Bulletin of the *Transilvania* University of Braşov • Vol 12(61), No. 2 - 2019
Series III: Mathematics, Informatics, Physics, 457-468
https://doi.org/10.31926/but.mif.2019.12.61.2.23

DECISION SUPPORT SYSTEM CHOOSING SOFTWARE TESTING STRATEGY

Vladimir CHERNOV [1], Liudmyla DOROKHOVA [2], Oleksandr DOROKHOV[3], and Galina EGOROVSKAYA [4]

Abstract

The problem of choosing the optimal software testing strategy is considered. The classification of testing approaches is given. The factors influencing the choice of type of testing are investigated. It is shown how these factors can be used to make decisions. Algorithms for choosing the type of testing are developed. The proposed and developed decision support system is implemented in practice.

2000 *Mathematics Subject Classification:* 68U35, 91B06, 94C12
Key words: quality indicator, quality model, software testing, factors for choosing test approaches, types and methods of testing, release frequency, software defect, decision support system.

1 Introduction

The rapid increase in the complexity and size of modern software packages while increasing the responsibility of the functions performed has dramatically increased the requirements on the part of customers and users for their quality and safety of use. A tested means of ensuring high efficiency and quality of functioning of programs and software systems is the process of testing them.

Testing is an expensive but necessary process. Many types and methods of testing in practice cannot be used due to the limitations of time and money, financial and human resources. To improve the organization of the testing process, it is necessary to choose the most appropriate methods for this particular project.

The aim of this work is to develop an algorithm for choosing testing approaches to determine the testing strategy for each specific software product in accordance with the specified indicators of software quality.

[1]Vladimir State University, Russia, e-mail: chernov.vladimir44@gmail.com
[2]National Pharmaceutical University, Ukraine, e-mail: liudmyladorokhova@gmail.com
[3]Kharkiv National University of Economics, Ukraine, e-mail: aleks.dorokhov@meta.ua
[4]Vladimir State University, Russia, e-mail: chernov.vladimir44@gmail.com

2 Main software product quality criteria

The quality of software is the entire range of signs and characteristics of software products, which refers to its ability to satisfy established or anticipated needs [1]. According to the main literature sources, software quality criteria are a set of properties (attributes) of software products according to which all quality is evaluated and described [12, 13]. A quality model is a certain set of characteristics and the relationships between them that provide the basis for determining quality requirements and assessing quality.

The product quality model reduces the quality properties of a software product to eight characteristics, which are: functional suitability; level of performance; ease of use; compatibility; security; reliability; portability (mobility) and accompaniment. In turn, each characteristic consists of a series of corresponding sub characteristics (Figure 1).

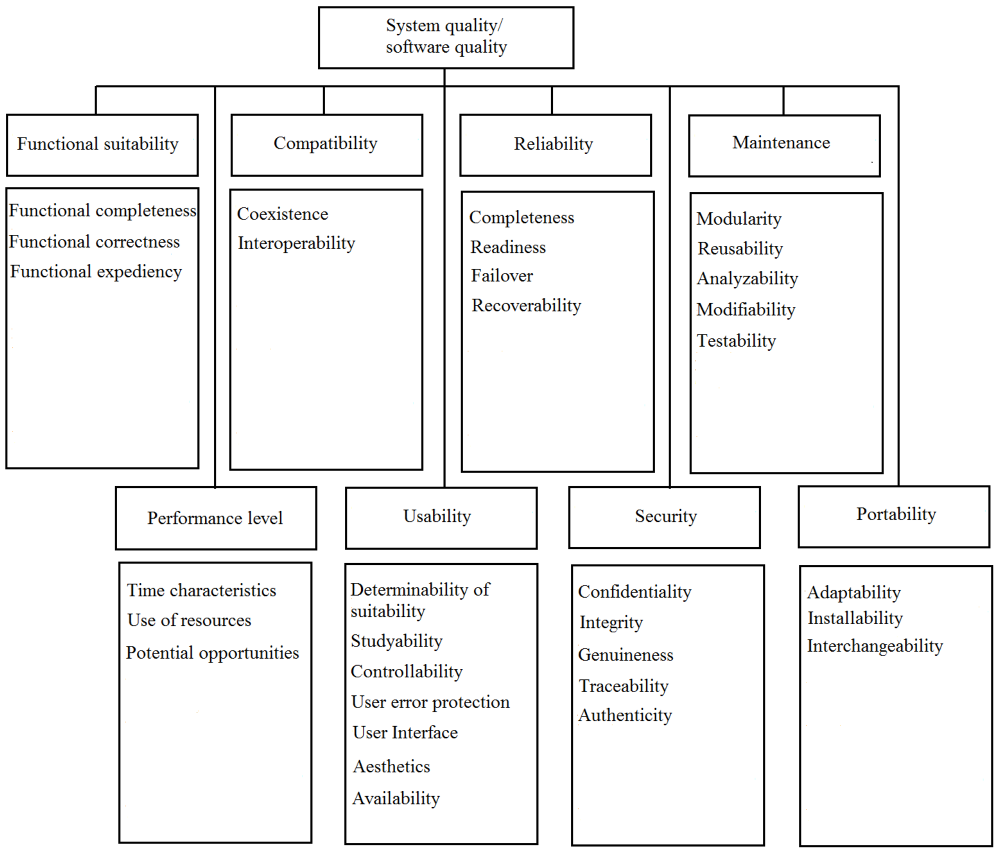


Figure 1. Software product quality model

Some of these quality criteria are set when writing a technical task, some are obvious, some can be determined by the parties in the process of developing a software product, but in many cases functionality and reliability are mandatory quality criteria for a software product, and ensuring reliability is a red thread through all stages and development processes of a software product [2].

The remaining criteria are used in accordance with the requirements for the software product depending on the needs of users. The quality indicators discussed in this section can be measured or evaluated by testing.

3 Testing and development of an algorithm for choosing test processes, methods and approaches

Testing is a way to assess the quality of software in terms of defects found, both for functional and non-functional requirements and characteristics of the software.

Testing a software product allows you to ensure that software projects meet specified quality parameters throughout the entire software life cycle.

The main purpose of testing is to determine deviations in the implementation of functional requirements, to detect errors in the execution of programs and to correct them as early as possible in the course of the project, that is, to reduce the development cost by early detection of defects.

There are many types and methods of testing [6, 9, 10], and its classification is possible by:

- access to the code and application architecture (white, black, and gray box methods);
- execution code execution (static and dynamic);
- degree of automation (manual and automated);
- the level of detail of the application (modular, integration and system);
- techniques and approaches (positive and negative testing);
- the importance of the tested functions (smoke, critical path testing, advanced);
- attracting end-users (alpha, beta, gamma testing);
- goals and objectives (positive and negative testing, functional and non-functional, testing of reliability, recoverability, fault tolerance, performance, interface, security, compatibility, load, stress, usability testing and regression testing), etc.

But in practice, all types of testing cannot be used due to time and cost limitations.

To optimize the testing process, it is necessary to draw up a testing plan in which you need to choose the most appropriate methods for this particular project [8, 14].

To solve this problem, it is first necessary to formulate significant factors on the basis of which conclusions can be drawn about which methods to use.

As a result of the work done, the following factors are summarized and grouped: release frequency, nature of the system, the criticality of defects, the complexity of the system and the quality indicators.

The first such factor is the frequency of releases.

Release of a software product version - a fixed state of changes in a software product related to the correction of identified errors in the functioning of software (comments) or the implementation of additional requirements of the Customer (proposals) that do not lead to a change in the ideology of software product development for its specific version.

The second factor is the nature of the system.

Online systems and single-user applications need to be tested differently.

In the first case, special attention should be paid to load testing and around-the-clock system availability.

In the second case, first of all, it is necessary to test the performance of the application and its stability.

Therefore, systems of a different nature need to be tested with different approaches.

The third factor considered is the criticality of defects.

A defect is a software bug that leads to failure. The criticality of defects can be of three levels (high, medium, low).

If the presence of defects in the system is critical, then it is necessary to devote more time to testing, take into account more details, carry out double control, test each task and test the functionality, reliability and fault tolerance of the system (based on RSTQB materials and tester experience).

If defects are not critical, then it is enough to conduct testing on tasks and at the end of the project regression testing.

Another factor is the software development life cycle.

The place of testing is different in different models of software development (waterfall, v-shaped, iterative, spiral, flexible model).

The next factor is risk level and type.

The risk level is determined by the probability of an adverse event and its impact.

Risks are used to determine where to start testing and what aspects to pay more attention to. Testing is used to reduce the risk of adverse effects or their consequences.

Another factor is quality indicator.

Depending on the quality criterion that is checked (reliability, efficiency, performance, etc.), it is necessary to choose different testing methods.

The next factor is the complexity of the program.

The complexity of the program is determined by the following characteristics: the number of lines of the code and the mathematical complexity.

If the program performs many functions and contains many modules, then it is best to divide it into task blocks and apply the testing approach to tasks.

Also, the criteria that determine the testing approach are regulatory standards, customer or contract requirements, test objectives, accessible documentation, testers' knowledge, time and budget, and previous experience about the types of defects found.

Using the studied factors, we can develop an algorithm for choosing approaches to testing. Depending on the frequency of releases (every week, once a month or once a year), one should choose different approaches and devote a different amount of time for testing.

If releases are held once a week, then there is no way to conduct long and frequent testing.

Therefore, based on the many years of experience of testers, it must be carried out every 1-3 days.

And after it, regression testing a day before the release.

With releases once a month, you can test every 3-7 days before release.

And also there is time to test each task.

If releases are held once a year, then regression testing should be carried out every month, as well as testing of functionality and testing of tasks a month before release.

Figure 2 shows an algorithm for choosing an approach to testing depending on the frequency of releases.

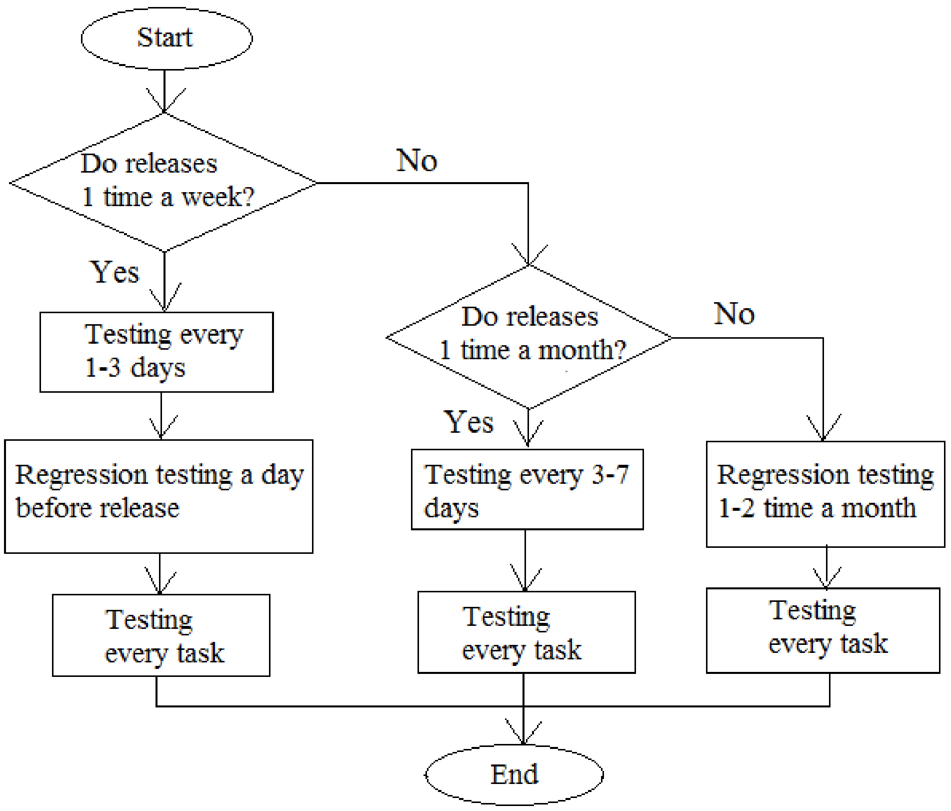

Start
Do releases 1 time a week?
No
Yes
Testing every 1-3 days
Regression testing a day before release
Testing every task
Do releases 1 time a month?
No
Yes
Testing every 3-7 days
Testing every task
Regression testing 1-2 time a month
Testing every task
End

Figure 2. Algorithm for choosing approaches to testing depending on the frequency of releases

If we are dealing with a mobile application, then it is necessary to conduct testing on an emulator and regression testing on a real device.

In the case of a web application, you need to test the load, functionality, and usability.

If this is a local service, then the following approaches should be applied: testing of functionality, testing of fault tolerance and load.

In the case of a web service, you must also test the interaction interface.

If we are dealing with a local application, then it is necessary to test the functionality of the algorithms and test the code.

If this is a multi-user LAN application, then you need to test the load, reliability, and functionality.

Figure 3 shows the algorithm for choosing an approach to testing depending on the nature of the system.

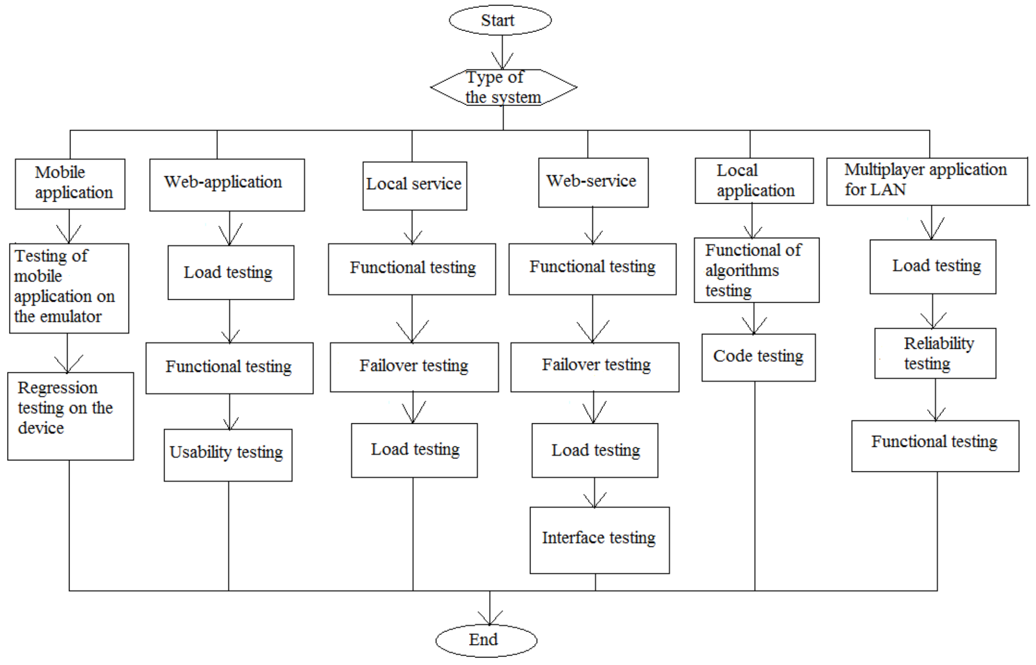


Figure 3. Algorithm for choosing approaches to testing depending on the type of system

The algorithm for choosing testing methods, depending on the quality indicators important for the system being developed, is as follows:

If performance is being tested, then load testing, scalability testing, volumetric, stressful, competitive testing, and resource use testing should be performed.

When checking compatibility, you need to conduct configuration and cross-browser testing. If usability is being tested, then usability, accessibility, interface, internationalization, and localization testing need to be done.

In the case of reliability testing, fault tolerance, recoverability, and reliability must be verified.

System security can be verified through security testing. In portability testing, adaptability, installability and interchangeability testing must be performed.

After you have tested all the necessary quality indicators, you need to test the functionality of the system (Figure 4).

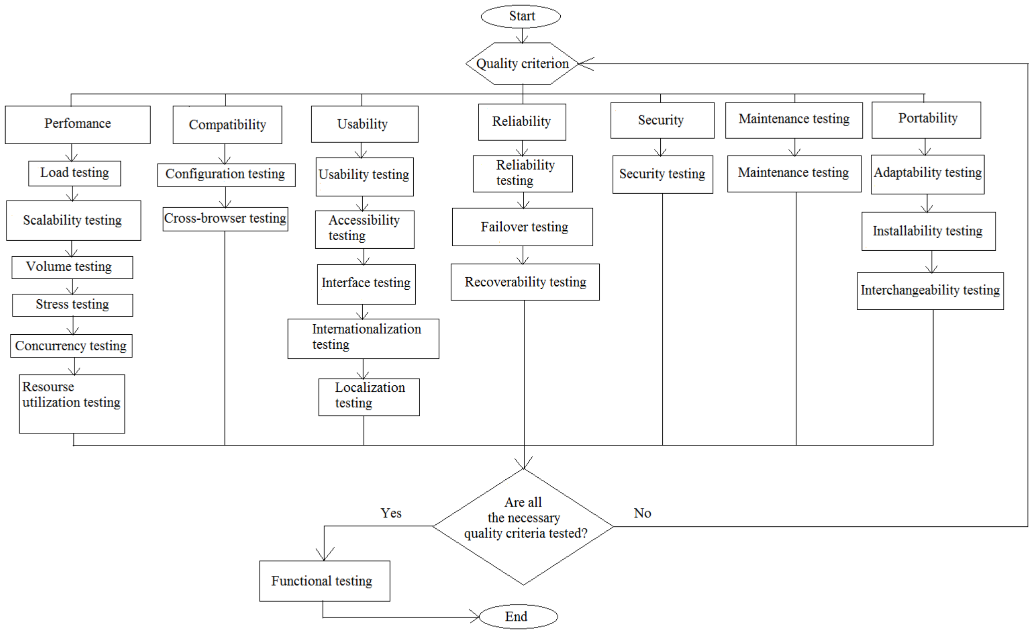


Figure 4. Algorithm for choosing approaches to testing depending on the quality indicator being tested

Based on the foregoing, it can be concluded that it is advisable to develop a decision support system for choosing test methods for a particular software project.

4 The development of the corresponding applied decision support system

Based on the proposed approaches and correspondingly developed algorithms, based on fuzzy logic and multicriterial decision making approaches [3, 4, 5, 7, 11], an applied decision support system was created to draw up an optimal strategy for testing software products.

The goal of creating this decision support system is to automate the construction of a software testing strategy.

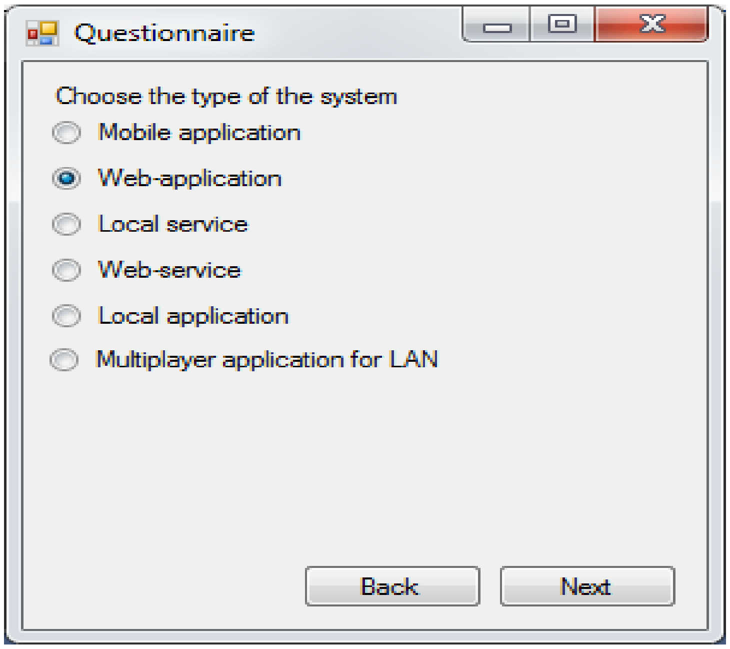
System tasks include: optimize the testing process; reduce the time and financial cost of testing; determine a software testing strategy based on specified quality indicators.

The application under development is a local single-user. To work with it, a database was implemented, in which recommendations for testing are stored depending on the values of indicators chosen by the user.

Special software was developed in the form of appropriate algorithms. UML diagrams have been developed to design the application architecture, namely the use case diagram, activity diagram, class diagram, and deployment diagram.

The system is written in the C Sharp programming language. For its development, the following were used: VisualStudio development environment, SQLServer database management system, CrystalReports report generator for creating reports and printed forms.

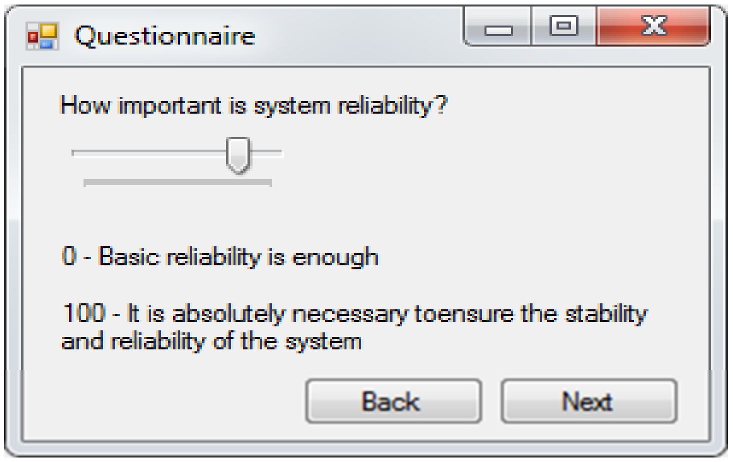
Figures 5 and 6 show some steps of working with the program (fragments). At each step, the user selects a criterion value. The corresponding result is stored in the database.


Questionnaire

Choose the type of the system

- Mobile application
- [✓ Web-application]
- Local service
- Web-service
- Local application
- Multiplayer application for LAN

Figure 5. The choice of system character



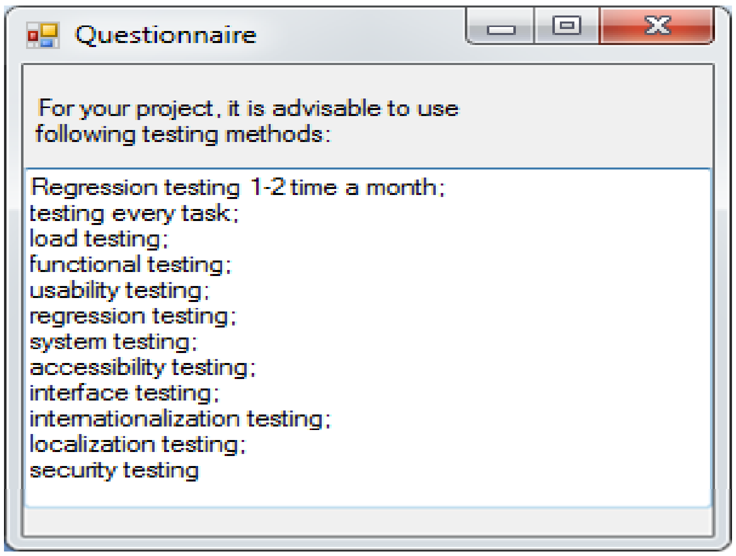
How important is system reliability?

0 - Basic reliability is enough

100 - It is absolutely necessary toensure the stability and reliability of the system

Figure 6. Assessment of the degree of importance of system reliability

After the decision-maker has gone through all the steps, the application will generate and output the appropriate testing recommendations from the database to the TextBox component (Figure 7).



For your project, it is advisable to use following testing methods:

Regression testing 1-2 time a month;
testing every task;
load testing;
functional testing;
usability testing;
regression testing;
system testing;
accessibility testing;
interface testing;
internationalization testing;
localization testing;
security testing

Figure 7. The result of the program

5 Conclusions

Modern testing methods are studied, models and quality indicators of software are considered and factors influencing the choice of testing methods for each specific software product are identified.

Algorithms are developed for choosing a testing strategy based on specified quality indicators.

A decision support system has been developed that allows us to find the best option for the testing process, taking into account specified quality indicators, and reducing the time and financial cost of testing.

References

[1] Bajaj, A., Sangwan, O., *A Systematic Literature Review of Test Case Prioritization Using Genetic Algorithms*, IEEE Access **7** (2019), 355-375.

[2] Chen, T., Thomas, S., Hemmati, H., Nagappan, M., Hassan, A., *An Empirical Study on the Effect of Testing on Code Quality Using Topic Models: A Case Study on Software Development Systems*, IEEE Transactions on Reliability **66** (2017), no. 3, 806-824.

[3] Chernov, V., Dorokhov, O., Malyaretz, L., *Construction of Estimates in the Choice of Alternative Solutions by Using the Fuzzy Utilities*, Transport and Telecommunication **13** (2012), no. 1, 11-17.

[4] Dorokhov, O., Chernov, V., Dorokhova, L., Streimkis, J., *Multi-Criteria Choice of Alternatives Under Fuzzy Information*, Transformations in Business and Economics **17(2)** (2018), 95-106.

[5] Dorokhov, O., Dorokhova, L., *Fuzzy Model in Fuzzy-Tech Environment for the Evaluation of Transportation's Quality for Cargo Enterprises in Ukraine*, Transport and Telecommunication **12(1)** (2011), 25-33.

[6] Huang, Y., Bhunia, S., Mishra, P., *Scalable Test Generation for Trojan Detection Using Side Channel Analysis*, IEEE Transactions on Information Forensics and Security **13(11)** (2018), 2746-2760.

[7] Hui Liu, H., Yan, F., Jiang, J., Song, J., *Energy Consumption Fuzzy Estimation for Object-Oriented Code*, IEEE Access **6** (2018), 62664-62674.

[8] Justo, J., Araujo, N., Garcia, A., *Software Reuse and Continuous Software Development: A Systematic Mapping Study*, IEEE Latin America Transactions **16(5)** (2018), 1539-1546.

[9] Marenbach, R., Albert, M., *Regression Test Approach for Testing of Protection Ieds to Improve Field Testing Quality and Support Knowledge Management*, The Journal of Engineering **15** (2018), 1023-1026.

[10] Marinho, E., Resende, R., *PLATEM: a Method for Mobile Applications Testing*, IET Software **11(6)** (2017), 319-328.

[11] Pietrantuono, R., Potena, P., Pecchia, A., Rodriguez, D., Russo, S., Fernndez, L., *Multiobjective Testing Resource Allocation Under Uncertainty*, IEEE Transactions on Evolutionary Computation **22(3)** (2018), 347-362.

[12] Ramasubbu, N., Kemerer, Ch., *Integrating Technical Debt Management and Software Quality Management Processes: A Normative Framework and Field Tests*, IEEE Transactions on Software Engineering **45(3)** (2019), 285-300.

[13] Tao, Ch., Gao, J., Wang, T., *Testing and Quality Validation for AI SoftwarePerspectives, Issues, and Practices*, IEEE Access **7** (2019), 164-175.

[14] Vizarreta, P., Trivedi, K., Helvik, B., Heegaard, P., Blenk, A., Kellerer, W., Machuca, C., *Assessing the Maturity of SDN Controllers With Software Reliability Growth Models*, IEEE Transactions on Network and Service Management **15(3)** (2018), 1090-1104.