UDC 004.9: 004.05
https://doi.org/10.31891/csit-2023-1-5

Iryna ZASORNOVA, Tetiana HOVORUSHCHENKO, Oleg VOICHUR
Khmelnytskyi National University

STUDY OF SOFTWARE TESTING TOOLS ACCORDING TO THE TESTING LEVELS

Recently, software has been intensively used in almost all areas of business. Testing is an integral process of the software life cycle, during which it is proved that the software meets the specified requirements and needs of the customer, thereby ensuring the quality of the software. The article analyses the tools for software testing with their generalisation by levels of testing.

The study has shown that there are a number of studies aimed at reviewing and classifying software testing tools. The correct choice of software testing tools is one of the vital elements to ensure the quality of the entire project. However, most studies in the field of testing focus on describing testing methods without directly connecting to the tools that are based on these methods.

A specialist's approach to software testing requires additional information about the testing tools currently available. With the increasing complexity of software products and shorter development cycles, it is clear that manual testing cannot deliver the level of quality required by the market. Choosing the wrong testing tools for a project leads to inadequate quality measurements or tool changes during the project. Both wrong choice and change of testing tools during the development process affect the quality of the software product and, as a result, the success of the project as a whole.

The classifiers discussed in this paper can be used to select software testing tools appropriately. On the one hand, it can be useful for navigating a wide range of testing subjects, reducing the time required for specialists to find the right solution. On the other hand, it can be used as a short introduction to the rapidly developing field of testing and available testing tools for those who are not experts in this field. The classification can be applied to testing various software projects, depending on the type of software and development methodology.

Keywords: software, software testing, manual software testing, automated software testing, levels of software testing.

Ірина ЗАСОРНОВА, Тетяна ГОВОРУЩЕНКО, Олег ВОЙЧУР
Хмельницький національний університет

АНАЛІЗ ІНСТРУМЕНТІВ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВІДПОВІДНО ДО РІВНІВ ТЕСТУВАННЯ

Останнім часом програмне забезпечення (ПЗ) інтенсивно використовується майже в усіх галузях підприємництва. Тестування є невід'ємним процесом життєвого циклу програмного забезпечення, під час якого доводиться, власне, відповідність ПЗ заданим вимогам і потребам замовника, тим самим забезпечується якість ПЗ. В статті проведено аналіз інструментів для тестування ПЗ з узагальненням їх по рівнях тестування.

Дослідження показало, що існує ряд досліджень, спрямованих на огляд і класифікацію інструментів тестування ПЗ. Коректний вибір інструментів для тестування ПЗ є одним із життєво важливих елементів для забезпечення якості усього проєкту. Проте більшість робіт у галузі тестування зосереджені на описі методів тестування без прямого підключення до інструментів, які базуються на цих методах.

Підхід фахівця до тестування ПЗ вимагає додаткової інформації про доступні на даний момент інструменти тестування. Із зростаючою складністю програмних продуктів та коротшими циклами розробки стає очевидним, що ручне тестування не може забезпечити рівень якості, необхідний для ринку. Неправильний вибір інструментів тестування для проєкту призводить до неадекватних вимірювань якості або заміни інструментів під час проєкту. Як неправильний вибір, так і зміна інструментів тестування в процесі розробки впливають на якість програмного продукту і, як наслідок, на успіх проєкту в цілому. Класифікатори, які розглянуто у роботі, можна використовувати для відповідного вибору інструментів тестування ПЗ. З одного боку, це може бути корисним для орієнтації в широкому предметному полі тестування, скорочуючи час, необхідний спеціалістам для пошуку вірного рішення. З іншого боку, його можна використати як короткий вступ до галузі тестування, що швидко розвивається, і доступних інструментів тестування для тих, хто не є експертом у цій галузі. Проведена класифікація може бути застосована для тестування різноманітних програмних проєктів, залежно від виду ПЗ та методології розробки.

Ключові слова: програмне забезпечення (ПЗ), тестування ПЗ, ручне тестування ПЗ, автоматизоване тестування ПЗ, рівні тестування ПЗ.

Introduction

Recently, software has been intensively used in almost all branches of business [1]. Testing is an integral process of the software life cycle, during which the compliance of the software with the specified requirements and needs of the customer is proven, thereby ensuring the quality of the software [2, 3].

Software testing can be manual or automated. In manual testing, testers perform tests manually without using any means of automation. Manual testing is a low-level and simple type of testing that does not require a lot of additional knowledge. However, before you can automate the testing of any application, you must first run a series of manual tests. Manual testing requires more effort, but without it, we cannot be sure whether automation is possible at all. One of the fundamental principles of testing is that 100% automation is impossible. Therefore, manual testing is a necessity [4].

Automated software testing is part of the testing process at the quality control stage in the software development process; it is a type of testing in which testing is performed using various automation tools and scripts. It uses software tools to execute tests and verify execution results, which helps reduce testing time and simplify the testing process.

Automated testing involves the use of special software to control the execution of tests and compare the expected and actual results of the program. This type of testing helps to automate activities that are often repeated, but which, at the same time, are necessary for maximum test coverage of the task. To compile automated tests, a QA specialist must be able to program. Automated tests are full-fledged programs that are simply designed for testing [4].

There are several main types of automated testing: Code-driven testing automation – testing at the level of software modules, classes and libraries (actually, automatic unit tests); graphical user interface testing automation (Graphical user interface testing) – a special program (testing automation framework) allows you to generate user actions – button presses, mouse clicks, and monitor the program's reaction to these actions – whether it meets the specification; automation of API (Application Programming Interface) testing – the software interface of the program, which is used to test interfaces intended for interaction, for example, with other programs or with the user. Here, again, as a rule, special frameworks are used [4].

The advantages of automated testing include:

– speed of execution of test cases is greater than with manual testing;

– lack of influence of the human factor in the process of execution of test cases;

– cost minimization during repeated execution of test cases (i.e. minimal human participation);

– the ability to perform such test cases that cannot be performed manually;

– the ability to collect, store, analyze, aggregate and present large volumes of data in a form convenient for human perception;

– the ability to perform low-level actions with the application, operating system, transmission channels, etc.

Disadvantages of automated tests include:

– the need for highly qualified personnel;

– high costs for complex automation tools, development and maintenance of test case code;

– automation requires more careful planning and risk management;

– complex selection of automation tools, the need to train staff (or find specialists);

– existing test cases may be unusable and outdated in the case of significant changes in requirements, changes in the technological domain, redesign of interfaces (both user and software), etc.

Currently, much attention is paid to the transition to fully automated software testing, as it provides better results, especially for large and super-large software projects, saves time and budget, allows to reduce the routine manual actions of the tester, and reduces the complexity of software testing [5]. Therefore, there is a rapid development of testing tools and methods. Both manual and automated testing can be used at different levels of testing, as well as be part of other types and types of testing.

Therefore, increasing the use of automated software testing is *an actual task* today.

The purpose of this study is to analyze software testing tools according to the level of testing.

Study of Software Testing Tools According to the Testing Levels

Software testing is usually performed by a team of testers or developers in accordance with set requirements (specifications). Depending on the type of software product, the testing method and appropriate software are chosen.

By level, testing is divided into Unit (first level), Integration (second level), and System (third level) [6]. However, acceptance testing (fourth level) is often added to this list [7]. In order to carry out quality testing, it is necessary to make a reasonable choice of testing tools. Using the classification, it is possible to choose the necessary testing tools.

Software testing tools are mostly designed to support one or more methods of testing a software product and can perform different tasks depending on the level of testing.

The conducted research showed that there is a sufficient number of testing tools designed for unit, integration, system, and acceptance testing.

Unit testing can be considered a basic level of testing with the ability to test several modules at the same time, which makes it possible to automate them. The free and open-source software xUnit is for the first level of testing like *Unit testing*, and depends on the programming language. For example, JUnit is designed for the Java programming language, NUnit for .NET, CppUnit, CUnit for C, C++, etc. [8]. In most cases, unit testing is performed by software product developers using automated tests.

The smallest number of tools was found for *Integration testing*. Testing tools for Unit testing (xUnit) are also used at the next (second) level – during Integration testing. This is related to the method of conducting Integration testing. Also using Rational Rose and Cantata++.

It is known that the behaviour of a software product changes to one degree or another after adding to it a new module that was tested at the first level of testing. Therefore, as part of Integration testing, *Regression testing* is carried out in order to check the functionality of the assembled parts (build) of the software product. Such tests are recommended to be performed using automation tools (Selenium, SilkTest, Rational Functional Tester and QEngine, etc.) since their set is constantly increasing during the development of a software product and manual testing is impractical.

System testing includes a set of the following tests: Recovery, Security, Stress and Performance testing [9]. A large number of tools have been identified for System testing, as this level of testing includes several more testing methods.

Acceptance testing is carried out at the last level before deployment. At this stage, a conclusion is made about the acceptance or rejection of the software product. At the same time, developers can be involved only to correct code errors, if any are found. As the project grows, the number of Acceptance testing will also increase. Therefore, for Acceptance testing, it is recommended to use automated testing tools, for example, FitNesse. FitNesse allows the customer of the software product to enter specially formatted input. Input data is interpreted and tests are created automatically. These tests are then run by the system and the output is returned to the customer. The advantage of this approach is fast feedback. The customer can write tests in the form of HTML tables and add, if necessary, additional text. The tool then analyzes the tables, runs the tests, and provides the results in an HTML document. FitNesse supports Java, but versions for C++, C#, Python, Ruby, Delphi, etc. are currently available.

In addition, it is worth noting that acceptance testing is not sufficiently automated today, as there is a lack of tools for this testing method. Therefore, the use of tools for system testing and acceptance testing is quite limited.

At the same time, it should be taken into account that the software for conducting automated testing is constantly improved, supplemented and expanded. Regarding the future solution, the tools presented in the study can be applied at different levels of testing, providing a practical demonstration of their use. Another possible improvement is to conduct a comparison between testing tools that belong to a similar group to present the advantages and disadvantages of a particular tool [10–11].

Consider the following types of testing. Thus, API testing checks the correctness of interaction between system components, UI testing tests the correctness of the graphical interface, and network protocol level testing checks the correctness of data transmission between computers. All these types of testing are important to ensure the quality of the software and its correct operation.

UI testing, i.e. the user interface, can be performed using the following tools: Abbot Java GUI Test Framework, Business Process Testing (BPT), QF-Test, cPAMIE, Jemmy, Quick Test Professional, Rational Functional Tester, Rational Robot, Selenium, SilkTest, TestComplete, TestPartner, TOSCA, Oracle Functional Testing.

The following tools are used for *API testing*: APL Sanity Autotest, Cantata++, CppTest, CppUnit, CUnit, DTM Data Generator, FitNesse, JProbe, JUnit, NUnit, JVerify, TestNG, VectorCAST/C++.

To *test the network protocol level* – in this case, the tool simulates the client part of the system that interacts with the object through network protocols. Tools: Conkormiq Qtronic, DTM DB Stress, HttpUnit, JMeter, LoadRunner, MessageMagic, NeoLoed, Nikto, OpenSTA, OpenTTCN Tester, Oracle Load Testing, QALoad, Rational App Scan, Rational Performance Tester, SilkPerformer, soapUI, The Grinder, WAPT, Weblnspect.

The listed tools have certain *disadvantages*. For example, although *Selenium* software is one of the most popular tools for automated website testing, it has some drawbacks:

– resistance to change: if the website under test undergoes changes in its structure or code, test scripts written using Selenium may become unusable, requiring the rewriting of tests;

– complexity of configuration: to use Selenium, you need to have basic knowledge of programming and environment configuration;

– limitations in operating systems: Selenium can only work on operating systems that support the browsers it works with;

– slow test execution speed: Selenium can sometimes lag during test execution, which can lead to a longer testing process;

– the need to support scripts: in order for Selenium to be able to work with some elements of the website, such as AJAX, it is necessary to have scripts that can be executed in the browser;

– limitation of liability: Selenium is not able to guarantee responsibility for the completeness of testing, so its use should be supplemented by manual testing and other software quality control methods.

Overall, although Selenium has its flaws, it is a pretty effective tool for automated website testing and performance testing.

Let's consider the advantages and disadvantages of the *Business Process Testing (BPT*) tool. Because of the complexity of testing business processes and the many applications involved, using code-based test automation is problematic. Coded test automation takes time to develop and test. With BPT and multiple scenario testing, the time spent building automated coded testing makes it slow and a significant barrier for organizations. Testing multiple applications requires expertise and knowledge of each application. Coders who develop automation do not have deep applied knowledge. With BPT and multiple applications, this increases testing slowdowns. Advantages of BPT: eliminates the need to create a separate automation system; automated testing becomes structured using business components; reduces the effort required to write and maintain test automation scripts BPT is independent of the detailed test script; high reusability with data-driven components: testers do not need technical knowledge in automation. Disadvantages of BPT: an additional license for the BPT Framework must be purchased for test scripts; The BPT Framework can only be used if you have access to Application Lifecycle Management (ALM).

QF-Test from *Quality First Software* is a cross-platform graphical user interface test automation software tool specializing in Java/Swing, SWT, Eclipse and RCP plug-ins, Java applets, Java Web Start, ULC and cross-browser static and dynamic test automation web applications. A sophisticated recognition mechanism provides extreme serviceability and low maintenance costs, which is the most important factor in software testing automation.

The results of the analysis of tools for software testing with their generalization by testing levels are presented in Table 1.

Table 1

Comparative analysis of tools for automated software testing

№	Tools	Criteria					
		stable to change	**easy to set up**	**supports various OS**	**test performance speed**	**script support**	**URL**
				UI testing			
1	Abbot Java GUI Test Framework	stable	difficult	works with all platforms that support Java	low	no	http://abbot.sourceforge.net/
2	Business Process Testing (BPT)	unstable if business processes change regularly	complex in case when many business processes need to be configured	may be limited in the OS on which its components can be used	low	may require support for scripts used to configure tests and automate their execution	http://www.hp.com/
3	QF-Test	stable, but if the PP changes frequently, the tests may require frequent modification	difficult, but if the PP changes frequently, the tests may require frequent modification	supports, but some functions may be limited	low	yes	http://www.qfs.de/en/qftest/index.html
4	cPAMIE	stable if the page does not contain dynamic content	medium difficulty	Windows, Linux	very low	can support JavaScript	http://pamie.sourceforge.net/
5	Jemmy	very stable	medium difficulty, requires some additional components and libraries NetBeans RCP	Windows, Linux, Mac OS X	low	supports based on Java	https://jemmy.dev.java.net/
6	Quick Test Professional	stable, but with frequent requires checking and correction	difficult	Windows	low	scripting support is required	http://www.hp.com/
7	Rational Functional Tester	stable	easy, but you need to have experience in test automation	supports	high, but depends on the volume of test scenarios and the complexity of the application	uses Java scripts	http://www.ibm.com/
8	Rational Robot	stable in the	easy	Windows	high	scripting	http://www.ibm.com/

		absence of significant changes in the user interface				support is required	
9	Selenium	stable	easy	supports	high, but depends on the size of the web page, the number of elements	yes	http://seleniumhq.org/
10	SilkTest	unstable	easy	supports	high, but depends on the complexity of software and the number of tests	yes	http://www.borland.com/
11	TestComplete	very stable	easy	supports	high	yes	http://automatedqa.com/products/testcomplete/
12	TestPartner	stable	easy	Windows	high	yes	http://www.microfocus.com/
13	TOSCA	stable	easy	supports	high	yes	http://www.tricentis/com/
14	Oracle Functional Testing	very stable	easy	supports	high	yes	http://www.oracle.com/
15	Canoo WebTest	stable if the structure of the web pages or web application does not change dramatically	easy	Windows, Linux, Mac OS X	high, but depends on the size of the test suites and the complexity of the web application	yes	http://www.testingtoolsguide.net/tools/canoo-webtest/
16	QEngine	stable	easy	Windows, Linux, Mac OS X	high, but depends on the size of the test sets and the complexity of the software	yes	http://www.manageengine.com/products/qengine/index.html
				API testing			
17	APL Sanity Autotest	stable	easy	Windows, Linux, Mac OS X	high	yes	http://ispras.linux-foundaton.org/index.php/API_Sanity_Autotest
18	Cantata++	stable if additional functions do not require significant changes to the source code	easy	Windows, Linux, Mac OS X	high	yes	http://www.ipl.com/products/tools/pt400.uk.php
19	CppTest	stable	easy	supports	very high	no	http://cpptest.sourceforge.net/
20	CppUnit	stable	easy	supports	high	no	http://cppunit.sourceforge.net/
21	CUnit	stable	easy	supports	very high	no	http://cunit.sourceforge.net/
22	DTM Data Generator	stable	easy	Windows or with the help of virtual machine	high	no	http://www.sqledit.com/dg/

23	FitNesse	very stable	easy	Windows, Linux, Mac OS X	low	так	http://www.fitnesse.org/
24	JProbe	unstable	easy	supports	high	yes	http://www.quest.com/jprobe/
25	JUnit	stable	easy	Windows, Linux, Mac OS X	high	no	http://www.junit.org/
26	NUnit	stable	easy	Windows	very high	yes	http://www.nunit.org/
27	TestNG	very stable	easy	Windows, Linux, Mac OS X	high	yes	http://testing.org/
28	VectorCAST/ C++	stable	easy	Windows, Linux, Mac OS X	high	yes	http://www.vectorcast.com/software-testing-products/c++-unit-testing.php
29	JVerify	stable	easy	supports	high	no	https://jverify.us/
Network protocol level testing							
30	Conkormiq Qtronic	stable	easy	Windows, Linux, Mac OS X	high	yes	http://www.conformiq.com/qtronic.php
31	DTM DB Stress	stable	easy	Windows, Linux, Mac OS X	high	yes	http://www.sqledit.com/stress/index.html
32	HttpUnit	stable, if the server code changes, the tests may need to be updated	easy	Windows, Linux, Mac OS X	high	yes	http://httpunit.sourceforge.net/
33	JMeter	very stable	easy	Windows, Linux, Mac OS X	high	yes	http://jakarta.apache.org/jmeter/
34	LoadRunner	stable, but changing existing tests can be quite difficult	easy	Windows	high	yes	https://www.hp.com/
35	MessageMagic	stable, but depends on external libraries	easy	supports	high	yes	http://www.elvior.com/messagemagic/
36	NeoLoed	very stable	easy	supports	very high	yes	http://www.neotys.com/
37	Nikto	stable	easy	Windows, Linux, Mac OS X	very high	no	http://cirt.net/nikto2
38	OpenSTA	stable	difficult	Windows, Linux, Mac OS X	low	yes	http://www.opensta.org/
39	OpenTTCN Tester	stable	easy	Windows, Linux, Mac OS X	high	yes	http://www.opentten.com/
40	Oracle Load Testing	stable	easy	Windows, Linux	high	yes	http://www.oracle.com/
41	QALoad	stable except for new versions	easy	Windows, Unix	high, but depends on the complexity of the tests	yes	http://www.microfocus.com/

					and the amount of data		
42	Rational App Scan	stable if the web application does not undergo significant changes in structure or design	easy	Windows, Linux	high	yes	http://www.ibm.com/
43	Rational Performance Tester	stable	easy	Windows, Linux, Mac OS X	high	yes	http://www.ibm.com/
44	SilkPerformer	stable if changes are not very significant and the test script should not be completely reworked	easy	supports	high	yes	http://www.borland.com/
45	soapUI	stable	easy	Windows, Linux, Mac OS X	high	yes	http://www.soapui.org/
46	The Grinder	stable	difficult	Windows, Linux, Mac OS X	high for small projects	yes	http://grinder.sourceforge.net/
47	WAPT	stable if the changes are not very significant	easy	Windows, Linux, Mac OS X	high	yes	http://loadtestingtool.com/
48	WebInspect	stable if the changes are not very significant	easy	Windows, Linux, Mac OS X	high, but depends on the complexity of the web application, the number of tests and the amount of data	yes	http://www.hp.com/

Results & Discussion

When performing an analysis of testing tools, the following was found: for UI testing – 16 tools; for API testing – 13 tools; for testing the network protocol level – 19 tools.

In general, there are 9 tools for Unit testing, 4 for Integration testing, 15 for System testing, and 20 for Acceptance testing. The diagram of the distribution of tools depending on the level of testing is presented in Fig. 1.

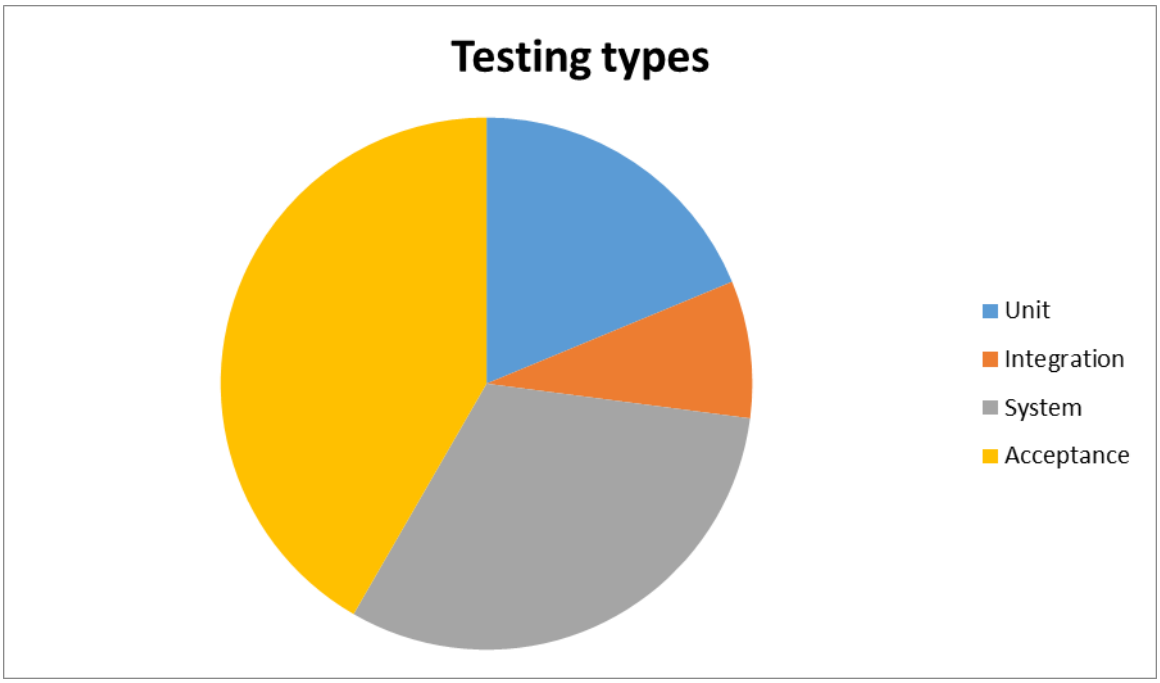

Fig. 1. Diagram of the distribution of tools depending on the level of testing

The conducted research can be used when choosing tools for testing software products.

Conclusions

Recently, software has been intensively used in almost all branches of business. Testing is an integral process of the software life cycle, during which the compliance of the software with the given requirements and needs of the customer is proven, thereby ensuring the quality of the software. The article analyzes software testing tools with their generalization by testing levels.

The study showed that there are a number of studies aimed at reviewing and classifying software testing tools. The correct choice of tools for software testing is one of the vital elements for ensuring the quality of the entire project. However, most of the work in the field of testing focuses on describing test methods without directly connecting to the tools that are based on these methods.

A specialist's approach to software testing requires additional information about currently available testing tools. With the increasing complexity of software products and shorter development cycles, it is becoming apparent that manual testing cannot provide the level of quality required by the market. Choosing the wrong test tools for a project leads to inadequate quality measurements or tool replacement during the project. Both the wrong choice and the change of testing tools during the development process affect the quality of the software product and, as a result, the success of the project as a whole.

The classifiers considered in the work can be used for the appropriate selection of software testing tools. On the one hand, it can be useful for orientation in a wide subject field of testing, reducing the time needed by specialists to find the right solution. On the other hand, it can be used as a brief introduction to the rapidly developing field of testing and available testing tools for those who are not experts in the field. The conducted classification can be used for testing various software projects, depending on the type of software and development methodology.

References

1. L. Li, Y. Tian. Application of Data Guidance Site Generation Technology in the Cloud Platform Supporting the Construction of Subject Teams in Finance and Economics Applied Universities. 2022 International Conference on Edge Computing and Applications: Proceedings (Tamilnadu (India), 2022). Tamilnadu, 2022. Pp. 19-22.
2. T. Hovorushchenko. Methodology of Evaluating the Sufficiency of Information for Software Quality Assessment According to ISO 25010. Journal of Information and Organizational Sciences. 2018. Vol. 42. No.1. Pp. 63-85.
3. T. Hovorushchenko, O. Pavlova, D. Medzatyi. Ontology-Based Intelligent Agent for Determination of Sufficiency of Metric Information in the Software Requirements. Advances in Intelligent Systems and Computing. 2020. Vol. 1020. Pp. 447-460.
4. Manual and automated testing. URL: https://qalight.ua/baza-znaniy/ruchne-ta-avtomatizovane-testuvannya/.
5. M. Rennhard, M. Kushnir, O. Favre, D. Esposito, V. Zahnd. Automating the Detection of Access Control Vulnerabilities in Web Applications. SN Computer Science. 2022. Vol. 3. Paper no. 376.
6. M. Surendra Naidu. Classification of defects in software using decision tree algorithm. International Journal of Engineering Science and Technology. 2013. Vol. 5. Paper no. 06.
7. M. Felderer, P. Zech, R. Breu, M. Büchler, A. Pretschner. Model-based security testing: a taxonomy and systematic classification. Software Testing, Verifiation & Reliability. 2016. Vol. 26. Issue 2. Pp. 119-148.
8. S. Nair, J. L. de la Vara, M. Sabetzadeh, L. Briand. Classification, Structuring, and Assessment of Evidence for Safety - A Systematic Literature Review. 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation: Proccedings (Luxembourg (Luxembourg), 2013). Luxembourg, 2013. Pp. 94-103.

9. M. Felderer, B. Agreiter, P. Zech, R. Breu. A Classification for Model-Based Security Testing. The Third International Conference on Advances in System Testing and Validation Lifecycle: Proceedings (Madrid (Spain), 2011). Madrid, 2011. Pp. 109-114.

10. S. Kumar, S. Mane, S. Mali. A Comparative Study of Machine Learning Algorithms for Software Quality Classification. Journal of King Saud University - Computer and Information Sciences. 2021. Vol. 33. Issue 3. Pp. 308-315.

11. M. Gokhale, S. Shukla. Improving the Software Development Process with SWEBOK and Agile Methodologies. The 4th International Conference on Computing Methodologies and Communication: Proceedings (Erode (India), 2021). Erode, 2021. Pp. 476-484.

Iryna Zasornova **Ірина Засорнова**	Associate Professor of Computer Engineering & Information Systems Department, Khmenlnytskyi National University https://orcid.org/0000-0001-6655-5023 e-mail: izasornova@gmail.com	Кандидат технічних наук, доцент, доцент кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет
Tetiana Hovorushchenko **Тетяна Говорущенко**	DrSc (Engineering), Professor, Head of Computer Engineering & Information Systems Department, Khmenlnytskyi National University https://orcid.org/0000-0002-7942-1857 e-mail: govorushchenko@gmail.com	Доктор технічних наук, професор, завідувач кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет
Oleg Voichur **Олег Войчур**	Lecturer of Computer Engineering & Information Systems Department, Khmenlnytskyi National University https://orcid.org/0000-0001-8503-6464 e-mail: o.voichur@gmail.com	Асистент кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет