UDC 004.415.53 **DOI: 10.20998/2079-0023.2022.01.13**

N. V. GOLIAN, V. V. GOLIAN, I. V. AFANASIEVA

BLACK AND WHITE-BOX UNIT TESTING FOR WEB APPLICATIONS

The types of the testing methods were analyzed from the side of development. If changes are made to the code, even minor ones, bugs can appear anywhere in the system. The cost of fixing bugs increases with the time they are discovered. Using effective web testing can prevent unexpected costs. Modern web applications are indeed a place where all the type of the testing is vital for the high-quality product. At the moment, traditional front and back office applications are increasingly moving from desktop computers to web interfaces, so testing web applications is becoming extremely relevant. Thus, learning what to automate and how is an important component of successful web application testing. Web application testing is a software practice that is designed to ensure quality by verifying that the functionality of a particular web application works correctly or according to fixed requirements. Web testing makes it possible to find bugs at any point in time, before release or on a daily basis. On the one hand there are wide range of tools that could be used to pursue this goal by means of using best decisions present for now but on the other hand there are essential advantages and disadvantages present in these tools, which are all in the approach to it, so there are always cons and pros of using one or another. For now, despite on having the ability to test using both black and white box testing it looks like the second one is not the best choice. There are several points on each board for both of them, but black box approach that is being represented with react testing library is more successful and efficient way to cover and application with high- and low-level tests, that could be easily maintained and understood. But white box testing is now the most used decision due to the historic development of the industry. So, it also has some great features and could be chosen to be used on the project, but it should be precise choice with the understanding of all the consequences standing behind.

Keywords: manual testing, automated testing, black-box testing, white-box testing, enzyme library, react testing library.

Н. В. ГОЛЯН, В. В. ГОЛЯН, І. В. АФАНАСЬЄВА

МОДУЛЬНЕ ТЕСТУВАННЯ БІЛОЇ ТА ЧОРНОЇ СКРИНЬКИ ДЛЯ ВЕБ ЗАСТОСУВАНЬ

Види методів тестування були проаналізовані з боку розробки. Якщо в код вносяться зміни, навіть незначні, помилки можуть виникнути будь-де системі. Вартість виправлення помилок зростає з часом їх виявлення. Використання ефективного веб тестування може запобігти непередбаченим витратам. Сучасні веб застосування дійсно є місцем, де весь тип тестування є життєво важливим для якісного продукту. На даний момент традиційні фронт та бек-офісні програми все частіше переходять з настільних комп'ютерів на веб інтерфейси, тому тестування веб застосувань стає вкрай актуальним. Таким чином, вивчення того, що і як автоматизувати, є важливим компонентом успішного тестування веб застосувань. Тестування веб застосувань є практикою програмного забезпечення, яка призначена для забезпечення якості за рахунок перевірки того, що функціональні можливості певного веб програми працюють правильним чином або відповідно до зафіксованих вимог. Веб тестування дає можливість знаходити помилки в будь-який час, до релізу або щодня. З одного боку, є широкий спектр інструментів, які можна було б використовувати для досягнення цієї мети за допомогою використання найкращих рішень, присутніх на даний момент, але з іншого боку, є суттєві переваги і недоліки, присутні в цих інструментах, які всі в підході до нього, тому завжди є мінуси і плюси використання того чи іншого. Поки що, незважаючи на можливість тестування з використанням як чорної, так і білої скриньки, останній метод має багато недоліків та не рекомендується до використання. Ці два методи мають місце бути у окремих проєктах, але підхід чорної скриньки, який представлений бібліотекою тестування React є найбільш успішним і ефективним способом покрити і застосування з високими і низькими тестами, які можна легко підтримувати і розуміти. Але тестування білої скриньки в даний час є найбільш вживаним рішенням у зв'язку з історичним розвитком галузі. Отже, воно також має деякі позитивні риси і може бути обрано, щоб бути використаним на проєкті, але повинно бути точним вибором з розумінням всіх наслідків, що за ним слідують.

Ключові слова: ручне тестування, автоматизоване тестування, метод чорної скриньки, метод білої скриньки, бібліотека enzyme, бібліотека тестування react.

Introduction. Since the very beginning of the development process the ability to tests your application is one of the edge things one would like to have to be sure that high quality product is being provided. The main reason for this is to assure the developer that the application works as expected, but also it does a lot of other aspects of work.

Namely finding defects that could appear while development process of the software, keep the level of quality of the product that may be documented, also being able to prevent the defects, test that the final result corresponds to the business requirements as well as System Requirements Specification and Business Requirement Specification, and less but not the least is to assure product owner and stake holders that the product is good enough to be driven in production mode [1].

Types of testing. So it came to the question of what and how should be tested. The main very abstract division between two types is automated and manual testing [2].

Manual testing. Manual testing is done by some real person that tries to interact with the application by means of clicking it or using some special tools for the tests that makes it way easier to find some inappropriate in terms of the quality parts out. As it is for now this type of testing is the most expensive for now.

Manual Testing is part of the testing process at the quality control stage in the software development process [3]. Testers or ordinary users carry it out by simulating possible user action scenarios.

Manual testing is to perform a documented procedure, where the method of performing the dough is described. The technique sets the test order and for each test – a list of parameter values that is fed to enter the list of results at the output. Since the procedure is intended for a person performing, in its description, some default values can be used for brevity, or references to information stored in another document.

The task of the tester is to find the largest number of errors. It should be good to know the most frequently allowed errors and be able to find them for the lowest time. The remaining errors that are not typical are detected only by carefully created test sets. However, it does not follow from this that for typical errors it is not necessary to make tests.

Automated testing. As a completely another approach to testing the automated tests are done with the machine, that goes through the product using some scripts that test it. Here it comes the main necessity of this type of tests which is the ability to test way more parts, as automated tests could have different level of the complexity. The complexity itself differs basing on the aim of it which are the different approach of it. So, by means of the automated tests we can cover almost everything starting with the class method or a whole flow of the user.

But the main problem here is that this type of tests is deeply depended on the quality of them that means that the higher quality of test's script the higher quality the coverage is at the end.

Types of automated tests. Continuous testing accelerates the supply of software, making the entire testing process faster. And thanks to the immediate feedback, which helps in the earliest stages to identify errors and other problems in the application, ensures that development teams will create high-quality and reliable applications. In addition, the ability to organize and conduct effective testing can significantly reduce the costs of the company, both by saving the time of developers and due to the creation of a qualifying supply conveyor in which they can quickly make changes to the code with minimal risks of impaired application performance in the productive environment.

There are some types of automated test should be mentioned to be understood the final goal of the unit ones.

Functional testing. Functional testing is the one of the most popular types of them being used to test the main functions of the system its usability, the ability for the different types of users to use the application and so on. This type of test it is the most abstract one and it includes unit testing, integration testing, smoke testing, and user acceptance testing.

Unit tests. Unit tests is the cheapest type of the tests, as it is being run over individual parts of the application that are called components in isolation [4]. For example, the application that is calculator could have the division operation F, that consists of some function A and B. It means that one should test F, A, B separately without trying to depend one on another.

This type of tests bring tons of the benefits to the quality of the application because it says one the parts the risky as the bugs may appear on the spots that are detected by them on the deepest level possible.

It also gives one the ability to understand the logic of application or its parts without digging to deep in it, so developer or another stakeholder person could not even have no idea of what is going on in the code but can easily understand what this part does.

Integrational testing. The third one is integrational testing which is on another level of the testing pyramid, which is a fundamental in underatanding of how and what should be tested [5].

Testing Pyramid – Concept, according to which there are several levels, on which test automation is possible: module level, service level, E2E (system level) [6].

As it is shown on the fig. 1 the higher, we get from the unit testing the more interactions are being done to test it, this is actually the main purpose of such a testing. One is trying to go higher and connect different isolated parts of the application to the more complex one and test it, not relying on the unit cases.

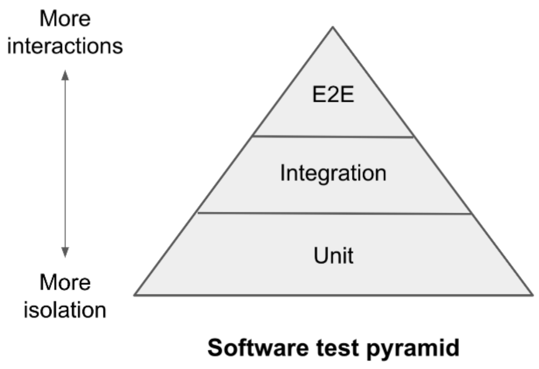

Fig. 1. Testing pyramid

This type of tests is more expensive than the previous one, but on the other hand it brings a developer the ability to understand what happens in more precise way which means that the developer could build essential part of the application and test is isolated.

End-to-end testing. This is the last but not the least type of testing used to verify whether the application works as expected and matches business requirements. Also, to be mentioned that this testing is the most expensive among all the automated tests.

It is as it is due to the number of resources being involved to test one essential part of the product because end-to-end tests scripts requires the high-level understanding of the application logic and also the higher level of the person that writes it. The script itself tests some whole flow of the user, for example it could go through a page and interact with the interface trying to pretend user's behavior.

So, it makes one to think about the flow and script as a user which involves way more resources makes it way more expensive than mentioned above types.

Root problem. So, it gives a developer a space to do a lot with tests, starting with the way an application could be testing and ending with the level of the tests, it also should be considered that the type itself requires different level of the expenses and skills of the one developing tests.

So, the main aim of the automated testing is to be written in the most effective and cheapest way as it should cover as much as possible and also cost the better the less.

Good thing is that any developer of the modern user web-application could use wide range of tools to do that. There are several approaches in testing that are widely used and the main two are black and white-box testing.

The most important and useful for now actually basic one is unit testing where it is always the widest range of how and by means of what it could be done.

Black-box testing. Black box testing is a technique of checking out software program wherein the inner workings (code, architecture, design, etc.) are not known by the tester. Black container checking out makes a specialty of the conduct of the software program and includes checking out from an outside or end-person perspective. With black container checking out, the tester examines the software program's capability without searching on the code or having any understanding of the application's inner flows. Inputs and outputs are examined with the aid of using being as compared to the predicted output. If the output does not in shape the predicted output, a Trojan horse has been found.

The term "black box" is used due to the fact you do not dig into the application. For this reason, non-technical specialists regularly use black box tests. Types of black field trying out consist of purposeful trying out, gadget trying out, usability trying out, and regression trying out.

Advantages: allows you to quickly find bugs in the developed functionality of software: the tester does not have to have a narrow-profile specialty, check runs from the end of the end user, you can develop test cases immediately after completion with the specification.

With this test method, you can perform the following checks: functional verification software, regression inspections, usability testing, smoke testing, check the graphical user interface.

White-box testing. White box testing is a way of trying out software program wherein the inner state (code, architecture, design, etc.) are recognized to the tester [7]. Therefore, it validates the inner shape and frequently specializes in enhancing safety and making the waft of inputs and outputs greater green and optimized. In this type of trying out, the tester appears for inner safety holes and damaged or poorly based coding paths. The term "white field" refers to having visibility into the inner workings. Because of this, a greater technically professional individual conducts the tests. Different forms of this manner consist of unit trying out and integration trying out.

So, it was just some terms to understand what we are going to go through next. The issue that is going to be used to cover the main problem of a discussion for this theme is black and white box testing approaches in modern web applications, where as an example we are going to use the most popular library for that which is namely ReactJS and the most popular framework for testing Jest and React Testing Library and Enzyme libraries for black and white box testing respectively [8].

As was already described above white box testing is a testing that requires the knowledge of the internal implementation of your component, which is being tested. This leads us to some issues. First of them is over specification, which means the when writing test one is being concentrated on how the component is constructed the details of the implementation actually, when the main purpose of testing itself is to test component behavior. This leads to construction of the application that is fragile and is hard to change. In addition, this gives us a space to use code smells i.e., calling some method of the component directly, which means that you are not testing component and its behavior it means that one is testing inner parts of the component.

So to sum up, testing a white box is a special method of checking the software that implies that the internal structure and technical features are thoroughly known to be checked.

Checking the white box consists of several complementary test types used to assess the ease of use of the web product, part of the code or special software functional.

Based on such testing, you can perform the following checks: unit tests, integration checks, system test components, checking the security architecture of software [9].

Traditionally, programmers are engaged in a similar type of testing, since for such inspections the specialist must have a high technical qualification.

Basic advantages of such a test method: optimization of the program code by searching for hidden bugs, creation of automated test cases, using the most suitable type of input data used for a qualitative testing process.

Grey-box testing. Grey-box testing – a special method of testing software with incomplete knowledge of its internal device. To perform a similar type of test, you do not need to have access to the source code of the software.

All tests are created based on simple knowledge of algorithms, architectures, and other high-level characteristics of the behavior of the product.

Types of grey box tests: regression inspections, matrix checks, template testing, check on using an orthogonal array, the main advantages of the method:

It has some features of the black and white box. In other words, the tester looks at the object of checking from the position of the black box, but analyzes the system with an accurate calculation of the data that it is pre-known (white box).

The QA specialist can create and apply more complex test scenarios.

This check allows the programmer to enlist sufficient time to correct bugs.

The programmer interacts with the tester at the initial level, which makes it possible to immediately remove unnecessary and excess test cases.

Disadvantages: analysis of the program code is limited since there is no access to the source code at the test, there is no time to test all the flows of input and output information, as it will take a lot of time., a situation is possible when testers can be superfluous (when not only the QA specialist but also the programmer checks its code with the help of unit tests).

Enzyme testing. So, for example using white box approach pushes us having Enzyme library in React JS application makes us think about the component meant to be tested as some ready isolated part of the application could be also called class or method. The library itself in connection with Jest framework knows about the component internals and could easily interact with the API of the component, so almost all convectional tests written with Enzyme considered black-box tests [10].

On the one hand it brings a lot of benefits to the platform and also essential ones covering all the main aims of the testing itself which are prevention of the bugs in code, unexpected behavior of the component, the ability to verify production code vulnerabilities, saves developer time by means of making one not bothering about the impact of the previous task and concentrate on the current one, also reduces the amount of work for the manual testers if present.

On the other hand, platform pushes the developer to write white-box tests, it also consumes a lot of resources involved into the testing process which leads us to the next issues that is performance.

One of the main issues is the maintainability as it was mentioned because platform problem is that the written test script for the component more than two or three functions influences the amount of script code and it grows exponentially.

Another issue present is that with the time it takes more and more time and money to maintain the tests which bounds developers from being rotated to another position, so it increases the cost of the project.

The one issue that is not being mentioned consists of two parameters which are the complexity of the tests as a separate module and inability to get deep into them if it wasn't written by the developer that investigates one. This is vital issue for newcomers that spend a lot of time just to understand what is going on with the code.

Also, it brings here another point which is immersive tools variety present in the library that requires high level of the writer of the code, so the person that develops scripts for this type of tests should have impressive understanding of the script writing techniques.

This leads us to the problem of the lack of the resources of the project because it starts to consume more and more resources being introduced into the testing process. It also exponentially increases the amount of money being spent so this type of testing is not the best decision despite on being black-box approach testing.

React testing library. There is present another choice for testing of the modern user web-application that is more effective and uses the whole another approach in testing.

This type of testing is meant to be black-box testing and gives a developer limited but wide range of the tools that could be used to write tests.

The main advantages of the react testing library despite being white-box testing library is that it is the fastest and the most maintainable way to write test scripts for now in the web-applications.

So black-box approach present in this type of tests pushes a developer to think about component from user perspective it means that it should be thought as what is done but not how.

Also, it gives tons of advantages in comparison to black-box testing present for now in React [11]. Namely, it has limited tools to be able to test with, so it is fully backward compatible, it takes away excessive work required to cover a component with tests, it is not meant to use only for React applications, that brings here one more advantage which is the most flexible approach to testing.

As well as mentioned above it is a good way to keep high level of the accessibility of the application, because if react testing library used tests of that could not be written in case of the bad code quality, so it also controls the code quality subconsciously. Here is one from the other consequences of using this approach is that a developer should not think about implementation details [12].

For example, the one trying to extract some component and test it this should be done by means of using some unique identifier but not text or styles as it is done in white-box approach. It makes tests way more flexible and maintainable because code could change and inner implementation of it as well, but the tests are going to still be valid. So it saves essential amount of time and resources including project budget in perspective.

Black-box testing uses whole different approach in tests implementation, and there are main ones: it deals with directly DOM nodes, and not platform specific components that are rendered into DOM node, increases the level of testing pyramid from only unit tests written with Enzyme.

So, to be mentioned one of the best advantages of this library which is easy to start and easy to dig into tests, so a developer with a superficially understanding of the code and tests could understand them without spending too much time on the investigation.

It makes it easier to write and maintain code, also saves vital amount of time and expenses that grow with the time as project expands.

Conclusions. Modern web-applications are indeed a place where all the type of the testing is vital for the high-quality product. On the one hand there are wide range of tools that could be used to pursue this goal by means of using best decisions present for now but on the other hand there are essential advantages and disadvantages present in these tools, which are all in the approach to it, so there are always cons and pros of using one or another.

For now, despite on having the ability to test using both black and white box testing it looks like the second one is not the best choice. There are several points on each board for both of them, but black box approach that is being represented with react testing library is more successful and efficient way to cover and application with high-level and low-level tests, that could be easily maintained and understood.

But white box testing is now the most used decision due to the historic development of the industry. So, it also has some great features and could be chosen to be used on the project, but it should be precise choice with the understanding of all the consequences standing behind.

References

1. Bentley J., Bank W., Charlotte N. C. *Software Testing Fundamentals – Concepts, Roles, and Terminology. Planning, Development and Support.* URL: https://support.sas.com/resources/papers/proceedings/proceedings/sugi30/141-30.pdf (accessed: 10.05.2022).
2. Jenkins N. A. *Software Testing Primer v.2.* OPENLIBRA, 2017. 55 p.
3. Myers G. J. *The art of software testing.* New York: Wiley, 2011. 256 p.
4. Gotel O., Cleland-Huang J., Hayes J., Zisman A., Egyed A. *Software and Systems Traceability.* London: Springer, 2012. 152 p.
5. Rubin K. S. *Essential Scrum: A Practical Guide to the Most Popular Agile Process.* Addison-Wesley Professional, 2012. 452 p.
6. Mike Cohn. *Succeeding with Agile: Software Development Using Scrum 1st Edition.* Addison-Wesley Professional, 2009. 512 p.

7. *Important Software Test Metrics and Measurements.* URL: http://www.softwaretestinghelp.com/software-test-metrics-and-measurements (accessed: 10.05.2022).
8. Crump, S. C. *Simplify Testing with React Testing Library: Create maintainable tests using RTL that do not break with changes.* London: Packt Publishing, 2021. 246 p.
8. Daniel Irvine. *Mastering React Test-Driven Development: Build rock-solid, well-tested web apps with React, Redux and GraphQL.* London: Packt Publishing, 2019. 496 p.
10. David Flanagan. *JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language 7th Edition.* London: O'Reilly Media, 2020. 706 p.
11. Carlos Santana Roldán. *React Design Patterns and Best Practices: Design, build and deploy production-ready web applications using standard industry practices 2nd Edition.* London: Packt Publishing, 2019. 350 p
12. Trevor Burnham. *Test-Driven React: Find Problems Early, Fix Them Quickly, Code with Confidence 1st Edition.* London: Pragmatic Bookshelf, 2019. 192 p.

References (transliterated)

1. Bentley J., Bank W., Charlotte N. C. *Software Testing Fundamentals – Concepts, Roles, and Terminology. Planning, Development and Support.* Available at: https://support.sas.com/resources/papers/proceedings/proceedings/sugi30/141-30.pdf (accessed: 10.05.2022).
2. Jenkins N. A. *Software Testing Primer v.2.* OPENLIBRA, 2017. 55 p.
3. Myers G. J. *The art of software testing.* New York: Wiley, 2011. 256 p.
4. Gotel O., Cleland-Huang J., Hayes J., Zisman A., Egyed A. *Software and Systems Traceability.* London: Springer, 2012. 152 p.
5. Rubin K. S. *Essential Scrum: A Practical Guide to the Most Popular Agile Process.* Addison-Wesley Professional, 2012. 452 p.
6. Mike Cohn. *Succeeding with Agile: Software Development Using Scrum 1st Edition.* Addison-Wesley Professional, 2009. 512 p.
7. *Important Software Test Metrics and Measurements.* Available at: http://www.softwaretestinghelp.com/software-test-metrics-and-measurements (accessed: 10.05.2022).
8. Crump, S. C. *Simplify Testing with React Testing Library: Create maintainable tests using RTL that do not break with changes.* London: Packt Publishing, 2021. 246 p.
9. Daniel Irvine. *Mastering React Test-Driven Development: Build rock-solid, well-tested web apps with React, Redux and GraphQL.* London: Packt Publishing, 2019. 496 p.
10. David Flanagan. *JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language 7th Edition.* London: O'Reilly Media, 2020. 706 p.
11. Carlos Santana Roldán. *React Design Patterns and Best Practices: Design, build and deploy production-ready web applications using standard industry practices 2nd Edition.* London: Packt Publishing, 2019. 350 p.
12. Trevor Burnham. *Test-Driven React: Find Problems Early, Fix Them Quickly, Code with Confidence 1st Edition.* London: Pragmatic Bookshelf, 2019. 192 p.

Received 13.05.2022

Відомості про авторів / About the Authors

Голян Наталія Вікторівна – кандидат технічних наук, доцент, Харківський національний університет радіоелектроніки, доцент кафедри програмної інженерії; м. Харків, Україна; ORCID: https://orcid.org/0000-0002-1390-3116; e-mail: nataliia.golian@nure.ua

Голян Віра Володимирівна – кандидат технічних наук, доцент, Харківський національний університет радіоелектроніки, доцент кафедри програмної інженерії; м. Харків, Україна; ORCID: https://orcid.org/0000-0001-5981-4760; e-mail: vira.golan@nure.ua

Афанасьєва Ірина Віталіївна – кандидат технічних наук, доцент, Харківський національний університет радіоелектроніки, доцент кафедри програмної інженерії; м. Харків, Україна; ORCID: https://orcid.org/0000-0003-4061-0332; e-mail: iryna.afanasieva@nure.ua

Golian Nataliia Viktorivna – Candidate of Technical Sciences (PhD), Docent, Kharkiv National University of Radio Electronics, Associate Professor at the Department of Software Engineering; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0002-1390-3116; e-mail: nataliia.golian@nure.ua

Golian Vira Volodymyrivna – Candidate of Technical Sciences (PhD), Docent, Kharkiv National University of Radio Electronics, Associate Professor at the Department of Software Engineering; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0001-5981-4760; e-mail: vira.golan@nure.ua

Afanasieva Iryna Vitayivna – Candidate of Technical Sciences (PhD), Docent, Kharkiv National University of Radio Electronics, Associate Professor at the Department of Software Engineering; Kharkiv, Ukraine; ORCID: https://orcid.org/0000-0003-4061-0332; e-mail: iryna.afanasieva@nure.ua