UDC 004.05
DOI https://doi.org/10.32782/tnv-tech.2024.4.1

FEATURES OF AUTOMATED TESTING USING FRAMEWORKS

Antonenko A. V. *– PhD in Technical Sciences, Associate Professor, Associate Professor at Department of Standardization and Certification of Agricultural Products of the National University of Life and Environmental Sciences of Ukraine*
ORCID ID: 0000-0001-9397-1209

Vostrikov S. O. *– Postgraduate Student at the Department of Computer Engineering of the State University of Information and Communication Technologies*
ORCID ID: 0009-0008-8425-8872

Burachynskyi A. Yu. *– Postgraduate Student at the Department of Computer Engineering of the State University of Information and Communication Technologies*
ORCID ID: 0009-0003-7913-2152

Tverdokhlib A. O. *– Postgraduate Student at the Department of Computer Engineering of the State University of Information and Communication Technologies*
ORCID ID: 0000-0002-6591-2866

Balvak A. A. *– Postgraduate Student at the Department of Computer Engineering of the State University of Information and Communication Technologies*
ORCID ID: 0000-0002-6441-8225

Slobodian O. A. *– Master of the State University of Information and Communication Technologies*
ORCID ID: 0009-0008-9388-3629

The article examines three of the most common problems that automatizers face, as well as ways to solve them. In order to successfully enter the market and maintain a stable position, each product is tested at different stages of its life cycle and in different aspects. This is a critical step in the development and support process, because as the IT industry evolves, the demand for the end product to remain competitive increases. Today, important features such as stability, ease of use, and thoughtful user interface design have become standard. In order to consistently attract new customers, it is necessary to provide more than just the basic requirements; each characteristic or specification needs detailed verification. It is estimated that the total cost of software testing can be 15 to 25% of the total project cost, so it is important to approach this process with a clear plan and preparation. Products can be conventionally classified into two categories: those that will not be refined after entering the market, and those that will undergo constant improvements throughout their life cycle. In the first case, it is enough to carry out full manual testing, and in case of a successful result, release the product. In the second, testing must be carried out regularly, with each change or update, which takes a significant part of resources and budget. That is why such projects usually use test automation, which reduces the amount of manual work. This can be more cost-effective, and can significantly save time and resources, while providing greater accuracy and efficiency compared to repeated manual testing. However, creating a new system for developing and running test cases for each project is quite difficult and expensive. In this regard, specialists have developed universal frameworks for automated testing, which are flexibly adjusted to specific needs. The use of such frameworks is accompanied by certain challenges that can affect the effectiveness of testing.

Key words: *test framework, testing, automation, test case, automation tools.*

Антоненко А. В., Востріков С. О., Бурачинський А. Ю., Твердохліб А. О., Балвак А. А., Слободян О. А. ***Особливості автоматизованого тестування з використанням фреймворків***

У статті розглянуто три найбільш поширені проблеми, з якими стикаються автоматизатори, а також шляхи їх вирішення. Для успішного виходу на ринок та стабільної утримання позицій, кожен продукт проходить тестування на різних етапах свого життєвого циклу та в різних аспектах. Це критично важливий етап у процесі розробки та підтримки, оскільки з розвитком ІТ-індустрії зростає вимога до кінцевого продукту, щоб він залишався конкурентоспроможним. Сьогодні такі важливі характеристики, як стабільність, зручність у використанні та продуманий дизайн інтерфейсу користувача стали стандартом. Щоб постійно залучати нових клієнтів, необхідно забезпечити більше, ніж просто базові вимоги; кожна характеристика чи специфікація потребує детального перевіряння. За оцінками, загальні витрати на тестування програмного забезпечення можуть становити від 15 до 25% загальної вартості проекту, тому важливо підходити до цього процесу з чітким планом та підготовкою. Продукти можна умовно класифікувати на дві категорії: ті, які після виходу на ринок не будуть допрацьовуватись, і ті, що зазнаватимуть постійних поліпшень протягом всього свого циклу. У першому випадку досить провести повне мануальне тестування, й у разі успішного результату випустити продукт. У другому – тестування необхідно проводити регулярно, з кожною зміною чи оновленням, що займає значну частину ресурсів і бюджету. Саме тому на таких проектах зазвичай застосовується автоматизація тестування, яка знижує обсяги ручної роботи. Це може бути економічніше, а також дозволяє суттєво зекономити час і ресурси, забезпечуючи при цьому більшу точність та ефективність в порівнянні з повторним ручним тестуванням. Проте створити нову систему для розробки та запуску тест-кейсів для кожного проекту – досить складно та витратно. У зв'язку з цим фахівці розробили універсальні фреймворки для автоматизованого тестування, які гнучко налаштовуються під специфічні потреби. Використання таких фреймворків супроводжується певними викликами, які можуть впливати на ефективність тестування.

Ключові слова: *тестовий фреймворк, тестування, автоматизація, тест кейс, інструменти для автоматизації.*

Introduction. In today's world, where the speed of software development is key, software testing is undoubtedly an integral part of the software development life cycle (SDLC). With the development of Agile and DevOps methodologies, as well as the desire of enterprises for rapid releases and quality products, there has been a need for software testing methods that are faster and more efficient than manual testing. It is at this moment that the architecture of automation of software testing, which is implemented with the help of frameworks, is becoming more and more widespread and today is the leader in software testing [1-3].

The Test Automation Framework is not a single tool or process, but a set of tools that work together to support direct automated testing, test case development, report generation, or integration with other tools. It combines various parts such as libraries, test data and various reusable modules. It is a conceptual part of automated testing that helps testers use resources more efficiently. A framework can be defined as a set of rules or best practices that can be followed systematically to ensure the desired results are achieved. In general, it is a platform developed by integrating various hardware and software resources, as well as using various tools based on qualified selection. It allows you to effectively design and develop automated testing scenarios, and also provides a reliable analysis of problems or errors for the object under test [4-6].

When developing and using frameworks, automatizers will face a number of problems that can affect the quality of testing and its costs. Some of these problems will be unique to the project being developed, but many of them will be either the same or similar in meaning or origin, and therefore it will be sufficient to adapt an existing solution instead of developing a new one. Therefore, in order to bring the greatest benefit and be able to correctly allocate project resources, it is important to understand the problems that often occur and to be able to adjust existing solutions to your own needs, because frameworks remain an important tool for ensuring software quality. Using the right approach, you can significantly reduce testing costs and reduce the time required to release new software versions. Below we will consider the main difficulties that are often encountered in the development and use of frameworks for automated testing, as well as several possible ways to solve these problems, with a separate analysis of each of them. This will allow you to deepen your understanding, as well as to have in your arsenal several options for approaches that can be combined to achieve the most beneficial result.

Formulation of the problem. One of the main challenges in automation is the need to ensure that tests cover various requirements, both functional and non-functional, while maintaining a high speed of test execution and maximum autonomy from humans. Therefore, it requires more knowledge and skills from automation engineers, since improper use and configuration of the framework can lead to deterioration of the testing process, which will have a negative impact on the entire project [7-10].

The problems considered in this article have an important practical connection with tasks in the field of software development and testing. After all, the developers of test frameworks always face the task of providing the best possible support and quality control of the product

The aim of the study. The purpose of the work is to deepen the understanding of the problems associated with the use of frameworks for automated testing and to provide information on possible ways to solve them.

The object of research is frameworks for automated testing.

The subject of the research are problems related to the use of frameworks for automated testing and ways to solve them.

Analysis of recent research and publications. The analysis of information sources related to the problems of development and use of frameworks for automated testing confirms the importance of this topic and helps to gain a deeper understanding of the existing issues.

In the scientific work «Challenges in Test Automation Framework Design and Development» in the International Journal of Advanced Research in Computer Science and Software Engineering [5], he examines the problems that arise in the development and use of frameworks for automated testing. The article describes issues with data usage, issues with managing test scenarios and scripts, and issues with test environment variability.

The study «Issues and Challenges of Test Automation: A Systematic Literature Review» [6] examines the problems associated with automated testing in general. The study confirmed that many companies use frameworks for automated testing and often face problems in their development and maintenance. The study notes that the lack of standards and norms in the development of test scenarios is one of the main problems.

In the scientific work «The challenges and benefits of continuous integration in software engineering» in the journal Information and Software Technology [11], he considers the problems associated with the implementation of Continuous Integration in the software development process. The article states that the integration of test frameworks is one of the most difficult tasks when implementing Continuous Integration. The article also describes the benefits of Continuous Integration, such as reducing software release risks and reducing the time it takes to identify and fix bugs.

The study «Automation testing challenges and solutions: A review» in the International Journal of Computer Applications [12] explores the challenges associated with automated testing in general. The article describes problems that may arise when using frameworks, such as the difficulty of debugging test scripts and problems with the variability of the test environment. The article also provides various solutions to solve these problems.

After analyzing the sources, the questions that most often affect the testing process were selected for consideration:

1. Creation and maintenance of test scripts.
2. Lack of generally accepted standards and norms for the development of test scenarios.
3. Integration of test frameworks into the process of Continuous Integration / Continuous Development.

These problems lead to errors and delays in the software development process, so studying and solving them is an important topic for those involved in software development and testing. Knowing about these issues will help you develop an effective testing strategy and choose the tools that are most suitable for a particular project.

Presentation of the main research material. The first problem that automatizers face is the creation and maintenance of test scripts. This task takes a lot of time and effort of the automatizer, because the system under test is often changed and supplemented, and therefore, previously written tests may already be out of date and need to be updated. When there are thousands of such tests, it is not an easy task, so in order not to waste time and project resources, it is important to understand the problem and approaches to solving it. Let's analyze the possible solutions.

The first way is to use design patterns and code refactoring. Conventionally, three levels of re-use of parts of the code can be distinguished. The lower one is a collection of classes, libraries, and modules. Frameworks are at the highest level, because only architecture is important for them. A framework is usually much larger than a single class. It allows you to specify the desired behavior, and then, when certain conditions are met, it itself causes it. For example, JUnit calls your class when it needs to run a test. Everything else happens inside the framework. At the middle level, you can place patterns that are more abstract than frameworks, and at the same time have less binding to the programming language. They are descriptions of how certain classes or methods interact with each other. The main advantages of using templates are as follows:

1. Save time and effort
2. Reduce maintenance costs
3. Improve code reuse
4. Increase reliability
5. Help create structured code that facilitates the automation process
6. Improve communication and understanding between engineers.

They help separate the internal implementation of the framework from the implementation of tests or solve the problem of code standardization and allow you to use solutions that have already been tested by a huge number of engineers, adapting them to your own needs.

Templates often used in automation are Page Object Model, Factory and Singleton.

Page Object Model (POM) allows you to separate the structure of the web page from the test script, which provides greater stability and convenience of test case support. With the help of POM, the structure of a web page is displayed as objects whose methods abstract requests and settings, giving only a user-friendly readable interface to the outside. Another advantage is the possibility of reusing these objects. If the structure of the web page changes, then only the implementation of the object needs to be changed, and the test cases will remain unchanged.

In the factory design pattern, there is a class with a factory method that handles all the processes of creating objects. This pattern has a superclass with several subclasses, and based on user input at the test class level, it returns one of the subclasses. A class that extends the parent class is responsible for the implementation logic, so it hides complex code at the testing level. As a user, we just need to create an object of this class and use it in the test to call the corresponding method containing the business logic.

The Singleton design pattern is one of the simplest and most straightforward patterns to implement in an automation framework. This pattern is used when you want to use the same class object in different places. It limits the possible instances of a class to a single instance. To implement it, you need to declare the constructor of the class as private so that no one can create an instance of the class outside of it, then declare a static reference variable of the class and a static method that returns an object of the same class. Also, this method should check whether the object has already been created once [13].

The next solution is to use tools that allow you to write test scripts. Such tools automatically remember the actions performed by the user. An example is Selenium IDE, which allows you to record test actions in Selenese format. This is very convenient for automators without sufficient programming knowledge, however, research shows that using such tools is ineffective in case of complex test scenarios or when changing the structure of a web page. Also, the lack of programming skills in the automator can lead to the generation of incorrect test scripts, so this approach requires attention and a mandatory review of tests by more experienced colleagues.

Another way is to use parameterization. This approach creates a single (perhaps slightly more complex) test script that accepts a specific set of test variations, allowing for many more tests. A good example is testing any input field. You can create a single test script that will accept various combinations of text as input, and check the behavior of the site in each of the variations. But it is worth noting that the use of parameterization is not always necessary and it depends on the situation. The best scenario for such tests is when we have many test variations for a single test. On the other hand, parameterized tests should not be used in situations where only one set of data is tested. Separating the data from the test would be an overcomplication.

Advantages of using parameterization:

1. Only one test is needed for many test cases.
2. Test logic is separated from test data.
3. The data file can be easily shared with other team members, especially useful if you need to share with manual testers who do not know the programming language.
4. They reduce code repetition.

Disadvantages of using parameterization:

1. Separating test logic from test parameters requires additional work.
2. Redundancy for tests with a small number of test cases to verify.
3. It is often necessary to maintain an additional file with test cases.

A study by Sauce Labs showed that the use of parameterized test scripts allowed to reduce the time required to maintain test scripts by 70% compared to test scripts without parameterization, and therefore, in the event of changes in the system, much less effort is needed to fix the test scripts [14].

Also, when developing test scripts, you should be guided by the 7 principles of testing according to ISTQB:

1. Testing reduces the chance that software will have undetected defects, but even if no defects are found, testing is not proof that the program is correct.
2. Testing all combinations of input data and prerequisites is impossible, except in trivial cases. Rather than attempting exhaustive testing, risk analysis, test methods, and priorities should be used to focus testing efforts.
3. To detect defects at an early stage, static and dynamic testing should begin as early as possible in the software development life cycle. Early testing is sometimes called a shift to the left. Testing early in the software development life cycle helps reduce or eliminate costly changes.
4. A small number of modules usually contain most of the defects found during pre-release testing or are responsible for most of the operational failures. Predicted defect clusters and actual detected defect clusters during testing or operation are important inputs to the risk analysis used to focus testing efforts (as outlined in principle 2).
5. If you repeat the same tests over and over again, these tests will eventually stop revealing new defects. To detect new defects, it may be necessary to modify existing tests and test data, as well as to write new tests.
6. Testing is done differently in different contexts. For example, safety-critical industrial control software is tested differently than an e-commerce mobile application.
7. Some organizations expect that testers can perform all possible tests and find all possible defects, but principles 2 and 1, respectively, say that this is not possible. Furthermore, it is a mistake to expect that simply identifying and correcting a large number of defects will ensure the success of the system. For example, thoroughly testing all identified requirements and fixing all defects found may result in a system that is difficult to use, does not meet user needs and expectations, or is inferior to other competing systems.

Development of test scripts is a time-consuming process that requires constant improvement. However, using the right technologies and approaches will help ensure a more efficient use of time and overall better software testing.

The second problem that needs attention is the lack of generally accepted standards and norms for the development of test scenarios. Often, automators can have different approaches to development, which leads to difficulties in maintaining such code by other QA engineers. There are no universally defined rules that everyone must follow in the process of developing test scripts, and therefore each person may have his own implementation of a solution to this or that problem. This isn’t always a bad thing, but it’s definitely worth spending time on when you’re starting to develop a test framework.

For example, the study «A Systematic Review of Test Automation Tools and Frameworks for Web Applications», published in the journal IEEE Access, showed that most of the 50 analyzed tools and frameworks for automated testing of web applications use their own testing methodologies, which can lead to inequality and causes the need for possible retraining of existing specialists [15].

A good solution to this issue is to analyze existing standards from related fields, such as development or manual testing, and develop standards based on them for a single team, or a group of teams working on the same project. Such standards may include rules for creating test scripts, rules for formatting code to make it look consistent, using testing techniques, and unifying error reporting and tracking processes. All these actions will help to ensure standardization of various test artifacts throughout the test life cycle, and increase the understanding for all team members. For example, if the development of tests takes place in the Python programming language, you can use PEP8. This is a style guide for Python code that is recommended for use by developers of the language, although it is not required. Many other programming languages have similar sets of rules and guidelines. As for reporting, it all depends on how the reports are generated. Allure Framework is very popular today. It is a flexible, lightweight, multilingual test reporting tool that not only shows a very concise representation of what was tested in the form of a web report, but also allows everyone involved in the development process to extract the most useful information from day-to-day test execution.

But these solutions will not be able to fully exist without a process of review and verification. After developing new test cases or parts of the framework, the written code must be reviewed by at least one more automatizer from the team. This will help identify errors and correct them immediately, as well as maintain adherence to accepted standards. This process is described by the ISTQB organization. There are four types of reviews, from informal to the most standardized, and therefore it is important to understand when and which review to apply, because this process takes the time of two or more engineers at the same time.

1. Informal:

– can take the form of pair programming or a technical lead who reviews the design and code;

– results can be documented;

– usefulness varies depending on the reviewers;

– the main goal: an inexpensive way to get a certain benefit;

2. Step-by-step introduction:

– a meeting led by the author;

– can take place in the form of scenarios, trial runs or group participation;

– open sessions;

– optional preparation of reviewers before the meeting;

– optional review report;

– main goals: learning, gaining understanding, finding flaws;

3. Technical review:

– a documented, defined defect detection process;

– under the guidance of a trained moderator;

– preparation of reviewers for the meeting;

– optional use of checklists;

– preparation of the review report;

– in practice it can vary from quite informal to very formal;
– main goals: discussion, decision-making, evaluation of alternatives, identification of shortcomings, resolution of technical problems and verification of compliance with specifications, plans, rules and standards.
4. Inspection:
– under the guidance of a trained moderator;
– usually conducted as an expert assessment;
– collection of metrics and various data;
– a formal process based on rules and checklists;
– defined input and output criteria for software product acceptance;
– preparation for the meeting;
– inspection report, including a list of conclusions;
– formal process of further actions;
– main goal: detection of defects.
To be successful in revue, you should follow the following rules:
– clearly define goals;
– fill in and use the documents accepted for the project;
– attract only those people who are really needed in order to get the maximum benefit and not waste time;
– try to identify defects, but express them objectively;
– consider human issues and psychological aspects;
– apply appropriate review techniques and review in an atmosphere of trust;
– conduct trainings on assessment techniques;
– focus on learning and improving the process;
– follow the rules.

It is important to conduct regular code reviews of test scripts to identify inconsistencies and deviations from standards. This requires using static code analyzers to detect potential flaws. They find problems such as undefined variable access, inconsistent interfaces between modules and components, unused or incorrectly declared variables, unreachable (dead) code, missing and erroneous logic (potentially infinite loops), overly complex designs, violations programming standards, vulnerabilities in the security system and violations of code syntax and software models. Static analysis must be included in the overall testing process, because it covers and finds those problems that the dynamic approach is unable to find.

From the company's point of view, a possible solution is to conduct trainings and seminars so that the automatizers become familiar with the common standards and techniques used in the team. To solve the problem of the lack of standards and norms in automation, it is necessary to establish common approaches and standardize possible testing processes, conduct regular code checks and trainings for traffic jams [16].

An important issue to consider is the integration of test frameworks into the Continuous Integration / Continuous Development process. This process is important in software development, because it allows you to automate the process of running tests and ensure fast detection of errors. However, integration with test frameworks has some complexities that need attention.

The lack of communication between the test framework and the version control system leads to loss of time in the detection and correction of errors, since there is no possibility to automatically track changes in the code and the corresponding changes in the tests. The solution to this problem is to use tools like Jenkins that support integration with version control systems, for example Git, which is the most popular system. This allows you to automatically run tests when the code changes and detect and fix defects accordingly [17].

Lack of ability to run tests in different environments. Software is often designed to run on different operating systems and with different configurations. A solution approach is to use containers such as Docker. They allow you to create isolated environments for running tests. This allows you to ensure the same conditions for running tests on different environments and identify and fix problems in different configurations accordingly. A container is a standard unit of software that packages code and all its dependencies so that the program runs quickly and reliably, regardless of the system on which it is run. A Docker container image is a lightweight, self-contained, executable software package that includes everything needed to run an application: code, runtime, system tools, system libraries, and configuration. The advantages of Docker containers in particular are that they are standardized (Docker created an industry standard for containers so that they can be moved anywhere), lightweight (containers use the core of the machine’s operating system, and therefore do not need a separate operating system for each application, which increases efficiency server workloads and reduces server and licensing costs) and secure (containerized applications are more secure and Docker provides the industry’s strongest isolation capabilities by default).

Testing takes a lot of time, so not being able to run tests in parallel can also be a problem, because it reduces the speed of software development and release. The solution is to use parallel test execution tools like Selenium Grid or TestNG or similar. They make it possible to run a larger number of tests at the same time, and accordingly reduce the time spent on testing. It is important to remember that automatic tests must be developed so that they do not depend on each other, because otherwise parallel running will not bring the desired benefit [18].

Another important issue in the CI/CD process is the ability to save test results, as it is difficult to track reports and identify problems. To solve this, you need to use test data collection tools like JUnit or TestNG. These tools allow you to collect test information such as execution time, status, and results and save it as a report. This allows for detailed analysis of launch results.

Research by the DZone company showed that the use of CI/CD tools reduces the duration of testing by 30-50%, increases the frequency of releasing new software versions, and reduces the number of errors detected during the testing phase [19]. This proves that the integration of test frameworks into the CI/CD process can reduce the duration of software development and release, increase product quality and reliability, and reduce development and testing costs.

Sauce Labs also conducted a study that showed that using containers can reduce testing time by an average of 25%, increase the number of tests that can be run in parallel, provide greater test stability, and reduce system impact [20, 21].

Integrating test frameworks into the CI/CD process is an important stage of software development that allows you to automate the testing process and ensure fast and efficient error detection. CI/CD tools, containers for running tests, tools for running tests in parallel and collecting data about their results can be used to solve problems that may arise during the integration of test frameworks. Studies show that the use of such tools can reduce testing time, increase the quality and reliability of software, and reduce costs for its development and testing [22, 23].

Conclusions. There are a number of challenges in developing and using automated testing frameworks that affect team performance, test quality, cost, and more. Automation has many benefits, but if the above issues are addressed during planning and implementation, or, failing that, their impact on the project is minimized. This article analyzed the most common problems and described ways to solve them based on research by various companies and generally accepted development and testing standards. Although the needs of different projects are very different, technologies and approaches to solving possible problems have been given that can help or suggest in which direction to look for a solution to a particular issue. The methods described above will help increase the efficiency and volume of automation, improve the quality of the final product.

BIBLIOGRAPHY:

1. Hardik S. Software Testing Cost, 2022. URL: https://www.simform.com/blog/software-testing-cost/ (дата звернення: 02.04.2023).

2. Dudekula M., Katam Reddy K., Kai P., Benefits and Limitations of Automated Software Testing: Systematic Literature Review and Practitioner Survey, Автоматизація тестування програмного забезпечення (AST), 7-й міжнародний семінар з питань, 2012.

3. A. Contan, C. Dehelean and L. Miclea, «Test automation pyramid from theory to practice», 018 Міжнародна конференція IEEE з автоматизації, якості та тестування, робототехніки (AQTR), Клуж-Напока, Румунія, 2018, с. 1-5, doi: 10.1109/AQTR.2018.8402699.

4. Vogel-Heuser, B., Diedrich, C., Fay, A., Jeschke, S., Kowalewski, S., Wollschlaeger, M. and Göhner, P. (2014) Challenges for Software Engineering in Automation. Журнал програмної інженерії та додатків, 7, с. 440-451. doi: 10.4236/jsea.2014.75041

5. Shukla, P., Patel, D., Challenges in Test Automation Framework Design and Development. International Journal of Advanced Research in Computer Science and Software Engineering, 2016, с. 67-71.

6. Khan, S. R., Ali, T., Khan, S., Issues and Challenges of Test Automation: A Systematic Literature Review. Journal of Intelligent & Fuzzy Systems, 2018, с. 2097-2108.

7. Твердохліб А.О., Коротін Д.С. Ефективність функціонування комп’ютерних систем при використанні технології блокчейн і баз данних. Таврійський науковий вісник. Серія: Технічні науки, 2022, (6)

8. Цвик О.С. Аналіз і особливості програмного забезпечення для контролю трафіку. Вісник Хмельницького національного університету. Серія: Технічні науки, 2023, (1)

9. Новіченко Є.О. Актуальні засади створення алгоритмів обробки інформації для логістичних центрів. Таврійський науковий вісник. Серія: Технічні науки, 2023 (1)

10. Зайцев Є.О. Smart засоби визначення аварійних станів у розподільних електричних мережах міст. Таврійський науковий вісник. Серія: Технічні науки, 2022, (5).

11. Humayun, M., Iqbal, M. Z., The challenges and benefits of continuous integration in software engineering. Information and Software Technology, 2017, с. 153-167.

12. Bajaj, S., & Singh, S. (2017). Automation testing challenges and solutions: A review. International Journal of Computer Applications, 173(4), 23-28.

13. Олександр Ш., Занурення в патерни проєктування / за ред. М. Ельвіри, Refactoring.Guru, 2021.

14. M. Leotta, D. Clerissi, F. Ricca and C. Spadaro, «Improving Test Suites Maintainability with the Page Object Pattern: An Industrial Case Study,» 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops, Люксембург, Люксембург, 2013, с. 108-113, doi: 10.1109/ICSTW.2013.19.

15. Gunjan K., Page Object Model, URL: https://www.toolsqa.com/selenium-webdriver/page-object-model/ (дата звернення: 03.04.2023).

16. John Kent M. Sc, Test Automation: From Record / Playback to Frameworks, 2019, URL:https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=327a028cc53b0774671ee380e0e268e3ffae28b4 (дата звернення: 03.04.2023).

17. Sauce Labs, Parameterized Testing: A Practical Guide for Better Tests., URL: https://saucelabs.com/resources/articles/parameterized-testing-a-practical-guide-for-better-tests.(дата звернення: 01.04.2023)

18. Nikolai Tillmann, Jonathan de Halleux, and Tao Xie, Parameterized unit testing: 32nd ACM/IEEE International Conference on Software Engineering – Volume 2 (ICSE ‘10). Association for Computing Machinery, Нью-Йорк, США, с. 483–484. doi: https://doi.org/10.1145/1810295.1810441

19. Z. Ali, S. S. Awan, S. A. Khan, M. H. Shah, A Systematic Review of Test Automation Tools and Frameworks for Web Applications, 2019

20. ISTQB Glossary, URL: https://glossary.istqb.org/en_US/search?term= (дата звернення: 30.03.2023).

21. Sheekha J, What is Version Control System, 2021, URL: https://www.toolsqa.com/git/version-control-system/ (дата звернення: 07.04.2023).

22. Dzone, The State of Continuous Integration and Continuous Delivery: 2021 Report, 2021, URL: https://dzone.com/articles/ci-cd-tools-and-trends-survey-2019-2020-results (дата звернення: 30.03.2023).

23. SauceLabs, The Benefits of Containers in Agile Testing, 2021, URL: https://saucelabs.com/resources/white-papers/containerization-testing-landscape-report-2019 (дата звернення: 30.03.2023).

REFERENCES:

1. Hardik S. (2022) Software Testing Cost. URL: https://www.simform.com/blog/software-testing-cost/ (data zvernennia: 02.04.2023).

2. Dudekula M., Katam Reddy K., Kai P. (2012) Benefits and Limitations of Automated Software Testing: Systematic Literature Review and Practitioner Survey, Avtomatyzatsiia testuvannia prohramnoho zabezpechennia (AST), 7-y mizhnarodnyi seminar z pytan.

3. A. Contan, C. Dehelean and L. Miclea (2018) «Test automation pyramid from theory to practice», 018 Mizhnarodna konferentsiia IEEE z avtomatyzatsii, yakosti ta testuvannia, robototekhniky (AQTR), Kluzh-Napoka, Rumuniia, s. 1-5, doi: 10.1109/AQTR.2018.8402699.

4. Vogel-Heuser, B., Diedrich, C., Fay, A., Jeschke, S., Kowalewski, S., Wollschlaeger, M. and Göhner, P. (2014) Challenges for Software Engineering in Automation. Zhurnal prohramnoi inzhenerii ta dodatkiv, 7, s. 440-451. doi: 10.4236/jsea.2014.75041

5. Shukla, P., Patel, D. (2016) Challenges in Test Automation Framework Design and Development. International Journal of Advanced Research in Computer Science and Software Engineering, p. 67-71.

6. Khan, S. R., Ali, T., Khan, S. (2018) Issues and Challenges of Test Automation: A Systematic Literature Review. Journal of Intelligent & Fuzzy Systems, p. 2097-2108.

7. Tverdokhlib A. O., Korotin D. S. (2022) Efektyvnist funktsionuvannia kompiuternykh system pry vykorystanni tekhnolohii blokchein i baz dannykh. Tavriiskyi naukovyi visnyk. Seriia: Tekhnichni nauky, no. 6.

8. Tsvyk O.S. (2023) Analiz i osoblyvosti prohramnoho zabezpechennia dlia kontroliu trafiku. Visnyk Khmelnytskoho natsionalnoho universytetu. Seriia: Tekhnichni nauky, no. 1.

9. Novichenko Ye.O. (2023) Aktualni zasady stvorennia alhorytmiv obrobky informatsii dlia lohistychnykh tsentriv. Tavriiskyi naukovyi visnyk. Seriia: Tekhnichni nauky, no. 1.

10. Zaitsev Ye.O. (2022) Smart zasoby vyznachennia avariinykh staniv u rozpodilnykh elektrychnykh merezhakh mist. Tavriiskyi naukovyi visnyk. Seriia: Tekhnichni nauky, no. 5.

11. Humayun, M., Iqbal, M. Z. (2017) The challenges and benefits of continuous integration in software engineering. Information and Software Technology, c. 153-167.

12. Bajaj, S., & Singh, S. (2017) Automation testing challenges and solutions: A review. International Journal of Computer Applications, 173(4), 23-28.

13. Oleksandr Sh. (2021) Zanurennia v paterny proiektuvannia / za red. M. Elviry, Refactoring.Guru.

14. M. Leotta, D. Clerissi, F. Ricca and C. Spadaro (2013) «Improving Test Suites Maintainability with the Page Object Pattern: An Industrial Case Study,» 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops, Liuksemburh, Liuksemburh, s. 108-113, doi: 10.1109/ICSTW.2013.19.

15. Gunjan K., Page Object Model. URL: https://www.toolsqa.com/selenium-webdriver/page-object-model/ (data zvernennia: 03.04.2023).

16. John Kent M. Sc. (2019) Test Automation: From Record/Playback to Frameworks. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=327a028cc53b0774671ee380e0e268e3ffae28b4 (data zvernennia: 03.04.2023).

17. Sauce Labs, Parameterized Testing: A Practical Guide for Better Tests., URL: https://saucelabs.com/resources/articles/parameterized-testing-a-practical-guide-for-better-tests.(data zvernennia: 01.04.2023)

18. Nikolai Tillmann, Jonathan de Halleux, and Tao Xie, Parameterized unit testing: 32nd ACM/IEEE International Conference on Software Engineering – Volume 2 (ICSE 10). Association for Computing Machinery, Niu-York, SShA, s. 483–484. doi: https://doi.org/10.1145/1810295.1810441

19. Z. Ali, S. S. Awan, S. A. Khan, M. H. Shah (2019) A Systematic Review of Test Automation Tools and Frameworks for Web Applications.

20. ISTQB Glossary, URL: https://glossary.istqb.org/en_US/search?term= (data zvernennia: 30.03.2023).

21. Sheekha J. (2021) What is Version Control System, URL: https://www.toolsqa.com/git/version-control-system/ (data zvernennia: 07.04.2023).

22. Dzone (2021) The State of Continuous Integration and Continuous Delivery: 2021 Report. URL: https://dzone.com/articles/ci-cd-tools-and-trends-survey-2019-2020-results (data zvernennia: 30.03.2023).

23. SauceLabs (2021) The Benefits of Containers in Agile Testing. URL: https://saucelabs.com/resources/white-papers/containerization-testing-landscape-report-2019 (data zvernennia: 30.03.2023).