VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Lora Doykova

BENEFITS AND USAGE OF MANUAL AND AUTOMATED TESTING IN SOFTWARE DE-VELOPMENT

Tietojenkäsittely
2023

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Lora Doykova
Opinnäytetyön nimi	Manuaalisen ja automatisoidun testauksen hyöty ja käyttö ohjelmistokehityksessä
Vuosi	2023
Kieli	englanti
Sivumäärä	39
Ohjaaja	Antti Mäkitalo

Ohjelmistokehityksessä kiinnitetään tänä päivänä yhtä enemmän huomiota laatuun, jonka myötä ohjelmistotestauksesta on tullut yksi tärkeimmistä vaiheista ohjelmistokehityksessä.

Opinnäytetyön tarkoitus on lisätä tietoisuutta ja ymmärrystä manuaalisesta ja automatisoidusta testauksesta sekä niiden erilaisista tavoista ja vaiheista. Työssä käydään myös läpi niiden hyödyt ja haitat projektitestauksessa. Ensimmäisessä osassa tehdään pieni esittely testaukseen. Toisessa kappaleessa käydään läpi testausta yleisesti läpi. Kolmannessa kappaleessa syvennytään manuaaliseen testaukseen, sen eri tapoihin ja hyötyihin ja haittoihin. Neljännessä kappaleessa käydään manuaalisen kappaleen tavoin läpi automatisoitu testaus. Ja viimeisessä kappaleessa on hieman pohdintoja testauksesta asiakasprojekti näkökulmasta.

Lopussa käsitellään, millaista testaus on asiakasprojekteissa ja millaisia hyötyjä on huomattu sekä mikä olisi mielenkiintoinen jatkoaihe.

Avainsanat Robot Framework, automaatiotestaus, manuaalinen testaus

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittely

ABSTRACT

Author	Lora Doykova
Title	Benefits and usage of manual and automated testing in software development
Year	2023
Language	English
Pages	39
Name of Supervisor	Antti Mäkitalo

In software development, there is increasing attention being paid to quality, and as a result, software testing has become one of the most important stages in software development.

The purpose of this thesis is to increase awareness and understanding of manual and automated testing, as well as their different methods and stages. The advantages and disadvantages of project testing are also discussed. The first section provides a brief introduction to testing, while the second section covers testing in general. The third section delves into manual testing, its different methods, and its advantages and disadvantages. The fourth section covers automated testing in a similar manner to the manual section. The final section offers some reflections on testing from the customer project perspective.

In the end the thesis addresses what testing is like in customer projects, the benefits that have been observed, and what an interesting topic for further study might be.

Keywords Robot Framework, automated testing, manual testing

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

TERMS AND ABBREVIATIONS

API	Application programming interface
Appium	Software to design UI automation of applications
Azure DevOps	Software project management tool
Bug	Problem in the software
C#	Programming language
Database	Organized collection of structured information
E2E	End-to-End
Java	Programming language
JavaScript	Programming language
Open-source	Source code is available for everyone
PHP	Programming language
Python	Programming language
Query	Question or a request for information in format
Robot framework	Automation framework
Ruby	Programming language
Selenium Grid	Allows executions of WebDriver scripts on remote machines
Selenium IDE	Selenium Integrated Development Environment, open-source test automation tool that can record and playback actions on the web
Selenium Library	Web testing library
Selenium RC	Selenium Remote Control, testing framework that enables writing test in any programming language
Selenium WebDriver	Web framework that allows to execute cross-browser tests
User Interface	Contact between humans and computers

1 INTRODUCTION

In every software application there is something that needs to be improved or something that breaks or does not work as it should. To be able to fix the problem it must be found first. That is where testing comes along. The problem can be found in many ways, but two common ways are either with manual testing or with automated testing. These testing methods can be compared to users' perspective of using the application.

Software testing in general is greatly beneficial in software development because it helps to improve the quality of the product. It allows you to check if the product matches expected requirements and helps to identify bugs and unknows issues in the process better and quicker, for example before new release. (Yasar 2022.)

There are many ways and forms of testing, and this thesis examines manual and automated testing. The different types and methods of testing, how they work and the pros and cons of them. The situations in which to use which type and why are examined at the end.

Manual testing is when you manually test the software application while following a written test plan that specifies the steps and the outcome without any automated tool. The idea of testing is to find bugs, issues, or unusual behavior of the application. The benefit of testing the application manually is that it can catch both visible and hidden defects of the software application. (JavaTpoint.)

E2E testing means that the product is tested from the beginning to the end ensuring that the application meets the expectations as intended. Tests are executed with specific tools without any human interference. Automated testing is all about changing human activity with systems and devices. The main aim is to test the application with real user scenario from the user's perspective and making sure that all features and elements work properly together. (Katalon.)

The first chapter is an introduction to this interesting subject of several types of testing and their differences. The second chapter discusses different testing types and how they work. The third chapter will focus on manual testing. It has several types of testing and the advantages and disadvantages of it. The fourth chapter reviews automated testing and the chapter will go through the several types of testing and some tools but mainly focus on robot framework and E2E-testing. The thesis will end with conclusions.

2 TESTING IN GENERAL

Software testing in general is greatly beneficial in software development because it helps to improve the quality of the product. It allows checking if the product matches expected requirements and helps to identify bugs and unknows issues in the process better and more efficiently. Proper testing of the application ensures reliability and security, which is in the long run time saving, cost-effective and increases customer satisfaction. (Hamilton 2023.)

2.1 Testing methods

There are many ways to test a software, it can be on a unit level or then going through the whole application. Testing in software development is a very underestimated thing even though it has many benefits. Developers do not have time to test everything, and it is impossible for them to find and notice every bug. That is why there should always be a test specialist in the project. When there is one person focusing on testing the application and trying different variations of every functionality the odds of finding bugs in the alpha version are higher than without a test specialist. With this said there are plenty of testing methods, but below will be explained the most used methods.

2.1.1 Unit testing

"Unit testing is a type of software testing where individual units or components of a software are tested" (Hamilton 2023). To validate that each unit performs as expected the unit testing is usually done by the developer during the development. The benefit of unit testing is that when the new function is thoroughly tested, and the required fixes are made to it, it than making it part of a bigger ensemble in the software is safer. (Kasurinen 2013, chapter 4.) Unit testing can be done manually and automatically. If done manually the testing is done by a person who then tests the specific unit with all kinds of different scenarios. Automatically done there is used a Unit test framework where the section of code is then executed.

2.1.2 Integration testing

Integration testing is a type of software testing where units and components are integrated logically together and tested as a group. The purpose of this testing is to find bugs and defects in the interaction between these components. The testing is done by a tester. (Hamilton 2023). Usually, the starting point in integration testing is to think that in a fully functional ensemble there is added one part more and then we verify that the ensemble is still working as it should (Kasurinen 2013, chapter 4.)

2.1.3 Feature testing

Feature is defined in software development as an isolated functionality that performs its individual function to the existing software. Feature testing is testing the changes made to the software in the form of new features or modifications that they work correctly. It is used to validate the addition of new features and test its performance. (Buildd.)

The goal is to make sure that the new added or modified features meets all the specifications, functionality requirements and business requirements. Once merged the new feature has to work with the existing ones. (Buildd.)

2.1.4 Manual testing

Manual testing is a type of software testing where you have a manual test script that you must follow. Usually, the script is structured so that you test the application from beginning to end with an expected result. The test contains all kinds of scenarios so that the testing is done thoroughly. It may contain testing boundary values of an input field, the text that elements can contain and the interaction between elements. Application needs to be tested manually before deciding if it needs to be automated. Testing is done by the tester. Manual testing can be performed in many different ways depending on the wanted end result. (JavaTpoint.)

2.1.5 End-To-End testing

End-to-End testing is an automated way of testing software. The test is done and executed, for example using Robot Framework. To create the automated test there are ready made keywords or you can make your own keywords by using the ready keywords that for example Selenium library has already. Usually there is already a manual test case that is also made as an automated version.

E2E testing imitates real end-user scenarios that tests the application as a whole. Functions and components are ensured that they work properly and as expected together. (Katalon.)

3 MANUAL TESTING

Manual testing is the most primitive technique of software testing methods. When testing the software manually it means that the tests are executed manually without using any automated tools. Its purpose is to find bugs, issues, both visible and hidden defects in the software. Testing manually requires a lot more effort and time but it is necessary. Manual testing does not require any knowledge from the tested tool. “One of the Software Testing Fundamental is “100% Automation is not possible”. This makes Manual Testing imperative.” (Hamilton 2023.)

Test coverage should always be 100% but it requires maintenance and time. When a bug is found the developer fixes it and the tester re-tests the application to ensure it is working properly again. With this testing method the quality is checked, and a bug-free product is released to the customer. (Hamilton 2023.)

Manual testing needs a lot of patience and creativity. Because the application is used by humans it is vital to have humans also testing it because the application must always be tested from end users’ perspective.

3.1 Manual testing types

There are several types of manual testing. In this chapter we will cover some of the types such as black box testing, white box testing, grey box testing, alpha testing, beta testing and exploratory testing. These are types of testing that all have different purposes and wanted results. Unit testing and feature testing can also be done manually but those will be covered in the next chapter because they are more commonly used as automated tests.

3.1.1 Black box testing

The software is tested in a way where you just give inputs and wait to see how the software reacts without paying attention to what happens inside the application. Usually, the testcases have all the information on how to use the application and what is expected from it and the steps and inputs for the testing. This testing method can be used at any time of the testing process. So, it basically covers all testing levels and its' work steps from the beginning. When doing black box testing it is important for the inputs to ensure that the tested functionality is working as expected in every situation possible there is. This is why the test should contain a lot of different scenarios and combinations. **Figure 1.** reflects the black box testing process. (Kasurinen 2013, chapter 4.)

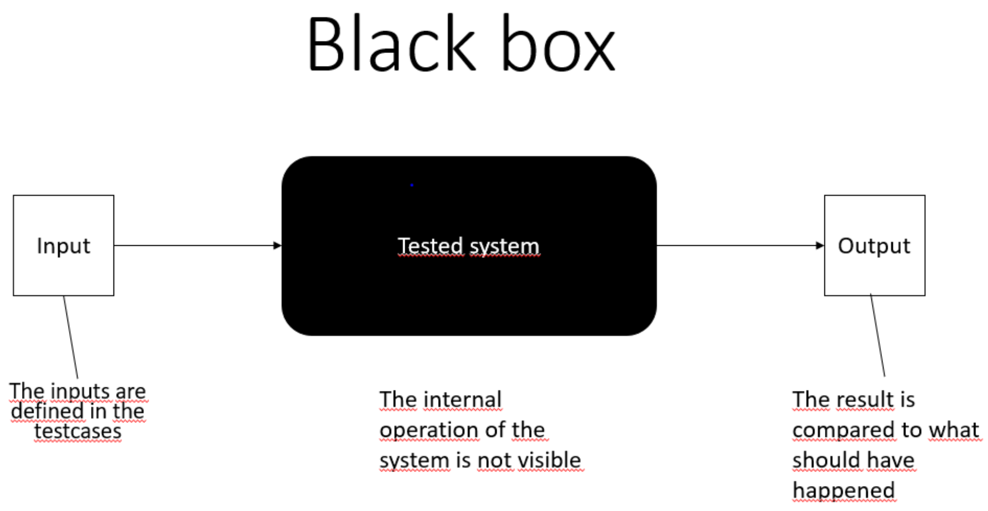

Black box
Input
Tested system
Output
The inputs are defined in the testcases
The internal operation of the system is not visible
The result is compared to what should have happened

Figure 1. Black box

3.1.2 White box testing

This is like a complete version of black box testing. There are inputs given to see how the application reacts but also to see what is happening inside the application and if it is working as intended. With this testing method the tester can see what exactly is happening inside the application and can track the issue down to source code level. In other words, white box testing is a deeper and more precise way of testing applications than black box testing. **Figure 2.** reflects the white box testing process. (Kasurinen 2013, chapter 4.)

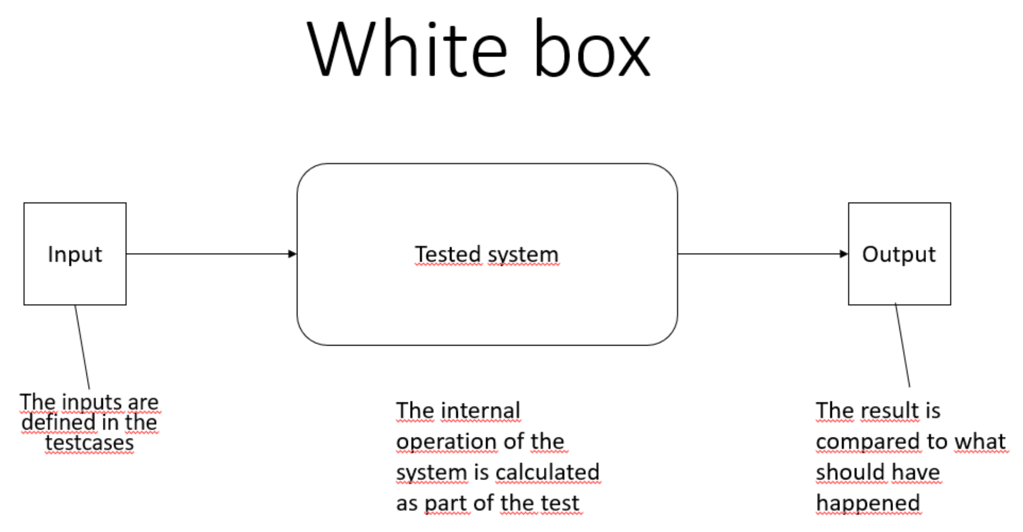

White box
Input
Tested system
Output
The inputs are defined in the testcases
The internal operation of the system is calculated as part of the test
The result is compared to what should have happened

Figure 2. White box

3.1.3 Grey box testing

When using grey box testing you combine the best parts of black box and white box testing. The basic idea is to go through the test cases made in the requirements definition with the ability to view the system from inside too. With grey box testing it is tried to test model and test coverage and that all the requirements are met in the system. It is a great method, for example in situations where you cannot test the system comprehensively on white box level, but all the tested components are accessible. (Kasurinen 2013, chapter 4.)

3.2 Alpha and Beta Testing

In software development, the concept of alpha and beta testing is often associated with classic waterfall design models in which one development phase must be completed before the next one can begin. (Semilof 2021.)

Alpha testing can be said to be the first step of testing the application as whole. In the alpha version all the functionalities are completed but there is no certainty that they are working as intended so that is why the alpha version must be tested to see if the application is steady enough to be released to beta version. (Kasurinen 2013, chapter 4.)

Beta testing is done by the end-user to validate the product. for functionality, reliability, compatibility, and usability. It is one of the acceptance testing types and because the end-user does it adds value to the products. (Software Testing Help 2023.)

3.2.1 Alpha testing

Alpha testing is performed early in the development process, usually by internal staff. It is the phase where the new product is tested to see if it performs as expected. Alpha testing is basically white box testing. The inner workings of the technology itself are tested during alpha testing. The purpose is to find major bugs and feature flaws that are critical to the use of the application. Alpha testing is often done by using testing environment.(Semilof 2021.)

Alpha testing provides early feedback of the software in the early phase of the process that can be used to improve the software. It also helps find issues and resolve them. It reduces the risk of major problems at the later stage. Overall, it helps to ensure that the released product meets its expectations. (Semilof 2021.)

3.2.2 Beta Testing

Beta testing is a pre-release type of acceptance testing which is performed by the end-user. It is done in a production environment to uncover bugs or other issues regarding the functionality of the application before the general release. This testing method is black-box type, and it does not require a testing environment because the testing is done on a functional version of the product. When testing is done by the end-user it provides a complete overview of the experience while using the application. With the help of Beta testing, it is validated that the users are happy with the product and gathers feedback for users that can help improve the application. (Software Testing Help 2023.)

With real end-users testing it provides valuable feedback that helps to solve the issues and even find them. By involving customers in the testing, it builds trust, loyalty and excitement over the new product that then increases customer satisfaction. Beta testing reduces the risk of post-release costly fixes if there happens to be any, because the product is tested by a wide audience. (Software Testing Help 2023.)

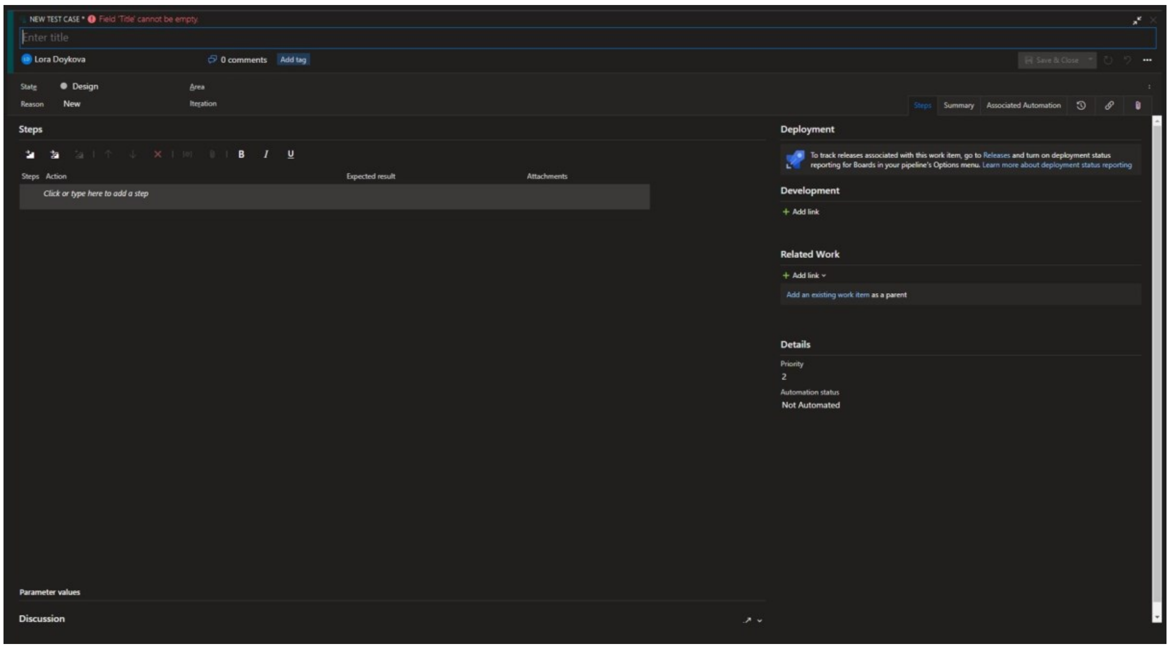
3.3 The structure of manual test case

When creating a manual test case, it is usually quite simple to do it. The test case should define the steps and the expected results clearly so that anyone who might use the test case knows how to execute it.

What comes to the structure of the test case it should always have a brief, descriptive title that summarizes what the test case is about. If creating test case in Azure DevOps, which is a set of tools offered by Microsoft that supports the software development lifecycle, including planning, coding, building, and testing, the test case ID is generated automatically by Azure as you can see from **Figure 4.** when comparing it to **Figure3**. Then when it comes to the test steps there is the action itself and then the expected result from the action. They should be clear enough to the executer that even someone who is not familiar with the application should know how to execute them. Each action has its own result so that should also be clear what is expected to happen after the action or what should be seen. (Microsoft 2023.)

The test case can also be linked to another existing backlog item or task. Test can also be tagged with wanted tag, for example if it is automated manual test case you can set “automate” tag to it or if you have query where you want to include the test there is usually defined tag in the query so you can use it in the test so it will appear in the query then. (Microsoft 2023.)

When doing manual testing you usually execute them by testing some kind of application or software. So, there is no need for any specific tools to practice this kind of testing. You test what is necessary depending on what is the expected result and include or exclude things while doing it but if you have clear structure and test case it has to be followed for reliable results. As you go on and notice some kind of issues or bugs in the system you are testing, you report it. First to the team and afterwards create a bug ticket for it so that it is marked up in the backlog ready to be fixed.

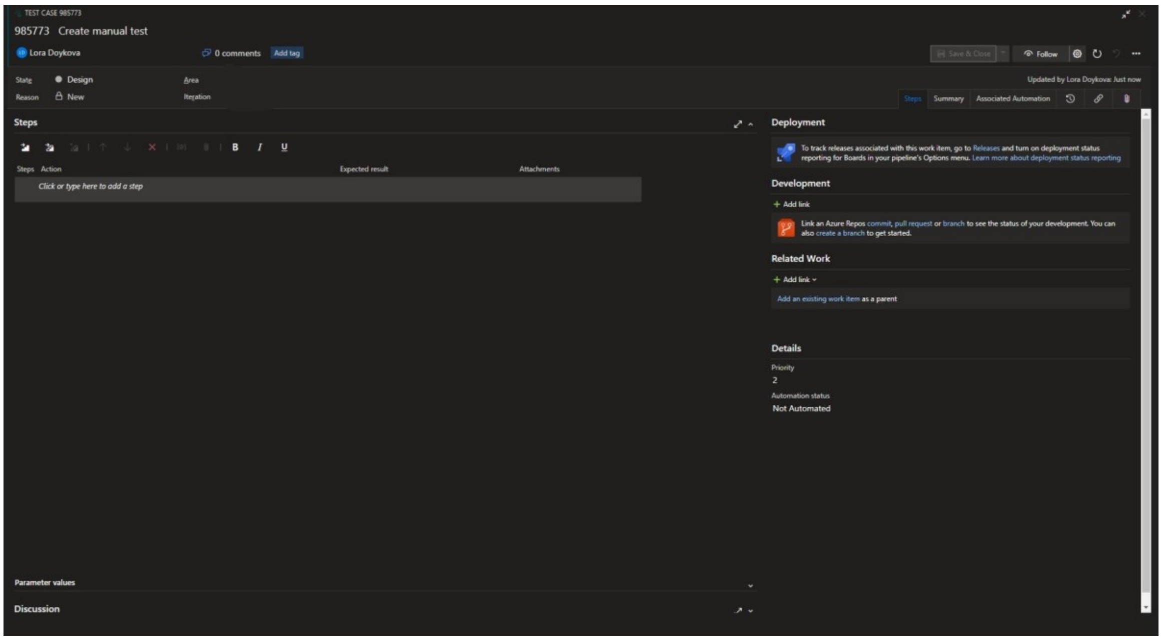


State: Design
Reason: New
Area
Iteration

To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting

Priority: 2
Automation status: Not Automated

Figure 3. Azure DevOps manual test case without id



985773 Create manual test

State: Design
Reason: New
Area
Iteration

To track releases associated with this work item, go to Releases and turn on deployment status reporting for Boards in your pipeline's Options menu. Learn more about deployment status reporting

Link an Azure Repos commit, pull request or branch to see the status of your development. You can also create a branch to get started.

Priority: 2
Automation status: Not Automated

Figure 4. Azure DevOps manual test case with id

3.4 Exploratory testing

Exploratory testing is a type of software testing where testers do not work on any previously created test cases. The test cases are designed in the fly while exploring the software. It “helps verify that the system works as expected and is an easy and pleasant user process” (Bose 2023.). In this method it is tried to mimic the end-user’s freedom and curiosity. Exploratory testing is all about discovery and investigation. While testing the tester makes spontaneous decisions during the exploring for example, which features and actions to test and with what kind of values.(Bose 2023.)

The goal is to find as many issues and bugs as possible in the short time of testing while also gaining understanding of the software. With this type of testing, it is more likely to find more defects than with traditional manual testing. Also, anyone in the team can execute this kind of testing, no matter if you are tester, developer, or end-user.

3.4.1 Types of exploratory testing

There are a few different types of exploratory testing. They can be divided into three different testing types based on the wanted outcome or structure of the testing. There is freestyle exploratory testing, scenario-based exploratory testing, and strategy-based exploratory testing. (Bose 2023.)

Freestyle exploratory testing has no rules, structures, or organization. Testing is done by testers just for them to go through the application quickly using whatever scenarios they want. The purpose of this is to do it quickly and investigate bug defects or verify previously done testing. (Bose 2023.)

Scenario-based exploratory testing is based on real user scenarios. Tester goes through each scenario by exploring the software in all possible ways to try and match the scenario. The goal is to test as many scenarios as possible to achieve maximum test coverage. (Bose 2023.)

Strategy-based exploratory testing is done by someone who is already familiar with the software that is being tested. It includes a deeper way of testing. It includes boundary value analysis, equivalence technique and risk-based technique to identify more challenging bugs and issues. (Bose 2023.)

3.4.2 Advantages and disadvantages of exploratory testing

By testing the software and its functionalities without any pre-made tests it is more likely to uncover defects that were missed in scripted testing. Communication and feedback are better because testing starts right away without any formalities. Exploratory testing is very flexible, and it allows testers to adapt to the test approach depending on their finds. It also helps gain better understanding of the system and user experience.(Software Testing Help 2023.)

Because testing is done without any scripts it makes it hard to replicate the steps when a bug is found. So, lack of repeatability is the biggest disadvantage, making it hard to track the bugs and verify fixes. Testing is based on users' knowledge and skills, so it limits quite a bit the coverage and leaves some areas untested.

3.5 Conclusion of manual testing

Overall manual testing is a particularly important and valuable part of the testing process. It provides a human touch with their knowledge and experience to identify bugs. It is also flexible because you can include or exclude things as you test the application. When bugs are detected in the early stage of development it is cheaper and easier to fix so in that sense it is also cost-effective.

When an application is tested manually it offers flexibility in the test scenarios. There is the possibility to modify and add things as you go. Also, you can test the application wider and from different aspects. With the human touch the manual testing is comparable to the end-user experience. Also, when a human does the testing, they can use their knowledge and expertise about the application to detect bugs and issues. (Da Silva 2022.)

Manual testing does require a lot of hours to manually check every feature and function of the application. It takes more time than automated testcases because those can be set to run overnight so it is also costly in its own way to perform manual testing. There is always room for human errors when doing manual testing because of lack of focus and distraction. Maintaining test cases is also very time consuming. Also, the tester has to have a good knowledge of the tested software to guarantee good coverage. (Da Silva 2022.)

4 AUTOMATED TESTING

Automation testing is a software testing method where there is a special automated testing software tool used to execute the test case suites that are once written by tester. By using automated testing, the testing process is more effective and time saving. It speeds up test executions and helps increase test coverage. (Hamilton 2023.)

Automated test cases are reusable and with them the testing process is consistent. When tests are run, for example every night the bugs and issues are detected in early phase and fixing them becomes easier. Once the test is automated it does not require human intervention. Only if something changes in the UI for example then the test needs to be updated. It also required fewer human resources than manual testing. (JavaTpoint.)

4.1 Testing methods

As there was mentioned before in the manual testing chapter that there are many different testing types and like in manual testing, they are almost the same in automated testing. The most common ones are integration -, unit -, regression – and performance testing (Ranorex 2022.)

4.1.1 Integration testing

Integration testing is a method where different components in the application are tested together to see if they work as intended. It involves use of automated tools to simulate the interactions between the modules and detect any issues. With this method it is ensured that the individual modules work as intended together without causing unexpected behavior. (Liu 2018.)

Testing is done by creating test scripts that replicate different scenarios and interactions between modules. These steps are then automated by using testing tools that then run the test automatically and generate a report if any issues are detected. (Liu 2018.)

"In this workflow, we have test input data that is provided to system component A, which then interacts with components B and C. The result of the output is then verified with the expected output" (Liu 2018.). This is demonstrated in **figure 5.**

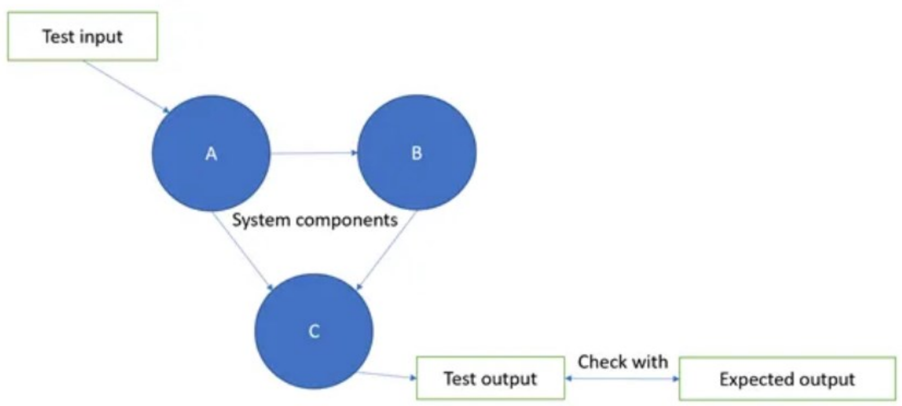


Figure 5. Integration testing example (Liu 2018.)

4.1.2 Unit testing

Automated unit testing is a method to test individual units or components of a software. Small sections of code are closely tested to ensure they work as expected. The goal is to validate that each small unit of a larger software project works as expected. Unit tests are made during the development phase by the developers.

A unit may be an individual object, procedure, module, method, or function. It is done before integration testing, and it is classified as white box testing. Properly done unit testing in early development is guaranteed to save money in the end. Unit tests also provide information to the developers to understand the code base. They are also counted as documentation. (Hamilton 2023.)

To execute the unit test, you need Unit test framework. The section of code to test a specific function is written by the developer and then executed with the help of the framework. When tested and it is ready to be merged the code is removed. The code can also be isolated to test the unit more rigorously. It helps reveal unnecessary dependencies between the code that is tested and other units. (Hamilton 2023.)

Test driven development is a methodology in software development where the emphasis is placed on writing test cases before the actual feature or function is written. It combines building and testing. It helps to ensure the correctness of the code and indirectly evolve the design and architecture of the project. (Test-Driven.io.)

There are a few steps that are followed when performing test driven development. First you add the test to the suite. Then you execute all the tests and ensure that the new one fails. Then we write just enough code for that one single test to pass. Then we ran all the tests again. After that refactoring to keep all the tests passing. The process is then repeated when a new test comes along. This is called "Red-Green-Refactor" cycle that is also shown in **Figure 6.** (TestDriven.io.)

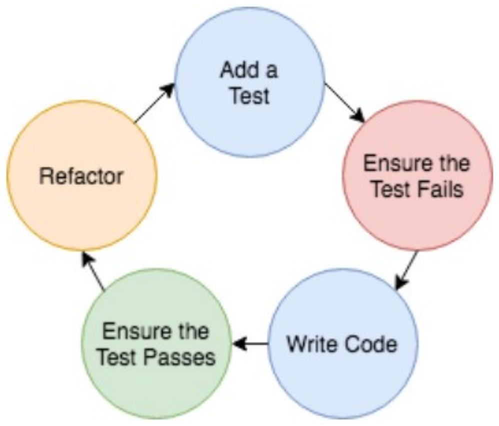


Figure 6. Test driven development cycle (TestDriven.io.)

4.1.3 Regression testing

Regression testing is a type of software testing that re-runs functional and non-functional tests to verify that the software works as expected after any code changes, updates, or improvements. It provides stability by checking the functionality of existing features to ensure that they can withstand frequent improvements. (Kanade 2022.)

Functional testing is a type of testing that verifies that every function of the software works in accordance with the required specifications. It mainly involves black box testing. Every functionality of the system is tested by providing inputs, verifying outputs, and comparing the results with expected results. In this type of testing, it is focused on user interface, APIs, database, and security. (Hamilton 2023.)

Non-functional testing is a type of software testing where non-functional aspects as performance, usability and reliability of a software are checked. Its focus is to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing. (Hamilton 2023.)

4.1.4 Performance testing

Automated performance testing is a non-functional testing type that checks the quality of the software under a particular workload by leveraging automation. It tests the software's speed, response time, reliability, resource usage, and scalability. With the help of automation all this is done faster than manually. (Kinsbruner 2020.)

The reason performance testing is important in the development process is because with this type of testing it can be ensured that the software meets the expectations as well as delivering positive user experience. This method can be cost prohibitive but continuous performance testing is the key to the success of an effective strategy. (Tricentis.)

Performance testing has many types, but the most important ones are load, stress, soak, and spike tests. Load testing simulates the application's response time when there are several users using the application at the same time and still operating at a high level of performance. Stress testing evaluates the behavior of the system when reaching peak activity when users are greatly exceeding expectations. Soak testing evaluates the performance of an application when there is high traffic for an extended period. It evaluates if intense and sustained activity shows a drop in performance levels. Spike testing compared to stress testing tries to understand the operation of the system when the activity levels are above average. (Tricentis.)

4.2 End-to-end testing

End-to-end testing is a software testing technique that verifies the performance and functionality of the entire software application from start to finish. It is done by creating real-world user scenarios. Its purpose is to identify bugs and issues that arise when all the components are working together. By doing E2E-testing we ensure that the application functions and delivers expected outputs when tested as a whole. (Katalon.)

There is a clear need nowadays to test the whole application from start to finish. Software today is becoming more complex. Applications are built on entire networks of subsystems and layers, including user interface and application programming interface and databases. All these must work together, they might work individually but conflict when connected, if one fails so does the entire product. E2E testing should be done in both manual and automated ways to maximize the findings and test coverage.(SmartBear.)

Performing E2E testing has many benefits in software development. It will validate your software functionality at every level from front-end to back-end. It also provides perspective on its performance across different environments. Bug detection and issue findings are easier to fix when found in development face and not in production. E2E testing improves the quality of the application and the workflow. Because automated E2E testcases can be set to run for example overnight there is no need to use time to run the tests over by human so the fails and changes will still be caught, and time and money is saved at the same time. The front end is also verified ensuring that the application works as intended across a wide range of browsers, devices, and platforms. (SmartBear.)

The lifecycle of E2E software testing consist of four steps that are test planning, test design, test execution and result analysis. When the integration test is done, and the components are tested to ensure that they work as expected we decide which all functions and object should be included in the E2E test. The test environment is then chosen and set up based on the requirements. Manually when executed it can be expected to use the alpha version of the tool. Automatically run you can use, for example robot framework to make and execute the tests. Test cases are then executed remotely and locally. After the tests have run the results are analyzed and the root causes of the bugs, if any, are solved. Then everything is reported to the team for improvements. (Katalon.)

4.2.1 Types of E2E testing

There are two types of E2E testing, horizontal tests, and vertical tests. Horizontal testing focuses on ensuring that each individual workflow works correctly. With this type of testing the testers set themselves in the position of the end-users and want to experience it from their aspects. (Katalon.)

Vertical testing is a more technical approach. In this the testers do not only focus on frontend but on backend as well. Testers ensure that the right data is transferred to the right place at the right time. Verifying that the background processes keep running so that the application can be executed smoothly. (Katalon.)

4.3 Advantages and disadvantages of automated testing

Using test automation in software development has been seen to be very beneficial. Because testing is a continuous process you will have to be testing the application every time something changes. It means enormous testing effort and higher cost if done manually. So, if tests are automated, they run automatically and repeatedly with no additional cost and at the same time catching all the fails and bugs. With automation it is also possible to test applications of any size, end-to-end, in just a few minutes or hours. It is a much faster way of executing a test. This enables us to test more features which means enhanced coverage and higher quality. When the test coverage is higher it means that bugs and problems are found in early phase which means that fixing them is easier and user experience gets better. (Avo Automation 2021.)

With test automation it can be ensured that the steps will be executed correctly. When tested manually there is always room for errors, steps can be missed or data input wrong, but automation takes that stress away with expected result every time. Communication and feedback also improves and increases between testers and developers resulting in faster bug fixes and excellent customer experience. (Avo Automation 2021.)

Everything always has its disadvantages and so does automated testing. Developing automated test cases can take longer than manual test cases. If the tests are hard to maintain or too complex it affects the quality of the test suites. One of the biggest cons in automate testing is that it takes a significant amount of time and money to implement and maintain. Tests must be updated every time there is a change for the tests to pass and the results to be reliable.(JavaTpoint.)

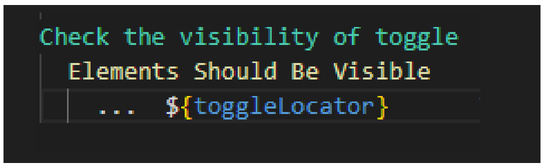
4.4 Different test automation tools

When deciding what test automation tool to choose for your project there is a few things to consider. The tools has to meet your need so think what tests you need to automate. Do research on the tools to make it easier to choose what is suitable for you. Here is a few examples.

4.4.1 Robot framework

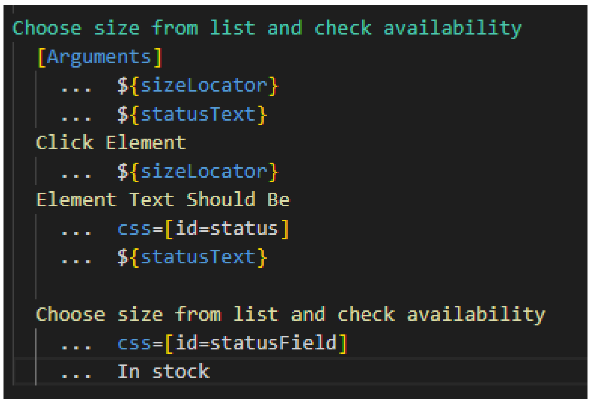
Robot framework is a Python based open-source test automation framework that uses a keyword-driven approach. It has easy syntax with human-readable keywords. The code aims to be easily readable so that anyone can understand what the code does. Its capabilities can be extended with libraries. (Robocorp 2022.)

Keywords are the operations that the robot executes to accomplish various tasks and are the foundation of any robot script. They are re-usable and can be composed which means that there can be keywords that are calling other keywords. Robot framework uses tabular syntax which is very important because the indentation tells the framework Check the visibility of toggle is a keyword with another keyword in it taken from Selenium library as shown in **figure 7.** The toggle locator is stored as a variable. Keywords can also require one or more arguments. Arguments are a way of providing more information to a function so with argument you pass data to the function. In **figure 8.** the keyword is expecting two arguments. When the keyword is used you only give the arguments to it. The indentation is also required between the keyword and the arguments and between the arguments itself. You can also set a variable as an argument. Variables are like containers that hold information. They are used to store information and referenced in computer programs. With variables you can label the data with a more descriptive name. This stored data can be used throughout the program. (LaunchSchool; Robocorp 2022.)



```
Check the visibility of toggle
    Elements Should Be Visible
    ...    ${toggleLocator}
```

Figure 7. Keyword structre



```
Choose size from list and check availability
    [Arguments]
    ...    ${sizeLocator}
    ...    ${statusText}
    Click Element
    ...    ${sizeLocator}
    Element Text Should Be
    ...    css=[id=status]
    ...    ${statusText}

    Choose size from list and check availability
    ...    css=[id=statusField]
    ...    In stock
```

Figure 8. Keyword with arguments

The framework uses extensions so that it can be easily extended to work with any target systems because it is not built to interact with one specific system. These extensions are called libraries. Robot framework comes with built-in support for a wide range of test libraries that can handle common cases. The libraries come in packages that you must add for them to work with your robot. Once the library is added you reference it in your script. When the library is added you get access to all the keywords it contains. (Robocorp 2022.)

Robot framework's ability to generate easy-to-read reports including logs and html reports makes it easy for testers to figure out where and why the test is, for example failing and just to understand the results of test execution. It can also run on Windows, Linux and macOS. (Robocorp 2022.)

4.4.2 Selenium

Selenium is a suite of tools that is used in cross-browser testing. It cannot automate desktop applications; it can only be used in browsers. It is one of the most preferred tools suites for automated testing. Selenium offers compatibility with many different programming languages C#, Java, JavaScript, Ruby, Python and PHP. So, it is flexible for users to use which language they prefer themselves compared to cypress for example that uses only JavaScript. (Unadkat 2023.)

Selenium contains four main components, Selenium Integrated Development Environment, Selenium Remote Control, Selenium WebDriver, and Selenium Grid. Selenium Integrated development environment is a record/run tool. It is an add n or an extension for both Firefox and chrome that generates tests quickly. Selenium Remote control allows you to develop responsive design tests in any language you prefer. But you must have a good knowledge of the chosen language. Libraries and servers are the two main components of Remote Control, and it has complex architecture and restrictions. Selenium WebDriver is a web framework that permits to execute cross-browser tests. It is used for automating web-based testing. It is an enhanced version of Selenium Remote Control. It was made to overcome the limitations faced in Remote Control. Even though it is just an advanced version of the Remote Control it still has a completely different architecture, and it supports multiple different programming languages too. Selenium Grid is used for concurrent execution of test cases on different browsers, machines, and operating systems simultaneously. It makes cross-browser compatibility testing very easy. (Unadkat 2023.)

4.4.3 Appium

Appium is an open-source, cross-platform test automation tool for mobile web, native, hybrid and desktop apps. It was initially created to automate iOS and android mobile applications, but it has been improved to a full-featured platform that provides WebDriver based automation possibilities for different mobile and desktop platforms. “Appium is mobile application testing tool that is currently trending in mobile automation testing technology” (JavaTpoint). The demand for mobile applications is high even more websites are converted to mobile apps. (GitHub.)

Appium is similar to Selenium WebDriver so if Selenium is familiar Appium is then easy to learn. It supports several programming languages such as Java, PHP, C#, Python, JavaScript with node.js and Ruby and many more that have Selenium libraries. Appium does not require source code or library and it allows us to run the same test on multiple platforms. (JavaTpoint.)

4.4.4 Cypress

Cypress is front end testing tool. It is a free and locally installed application that also has Cypress cloud to record your tests. It can test anything that runs in a browser and enables to write E2E-, component -, integration – and unit tests. The most typical users are developers and quality assurance engineers who use JavaScript frameworks. This testing tools mission is to build open-source ecosystem that enhances productivity, speed and quality without stress and anxiety for the developers. (Cypress 2023.)

Cypress has its own features that no other testing framework has. It takes snapshots as the tests are running. By hovering over commands in the log you can exactly what happened on each step. Their readable errors and stack traces makes the debugging very fast. Automatic waiting, cypress adds automatically waits or sleeps to the tests for commands before moving on. You can verify and control the behavior of functions, server responses and timers. (Cypress 2023.)

Cypress is not Selenium-based tool as most of the end-to-end testing tools are. The difference is where Selenium executes remote commands through the network, cypress runs in the same run-loop as the application. Cypress focuses on one thing, and it is end-to-end testing. Cypress tests anything that runs in a browser, and it is built to handle modern JavaScript frameworks. Tests are written using only JavaScript. When using Cypress there is no need to install many different tools and libraries, everything comes in one. (Cypress.)

4.5 When to automate test cases?

100% automation is never recommended. Manual testing should not be left out completely. This is why there is few things to consider when deciding should the test be automated or not.

When tests needs to be run repeatedly it is best if they are also automated. It is more time efficient than running them manually many times. It gives out more reliable results and won't lead to as many errors as manual testing can because of lack of attention. Also, if the test is too complex or time consuming it is better to automate it because it reduces the time and effort required to run the tests and gives time to do something else instead. (Lozowska-Pereira 2022.)

4.6 Conclusion of automated testing

Automated testing can be very beneficial in the long run. Even though it is a bit costly it has been proved to be a positive addition to the software projects. When tests are automated there is no chance for errors and because of that the results are reliable. Thanks to the automation the test can run while you focus on something else. It has a positive impact in time management and the application can be tested more thoroughly when there is time to do manual testing at the same time.

5 CONCLUSION

This thesis studied manual and automated testing in software development. It covered many different ways of testing types and frameworks and their usage and benefits. I have worked as a software tester for two years now, so I am familiar with the testing process in a software development project. Writing this thesis has been interesting and very instructive to myself and will definitely help me develop as a tester.

It is proven by this thesis that testing the software has a major impact and benefit in its quality. The testing has to be done in every step of the way to ensure that end-users experience is smooth as possible. From my experience as a tester, I can say that testing separately first the units and features by developer and after that implementing the E2E-testing to it has had a major impact on the product. The whole team has noticed that when the E2E-test is started to implement, bugs have been found - and that is what is wanted in the end. To find every issue and bug before it gets released. Also, when a new beta version is released, the verification to it is done before it is validated to be released. The number of bugs and issues found still in beta version after all the testing that is done before is incredible. However finding them and getting them fixed before the release is what guarantees the best possible quality. It's not said that the product is bug free when released but everything is done to at least to find all the most critical ones so that the user can enjoy the application without any major issues.

In the project I am working on every newly implemented feature gets tested. Usually, it is done first manually and then with the help of robot framework it is automated. My focus is on the E2E-testing, so I test the feature that was implemented, how it works with the whole application. But there is exceptions, not every test gets automated. Usually, the automated ones consists of a lot of repetition and usage of many features. Those kinds of tests are very beneficial to be automated so that we can see every day from the results if something is failing. The ones that stay manual are the ones where there is not a lot of inputs, or the feature is used so many times in other tests that it still gets tested properly.

Improvements from the perspective of the project I am working in is documentation. There are situations where no one else except the tester knows what has been tested and what not and how the results look. There should be a clearer way of communicating about it. The thing that have come to mind is that should the areas that are covered with testing be listed somewhere but is that then clear enough. Of course, the items are linked to each other, but it is still clear to everyone. Daily meetings are held where the issues and what has been done are discussed. There the result of, for example of nighty build can also be reported but that still does not give a very clear picture of how the results look to the rest of the team and there is always a little question mark about the covered areas.

Next interesting thing to look into could be what will the benefits be if the testing of the software was outsourced. Would it affect in the quality positively or negatively? It could be viewed from such perspective that the result from the internal testing to the outsourced testing could be compared.

REFERENCES

Avo Automation. 2021. 10 Benefits of Test Automation. Referred 11.5.2023. https://avoautomation.ai/10-benefits-of-test-automation/

Appium GitHub. Referred 11.5.2023. https://github.com/appium/appium

Basic concepts of robot framework. Referred 7.5.2023. https://robocorp.com/docs/languages-and-frameworks/robot-framework/basics

Bose, S. 2023. Exploratory Testing: A Detailed Guide. Referred 6.5.2023. https://www.browserstack.com/guide/exploratory-testing

Buildd. What Is Feature Testing. Referred 6.5.2023. https://buildd.co/marketing/feature-testing

Cypress. Why Cypress? Referred 11.5.2023. https://docs.cypress.io/guides/overview/why-cypress

Cypress. 7 Ways Cypress Is Different. https://www.cypress.io/how-it-works/

Da Silva, M. 2022. Pros And Cons Of Manual Testing. Referred 25.4.2023. https://uilicious.com/blog/pros-and-cons-manual-testing/

Hamilton, T. 2023. Integration Testing: What is, Types with Example. Referred 6.5.2023. https://www.guru99.com/integration-testing.html

Hamilton, T. 2023. Manual Testing Tutorial: What is, Types, Concepts. Referred 7.5.2023. https://www.guru99.com/manual-testing.html

Hamilton T. 2023. What Is Software Testing? Definition. Referred 13.10.2022. https://www.guru99.com/software-testing-introduction-importance.html.

Hamilton, T. 2023. What Is Automation Testing? Test Tutorial. Referred 2.5.2023. https://www.guru99.com/automation-testing.html

Hamilton, T. 2023. Functional Testing Vs Non-Functional Testing – Difference Between Them. Referred 4.5.2023. https://www.guru99.com/functional-testing-vs-non-functional-testing.html

Hamilton, T. 2023. Unit Testing Tutorial: What Is, Types and Test Example. Referred 6.5.2023. https://www.guru99.com/unit-testing-guide.html

JavaTpoint. Automation Testing. Referred 2.5.2023. https://www.javatpoint.com/automation-testing

JavaTpoint. Manual Testing. Referred 11.10.2022. https://www.javatpoint.com/manual-testing

JavaTpoint. Appium tutorial. Referred 11.5.2023. https://www.javatpoint.com/appium

JavaTpoint. Advantages and Disadvantages of Automated Testing. Referred 10.5.2023. https://www.javatpoint.com/advantages-and-disadvantages-of-automated-testing

Kasurinen, J. (2013). Ohjelmistotestauksen käsikirja. Jyväskylä: Docendo

Kanade, V. 2022. What Is Regression Testing? Definition, Techniques and Tools. Referred 3.5.2023. https://www.spiceworks.com/tech/devops/articles/what-is-regression-testing/

Katalon. What is End-to-end (E2E) testing? E2E Testing Full Guide. Referred 13.10.2022. https://katalon.com/resources-center/blog/end-to-end-e2e-testing

Kinsbruner, E. 2020. How To Shift Automated Performance Testing Left. Referred 9.5.2023. https://www.perfecto.io/blog/automated-performance-testing

Launch School. Variables. Referred 7.5.2023. https://launchschool.com/books/ruby/read/variables

Liu, Z. 2018. Automated Integration Testing. Referred 2.5.2023. https://medium.com/@allenliuzihao/automated-integration-testing-a295d21e513a

Lozowska-Pereira, M. 2022. Why And When To Use Automation In Testing. https://www.netguru.com/blog/automation-in-testing-importance

Microsoft. 2023. Create manual test cases. Referred 2.5.2023. https://learn.microsoft.com/en-us/azure/devops/test/create-test-cases?view=azure-devops

Ranorex. 2022. Types of Automated Testing. Referred 8.5.2023. https://www.ranorex.com/blog/types-automated-testing/

SmartBear. What Is End-to-End Testing? Referred 8.52023. https://smartbear.com/learn/automated-testing/what-is-end-to-end-testing/

Semilof, M. 2021. alpha testing. Referred 3.5.2023. https://www.techtarget.com/searchsoftwarequality/definition/alpha-testing

Software Testing Help. 2023. What Is Beta Testing? A Complete Guide. Referred 1.5.2023. https://www.softwaretestinghelp.com/beta-testing/

Software Testing Help. 2023. What is Exploratory Testing in Software Testing (A Complete Guide). Referred 6.5.2023. https://www.softwaretestinghelp.com/what-is-exploratory-testing/

TestDriven.io. What Is Test-Driven Development. Referred 10.5.2023. https://testdriven.io/test-driven-development/

Tricentis. Performance Testing: best practices, metrics and more. Referred 9.5.2023. https://www.tricentis.com/learn/performance-testing

Unadkat, J. 2023. Selenium WebDriver Tutorial: Getting Started With Test Automation. Referred 10.5.2023. https://www.browserstack.com/guide/Selenium-webdriver-tutorial

Uilicious. The Advantages of Using Manual Testing over Automated Testing. Referred 25.4.2023. https://uilicious.com/blog/pros-and-cons-manual-testing/#the-advantages-of-using-manual-testing-over-automated-testing

Yasar, K. 2022. Software testing. Referred 13.10.2022. https://www.techtarget.com/whatis/definition/software-testing