Czech University of Life Sciences Prague

Faculty of Economics and Management

System Engineering and Informatics

Bachelor Thesis

The Effectiveness of the Latest Methods of Software Testing

Anıl Tuncay

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

ANIL TUNCAY

Informatics

Thesis title

The Effectiveness of the Latest Methods of Software Testing

Objectives of thesis

The objective of this thesis is to evaluate the comparative effectiveness of automated software testing approaches in contrast to traditional manual testing methods. The research aims to assess the impact of automated testing on reducing defects and enhancing software reliability. It will involve a comprehensive analysis of existing literature and empirical studies to examine the advantages and disadvantages of both approaches. Additionally, interviews with software testing professionals will be conducted to gather insights and perspectives on the challenges and considerations associated with implementing automated testing in real-world software development processes. The findings of this research will contribute to a deeper understanding of the effectiveness of automated software testing and provide insights into the potential benefits and limitations of adopting automated testing methods.

Methodology

Qualitative methods of research will be used, involving employee interviews and questionnaire-based survey for a more comprehensive understanding of software testing methods used in selected company.

Interviews will involve observing activities, team interactions, and any challenges encountered during testing.

Data obtained through employee interviews and surveys will be analyzed using a theoretical framework to identify recurring patterns and themes related to software testing methods. The analysis will focus on identifying bottlenecks in the current software testing practices and exploring the advantages and efficiency of the latest methods adopted in the field.

To validate the findings of the qualitative part, a testing script will be developed. This testing script will be designed based on the identified themes and patterns from the qualitative analysis. It will serve as a tool to assess and evaluate the effectiveness and applicability of the software testing methods identified in the qualitative findings.

The proposed extent of the thesis

30-40 pages

Keywords

Functional Testing, Non-Functional Testing, Acceptance Testing,Automation Testing, Integration Testing, Waterfall methodology, Black-Box-Testing, User accessibility, Automation Testing, Selenium

---

Recommended information sources

BADGETT, T. MYERS, G. J. SANDLER, C. 2011. The Art of Software Testing, Hoboken: Wiley. 256p ISBN 978-1118031964

GAYATHRI, M, 2022. Full Stack Testing: A Practical Guide for Delivering High Quality Software. Sebastopol: O'Reilly Media, ISBN 978-1098108137.

Inflectra, Software Testing Methodologies – Learn The Methods & Tools (online) 21 February 2022. https://www.inflectra.com/Ideas/Topic/Testing-Methodologies.aspx. Accessed 24 May 2022.

P. C., DEVRIES, B. JORGENSEN, P. C 2021. Software Testing: A Craftsman's Approach. Boca Raton: Auerbach Publications. 550 p ISBN 978-0367358495.

PRAWIN, M. Software Testing Trends in 2021 (online). 17 Mar 2021. https://medium.com/tilicholabs/software-testing-trends-in-2021-10ad571c42d8. Accessed 24 May

RASNUSSON, J, 2016. The Way of the Web Tester: A Beginner's Guide to Automating Tests. Raleigh: Pragmatic Bookshelf, 243 p ISBN 978-1680501834.

Expected date of thesis defence

2022/23 SS – FEM

The Bachelor Thesis Supervisor

John McKeown

Supervising department

Department of Languages

Advisor of thesis

Mgr. Michaela Peroutková, Ph.D.

Electronic approval: 4. 3. 2024

PhDr. Mgr. Lenka Kučírková, Ph.D.

Head of department

Electronic approval: 11. 3. 2024

doc. Ing. Tomáš Šubrt, Ph.D.

Dean

Prague on 14. 03. 2024

Declaration

I declare that I have worked on my bachelor thesis titled "The Effectiveness of the Latest Methods of Software Testing" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on 15.03.2024 _____

Acknowledgement

Firstly, I would like to express my thankfulness to:

*To my deceased mother **Aynur Tuncay**, whose will is that I am a university graduate,*
*To my esteemed father **Bilgin Tuncay**, who was with me unconditionally both in the education process and in all areas of my life and for who has been waiting for 31 years to see my graduation.*
*To my precious grandmother, **Filiz Tuncay**, who took care of a naughty child like me for 3 years after my mother's death.*
*To my second mother, the precious **Ayten Tuncay**, who has been raising me since I was 9 years old.*
*To my precious and beautiful wife **Gözde Tuncay**, who studied with me for 3 more years even though she is a university graduate.*

Secondly, I would like to thank:

*My supervisor **John McKeown** for his support and time,*

*And **Ing. Martin Kozak** for his help and time during my study*

Also, I am grateful to everybody who supported me during work on this topic.

The Effectiveness of the Latest Methods of Software Testing

Abstract

Thanks to technological developments, software is used in all areas of our lives today. Software testing is the most crucial stage in getting the best software to the end user since it is the ultimate way to determine the quality of any piece of software. No doubt, testing has also been positively affected by all these developments in technology, and testing technology has advanced a lot today. This research aims to learn about different types of software testing with a focus on how effective modern methods are in software testing today. Software test life cycle, the importance of automation tests, Agile and Waterfall test processes will be investigated in detail.

Keywords: Functional Testing, Non-Functional Testing, Acceptance testing, Leverage Automation, Integration Testing, Waterfall methodology, Black-Box-Testing, User accessibility, End-To-End Test

Účinnost nejnovějších metod testování softwaru

Abstrakt

Díky technologickému vývoji se dnes software používá ve všech oblastech našeho života. Testování softwaru je nejdůležitější fází při získávání nejlepšího softwaru pro koncového uživatele, protože je to konečný způsob, jak určit kvalitu jakéhokoli softwaru. Není pochyb o tom, že testování bylo také pozitivně ovlivněno veškerým tímto technologickým vývojem a technologie testování dnes velmi pokročila. Cílem tohoto výzkumu je seznámit se s různými typy testování softwaru se zaměřením na to, jak efektivní jsou dnes moderní metody v testování softwaru. Podrobně bude zkoumán životní cyklus testování softwaru, význam automatických testů, agilní a vodopádové testovací procesy.

Klíčová slova: Funkční testování, Nefunkční testování, Akceptační testování, Automatizace pákového efektu, Integrační testování, Metodika vodopádu, Black-Box-Testing, Uživatelská dostupnost, End-To-End test

Table of content

1 Introduction

A software product must be of the best quality throughout the software development lifecycle. Evaluation of software quality is challenging, though. As a result, the software development process requires more and more quality and process management models and standards every day. Additionally, the need to manage an efficient software quality management process and ensure that this system is continuous has been driven by budget and time constraints in software projects, failures, and the fact that the final result is flawed or different from what was asked.

This thesis discusses the problem of raising software quality, primarily through testing. Software is a product that carries out a specific task, has inputs and outputs, runs on any hardware, and includes computer programs and documentation like usage and maintenance manuals and analysis and design models. The major objectives of software development projects are to create software that satisfies client needs, is bug-free, will increase the customer's marketability, and will be finished within the allocated budget and timeline. The absence of user elicitations, inadequate requirements/specifications, and requirement variations are the key reasons why some software development uncertainties result in project failure. Finding the problem in software projects is essential in light of the primary reasons mentioned. Early bug detection is crucial for minimizing budget and time expenditures and effectively achieving objectives like productivity, accuracy, dependability, usability, and good maintenance within the context of the software iceberg. Software development requires testing because:

- The test items under test don't always work as planned.
- The test items under test need to be validated.
- Decision-makers demand high-quality information.
- The validity of the test items being considered is required.

It is necessary to acknowledge that it is impossible to create bug-free software. It is crucial to test the software before distributing it to stakeholders in order to reduce risk. It is rather typical for software projects to occasionally contain bugs, flaws, and defects. When bugs are supplied to the end user without being promptly noticed, they may malfunction. These bugs are occasionally generated by people working on software development projects. If a flaw is not discovered when the supplied software product is being used by the end user, it will not have an impact on how the program functions. A user running into a software flaw

could have major repercussions; for instance, a mistake could affect public safety, have an impact on the economy, or have an impact on the environment. The main objective of testing is to reduce any residual risk connected to the duration of the test item. Finding problems with the test product before it is released for usage also lowers the risk of poor product quality to stakeholders. The purpose of this study is to assess the effectiveness of the latest testing techniques, this study proposes the following hypothesis: The incorporation of automated testing procedures enhances the effectiveness and efficiency of software testing, leading to faster results and enhanced overall quality.

2 Objectives and Methodology

2.1 Objectives

The objective of this thesis is to evaluate the comparative effectiveness of automated software testing approaches in contrast to traditional manual testing methods. The research aims to assess the impact of automated testing on reducing defects and enhancing software reliability. It will involve a comprehensive analysis of existing literature and empirical studies to examine the advantages and disadvantages of both approaches. Additionally, interviews with software testing professionals will be conducted to gather insights and perspectives on the challenges and considerations associated with implementing automated testing in real-world software development processes. The findings of this research will contribute to a deeper understanding of the effectiveness of automated software testing and provide insights into the potential benefits and limitations of adopting automated testing methods.

2.2 Methodology

Qualitative methods of research will be used, involving employee interviews and questionnaire-based survey for a more comprehensive understanding of software testing methods used in selected company. Interviews will involve observing activities, team interactions, and any challenges encountered during testing.

Data obtained through employee interviews and surveys will be analysed using a theoretical framework to identify recurring patterns and themes related to software testing methods. The analysis will focus on identifying bottlenecks in the current software testing practices and exploring the advantages and efficiency of the latest methods adopted in the field.

To validate the findings of the qualitative part, a testing script will be developed. This testing script will be designed based on the identified themes and patterns from the qualitative analysis. It will serve as a tool to assess and evaluate the effectiveness and applicability of the software testing methods identified in the qualitative findings.

3 Literature Review

In this section, a literature review about software testing will be made and information from various sources will be included.

3.1 Understanding Software Testing

Software testing is the step in the software development lifecycle where a software is checked to see if it is working as expected. It has advantages such as detecting and preventing bugs at an early stage, reducing development costs, and increasing performance.

According to a report by the US National Standards Institute, the damage to the US economy by software bugs is more than $59.5 Billion a year. The potential cost reduction from feasible infrastructure improvements is $22.2 billion. (Tassey, 2022) Software developers accounted for about 40 percent of total impacts, and software users accounted for the about 60 percent Modern testing solutions are evolving and revolutionising. QA is not just about finding bugs and errors anymore; it is a complex technical philosophy that also includes evaluation of a product idea, behavioural predictions, analysis of opportunities and threats, etc (Prawin, 2023).

3.2 The Testing Spectrum

Each stage of the software life cycle involves software testing, but the methods used and the goals of each stage of testing are distinct from one another. Unit testing is a type of code- based testing done by developers. Its major purpose is to test each and every unit separately. It is possible to perform unit testing on small pieces of code, typically no larger than a class. Because developers write unit tests each time they add a feature to the system, unit tests are where most teams start. Here, the goal is to test every potential weak point (Rasnusson, 2016). Integration testing, which tends to concentrate on the interfaces described in low-level design, verifies that two or more units or other integrations function together properly. System testing demonstrates that the system provides the business functions required in the high-level design from beginning to end in a setting similar to production. Business owners perform acceptance testing, which is intended to determine whether the technology satisfies their operational needs. Testing software after changes have been made is known as regression testing, and it is done to ensure the dependability of each software release and to ensure that the changes did not introduce any new flaws. Alpha

Evaluation It will typically be carried out on the developer's property while they are there. a test version completed at the client's location without the developer there. A finished application must undergo functional testing to ensure that it possesses all of the desired behaviours (Nidhra, 2012).

3.3 Testing Effectiveness

Test effectiveness is a measure of bug finding ability of the testing technique. Testing effectiveness can be measured by dividing the number of faults found in each test by the total number of faults, including those found after the test (QUADRI, 2010).

Test effectiveness can be calculated using the below formula:

Test Effectiveness = Number of valid bugs fixed/ (Bugs injected+ number of bugs escaped) *100

Test efficiency = (Total number of defects found in unit + integration + system) / (Total number of defects found in unit + integration + system + User acceptance testing) *100 (Chernak, 2001).

3.4 Existing Testing Methods

Since white box testing checks the internal organization of the program in addition to the software's functionality, it is a successful testing technique.

Programming skills are necessary to design the test cases in order to perform white box testing. Clear box and glass box testing are other names for white box testing.

All testing tiers, including unit, integration, and system testing, can use this type of testing. This type of testing also known as security testing meets the need to ascertain whether the information systems secure data and continue to perform as intended. Every logical decision is exercised, all loops are verified at each boundary level, and internal data structures are also exercised since this type of testing procedure uses the internal logical arrangement of the software. As a result, it is capable of testing all the independent paths of a module. White box testing, however, serves a purpose by being a sophisticated testing method because it incorporates programming skills (Swebok, 2004).

A testing method known as "black box testing" basically tests the functionality of the program without getting too specific about its implementation. Every testing stage of the SDLC is compatible with this method.

It primarily carries out the testing in such a way that it covers every functionality of the program to ascertain if it satisfies the user's initial needs or not. By evaluating each functional aspect's performance at the minimum, maximum, and base case values, it can detect improper functions. It is the most straightforward and often utilized testing method globally (Miller, 1981).

Grey box testing combines the benefits of both the White Box and Black Box Testing Techniques. This form of testing is necessary because the tester is aware of the internal organization of the program, which allows for better functionality testing by taking the internal organization of the application into account.

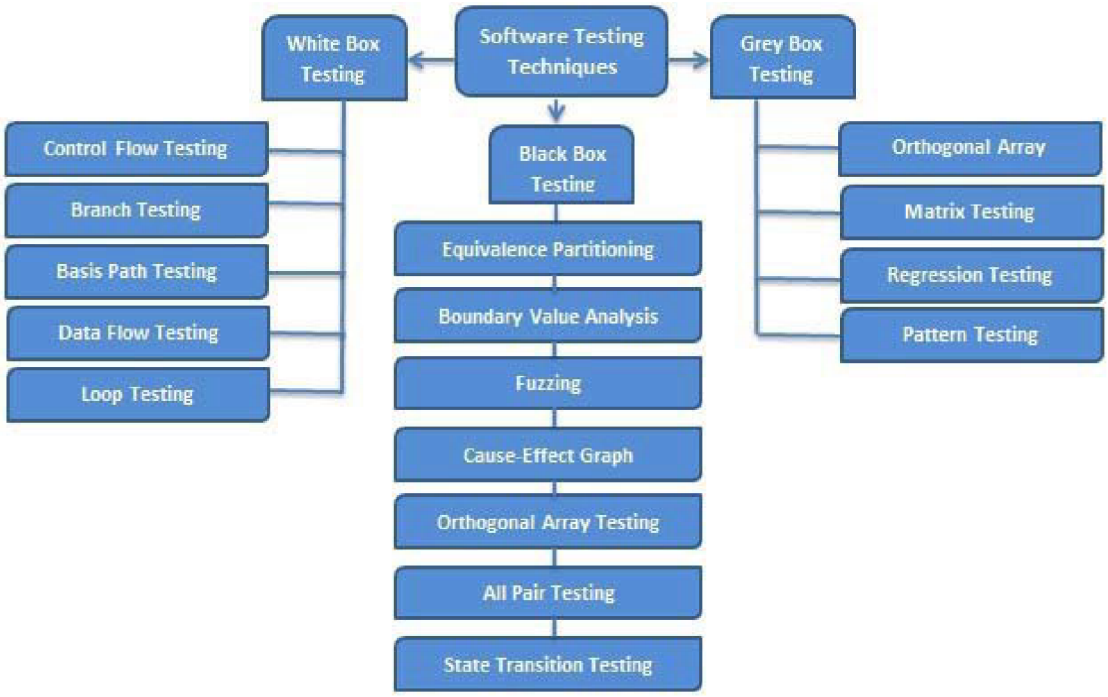


Figure 1Software testing methods and techniques (J.Irena, 2008)

3.4.1 Unit Testing

Unit testing is a method for testing where individual units or modules of a software are tested, whether they are working as expected. In unit testing, every unit, which is normally the littlest testable piece of a product, is tested separately without its dependencies.

3.4.2 Integration Testing

Software testing that confirms the correct operation, compatibility, and communication between various parts or modules of a software system is called integration testing. In order to guarantee that these parts function as a unified system, it attempts to validate the integration and interaction of these parts.

3.4.3 Functional Testing

Functional testing verifies a software system's behavior and functional requirements. Its goal is to guarantee that the system operates as intended, meeting the defined use cases, business rules, and functional requirements.

3.4.4 Regression Testing

Regression testing is a testing method which can be used, for the previously tested functionalities after the new changes performed. It ensures that, performed change did not cause any other new issues.

3.4.5 Performance Testing

Performance testing is a method of software testing used to assess how well a software system responds, scales, remains stable, and uses resources under various workloads and circumstances. In order to find any bottlenecks, performance problems, and opportunities for optimization, it entails monitoring and evaluating the system's performance indicators.

3.4.6 Acceptance Testing

Acceptance testing is a testing method, and it tests whether a system satisfies requirements, user expectations, and corporate goals. It entails putting the system through stakeholder or end-user testing to make sure it meets their needs, functions as planned, and is prepared for deployment.

3.4.7 Security Testing

Security Testing is performed to detect possible security vulnerabilities of the software. It includes important security tests such as authentication, autherization, and data encryption.

Criteria	Unit	Integration	System	Acceptance
Purpose	The correct working of unit/module	The correct working of integrated units	The whole system works well when integrated	Customer's expectations are met
Focus	Smallest Testable part	Interface and interaction of modules	Interaction and working of all modules as one	Software working in accordance with given specifications
Testing Time	Once a new code is written	Once new components are added	Once the software is complete	Once the software is operationally ready
Performed By	Developer	Development Team	Testing Team	The development team and End-Users
Testing Techniques	Usually Whitebox, and Greybox	Whitebox, and Blackbox	Usually Blackbox, and Greybox	Black-box testing
Automation	Automable using Junit, PHPUnit, TestNG etc.	Automable using SOAP UI, Rest Client etc.	Automable using Webdriver	Automable using Cucumber
Scaffolding	Complex (require drivers and/or stubs)	Moderate (may require drivers and/or stubs)	No dirvers/stubs required	No drivers/stubs required.

Table 1 The Software Testing Levels compared (Umar, 2019)

3.5 Strengths and Weaknesses of Manual Testing

Manual testing allows to handle efficiently ad hoc or exploratory testing and requirements changes can be easily accommodated. Testers can offer subjective comments and evaluate the overall user experience by manually engaging with the system, which enables a thorough assessment. Additionally, manual testing gives testers the ability to cover edge situations and replicate real-world events, which results in comprehensive test coverage. Manual testing does, however, have several drawbacks. Large-scale or recurring test cases in particular can take a lot of time, and a substantial number of human resources must be allocated. Furthermore, because manual testing heavily depends on testers, it is susceptible to oversight, inconsistency, and human mistake. Moreover, manually conducted test cases cannot be easily reused for subsequent iterations or regression testing, leading to redundant work.

3.6 Automation Testing

Automation testing is the process of running tests and comparing actual results with predicted outcomes utilizing automation tools, scripts, and frameworks. In order to increase productivity, accuracy, and test coverage, it entails automating the tedious and manual processes of test execution, data setup, and result verification.

3.6.1 Selenium

Selenium is an open-source automation testing tool. It facilitates testing web applications on various platforms and browsers. It offers a wide range of functionalities for test scripting (in multiple programming languages, such as Java, Python, C#, etc.), element interaction, browser automation, and testing framework integration.

3.6.2 Appium

Appium is specifically made for testing mobile applications, Appium is an open-source automated testing tool. Testers can create and run tests for native, hybrid, and mobile web applications because it supports the Android and iOS platforms. Cross-platform compatibility and support for many programming languages are features offered by Appium.

3.6.3 HP UFT

Unified Functional Testing A popular commercial automated testing solution for functional testing, it formerly known as HP QuickTest Professional (QTP). It provides several tools for test scripting, object detection, test data management, test execution, and result reporting. It supports both UI and API testing.

3.6.4 JUnit

JUnit is a popular unit testing framework to create and run automated test cases for Java programs. It helps to planning and carrying out tests, it offers test runners, assertions, and annotations.

3.6.5 SoapUI

SoapUI is a popular testing tool for web services. It enables the creation, use, and validation of RESTful and SOAP web services by testers. It facilitates web service load, security, and functional testing.

3.6.6 JMeter

Apache JMeter is a performance testing tool. It is helpful to test a web application under the load and measure the performance. It supports multiple protocols, including HTTP, HTTPS, FTP, and others.

3.6.7 Robot Framework

Robot Framework is an all-purpose open-source automation framework. It can be applied to robotic process automation (RPA) and test automation. Robot Framework Foundation provides support for Robot Framework. Leading businesses in the sector use the tool for their software development. Robot Framework is adaptable and open source. Almost any other tool can be connected with Robot Framework to produce robust and adaptable automation systems. Robot Framework has no licensing fees and is available for free usage. Robot Framework uses human-readable terms in an easy-to-read syntax. Libraries written in Python, Java, or many other programming languages can increase its capabilities.

3.7 Software Testing Process

The software testing process consists of a series of actions that were planned and enforced, and the results were recorded and documented. This process focuses on the existence of errors in software projects developed (Gürbüz, 2007). According to Garousi et al. testing process is defined as "*a test process involves several steps from test planning [to test definition (test case designing), execution, and reporting, each of which can be either done manually or automated.*" (Garousi, 2016).The activities performed in the software testing process and the steps of the software testing life cycle are shown in Figure 3.

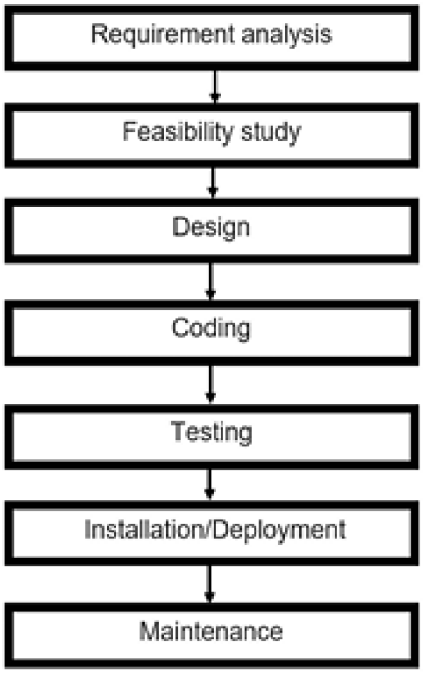


Figure 2 Steps of the software testing life cycle (Taley, 2020)

3.7.1 Data Collection

Test data is a collection of input data used to test any application. Several types and sizes of input data are used to test the application. For critical applications, the customer may also perform test data collection.

3.7.2 Test Planning

The what, when, how, who, and other details of a testing project are all outlined in a test plan for software testing. The scope of the testing, the test items, who will perform which testing task, the item test/pass criteria, the requirements for setting up the test environment, and much more are all covered in detail.

The initial step in the software testing process is planning. A test plan document provides a detailed blueprint for the complete testing procedure. It contains the rules for the testing procedure, including the strategy, the tests to be run, the environment required, the resources needed, the timeline, and the limitations. It provides detailed instructions on how to test the software in its entirety (Kaner, 1999)

3.7.3 Test Case Designing

The scope of the test, the test strategy, the testing environment, the software components to be tested, the expected test actions for the project, the resources, and a schedule are all included in a test case design document (Black, 2005).

After a software project's requirements are approved, the test case design phase begins. While system analysts, system engineers, and software engineers prepare the system requirements and software requirements, the test team determines the test strategy suitable for the project and updates the relevant test plans. Then, while system and software developers perform preliminary design and detailed designs, the test team starts to create test cases. The chart, which is presented in figure 3, explains the requirements and test case relationships.

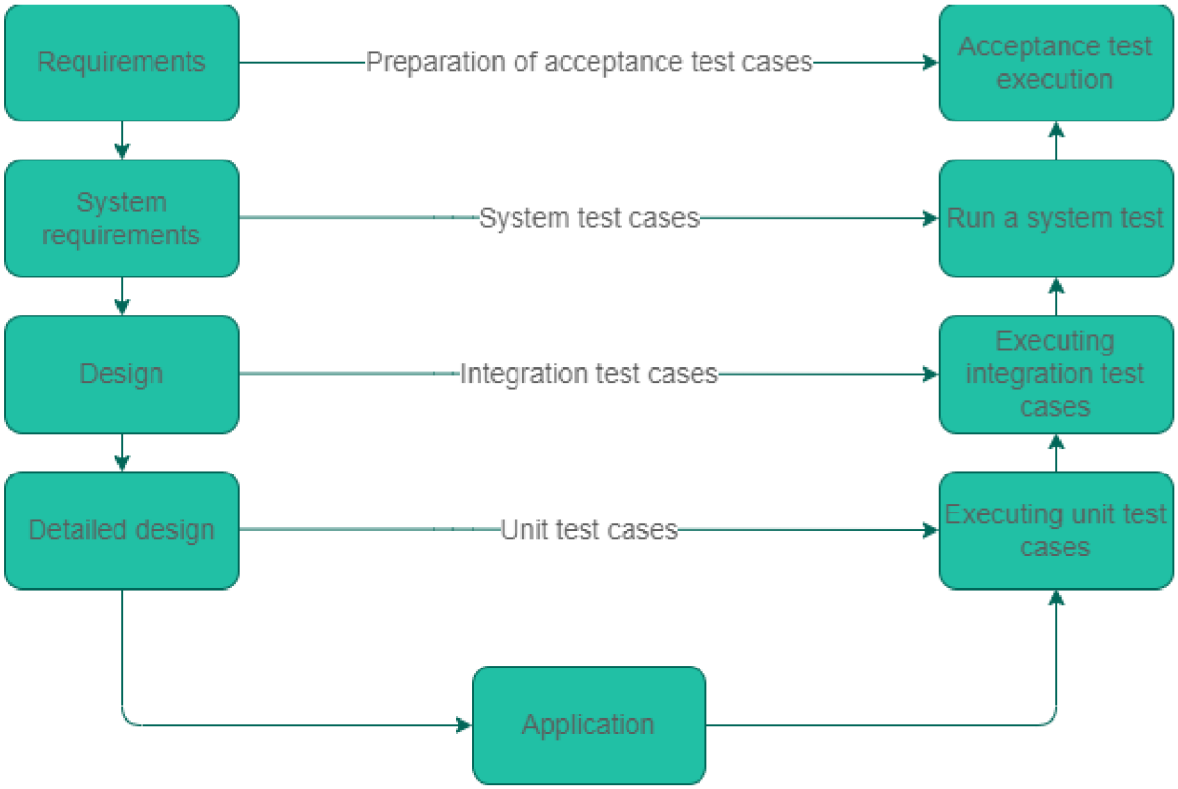


Figure 3 Test case relationships (Testim, 2022)

The importance of test-case design is explained with the following sentences in The art of software testing book “The most important consideration in program testing is the design and creation of effective test cases. Testing, however creative and seemingly complete, cannot guarantee the absence of all errors. Test-case design is so important because complete testing is impossible; a test of any program must be necessarily incomplete. The obvious strategy, then, is to try to make tests as complete as possible.” (Myers, 2011).

3.7.4 Test Executions

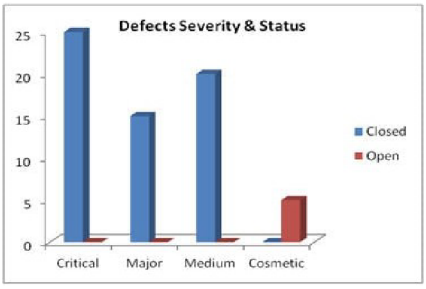
Test Execution is the running of a test to check whether the software to be tested meets the pre-defined requirements and specifications. Expected results and actual results are compared. The next steps are determined. If the results comply with the predetermined requirements, the tested product is ready to be released on the market. Otherwise, the product should be fixed and re-tested (ProfessionalQA, 2018).

3.7.5 Test Result Reporting

The Test Results report is a physical log that records details such as the environments in which the code was tested, by whom, when and how it was tested, and any errors that occur during testing (Alli, 2007).

Information such as an overview of the application, the scope of the test, metrics (a metrics example is presented in figure4), the types of tests run, the test environment and the tool used for the test, the errors that occur during the test and their solutions, the summary of the thesis are the information that should be included in the test report.

	Critical	Major	Medium	Cosmetic	Total
Closed	25	15	20	0	60
Open	0	0	0	5	5
					65



- **Defects distribution – module wise**

	Registration	Booking	Payment	Reports	Total
Critical	6	7	5	7	25
Major	4	5	2	4	15
Medium	6	8	2	4	20
Cosmetic	1	2	1	1	5
Total-->	17	22	10	16	65

Figure 4 Example of test result metrics (Shetageri, 2016)

3.8 Understanding the Agile and Waterfall

The traditional model of software development is the waterfall model, which is illustrated in Figure 5.

In the waterfall method, the activities in each step are performed completely. This is the condition to move on to the next step. A document is created at the end of each phase. Therefore, the waterfall model is document driven. The software process is linear, that is the activities in the previous stage must be completed in order to move on to the next stage. User participation is possible in the initial stage. User requirements are determined and detailed at this stage. While it has advantages such as clear limitation of phases, simple planning and control possibilities, and low cost, it has disadvantages in terms of sequencing, delimitation, and adequacy, as user participation is only possible in the first phase (Kramer, 2018).

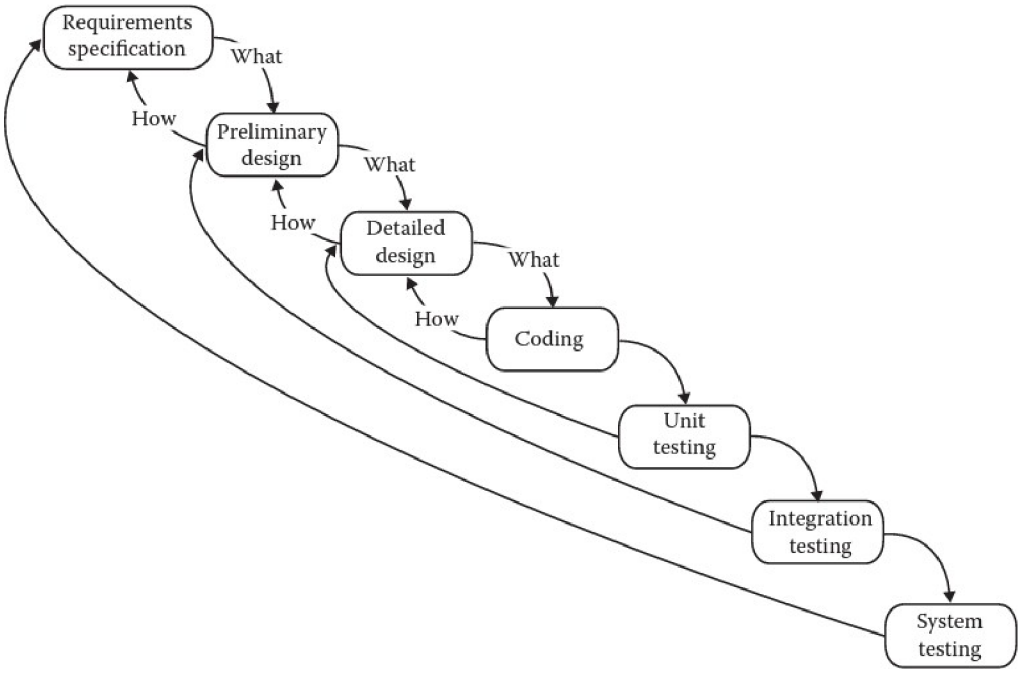


Figure 5 The waterfall life cycle (Jorgensen, 2013)

Agile methodology is much more flexible than traditional methods as shown in figure 6. It is built on the provision of early and continuous testing of the software. Thus, it has become possible to find and fix a bug that will appear in the product at an earlier stage.

In Agile methodology, there is a greater scope and time to accept feedback from end users and stakeholders. The user is in control of the development process from beginning to end, so there is much less documentation work. Of course, besides these advantages, Agile methodology also has some disadvantages. Since each iteration will go through its own development stages, development levels can overlap unnecessarily. Also, the repeated release of software pieces will mean higher costs (McCormick, 2012).

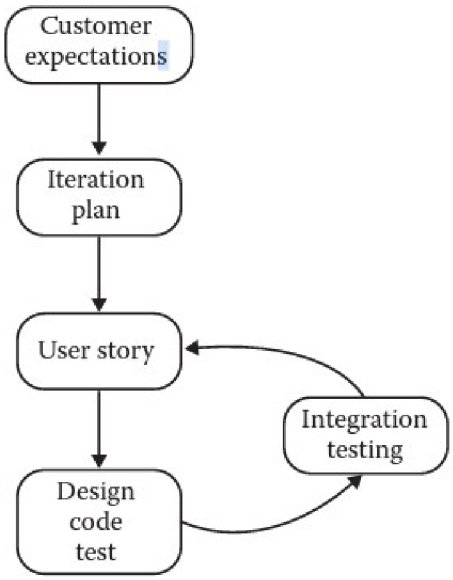


Figure 6 Agile Ligecycle (Jorgensen, 2013)

3.9 Future Directions in Testing

It would not be very accurate to make predictions about the future. However, we can provide some estimates with the information we have. Let's start with automation, we can say that automation is one of the most focused topics in software testing today. As the work continues in this direction, it will not be difficult to say that software testing will become more automated. Thanks to the development of automation, testing professionals will be able to focus on more complex issues, which will bring further developments. The rapid progress of artificial intelligence and the point where it comes from 10 years ago is astonishing. It will not be very difficult to predict that artificial intelligence will play a very important role in the software test in the future.

According to Hourani, H., Hammad, A. and Lafi, M artificial intelligence has the ability to automatically analyze complex data using intelligent models and algorithms AI has already shown that it can perform better in software testing In the near future, AI-powered testing will usher in a new era of quality assurance. Will manage and control most areas of testing and add significant value to test results and deliver more accurate results within a competitive timeframe There is no doubt that AI will influence and drive the quality assurance and testing industry in the future. Intelligent automation of software testing will improve software quality and have a significant impact on customer experience by delivering reliable and defect-free applications and solutions (Hourani, 2019).It is also possible to

predict that there will be improvements in Security Testing. According to the report prepared by the Identify Theft Resource Center affiliated to the Department of Justice in US In 2021, there were more data compromises reported in the United States of America than in any year since the first state data breach notice law became effective in 2003 (IDTheftCenter, 2022).

In addition, according to the information given by M. Gayathri in his book full stack testing, cybersecurity experts are thinking, the yearly worldwide cost of cybercrime (including both direct and indirect expenses to its victims) will rise from an anticipated $6 trillion in 2021 to $10.5 trillion in 2025 (Gayathri, 2022).These informations show us that security test will play a much more important role in the near future due to cyber attacks and data violations that are increasing day by day.

4 Practical Part

In order to validate the hypothesis, in this section a questionnaire will be performed to assess current testing practices and an automation script will be developed and evaluated.

4.1 Questionnaire and Analysis

In this section, information about the research method, the characteristics of the participants and the preliminary information about questionnaire will be provided.

4.1.1 Introduction the Research

In this chapter, a questionnaire will be examined with Software testers from different positions and experience levels working in different areas of software testing, working in the Software Testing & Validation Team in a large-scale company. A qualitative method was preferred as the research method. After collecting the answers, the data will be visualized, analysed and a conclusion will be formed in this direction. The company name (Hereinafter referred to as the company) and the names of the participants will not be disclosed. Participants will be named as participant 1, participant 2, participant 3.

4.1.2 Details Regarding the Participants

In the study, a pharmaceutical company with 68,000 employees and members of its Software Validation and testing team were selected. The company's headquarter is in New Jersey, and its offices are around the world. The most important reason for choosing this company is that the test team is exposed to working with a wide variety of software, tools, etc. within the company. The testing team is experienced in its field. Team members work in the same team, but in very different areas. This is the other most important reason for choosing this company and team. In this way, it is possible to obtain information about both their continuous communication and their experiences in different areas. While determining the experts to participate in the research, it was taken into account that they were from different fields, different experiences and different positions. Thanks to this diversity, it was aimed to reach a more accurate result. The information of the participants is presented in the Table 2.

Participant	Overall experience in Testing	Current Position	Testing Areas
Participant 1	12 years	QA Team Lead	E-Commerce website testing UI Testing Application Testing (Both on system and android device) Security Testing
Participant 2	4 years	QA Test Lead	Manual & Automated, Performance testing
Participant 3	2 years	QA Engineer	Regression Testing
Participant 4	8 years	Software Engineer	WEB UI Manual and, Automated tests
Participant 5	7 years	Senior Analyst	Performance Testing
Participant 6	7 years	Performance Test Lead	Performance testing
Participant	Overall experience in Testing	Current Position	Testing Areas

Participant 7	8 years	QA Team Lead	Integration Testing, System testing, E 2E Testing
Participant 8	11 years	Product QA Lead	Manual web, mobile testing
Participant 9	15 years	QA Teem Lead	Test automation
Participant 10	3 years	QA Automation Engineer	Automation system testing

Table 2 Details of the research and participants inputs (author's own analysis)

While some of the questions asked in the study tried to find answers to the shortcoming points of today's testing technologies, others sought answers for possible improvements.

In the study, the following questions were asked to the participants:

What is your overall experience in testing (as years)? In which testing areas do you have experience so far? What is your current position?

What do you think about the latest testing methods?

Prioritize the following issues of software testing that should be used more effective (Start from the biggest issue)?

- Incident Handling
- Test Design
- Test Efforts
- Test Tools
- Time

- Test Management
- Test Planning
- Test Execution

What are the most critical problems which you are facing (Start from the most you are facing)?

- Communication issues
- Impossibility of the complete testing
- Lack of requirements
- Unstable environment
- Lack of test tools' capability
- Management related issues
- Missing documentation
- Diversity in testing environments

What are limitations which you are facing?

Which testing processes should be automated ?

How to improve the use of software testing techniques and testing processes in order to ensure software quality ?

4.1.3 Analysis of Questionnaire

In this section, the answers of the participants will be examined, and results will be obtained.

In the very beginning of questionnaire expert were asked to specify their personal involvement to the testing activities. The most experienced of the participants has 15 years of experience and while he is working as a QA Team lead, least experienced one has been in the sector as a QA Engineer for 2 years.

The first question asked to the participants was what they thought about the latest testing methods. While participant 1 answered this question as Agile methodology is doing great in the latest testing methods, participant 5 answered that test methods are developing and becoming more and more efficient day by day. Participant 7 mentioned the contribution of automation testing to the development of the testing process, and according to participant 7, the most important point of the final testing methods is that defects and observations can be detected and reported earlier thanks to automation testing. Participant 9 answered this

question in a similar way to Participant 7. According to Participant 9, thanks to the development of automation, a simpler process, reaching the target more easily and collaborating with DevOps is more possible than before.

The second question was a ranking question and the participants were asked which topics should be given more importance in software testing. Topics to be included in the ranking were: Test Planning,Test Design,Test Efforts,Time,Test Tools,Test Management,Test Execution,Incident Handling. 6 out of 10 participants stated that Test Planning is the most important stage. After Test Planning, the most important issue was Test Design, while 2 of the participants evaluated Test design as the most important, it was the second choice of 3 of them. The order of importance according to the participants was determined in Figure below.

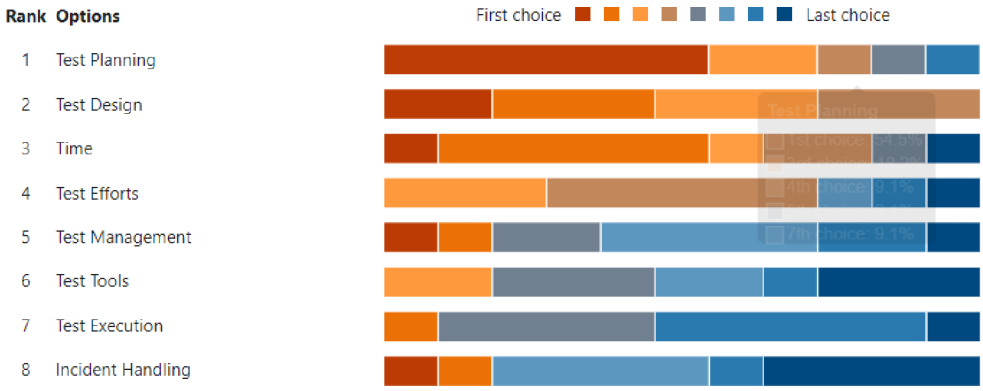


Figure 7 Which topics should be given more importance in software testing? (author's own analysis)

In the third question, another ranking question was asked to the participants, and they were asked to rank the problems they encountered the most among the following problems.

Lack of requirements, Missing Documentation, Communication Issues, Unstable Environment, Management Related Issues, Diversity in Testing Environments, Impossibility of the complete testing, Lack of Test Tools' capability. According to the data collected, the most common problem was the lack of requirements. Lack of requirements was the first choice of 2 participants, it was the second choice of 4 participants. Missing documents and communication issues followed the lack of requirements. The most common problems encountered by the participants are listed in Figure 8.

Figure 8 The most common problems encountered by participants (author's own analysis)

Question 4 was about the limitations participants faced and was a comment question. The similarity of the answers to this question was striking. Answers of participants 1,2,3,4,6 and 9 was same for this question. According to these participants, the biggest limitation is communication gap between business. The 3rd participant complained about the lack of documentation about the software to be tested. According to the 3rd participant, not having enough documentation about that software is the biggest limitation to be able to test a software in the best way. The 8th and 10th participants stated that time was the biggest limitation. According to these participants, the time given to testing a software is often insufficient.

In Question 5, users were asked which test processes and test types should be automated. Participant 3 Participant 4 and Participant 7 stated that regression testing packages should be automated due to repetitive execution throughout software testing. Participant 10 has stated that any task that can be use in every release should be automated. Participant 1 stated that the approval processes should be automated. Participant 9 has stated that the test cases that require constant execution, such as smoke test, component test, and integration test, can be executed with automation.The last question asked to participants in the questionnaire was how to improve the use of software testing techniques and testing processes to ensure software quality. Participant 9, the most experienced participant, provided a clear answer to this question Develop the scope, have detailed discussions with the developers. Participant 4 and participant 10 indicated the importance of prioritization, while participant 5 emphasized the need for effective conduct of examinations. Participant

7 answered this question by simply writing down all the steps of the Software development lifecycle.

Many test processes that were supposed to be done manually by people years ago can now be handled with automation, but clearly revealing the needs, documenting them well and explaining them to the software testers in the best way ensures that the processes are progressed and completed successfully. Another important finding is how important test planning and design is. If test planning, which is one of the first steps of the software test life cycle, is not done well, it causes many problems in the following processes. According to the participants, the biggest obstacle to the successful completion of the testing process is poor test planning and test design

The latest software testing methods are undoubtedly far ahead of the past and continue to evolve rapidly. The interviewers stated that automation testing has greatly enhanced the testing process by decreasing errors and increasing efficiency. Automation testing has also made continuous integration and delivery easier. The interview's takeaways highlight how crucial automated testing is to modern software testing. Latest testing methods have also brought about improvements in earlier detection and reporting of bugs. In addition, existing test tools play an important role in increasing the effectiveness of test methods.

4.2 Robot Framework Automation Test Script

In this section, an automation test script will be developed with the Robot Framework and the difficulties encountered during development, the advantages and possible disadvantages of the automation will be examined.

Spotfire is a BI tool which is developed by TIBCO.The test automation to be developed aims to automate repetitive cases that were performed manually on 76 separate servers in previous years, after the Spotfire upgrade. The script contains 29 cases. These cases will aim to test whether the correct configurations required for the application to work as expected have been made, whether the db upgrade has been completed successfully, whether the server has the minimum requirements specified in the application documentation, and whether all the components required for the application to run smoothly have been installed successfully. While creating the automation, Robot Framework, which is open-source Python library, Python and Powershell were used.

4.2.1 Implementation and Execution of Automation

All cases run in the Automation script were run through the testsuite.robot file. In each case, the script runs one of the Python or Powershell scripts in the background and compares whether the output is the same as the expected result. If the expected result and the actual result match, the test case is passed, otherwise it fails.

As previously mentioned in the overview part, Spotfire servers have different components, which called as Spotfire Server – Node Manager and Statistic Servers.

In the test script, while some cases are valid only some of the components, some of them needs to be executed in all the servers. It is defined with the tags feature of Robot Framework.

SS tag has been defined for Spotfire Servers.

NM tag has been defined for Node Manager servers and

STAT tag has been defined for Statistic Servers.

A Json file has been used to define server type in the execution server. After defining the server type, case with the relative tag has been executed.

```
*** Settings ***
Documentation     This file contains test cases for Spotfire IQOQ
Library     python_keywords
Resource     ../Script_Parameters/script_parameters.robot
Resource     ../Keywords/rf_keywords.robot
Library           OperatingSystem
#Library           Collections
Library           DateTime
Library           cx_Oracle

*** Test Cases ***

Run | Debug | Run in Interactive Console
Verify the User Directory is set to Ldap
    [Tags]    SS
    ${output}    Execute Powershell Script With Parameter    ../PowershellScript/LDAPauth.ps1    ${installation_pa
    ${status}    Pattern Should Match    ${output}    ldap
    Should Be Equal    ${status}    True

Run | Debug | Run in Interactive Console
Server has a supported operating system
    [Tags]    SS    NM    STAT
    ${output}    Execute Powershell Script    ../PowershellScript/SupportedOS.ps1
    ${status}    Run Keyword And Return Status    Should Contain Any    ${output}    Microsoft Windows Server 2016
    Should Be True    ${status}
```

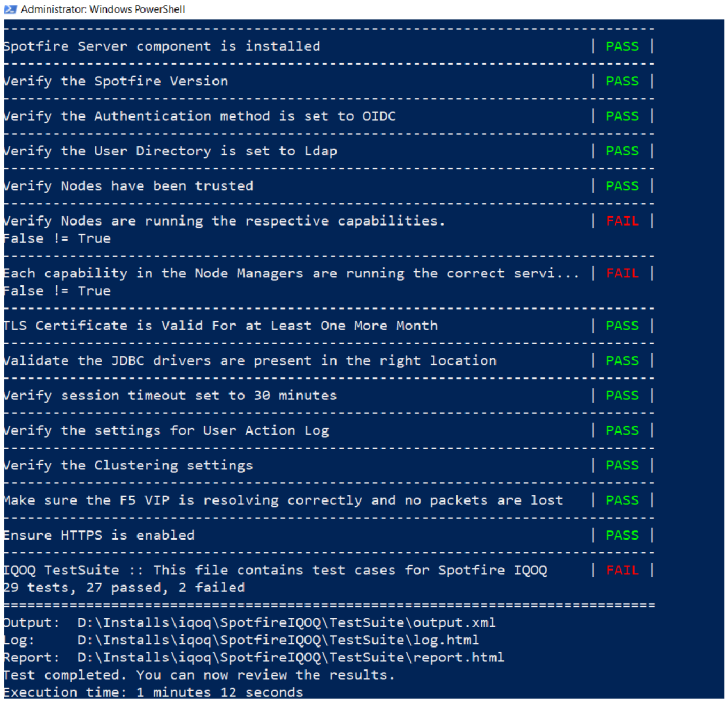
Picture 1 Test suite.robot file

Robot Framework provides built-in keywords for testing activities. All those keywords can be found under the official documentation of Robot Framework library. However, in some cases creation of custom keywords might be required. In our script, some keywords have been created customly as you can see below in Picture 2.

```python
def connect_to_oracle_db(username,password,host,query):

        connection = cx_Oracle.connect(user=username, password=password,
                                       dsn=host)
        cursor = connection.cursor()
        sql = (query)
        cursor.execute(sql)

for row in cursor:
    print(row)

connection.close()


def create_connection_to_server(path,cmd_command):
    output_object = subprocess.check_output("D: &&"+f"cd {path} &&"+f"{cmd_command}", shell=True)
    print(str(output_object,'UTF-8'))
    return str(output_object,'UTF-8')

def execute_powershell_script_with_parameter(path,parameter):
```

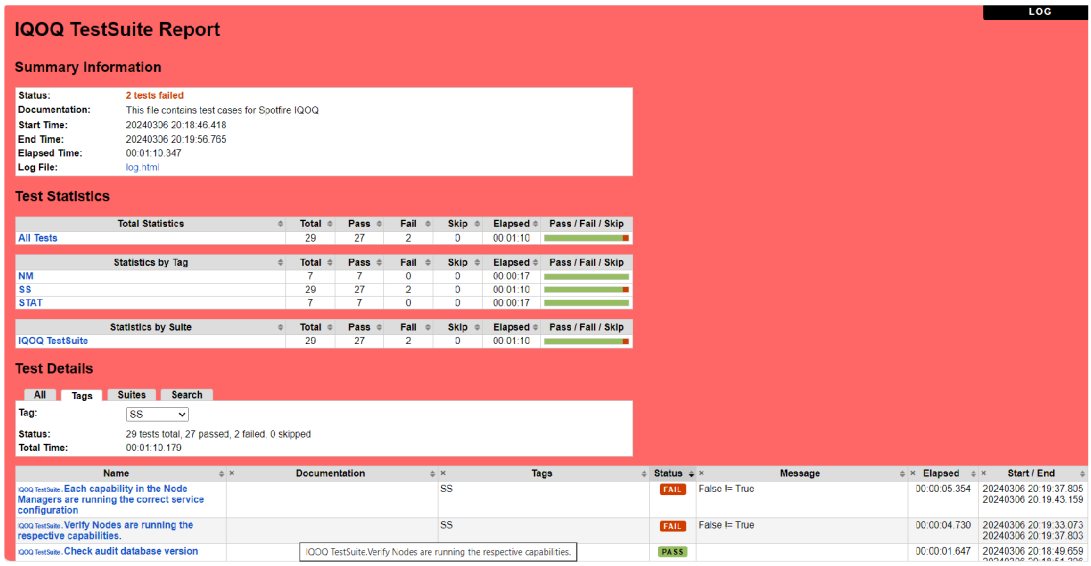
Picture 2 Python keywords

The developed test automation was run on a dev server and the execution was completed in 1 minute 12 seconds as shown in Picture 3.

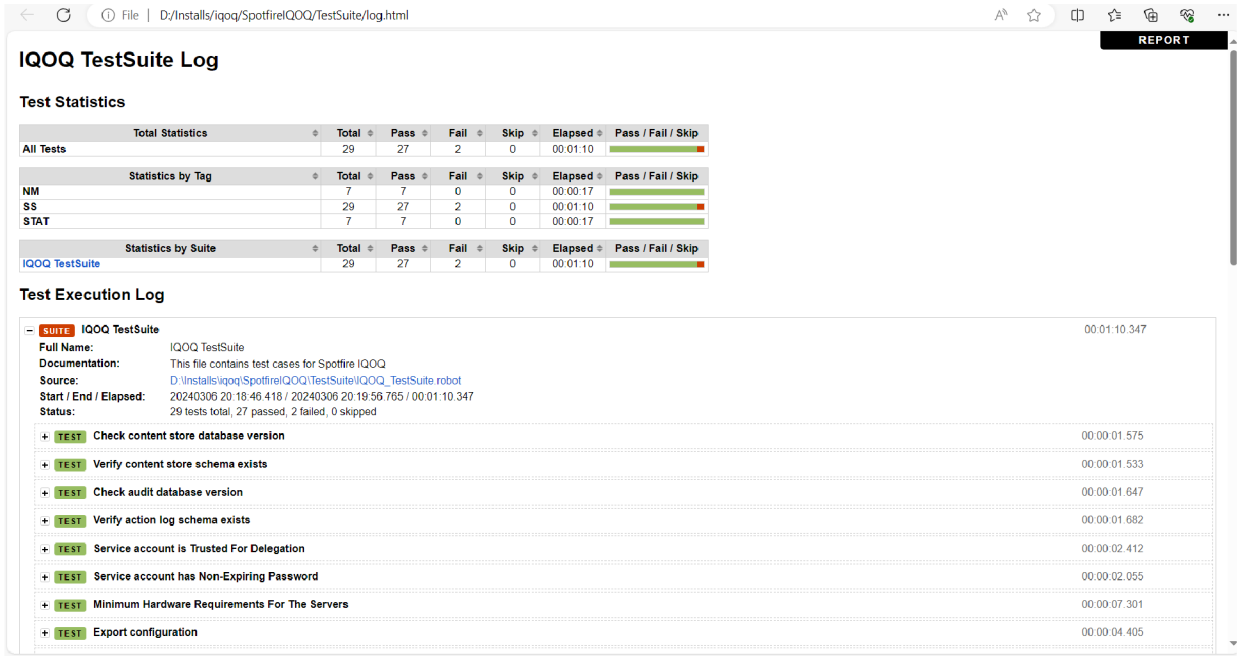


Picture 3 Script execution

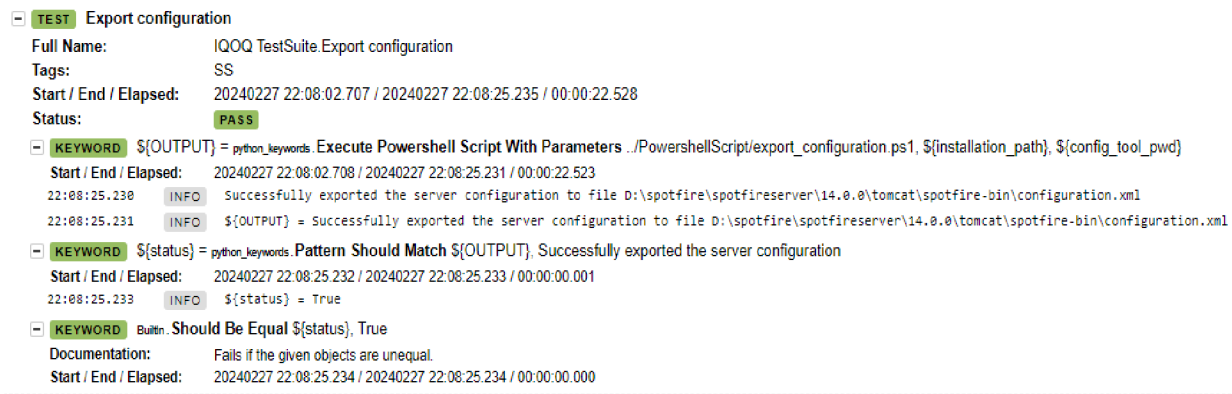
29 different cases were tested and 27 of them passed successfully, while 2 of them failed. Log.html , output.xml and report.html files were created by Robot Framework. An example from the Report.html file is shown in Picture 4, and an example from the Log.html file is shown in Picture 5. All the logs can be extended to see further details about the case by clicking the plus button near the case name.An example for the passed case shown in Picture 6 and an example for a failed case shown in Picture 7.



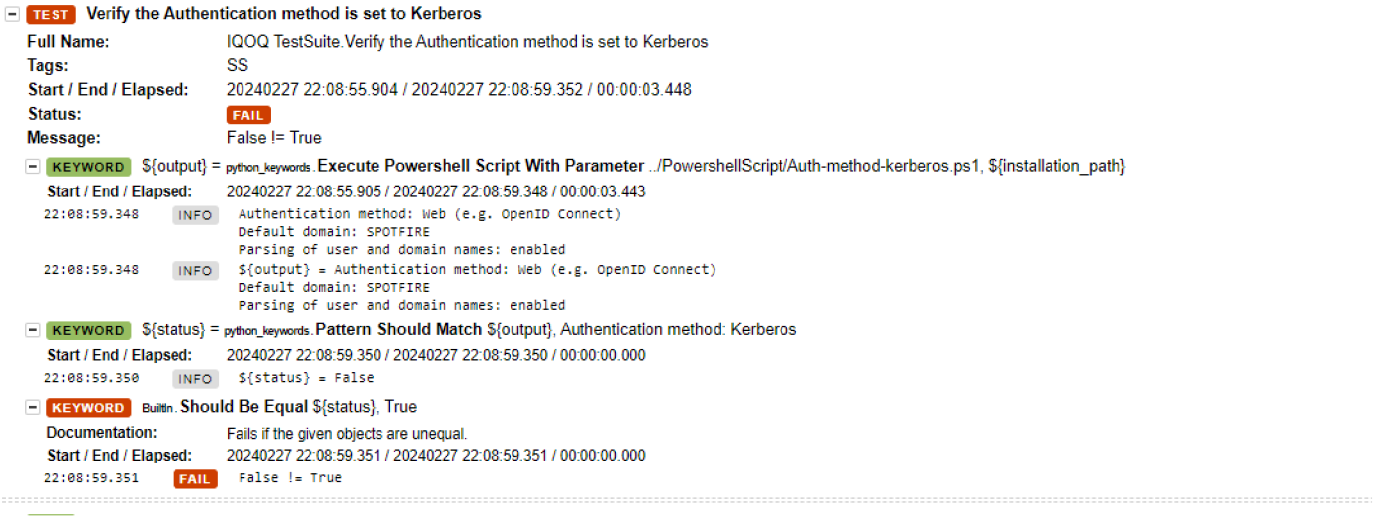
Picture 4 Report.html



Picture 5 log.html



Picture 6 Example for the Passed Case



Picture 7 - Example for the Failed Case

4.2.2 Evaluation of Collected Data

The script tested 29 different cases and created the report file in 1 minute and 12 seconds, resulting in an impressive performance.

According to a member of the team who carried out the same testing activities manually in previous years and according to the estimation documentation created in the previous year, it took around 1.30 hours to complete this test manually for each server, not considering any possible breaks. Considering that the script will be run on 76 different servers, a reduction in time of 112.5 hours and therefore in cost occurs. Since the developed script is designed to be reusable with very small maintanances, it is suitable for use after every upgrade, and the reduced cost in long run will increase even more. Details shown in Picture 8.

Execution Time Analysis in hours as per the Approach

Approach	Execution Time
Automation	1.52
Manual Testing	114.00

Picture 8 Execution time analysis

Picture 9 Cost analysis

A comparison of cost analysis is depicted in Picture 9. The development of the automation script has consumed approximately 75 hours. On the other hand, the cost of the manual testing was calculated qualitatively using previous year's documentation and a resource who involved the manual testing activities during last upgrade, which indicated that it required 114 hours of manual effort. The cost per hour of resource accepted as same at $35 for both manual testing and automation script development. During the cost calculation of the automation script, a tool cost did not be included because Robot Framework is an open source tool.

It is important to note that although some resources were utilized to gather these data, they should be considered hypothetical. The required time for automation script development may vary depending on the expertise of the developer, and the cost can differ under different circumstances.

5 Results and Discussions

The purpose of this thesis was to examine the effectiveness of the latest testing methods and to identify missing points. The literature review about software testing was conducted, existing testing methods were explained, and information was provided about terms such as verification and validation. Some predictions about the future of software testing were also shared as a result of the scientific literature analysis. All the steps of the software testing life cycle were examined, and information was presented from various scientific sources. Comparisons of methods such as Agile-Waterfall were made, and various opinions were gathered about the test methods applied today. The study allowed us to learn the ideas of experts with various degrees of experience involved in a work environment. Participants were selected based on criteria such as work experience and education level.

The results of the study revealed that the biggest limitations to software testing are communication gaps between testing experts and business users, bad documentation, and poor test planning. During development, communication problems were encountered, which is the problem that interviewers mentioned. Some cases that were not in the plan at first, and then requested, its integration took more time than it normally would. The goal was to run the script from a single central server via remote connection, but although this was technically possible and the developed script did not require any manual intervention, this was not possible due to company policy. For this reason, a manual connection will be made for each server and the script will need to be triggered. As another limitation, it can be mentioned that automation scripts can be created based on already known configurations or previously experienced problems. Automation scripts cannot take unexpected issues into account. Moreover, although it shortens the effort and cost, human intervention is required at certain points. For example, logs should be examined carefully and in case of any problem, necessary actions should be taken manually.

Furthermore, the study highlighted the difference between manual testing and automation testing. Automation testing emerged as an important part of the latest testing methods. It is a more efficient and reliable approach compared to manual testing. Automation testing utilizes various tools and technologies to automate test execution processes, which eliminates the need for repetitive manual effort. Additionally, automation tests are effective at capturing details that may be missed by human testers. In line with the objective of the thesis, an automation script was developed to provide practical evidence of

the effectiveness of automation testing. The automation script was designed to automate repetitive testing tasks and verify the functionality of the application under test. This script demonstrated the efficiency and accuracy of automation testing by significantly reducing the time and effort required for testing. The findings of the study indicate that automation testing offers several advantages over manual testing. It improves the overall efficiency and accuracy of the testing process, reduces testing effort and costs. Moreover, automation testing enables faster execution of tests and provides detailed and comprehensive test results. In conclusion, the development of an automation script as part of this thesis project provided practical evidence of the effectiveness and benefits of automation testing. The findings reinforce the importance of automation testing as an integral part of the latest testing methods. Organizations that embrace automation testing can expect improved efficiency, accuracy, and cost-effectiveness in their testing processes, ultimately leading to higher-quality software products.

This study makes it clear that, in order to get better and more reliable results from testing operations, detailed pre-definition of requirements and testing scope is essential. To guarantee that all testing operations are planned,executed and reported systematically, a comprehensive and well-documented testing process has to be developed. Organizations may increase the chances of finding and resolving any flaws early in the development process by making sure that all relevant tests are executed and by clearly specifying the objectives, test coverage, and success criteria for each testing phase. Furthermore, it is recommended to include stakeholders in the test planning phase in order to obtain insightful information and align testing activities with the overall project objectives and user expectations.

In addition, It's also crucial to set up the test environment correctly and take care of any compatibility, dependency, and access concerns before starting any testing. This entails configuring the required hardware, software, and network settings to faithfully simulate actual situations. To reduce interruptions during testing, any potential bottlenecks or restrictions in the test environment should be found and fixed. Organizations can improve the accuracy and repeatability of test findings, facilitating more efficient fault detection and resolution, by guaranteeing a stable and dependable test environment.

Team members who are involved in test planning procedures should also keep lines of communication open during the testing process. A common understanding of testing objectives, roles, and status reports is fostered by team members' regular and open

communication. It makes decision-making, coordination, and problem-solving more efficient. Creating avenues for collaboration—such as frequent team meetings, specialized communication apps, and document repositories—allows for efficient knowledge exchange and promotes a collaborative culture.

In summary, companies can improve the efficiancy of testing operations, which will improve software quality and lead to successful project outcomes, by placing an emphasis on precise pre-definition of requirements, a thorough testing approach, stakeholder involvement, proper test environment configuration, and open lines of communication.

6 Conclusion

In Conclusion, this research study has shown a number of noteworthy advantages and developments in the most recent testing techniques. A questionnaire was used in order to gather important ideas and suggestions and an automation test script was created.

First of all, the questionnaire gave participants a chance to submit information and comments, which made it possible to thoroughly analyze the effectiveness of the latest testing techniques as well as their bottlenecks. The results demonstrated the value of automated testing.

Second, the creation of the automated test script demonstrated how the findings were put to use in real-world scenarios. The testing procedure was made more efficient and error-prone by automating it. This demonstrated how automation technologies can improve software testing procedures in addition to validating the efficacy of the suggested approach.

In addition, this study journey has been incredibly fulfilling for me personally and has helped me advance both intellectually and professionally. The intense procedure of carrying out this study has improved my Robot Framework and Powershell proficiency. In addition, my enthusiasm for software testing has been stoked by the research experience.

In conclusion, this thesis has advanced our understanding of the effectiveness of contemporary testing methodologies by offering insightful information via the questionnaire and demonstrating the real-world use of automation through the creation of an automated test script. The industry may benefit from the research and discoveries, and my career path has been impacted by the personal growth I've experienced.

All things considered, this study has set the stage for more research in this area, and it is intended that future academics will be motivated to expand on these discoveries to improve software testing.

7 References

Alli, M. Z. (2007). *Tech Target*. Získáno 7. March 2023, z https://www.techtarget.com/searchsoftwarequality/tip/How-to-write-an-effective-test-report

Black, P. (2005). Test Plan, Test Design, and Test Case Specification.

Chernak, Y. (2001). Validating and improving test-case effectiveness. *IEEE Software*, 81-86.

Garousi, V. a. (2016). When and what to automate in software testing? A multi-vocal literature review. Information and Software Technology,. 92-117.

Gayathri, M. (2022). *Full Stack testing: A Practical Guide for Delivering High Quality Software.* O'Reilly Media.

Gürbüz, A. K. (2007). Pitfalls in Software Testing Tool Selection In III. National Software Engineering Symposium, Ankara. *1*.

Hourani, H. H. (2019). The impact of artificial intelligence on software testing. 565-570.

IDTheftCenter. (2022). *Data Breach Anual Report.* ID Theft Center.

J.Irena. (2008). Software Testing Methods and Techniques. 30-35.

Jorgensen, P. (2013). *Software testing: a craftsman's approach.* Auerbach Publications.

Kaner, C. F. (1999). Testing computer software.

Kramer, M. (2018). Best practices in systems development lifecycle: An analyses based on the waterfall model. 78-84.

McCormick, M. (2012). Waterfall vs. Agile methodology.

Miller, E. F. (1981). Introduction to Software Testing Technology", Software Testing & Validation Techniques. 4-16.

Myers, G. S. (2011). *The art of software testing.*

Nidhra, S. a. (2012). Black Box and White Box Testing Techniques - A Literature Review. *International Journey of Embedded Systems and Applications*, 29-50.

Prawin, M. (24. 05 2023). *Software Testing Trends in 2021*. Načteno z 2021: https://medium.com/tilicholabs/software-testing-trends-in-2021-10ad571c42d8

ProfessionalQA. (2018). *ProfessionalQA*. Získáno 7. March 2023, z https://professionalqa.com/test-execution

QUADRI, S. U. (2010). *Effectiveness of software testing techniques on a measurement scale.*

Rasnusson, J. (2016). *The way of the Web Tester: A Beginner's Guide to Automating Testing.* Pragmatic Bookshelf.

Shetageri, V. a. (2016). A cross-sectional study of depression and stress levels among school teachers of Bangalore. *Journal of Dental and Medical Sciences* , 21-27.

Swebok, A. (2004). *Guide to the Software Engineering Body of Knowledge, project of the IEEE Computer Society Professional Practices Committee.*

Taley, D. a. (2020). Comprehensive study of software testing techniques and strategies: a review. 817-822.

Tassey, G. (2022). The Economic impacts of Inadequate Infrastructure for Software Testing. 169-172.

Testim, B. (2022). *Testim*. Získáno 7. March 2023, z https://www.testim.io/blog/test-case-design-guide-for-qa-engineers/

Umar, M. A. (2019). Comprehensive study of software testing: Categories, levels, techniques, and types. *International Journal of Advance Research, Ideas and Innovations in Technology*, 32-40.

8 List of pictures, tables, graphs and abbreviations

8.1 List of pictures

8.2 List of tables

8.3 List of figures