, criticality significant)* *Risk 3*	**Administrator**: 1. Adding operations to a group from a pre-formed list *(probability- rarely, criticality significant)* *Risk 6*	
H/ Operations	**Administrator**: 1. Add operations *(probability- very rarely, criticality significant)* *Risk 3*	**Administrator**: 2. Delete operations *(probability- very rarely, criticality significant)* *Risk 3*	**Administrator**: 3. Edit operations *(probability- very rarely, criticality significant)* *Risk 3*		
I/ Customers	**Administrator:** 1. Add customers *(probability- very rarely, criticality minimal)* *Risk 1*	**Administrator**: 2. Delete customers *(probability- very rarely, criticality minimal)* *Risk 1*	**Administrator**: 3. Edit customers *(probability- very rarely, criticality minimal)* *Risk 1*	1. Automatic notifications are automatically sent to mail on email for completed tasks *(probability- rarely, criticality significant)* *Risk 6*	

TABLE II. ACC TABLE FOR THE ATTRIBUTES ACCESSIBLE AND SECURE FOR REPAIR SHOP

	3/ Affordable: allows connecting users with different roles to connect	*4/ Secure: protects information from against various threats*
A/ Search	**All**: 1. Search by job number and customer in current and completed work *(probability - very rarely, criticality minimal)* *Risk 1*	
B/ Repair Map	**All:** 1. View repair completion status *(probability – very rarely, criticality maximum)* *Risk 4*	
C/ In progress	**All**: 1. View all works in active and suspended states (*probability- rarely, criticality maximum)* *Risk 8*	

D/ Completed	**All:** 1. View the list of completed works, their completion date, customer, repair number *(probability - very rarely, criticality minimal) Risk 1*	**Administrator**: 1. Cannot change master and workers after work has started, which helps prevent scheduling conflicts *(probability- very rarely, criticality significant) Risk 3*
E/ Reports		**Director**: 1. Keep reports confidential *(probability - very rarely, criticality maximum) Risk 4*
F/ Users		**Administrator**: 1. Assign roles with limited access rights *(probability- rarely, criticality maximum) Risk 8*
G/ Groups		**Administrator**: 1. Keep operation groups confidential *(probability- very rarely, criticality maximum) Risk 4*
H/ Operations		**Administrator**: 1. Keep operations confidential *(probability - very rarely, criticality maximum) Risk 4*
I/ Customers		**Administrator**: 1. Keep customers confidential *(probability – very rarely, criticality maximum) Risk 4*

There can be many capabilities (tens or hundreds). They provide the result for which the user uses the system. Therefore, the correctness of their implementation should be verified. This means that each capability should be tested at least once.

Already, this matrix is useful as a source of information for building a testing plan "components-attributes-capabilities."

- Each capability requires at least one test. Therefore, the number of capabilities in a table cell indicates the minimum number of tests associated with that cell. It is easy to identify cells, rows, and columns that require maximum testing efforts.
- Each row and each column represent a certain logical integrity. All cells in a row (column) are connected. Therefore, it is logical to test them together. Therefore, each row and each column can serve as a test session assignment. This way, we eliminate duplication and ensure a high level of coverage.

However, the ACC analysis goes further. To increase the informativeness of the model, it involves a risk-oriented approach. This is done as follows.

So far, we have considered all capabilities equal in terms of testing. But in reality, this is not the case. Some capabilities are more significant, while others are less significant. It is necessary to test the more significant capabilities first. (There may not be enough resources to test everything indiscriminately.) The question is: how to determine the significance of capabilities from a testing perspective? ACC proposes to assess the risk of their failure.

Two characteristics are evaluated for each capability: the probability of failures and the degree of criticality of failures. The probability is assessed on a scale of "very rare - rare - sometimes - often." The criticality of failure is assessed on a scale of "minimal (the user may not even notice) - minor - significant - maximum (a blow to the product's reputation; will make the user stop using it)." In both cases, an even number of values is deliberately set on the scales. This is done intentionally to deprive the tester of the opportunity to choose an average option.

After evaluating the probability and criticality of failures for each capability, the risk value (the product of probability and criticality) is calculated and added to the "components-attributes-capabilities" matrix. For better visualization, the matrix is presented as a "heat map,": 1-2 - green risks, 3-4 - yellow, 6-9 - orange, 12-16 - red (5, 10, 11 cannot be).

(When there are multiple possibilities in one cell [16], it recommends averaging their risks. In the opinion of the authors of this article, this recommendation is strange. In our opinion, either the maximum or the sum should be taken.)

The informativeness of the matrix sharply increases. It provides information to answer questions such as:

- How to distribute the resources allocated for testing among different functions and components of the system? Which functions and components should receive more attention, and which less? What should be tested first?
- What is the criterion for completing testing? When do we have the right to say, "We have tested everything"?

Further, the description of the enhancement introduced by the authors in the ACC analysis begins.

III. Our suggestions for improvments of ACC analysis

A fourth dimension - actors, classes of system users - was added to the three dimensions of classical ACC analysis (attributes-components-capabilities).

All users of the Repair Shop system are divided into five classes (playing one of five roles): director, workshop manager, master, administrator, worker. Each role has its needs, its goals in using the Repair Shop system. Each action performed by the system (each capability of the system) is executed upon request of one or several roles. That is, each role has its own set of capabilities.

"Entering the fourth dimension" immediately provided a new perspective on the system, allowing it to be viewed from the user's standpoint. Another basis for grouping capabilities and assessing their risks emerged. What does this provide? Firstly, it is logical to conduct test sessions as the work of a specific role. Secondly, different users may have different qualifications. There should be a correspondence between the user's level of qualification and the level of riskiness of actions performed by them. Performing highly risky actions by low-skilled specialists increases the likelihood of failures. On one hand, there is an opportunity to specify requirements for implementing a specific action of the system (an action intended for low-skilled specialists should have low "riskiness"). On the other hand, "risk assessment" of actions allows determining the "riskiness assessment" of each role. Thus, defining requirements for the level of qualification of users performing that role (highly risky actions should only be performed by highly qualified specialists).

Ideas for further development of the method.

The fourth dimension - it does not necessarily have to be actors. Depending on the project, it can be something else.

We have moved from three-dimensional space to four-dimensional space. The logical next step is multidimensional space. It is possible to introduce consideration of the fifth, sixth, and further dimensions. The limitations here will be associated with the increasing complexity of the model. To combat this complexity, it is logical to use automation.

IV. The example of applying improved ACC method for analyzing the Repair Shop system

As mentioned earlier, a total of 4 attributes, 9 components, and 41 features were identified that intersect attributes and components. The probability and criticality of failures for the features were evaluated, and risk levels were calculated based on these assessments. User roles were assigned to the features that were accessible to them. Most features were accessible to one role, a few to all roles, and one feature had no role assigned as the corresponding action was performed automatically.

The features were sorted in descending order of risk level. There were a total of 5 red-level risks (three at level 16 and two at level 12), 6 orange-level risks (two at level 8 and four at level 6), 17 yellow-level risks, and 13 green-level risks.

The total sum of all risks was 231. This number can be considered as the overall risk level of the entire system.

The features were grouped by components, attributes, and actors. The total weights of the features by groups are presented in tables III, IV and V (The column "Sum after testing" will be explained later).

TABLE III. Component and risk sum table

Component	***Description***	***Sum of risks before testing***	***Sum of risks after testing***
Search	Search strings on different application tabs	4	4
Repair Map	Card with repair information	40	19
Work in progress	List of active and suspended repair cards	25	17
Completed	List of completed repair cards	34	26
Reports	Tab for viewing working hours and order information	20	20
Users	User data administration tab	30	26
Groups	Work group data administration tab	28	28
Operations	Possible engine operation administration tab	22	22
Customers	Customer work data administration tab	28	28

TABLE IV. Attribute and risk sum table

Attribute	***Description***	***Sum of risks before testing***	***Sum of risks after testing***
Simple	Intuitive actions	124	99
Convenient	Minimization of operations for frequently performed actions	66	54
Accessible	Allows connection for users with different roles	14	14
Secure	Protects information from various threats	27	23

TABLE V. Role and risk sum table

Role	***Sum of risks before testing***	***Sum of risks after testing***
Administrator	120	116
Master	42	21
Workshop Manager	32	16
Director	22	22
All	15	15

The discipline of session testing was chosen for testing.

The question arose of how to organize sessions based on what principle. Traditional ACC analysis suggests using rows and columns of a table, i.e. conducting testing "by components" or "by attributes". This is convenient for tracking the completeness of testing (it is sufficient to mark "closed" rows and columns). In our case, this order turned out to be not very convenient. The point is that each user should authenticate when logging into the system. This takes time. In the table, capabilities related to different roles often reside in the same row and column. So, to test one row (one column), it will be necessary to log in and out of the

system several times. To avoid this, it was more convenient to conduct testing "by roles". Although this complicates tracking the completeness of testing (capabilities of one role are scattered in the table in different places). Another argument in favor of testing "by roles" was the simplicity of building test scenarios. When testing "by roles", it's easy to do this (unlike testing "by components" and "by attributes").

At the same time, the question of the order of checking "actors" arose. It is noticed that Table V leads to an incorrect decision. The thought arises that it should be checked based on the reduction of the sum of risks related to the actor. This is incorrect. A large sum can be obtained not because it includes the most significant risks, but because it includes many less significant risks. This is precisely the situation reflected in Table IV. The role of Administrator carries the greatest weight here. However, the Administrator does not have any "red" risks. In terms of "red" risks, the roles of Master and Workshop Manager take the lead. The former has three "red" risks (16 + 12 + 12 = 40), and the latter has two (16 + 16 = 32). However, another factor intervenes in determining the order of "role testing": the order of filling the information base. According to this factor, the role of Administrator was brought to the forefront. Testing the capabilities of all other actors required a filled information base (operations, operation groups, users performing different roles). Therefore, it was decided to first check the Administrator's actions to fill and adjust the information base, and only after that to check the most risky capabilities. Thus, the table "components-attributes-capabilities" was used as the basis for building the testing plan.

Another role played by this table is the basis for building the testing completion criterion. The criterion chosen was the change in the level of system riskiness, i.e. the total sum of all risks. We proceeded from the assumption that as a result of testing, the probability of system failures would decrease. And this means that the magnitude of risks would decrease. (Testing will not be able to affect the criticality of failures.) The criterion for ending testing was chosen as a 15% reduction in system riskiness.

A total of 13 errors were found during testing. As these errors were corrected, the probability of failures was reassessed, and the level of system riskiness was recalculated. The new risk values are shown in Tables III, IV and V in the column "Risk Sums after Testing". After correcting the thirteenth error, the level of system riskiness decreased by 18%. This means that the criteria for ending testing were satisfied.

Table V shows that the most progress was achieved for those roles to which the most risky capabilities were attributed. The sum of risks for the Master decreased from 42 to 21, and for the Workshop Manager from 32 to 16.

In comparison to other testing methodologies such as RUP and IEEE, which focus more on test formatting advice, ACC addresses both the structure and content of the information system. Additionally, using a risk-oriented approach helps in creating efficient tests due to prioritizing capabilities based on their failure probability and criticality, considering the frequent time constraints.

CONCLUSION

The article presents proposals for improving the Activity-Components-Component (ACC) analysis method. In addition to the three dimensions of the traditional ACC analysis - "attributes-components-capabilities," it is proposed to add a fourth dimension - "actors" (roles). This provides a new perspective on the system - a user-oriented view, providing another opportunity for organizing testing.

The application of the enhanced ACC method is demonstrated in organizing the testing of a specific software system - a system for monitoring the technological operations of repairing electric motors. The addition of the "actors" dimension facilitated the optimization of test sessions organization. The main focus was on testing the most risky capabilities. During testing, 13 errors were identified and corrected, leading to an 18% reduction in the overall system risk level.

Further development of the ACC method may involve either replacing the "actors" parameter with another parameter or continuing to increase the number of dimensions, making ACC analysis five-dimensional, six-dimensional, etc. This will make the method more complex and raise questions about its automation.

REFERENCES

[1] Kulakov K. A., Dimitrov V. M. Fundamentals of software testing //Electronic textbook for students of the Institute of Mathematics and Information Technologies. /Petrozavodsk: PetrSU. - 2018.

[2] Safiulin R.Z. Development of testing technologies in education // Education management: theory and practice. 2015. №1 (17). URL: https://cyberleninka.ru/article/n/razvitie-tehnologiy-testirovaniya-v-obrazovanii (date of reference: 15.11.2023).

[3] Karpunin Aleksey Aleksandrovich, Ganev Yuri Mikhailovich, Chernov Maxim Mikhailovich Quality assurance methods in the design of complex software systems // NIKSS. 2015. №2 (10). URL: https://cyberleninka.ru/article/n/metody-obespecheniya-kachestva-pri-proektirovanii-slozhnyh-programmnyh-sistem (date of reference: 15.11.2023).

[4] Galimova, E. Yu. Methodology for selecting automated, manual and mixed way of testing a software product based on quality criteria // Proceedings of Tula State University. Technical Sciences. - 2019. - №. 7. - C. 248-256.

[5] Kulikov S. S. et al. Software testing: textbook. - 2019.

[6] Polevshchikov, I. S., Chirkov, M. S., Levanov, A. A. V. Automated system of test-plans development in software testing // Engineering Gazette of Don. - 2019. - №. 8 (59). - C. 29.

[7] Piven A. A., Skorin Yu. I. Software testing // Sistemy obrokobki informatsii. - 2012. - №. 4 (1). - C. 56-58.

[8] Kuvshinova E. A., Glazova V. F. TESTING AS IMPORTANT COMPONENT OF THE SYSTEM OF CONTROL OF SOFTWARE QUALITY // Applied Mathematics and Informatics: Modern Research in Natural and Technical Sciences. - 2017. - C. 305-308.

[9] Drobysh A. A., Santsevich S. N. Debugging and testing of software. - 2020.

[10] Shakirova A. I., Khasyanov A. F., Dautov E. F. Software testing time reduction // Modern Science-Intensive Technologies. - 2019. - №. 7. - C. 104-109.

[11] Vildanova K. I. Choice of software testing method // Scientific Notes of UlSU. Series" Mathematics and Information Technologies". - 2022. - №. 2. - C. 31-37.

[12] Moiseev D. A. Methodology and process of manual testing // Reliability and quality of complex systems. - 2017. - №. 3 (19). - C. 107-112.

[13] Aksenov D. O., Khafizov E. U., Ryabov M. A. Software Testing Management System.

[14] Ivanov E. C. Development of software testing methodology: master's thesis. - 2014.

[15] Viktorova V. S., Stepanyants A. S. Models and methods for calculating the reliability of technical systems //M.: LENAND. - 2016.

[16] Carollo J., Whittaker J., Arbon J. How testing is done at Google. - 2012.

[17] Plaksin M.A. Testing and Debugging Programs for Future and Present Professionals // Moscow: BINOM, 2023.