1862

RIGA TECHNICAL UNIVERSITY

Padmaraj Nidagundi

SOFTWARE TESTING STRATEGY UTILIZING LEAN CANVAS MODEL

Doctoral Thesis

RTU Press
Riga 2022

RIGA TECHNICAL UNIVERSITY

Faculty of Computer Science and Information Technology
Institute of Applied Computer Systems

Padmaraj Nidagundi

Doctoral Student of the Study Programme “Computer Systems”

SOFTWARE TESTING STRATEGY UTILIZING LEAN CANVAS MODEL

Doctoral Thesis

Scientific supervisor
Professor Dr. sc. ing.

MARINA UHANOVA
Professor Dr. habil. sc. ing.
$\boxed{\text{LEONĪDS NOVICKIS}}$

RTU Press
Riga 2022

Nidagundi, P. Software Testing Strategy Utilizing Lean Canvas Model. Doctoral Thesis. – Riga: RTU Press, 2022. – 181 p.

Published in accordance with the decision of the Promotion Council “RTU P-07” of August 25, 2021, No. 21-7.

DOCTORAL THESIS PROPOSED TO RIGA TECHNICAL UNIVERSITY FOR THE PROMOTION TO THE SCIENTIFIC DEGREE OF DOCTOR OF SCIENCE

To be granted the scientific degree of Doctor of Science (Ph. D.), the present Doctoral Thesis has been submitted for the defence at the open meeting of RTU Promotion Council on 21 March 2022 at 14:30 at online https://rtucloud1.zoom.us/j/93987854650 Riga Technical University.

OFFICIAL REVIEWERS

Professor, Dr. habil. sc. ing. Jānis Grabis
Riga Technical University

Dr. sc. ing Igor Lemberski
Ventspils University, Latvia

Assoc. Professor Dr. Dejan Jokic
International BURCH University, Bosnia & Herzegovina

DECLARATION OF ACADEMIC INTEGRITY

I hereby declare that the Doctoral Thesis submitted for the review to Riga Technical University for the promotion to the scientific degree of Doctor of Science (Ph. D.) is my own. I confirm that this Doctoral Thesis had not been submitted to any other university for the promotion to a scientific degree.

Padmaraj Nidagundi ……………………………. (signature)

Date: ………………………

The Doctoral Thesis has been written in English. It consists of introduction, 4 chapters, conclusions, 45 figures, 23 tables, 10 appendices: the total number of pages is 175.

ABSTRACT

Software testing is a fundamental issue in software engineering and software quality assurance in general. Many software testing strategies are available and are being used today in software testing processes. The testing strategy plays a vital role in the software testing process. Observation of testing strategy creation is a complex task, and in most cases, test documents are drawn in the text format. This is an essential problem, in software testing in particular. This makes the process of test strategy creation and updating in the agile development complicated, thus the resulting testing process takes additional time due to the need to process lengthy test documentation or to coordinate activities of a distributed remote team working on the same project. Without a robust test strategy, software development projects may face overrun budgets, the team may produce bad quality software product, or a project may fail due to software quality issues. In case of agile development, software release cycles are prompt. In order to fill the gap between software testing, test strategy creation, and simplification of the testing process itself, there is a need for visualised representation of the testing strategy.

The main goal of this Doctoral Thesis titled "Software Testing Strategy Utilizing Lean Canvas Model" is develop a visualised test strategy.

In order to accomplish this goal, the visualised test strategy has been created using the lean canvas. The author analysed the existing theoretical approaches, their drawbacks, and many other essential issues in software testing. At the next stage, the author used the principles of user experience design and user interface design to identify the appropriate design possibilities for test strategy creation. Using the lean canvas design, the author approached the problem "visualised testing strategy creation" adopting the novel approach TABB (Test activities building block) for visualised strategy created at this stage. In this process, Ontology and Use Cases were created and a practical experiment was conducted to analyse the results. In the course of research, the author discovered that the visualised strategy improves the testing process, work visibility, reduces lengthy documentation, facilitates test team resource handling, and improves remote team communication in both functional and non-functional testing. The supporting web portal was created to get feedback from individuals. The Doctoral Thesis consists of 175 pages, it comprises 43 figures, 24 tables and 10 appendices. Bibliography includes 199 literary sources.

TABLE OF CONTENTS

INTRODUCTION

Rapidly developing technology has brought many benefits to human life, technology is increasingly integrated within software today. Technology has now become an integral tool in addressing daily issues. Software development has moved towards the agile approach as outlined by Black in [2], and whenever software is developed, it needs to be tested and made free from defects before transferring it to end-users. Well-tested software can ensure customer satisfaction. Software testing has a long history and many authors have contributed to this topic, as discussed in [3], [4], [5] and [8]. A test strategy is a framework that describes the software testing technique in the software development life cycle [105] and [106]. However, in the field of software testing, the use of the lean canvas in software testing process and visualisation of the test strategy at different levels have not been sufficiently investigated, see [134] and [137].

The object of this research is visualisation of the software test strategy. Adopting lean canvas in strategy visualisation process allows reducing the cost of the testing process, speeding up documentation circulation and addressing test communication and management issues that arise during software product development process. It contributes to the reduction of costs in software development and optimisation of the testing process, as outlined in [19].

Statement of the Problem

The idea: According to the software engineering literature [30], [31] and [32], when a code was created, tests were carried out during the same process functionality, and the same developer did this work till the end. This activity was regarded part of the same development activity, i.e., the term *testing* still had no connotation. Just in the late 1950s, developers began to see the process of testing as a different activity in the process of software development. However, in the following decade, in the 1960s, testing started to be seen as an increasingly important task. This happened because specialists began to understand the importance and positive impact of testing on both cost and time. Thus, the author of the Thesis discovered that a rigorous process of detection of deficiencies in the developer’s program was carried out.

Already by the 1970s, more rigorous testing methods were introduced, and more resources started to appear. The term "Software Engineering" was coined, emerging from the subject of

"Software Testing". Testing became vital for those dedicated to the development of applications as a way to ensure that the created software was performing in the way it was intended. In the 1980s, there was a total shift in the way the subject was treated; the concept of "Quality" changed approach to software testing and became the focus of development efforts. According to [8], [11], quality became an important objective to achieve. As a result, software engineers began to use best practices that would lead to creation of quality standards, such as IEEE 829-2008, ISTQB, ANSI and ISO, the software testing process focused on ensuring "quality" of the final product, which led to continuous improvement. Testing methodologies not only test applications in their final stage, but also support development throughout all stages of the development life cycle.

In the 21st century, software development process has become complex. According to [29], complicated business requirements have created new challenges as well as opportunities for development of new solutions for the problems. In the software development life cycle, software testing plays a pivotal role in ensuring that software is defect-free and works with the functionality developed for the end customer. According to [6] rapid development of software and frequent release of new versions of the software have become challenging.

The reality: In the current working numerous Latvia based software development companies there is not any scientific research carried out to visualise software test strategy design, which is still following a traditional documentation plan, and not much investigation has been performed in this area.

For more than one reason, software testing is a time consuming and expensive process [176]. Besides, poorly written test strategy information in test documentation is difficult to interpret and implement within the project, according to [104]. Until now, there are no standard techniques for development of a software test strategy, and the existing strategies do not meet possible software testing criteria in the agile development [104] and [176]. Each software development project is unique and has different development methodology adopted for the project, and each of them has different testing requirements and possibilities.

The consequences: In software development industry, geographically distributed teams are working in different time zones [196]. In this situation, writing a code for complex requirements and testing it have become challenging [6]. At this point, test strategy visualisation offers significant benefits and assist develop the best quality software. Visualising test strategy speeds up the testing process and keeps the team on the same page, as discussed in sub-chapter 4.6.

Topicality of the Theme

The author has used personal theoretical knowledge and practical experience, which was acquired throughout the engineering career in software development, in the development of the Doctoral Thesis. The current research is dedicated to the software testing process, where visualised test strategy is created as a part of the research and is adopted in the test process.

In the numerous Latvia based software development companies run their internal testing processes and maintain documentation, guidelines and tools for software testing.

According to World Quality Reports [164] and [165] published annually, there is a persistent skill gap in test strategy design in the software testing industry. The benefits of the test strategy design include development of testing processes in order to speed up circulation of documentation, to address test management issues and to reduce costs and time of software development. Software development companies it is continue considering test strategy design practice as a separate process. The test strategy design practice is not always seen as too important and is often considered only as an aspect increasing operating costs [196]. A clear and adequate process or test strategy is adopted according to the research needs, as discussed in [104]. The present research intends to propose the lean canvas visualised design tool using test strategy visualisation that can serve these needs.

The research is valid because it proposes a new visualised test strategy for software development, that can reduce length of documentation. In this manner, a visible it is quickly created and managed, monitoring software development life cycles. If software testing is not done appropriately, the customer receives low-quality software, which leads to customer dissatisfaction with the end product. In some cases, organisations incur significant financial losses due to defects in the software.

According to [134], the software development industry requires an appropriate test strategy for the projects according to the scope and requirements of the project. Choosing the proper test strategy is a dominant factor in software test effort according to software test document IEEE 29119. In most cases, a software test strategy plan is a subsection of a test plan.

Within each project, different types of tests and test documentation are used. According to [10], [21] and [16] it is well known that many of these procedures are proprietary and very specific in relation to the company and the product, and complex for a small company to use. Moreover, they are not always well managed, because they use tools or manual updating documents that are not most suitable for test strategy creation for the project.

The research aims to develop a software test strategy design in terms of development of a new visualised strategy. It is based on the components of applications in the developed software or in the process of development focusing on the functionality of the product.

The proposed test strategy should primarily focus on visualization in the overall project. It also identifies the types of testing, essential software components, which are the most significant advantages and the highest risk, respectively, thus marking a difference from the traditional testing strategies. Using the new strategy withing a software development project, companies can support test strategy creation in a visualised form. In practice, adoption of proposed test strategy provides development team members to improvement and optimise their software testing processes in the project.

The adoption of the proposed test strategy design process provides a significant benefit to the software development life cycle to reduce the documentation effort. According to [164] and [165], software development companies continue viewing ‘test strategy design’ practice as an inessential part of the process and in order to address this problem, it is necessary to bridge the skills gap. In the investigation process, the author found that in Latvia based software development companies, the strategy design process is long documented, not updated according to the changing requirements, or is not adopted at all. The author of the research intends to propose the visualised software test strategy as a tool that can serve this type of company and, increasingly, promote the use of proper software testing practices. Adoption of the present research outcomes contributes to reducing the cost of testing in order to speed up circulation of documentation and to be able to address test management issues in software testing.

Aim of the Doctoral Thesis

The aim of the Doctoral Thesis is to develop a visualised software testing strategy utilizing lean canvas model. Software quality improvement is still a significant challenge in different types of software development life cycles (SDLCs) for many organisations, and it gets critical primarily when distributed teams work remotely and test complex products, according to [30]. The software testing process involves many activities, including different levels of testing and different resources. The author intends to scrutinize the possibilities for software testers to develop the visualized lean canvas approach and to examine its effect on the process of communication, collaboration, and circulation of documentation.

Tasks of the Doctoral Thesis

In order to reach the primary aim of the Thesis, the following tasks have been completed:

- To explore the existing software testing processes and test strategies, to analyze the need for visualisation in strategy creation;
- To identify the main advantages and disadvantages of the recent software strategies and to analyse strategy visualization;
- To develop visualised test strategy based on different real-life projects and software requirements;
- To develop criteria to evaluate the proposed visualised test strategy;
- To implement the visualised software test strategy in software development projects and to collect feedback;
- To investigate and document the benefits and limitations based on the practical results, to collect recommendations from other project and team members;
- To define the possible further development scope of the proposed strategy.

Object and Subject of Research

The object of the research is visualisation of the software test strategy.

The subject of the research is development of the visualised software test strategy within different software development life cycles and types of software testing.

Research Hypothesis

Software test strategy involves such stages as plan, design, execution, analysis and reporting within the software development life cycle (SDLC). Thesis statements to be defended:

- A visualised software test strategy improves the test process, work visibility, test team resource handling and remote team communication in both functional and non-functional testing;
- The proposed approach will help reduce the lengthy documentation effort and create an effective agile test strategy in a short time in a more visible way.

The hypothesis to be proven within the Doctoral Thesis is as follows: To overcome the above-mentioned challenges, the visualised test strategy has been purposed, adopted and evaluated.

Research Methodology

The following methods have been used in the Doctoral Thesis.

This research starts with the analysis of the existing different software testing strategies and their documentation. Qualitative research method (exploratory and observation) is used to find the strengths and weaknesses of different ways of visualizing the software testing strategy. Six different design options for strategy visualisation were analysed and limitations were identified, which are listed in Section 2.1. Next, this research developed the software testing strategy visualisation using the Lean Canvas testing activity building blocks in Section 3.2.

Requirement analyses done to collect the system design requirements, technology, system users, use cases to study and develop prototype system. In this research, the author collected the system design requirements, technology, system users, use cases to study and develop prototype system. From the collected requirements, the web portal was simulated in section 3.3.

The quantitative and qualitative methods (survey-based questionaries, online user reviews, use case method) are used data collected to determine the strengths and limitations of the author's proposed approach to visualising the testing strategy. Quantitative research methods to analysis of the case study results to analyse the length of documentation.

Finally, the analysis of the results of the case study resulted in the recording of the amount of documentation (up to) in pages after the implementation of TABB and the amount of documentation (up to) in each software version in hours.

Many of the processes of theoretical calculations and graphical representation of the results have been obtained utilising a menagerie of software systems, including:

• Wordpress.com (PHP framework for developing a content management system);

• MySQL (storage of user information and content management system);

• Github.com (version control for source code maintenance);

• Atlassian.com (software development and collaboration tools for issue and project tracking);

• Google drawings (TABB creation, storage, and updating of building block templates for testing activities);

• Google forms (survey and data collection);

• Microsoft word (analysis of existing documents and creation of new documents);

Scientific Novelty

Scientific novelty of the Doctoral Thesis:

- A new approach to test strategy visualization has been developed for implementation of software test strategy with the lean canvas;
- A new approach - TABB strategy (Test activities building block) has been used for finding the right test metrics for the lean canvas visualized board using transformation models defined;
- The new strategy has been evaluated within different projects and data have been collected.

Practical Significance

The author intends to implement the TABB strategy in the projects and collect constructive feedback in order to understand the research impact. The author has collected criticism and statistical data for the improvement of TABB strategy.

- Criteria for the TABB strategy have been developed. It has been explained how to identify the lean test metrics for the lean canvas board;
- Experiments have been conducted with an aim to utilize the lean canvas test strategy in software development projects in Latvia based software development companies;
- The collected practical results can be used in future by the software companies to improve software development and testing processes, mainly using test activity building block strategy on the lean canvas.
- The open-source project has been created, where the integrated strategy models can be used.

Approbation of Research Results

Research results have been presented at 8 international conferences in the Czech Republic, Ukraine, Lithuania, Austria, and Latvia.

1. Symposium for Young Scientists in Technology, Engineering and Mathematics, Kaunas, Lithuania, 28 April 2017;
2. Environment. Technology. Resources. Proceedings of the International Scientific and Practical Conference. Rezekne Academy of Technologies, Rezekne, Faculty of Engineering, Latvia, 29 April 2017;
3. Proceedings of the IRES International Conference, Vienna, Austria, 26 November 2017;
4. XII-th International Scientific and Technical Conference “Computer Science and Information Technologies”. Lviv, Ukraine, 6 September 2017;
5. CSOC 2017, Faculty of Applied Informatics, Tomas Bata University in Zlin, Czech Republic, 27 June 2017;

6. RTU 58th International Scientific Conference, Riga Technical University, Riga, Latvia, on 13 October 2017;

7. International Conference on Information Technologies, IVUS 2018; Kaunas; Lithuania, 27 April 2018;

8. RTU 59th International Scientific Conference, Riga Technical University, Riga, Latvia, 11 October 2018.

The results of the research have been published in the following publications:

1. Nidagundi P., Introduction to Investigation and Utilizing Lean Test Metrics. In: Agile Software Testing Methodologies. Int. Journal of Engineering Research and Applications. ISSN: 2248-9622, Vol. 6, Issue 4, (Part - 1) April 2016, pp.13-16. (INDEXED: Google Scholar).

2. Nidagundi P., Novickis L. Possibilities about the design lean canvas model and its adaptation in the agile testing. Symposium for Young Scientists in Technology, Engineering and Mathematics, Volume 1853, SYSTEM 2017, Kaunas, Lithuania, 28 April, pp. 20-23. (INDEXED: SCOPUS, Google Scholar).

3. Nidagundi P., Novickis L. Introduction to Lean Canvas Transformation Models and Metrics. In: Software Testing, APPLIED COMPUTER SYSTEMS, Year 2016, pp. 30-36. (INDEXED: Google Scholar, Web of Science).

4. Nidagundi P., Lukjanska M. Introduction to adoption of lean canvas in software test architecture design. Computational Methods in Social Sciences, Volume 4, Number 2, 2017, pp. 23-31(9). (INDEXED: Google Scholar, ProQuest, RePEc, DOAJ, EconPapers, Index Copernicus International).

5. Nidagundi P., Novickis L. Introducing Lean Canvas Model Adaptation in the Scrum Software Testing, Procedia Computer Science. Volume 104, 2017, pp. 97-103. (INDEXED: SCOPUS, Web of Science, ScienceDirect, Google Scholar).

6. Nidagundi P., Novickis L. Towards Utilization of Lean Canvas in the DevOps Software. Environment. Technology. Resources. Proceedings of the International Scientific and Practical Conference, ISSN 2256-070X, June 2017, pp.107-111. (INDEXED: SCOPUS, Google Scholar).

7. Nidagundi P., Novickis L. Towards Utilization of a Lean Canvas in the Testing Extra-Functional Properties, Software Engineering Trends and Techniques in Intelligent Systems June 2017, pp. 349-354. (INDEXED: SCOPUS, Web of Science, Google Scholar).

8. Nidagundi P., Stepanova V. Survey on Software Test Strategy. Proceedings of the IRES International Conference, Austria, Vienna, November 2017, pp. 26-27. (INDEXED: Google Scholar, WORLD RESEARCH LIBRARY).

9. Nidagundi P., Novickis L. New method for mobile application testing using lean canvas to improving the test strategy, XII-th International Scientific and Technical Conference "Computer Science and Information Technologies". Sept. 2017. (INDEXED: IEEE Xplore digital library, SCOPUS, Web of Science, Google Scholar).

10. Nidagundi P., Novickis L. Towards Utilization of a Lean Canvas in the Biometric Software Testing. A JOURNAL OF MULTIDISCIPLINARY SCIENCE AND TECHNOLOGY, IIOABJ, 2017, Vol.8 (Supple 3), pp. 32-36.

11. Nidagundi P., Uhanova M. Software application security test strategy with lean canvas design, International Conference on Information Technologies. April 2018, pp. 50-53. (INDEXED: SCOPUS, Google Scholar).

Outline of the Doctoral Thesis

The Doctoral Thesis consists of the introduction, four chapters addressing theoretical and practical issues, conclusion, bibliography, and appendices. The volume of the Doctoral Thesis is 175 pages, it contains 43 figures and 24 tables, there are 199 references in the bibliography.

The *Introduction* explains the motivation and purpose of the research and defines the research goal. It presents the aim, research hypothesis, research methods, practical implementation of the research, approbation and the main results.

Chapter 1 is devoted to the history and current industry practices in software testing and software test management. This chapter demonstrates that software testing is a vital part of the software development process. In the software development life cycle, software testing process depends on the requirement; according to software requirements, the tester needs different skillsets and process knowledge. In terms of technology change, process change, or people change in the software development process, software testing principles still remain stable. This chapter also considers statement of the problem, different methodologies in software development and identifies different test strategies and their limitations.

Chapter 2 visualized approaches are described, and it devoted to the introducing the existing business lean canvas design. The author investigates the solution from personas perspective, as well as considers interconnection between software testing and UI/UX, user experience evaluation techniques and lean whiteboard design.

Chapter 3 demonstrates ontology development and visualised test strategy generation. TABB (Test activities building block) is introduced, explanation and system design use case are developed. The pilot project is conducted and survey feedback is gathered; eventually, research data are evaluated, considering the importance of system design and its limitations.

In *Chapter 4* different types of development and testing project case studies are carried out and experimental evidence is collected.

Conclusion and Scope of Future Research summarizes the outcomes of this Doctoral Thesis - results of analysing the existing test strategy, results of new models used in the experiments and their strengths and weaknesses are evaluated. Further scope of the research on this topic is also discussed.

There are ten *appendices* to the Doctoral Thesis:

1. List of abbreviations;
2. List of figures;
3. List of tables;
4. Classroom experiment work by students of Riga Technical University (RTU);
5. Creation of a generic test activity building block template in GitHub;
6. Creation of a test strategy resource website for software testing community.
7. WWWWWH questionnaire survey documentation for test strategy.
8. STD (story, development, testing) test strategy
9. Company test process overview
10. Mathematics formulas

1. RESEARCH ON SOFTWARE TESTING AND TESTING MANAGEMENT

The work focuses on the detailed research of the proposed theme, study and analysis of the documented processes, strategies and test techniques.

The exploratory research on the possibilities of visualization of software test strategy creation has been conducted. [1],[2],[3] and [12] state that exploratory analysis seeks to discover new theories and practices that modify the existing ones.

The research based on the analysis of bibliographical references, such as scientific books and articles and the Internet, using definitions of processes, strategies and test techniques and the existing software testing processes has been carried out. The author intended to explore fundamental software processes, tools, actors, quality insights, test matrices and modern trends, such as agile, continuous delivery for testing. This research is dedicated to finding the possibilities of testing strategy visualization and gathering different essential viewpoints on testing. In the next chapter, the author tries to review these essential topics.

1.1 Definition of Software Testing

Software needs to be tested before it is transferred to the intended end customer's use. Despite intense software development effort, software defects are likely to occur.

Software testing try to find possible defects that a newly developed or already program may contain. These defects need to be corrected before software is released in the market making it available for public use. As a result of literature analysis of [1],[12],[5],[8] and [13], it has been found that there are more than ten different definitions that explain the concept of software testing. The common elements of the identified definitions have been considered and, as a result, a general definition of software testing and management process has been proposed.

Software testing includes different processes that identify different steps and methods to ensure product quality and to meet the stakeholder business objectives. Software testing focuses on evaluating software components under test.

Software testing fulfils the following main tasks:

- Software should meet business and technical requirements;
- Software works as expected;

- Software should meet stakeholder needs;
- Possibly shippable products are delivered to end customers;
- Application needs to work and react according to business requirements.

The definition of software testing is decomposed into the following parts to get an in-depth perspective.

- Types Of Strategies - [Analytical Strategy, Model-based Strategy, Methodical Strategy, Process Strategy, Dynamic Strategy, Consultative Or Directed Strategy, Regression-averse Strategy And Reactive Strategy];
- Comparison Of the Strategies Key Factors - [Risk factors, Type of testing, Skills of the testing team, Scope and objectives, Regulations];
- Software Development Methodologies - [Waterfall, Scrum, Extreme Programming, Test Driven Development, Prototyping, Spiral, Agile];
- Test Process Management - [TMap, ISO-IEC Organizational Process];
- Different Test Documentation - [Project-Based Type of Test Documentation, Different Test Level Documentation, Documentation In Software Development Life Cycle];
- Software Testing Complexity - [Human Resources, Technology, Method, Tools];

Examining sources [14], [15] and [17], the primary software testing definition and main tasks have been found out. Analysis of different literature is aimed at developing a possible strategy in the Thesis.

This sub-chapter has clarified software testing definitions collected by means of literature review and has provided a more comprehensive view of the software quality. In the next sub-chapter, the author presents the overview of test strategy.

1.2 Test Strategy Overview

Definition of the test strategy is as follows: A test strategy is a framework that describes the software testing technique in the software development life cycle.

It is designed to inform project managers, team members, testers, and developers about some essential issues of the testing process. In software testing process, the software test strategy is adopted either proactively or reactively [36].

a) Proactive strategy adaptation - In this process, the test strategy created in the early stage to discover and fix software defects before the software structure is created.

b) Reactive strategy adaptation - A process software testing strategy is not created until user interface design and coding are completed.

In a physical system, defects occur due to design defects, not due to manufacturing defects. If the software is not affected by design defects, it is affected by updating the code or changes in the system. Software defects always exist at some level in the code because not only due to the carelessness of a programmer or too many details that should be considered, but also due to complexity of the software and limited capacity of humans to scrutinizing it, as outlined by Spillner in [1]. Similar situation is observed in the physical system due to the complexity of products, hence, defects can occur at some time. Software can fail in many ways and detecting possible different failures is impracticable.

According to [70] steps should be made towards software quality continuous improvement. Nowadays software is used in both simple and very complex applications, if defects are found in critical applications very late, they can be costly to fix [82]. Software defects can cause damage to human life in a space shuttle, aeroplane, nuclear power plants, as well as cause financial loss in complex stock market applications.

In general, quality means conformance, when software fulfils the product or system requirements. Precise conditions also lead to the quality of software building.

According to [7] there are many pragmatic practice approaches that can be adopted to improve software quality, such as predefining quality criteria, identifying quality metrics, having a well-organised team, identifying the precise requirements, testing potential risks and most crucial areas first, developing a transparent and straightforward code that is easier to check. Redesign of the system using an automation tool reduces the number of repeated testing life cycles. While speaking with colleagues about strategies used in the industry, many questions cropped up, for example: What does this strategy do? Why do you use this strategy? What kind of strategy is it and how to design a strategy?

Thus, during the research considering [104], [105] and [106], many test strategies were investigated to find the answers to the above-mentioned questions and to determine the purposes of use and creation of various testing strategies. It is also necessary to clarify the differences among them.

The author has considered the definitions provided in ISO/IEC 12207, CMMI, IEEE 829, ISTQB and other resources, but they provide high-level description of test levels to be performed and describe testing within those levels for an organisation. Figure 1.1 presents the process of evaluation of software quality in ISO and CMM standards [168]. However, it was not sufficient for the purposes of the present research since these definitions are not specific enough. In software testing, one of the most critical missions is to collect information about the system to help managers run it. According to [30] and [31], within the projects, sometimes other people make decisions, and two questions arise: How do researchers uncover the most critical information about the system? Moreover, how do they do it efficiently and effectively. There are many constraints the author wants to consider, mainly they concern risks. The author considers that creating a model is a complex task.

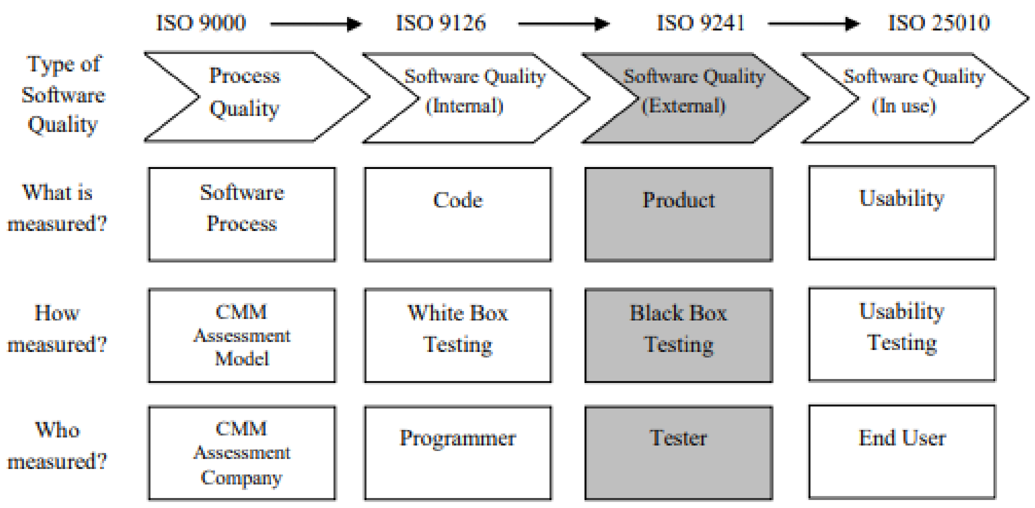


Figure 1.1: Evaluation of software quality [168].

In order to solve this problem, the author considered such missions or problems as what kind of information is available, what the most significant risks are, what the product owner wants to know [9]. For the author, the strategy is a solution to a complex problem. All testers use large test plans, which involves much documenting of the system. In most cases, a test strategy is created by test managers. It has been observed that usually team members agree to this strategy without asserting the proposed model. The author observed that after adopting the strategy proposed by the manager, team members struggle with this model, fist adopting it and then updating the test strategy document, which is a time-consuming effort.

1.3 Types of Strategies

In this sub-chapter, the author analyses the existing software test strategies. The different test strategies, such as an analytical strategy, model-based strategy, methodical strategy, process strategy, dynamic strategy, consultative or directed strategy, regression-averse strategy and reactive strategy in practice in the industry [34], [93], [95] [105] and [106]. All these strategies are described and analysed in this sub-chapter.

Test strategies are often used to cover both functional and non-functional test, which makes it easier for the tester to do better work by using the software toolset. When writing test strategy, there are many different ways of doing it; not everyone has the same method. Some people might use an existing test toolset, while others may prefer to create their own set of methods.

Analytical Strategy

This strategy focuses on the meaningfulness and discovery of test process design. In this process, different data are used, including historical test data. This process involves detailed research of the project documents, stakeholder input, essential planning, test estimations, risk, a human resource available for the project and previous experience with similar projects, such as collection and analysis of different multidimensional data within the test strategy created. A test strategist needs to be good at mathematics and statistics to put multidimensional data in one format.

There are many different types of risks such as tracking for the resources, estimation of time that exist. It is not possible to incorporate all of them into the analytical test strategy as they can cause an extensive amount information. It is important to be aware of the common risks which have been identified as well as those which have been documented in literature so that they can be effectively used as part of analysis [34].

An analysis is a process which aims to break down the problem or a situation through close examination and a study of underlying reasons rather than what is immediately obvious. An analytical test strategy is an analysis of the project which has been conducted to find out where risk factors exist and what can be done to remedy them.

Model-Based Strategy

In this strategy, the model of the system, runtime behaviour implementation under the test is checked against predictions made by a formal specification or model in terms of the general picture. The strategy state transition diagram (finite-state models) has been derived from the existing case studies or user cases. In this process, algorithms are used to extract the test cases, and these test cases execute system against the test. In the model-based test strategy, one can use formal or informal models for assessment of critical system behaviour [34]. The benefits of adopting the model-based strategy, such as automation of test case generation, a wide range of combinations of test cases, it is easy to manage the test cases, early defect detection, testing big projects with many test cases which cover all software [197] and [198].

Methodical Strategy

In this process, tests are well organized and pre-planned, for example, adopting ISO 9[34], 6 testing process for the significant part of the test process, and sometimes this process might be developed and adapted using in-house development and test practices. External ideas may also be adopted later for improvement of the testing [34].

It is important to design a test strategy considering various regulations in order to comply with all applicable regulations. Regulatory compliance is crucial for every business sector. Businesses are expected to meet stakeholder needs including governmental expectations, competitive demands, technological trends and customer expectations. It is important to consider standardization and industry testing standards in order to help ensure consistency, avoid possible misconceptions and allow for effective data analysis.

Process Strategy [standard-compliant]

In this strategy approach, users follow the process, for example, if the project is running under agile methodologies, such as scrum, then the testing strategy and standards are adopted according to the Scrum principles to improve the test process over time along with project growth.

This is focused primarily on regulations, but the discussion is still applicable to other processes because regulations, by definition, are set up to follow certain standards.

Regulations are inspected and approved by knowledge workers, who are part of the customer's organization. Once they are passed, these regulations will not be changed unless there is a budget or funding increase. Regulations affect testers in that they set up certain standards that must be met during the testing process [34] and [106]. The test process will be implemented by a variety of people. The stakeholder or person who will begin it will pass on the information to workers who will convert the information to information that is not only presentable but also able for software development team. This can be someone from outside of testing, such as a project manager, or from the testing team itself.

Dynamic Strategy

According to [34] Initially within this strategy, the basic set of lightweight testing guidelines is defined, and weaknesses for example, low performance of the software application of the software are identified.

Dynamic Testing Strategy covers risk factors by identifying various levels of risks to be mitigated either through light-weight techniques or exploratory techniques. In this manner, tests can be performed in a manner that allows for re-use of test data to identify any patterns that have been identified, thus allowing thorough analysis to be completed. After completing every phase of software testing, exploratory testing is done for the identified weaknesses of the software.

Consultative or Directed Strategy

In this strategy, non-testers may be consulted on the aspects that should be focused on in testing, for example, a developer and an end-user may be asked what specifically will need to be tested in the later stages [34] and [93].

The consultative test cover provides credible evidence of how their efforts are received by the rest of the organization before execution can proceed. There are circumstances when strategic test design is not appropriate due to time constraints or conflicting interests within an organization. However, where there is time and space for strategic test design to take place, the consultative test cover technique may be preferable.

Regression-Averse Strategy

It is a risk-based technique for regression testing. For example, all test case regressions for the previous release of the software and the least automated new software functionalities can be automated [34].

Regression testing can be described as giving the test software the ability of repeating sequential steps of its operation repeatedly. An example would be manually clicking through all of your favourite apps on your phone until you find the one that doesn't crash. This assist to identify any changes that might have occurred within the code or between releases, which then allows for a team to decide what changes need to be made before moving forward with a new product or release. By doing this testing, the chances of a product failing in the future can be lowered, leading to happier users and secure products. Regression testing should be performed on all code after every update and before any release.

Regression testing is a complex and often overlooked area of quality assurance (QA) as it plays such a major role in the overall security and stability of any software [37]. When performing regression testing, QA professionals make sure that the code runs as expected after every update and before every release, which makes it possible to save time and money in case there are any bugs to fix. However, regression testing is an extremely time-consuming process, and if not executed correctly, can result in failing tests that could have been prevented through early detection.

Regression-Averse test strategy focus on extensive automation of the testing process, as this allows effective and efficient end-to-end testing including possible risk-based scenarios.

Reactive Strategy

To apply this model, a test team waits for the design and implementation of the test of software components, when they are available for testing [34] and [22]. The components include units, classes, modules, and features. The development team plans and schedules each test and determines which pieces to test, and when within the overall process. This document will help the team to determine what risks there are with the use of this strategy that it will be able to overcome if they want to. In this process time is additional in this case possible risk-based scenarios identified.

Standard-Compliance Strategy

Within this test strategy, a team strictly follows standards, for example domain-level standards, organization standards. Compliance testing is any form of auditing that establishes conformance to a policy, regulation, or rule. Compliance testing may be used to test the quality of equipment and processes, including safety aspects. It can also be applied to personnel management practices [34].

Compliance testing is executed to find out if the particular product or process is conforming to that statutory or legal requirement or any other regulation. In this way it differs from quality assurance testing, which seeks only to ensure that a product meets its specified requirements. The test results may be used as evidence in a trial. In financial services, compliance testing of personnel has been used to restrict persons from holding certain financial management roles if they do not meet specific criteria, such as having a minimum level of education and certification.

1.4 Comparison of the Strategies

In this sub-chapter, the author created Table 1.1, which presents strategy types of key factors such as, risk factors, business goals, necessary skills of the testing team, objectives, regulations.

The different key factors affecting software testing strategies are selected according to [35].

Risk factors: Risk management plays a crucial role in testing allowing to analyse what type of risk persists and at what level.

Type of testing: Software testing is classified mainly into two subcategories (functional and non-functional), and the strategy needs to support any.

Skills of the testing team: A tester needs to have skills to create a test strategy, sometimes it is challenging to devise a strategy due to lack of knowledge.

Scope and objectives: System testing should meet the stakeholder’s predefined requirements and to achieve this, the testing team needs clear scope and objectives.

Regulations: In some businesses, the testing process should meet the stakeholder needs in compliance with certain regulations.

After considering the above-mentioned strategies and having selected the most appropriate strategy, it is necessary to appoint a right talented person who will execute the strategy, otherwise, the strategy will soon fail.

Table 1.1

Strategy Types and Standards

Strategy type	**Risk factors**	**Type of testing**	**Skills required from the testing team**	**Scope and objectives**	**Regulations**
Analytical	Yes	-	-	-	-
Model-based	-	Yes	-	-	-
Methodical strategy	-	-	-	-	Yes
Process- or standard-compliant strategies	-	-	-	-	Yes
Dynamic strategy	Yes	-	-	-	-

Strategy type	**Risk factors**	**Type of testing**	**Skills required from the testing team**	**Scope and objectives**	**Regulations**
Consultative or directed strategy	No	-	-	-	-
Regression-averse	Yes	-	-	-	-
Reactive test strategy	No	-	-	-	-
Standard-compliance test strategy	-	-	-	-	Yes

In the previous section, the author explained different types of test strategy and different factors that should be considered in selecting the appropriate strategy (see Table 1.1).

1.5 Author's Point of View on Test Strategy

Defining the strategy approach and quality assurance objective ensures seamless communication between stakeholders, quality assurance specialist, and developers in order to create a good quality software product. Having a detailed scrutinizing of the strategy that focus on what need to be tested and what not is the first step to deciding on the strategy.

At the initial stages of software engineering, qualitative documentation creation is the most important basic practice to be implemented during software development.

Considering the documentation for software test strategy, in the initial stage it contains the less page length, and in the later stages, it needs to be changed further frequently according to the changing requirements and scope of the project. Moreover, a tester starts testing and starts exploring, and thinks about the discovery of further information over time. Respectively, the test strategy grows in volume. Over time it becomes very difficult to manage such large document.

According to [44] and [183] the current industry trend is agile process manifesto, which focuses on building the working software over comprehensive documentation. In the author's point of view, it is necessary to think about possibilities of visualising the software test strategy itself. The visualised strategy is supposed to highlight the risks in testing, possible components of the test strategy document, as well as to increase the impact of team communication on reducing the testing costs and documentation effort.

The author of the Thesis suggests approaching problem-solving in a visualised way. The author addresses such problems as what the most significant risks are, what the product owner wants to know and what goal is to be reached [9]. In this way, the strategy offers a solution to a complex problem of how to meet the information needs of the stakeholders and how to do that in an efficient way. There are many aspects to consider on the way to a software test strategy.

Deliberations Before Creating the Test Strategy

Several important issues should be considered apart from risks, including also value.

- What makes a particular product or testing valuable for the customers?
- What are the stakeholders' goals and the missions to be achieved?
- What type of test is needed to deal with the software application?
- Which techniques are preferred?
- Which element the application poses risk in terms of testing?
- Establish the information sources, recognize the imported problem sets such as quality objectives or quality criteria;
- What is the type of software defect – functional or non-functional?
- Who is a product customer and how do they intend to use the products?
- What do the team members think about testing and what team skills will they apply to the types of testing?

This sub-chapter has provided an introduction and a broader view of the possible software test strategy considerations. The author's theses are mainly focused on test strategy development for agile methodology. According to [183], IEEE 29119 focuses on long software test documentation, whereas agile development process focuses on the lightweight documentation process. Figure 1.1 has presented different ISO standards for software testing and provides evaluation of software quality. In the end, the author's point of view on test strategy has been explained and deliberations to be taken into account before creating the test strategy have been discussed.

1.6 Software Development Methodologies

The test strategy depends on software development methodologies, and this sub-chapter describes some popular development methodologies [165]. According to [22], [53] and [133], the traditional methodologies are more documentation-oriented. They were used extensively in the past in the context of software development very different from the current one, based only on the mainframe and dumb terminals.

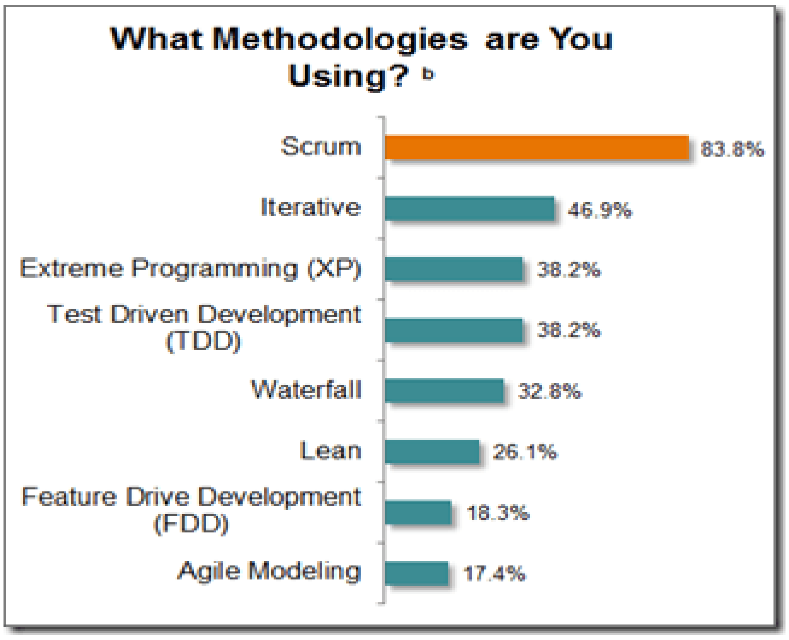


Figure 1.2: Most frequently adopted methodologies according to World Quality Report 2016 [165].

Waterfall Methodology

The waterfall methodology created to minimise the problems highlighted above is the cascade methodology. It represents a breakthrough in software development, especially if one considers that before the appearance of the first methodologies there used to be a persistent problem commonly known as "code breaks", when faulty or failure to comply with the requirements led to successive corrections, defects often led to the failure of the project.

Waterfall methodology is also known as sequential or linear methodology [53]. It is based on the succession of steps where each new step is initiated after the end of the one immediately preceding it. Development flows from the top - from systems engineering towards maintenance. Within this methodology, initially the developer tries to gain in-depth software development requirements of the problem to be solved, its requirements and its restrictions; then solutions are designed to meet all requirements and constraints. Once this step is completed, the implementation of the project begins, and when the entire implementation stage is completed, it is verified with the client whether the solution meets the established requirements, and finally, the product is delivered.

Scrum Methodology

Scrum is just one out of many iterative and gradual agile software development methods, according to [40], [42] and [60]. In the SCRUM methodology, sprint is the basic unit of development. Each sprint starts with a planning meeting, where sprint tasks are identified and anticipated commitments to sprint goals are made. The race ends with a review or session to review the progress and confirm the lessons for the next sprint. During each sprint, the team creates the finished part of the product.

In an agile methodology, every iteration requires a team to work through a complete software development cycle, including outlining, requirements investigation, design, coding, unit testing, and acceptance testing, when the work product is demonstrated to stakeholders.

In the development process, the team follows all steps in the SCRUM sprint, from requirements analysis to acceptance testing. It may be stated that the SCRUM sprint corresponds to an AGILE iteration.

Extreme Programming Methodology

Extreme programming is an agile software development methodology. According to [43], [44] and [52], it implies the process of creating software in an agile and productive way. It does not specialize in the management of the development process but mainly focuses on the engineering practices required to deliver the software.

When adopting scrum for software development purposes, engineering practices are taken from agile software development disciplines, most often from Extreme programming (XP).

Certain practices are adopted in a way that is separate from the rest of the development practices. The most frequently used methods are test-driven development, refactoring, pair programming, user stories, etc. which are not required in Scrum and are the only way to complete the task.

Test Driven Development Methodology

According to [49], [51] and [55], the concept of TDD at specific behavioural levels is implemented as follows. The first thing to do is to write a test and write something to pass the test, that is, write the test first, write the code that passes through it, and iterate it over to get feedback on how it works.

Test-driven development (TDD) is one of the existing developments, which creates test cases after the code design procedures are implemented for refactoring and then writes test cases differently by developing the actual code. For this reason, TDD is also called Test First Development.

Prototyping Methodology

Jeremy and Bond [116] discuss the prototyping methodology that emerged after the cascade. It enables the software development team to create a prototype application that can take three separate forms. The first is a paper ideal or even a computer that portrays man-machine intercommunication. The second alternative is to implement a feature that is previously in the scope of the software developed. Finally, there is a probability of using ready-made software that has some or each of the desired functionalities. This form most usually is adopted in software, although it may be available or partially ready, it has characteristics that need to be increased or improved in a further development effort.

Generally, the prototype works only as a mechanism to collect software specifications. In most cases, the first system developed is not entirely usable. Usually, it has a list of problems that are corrected in a redesigned version in which the deficiencies are corrected as well. The Cascade Prototyping methodology also has its negative points. One of them is that the customer may the prototype is already software available or in the completion phase and starts pressing for minor alterations and prompt delivery of the software. Faced with such a panorama, the development team often yields and the ultimate quality, as well as maintainability, can be arbitrated. Another downside sometimes implies that the development team can make short bargaining to get the prototype up and running that end up staying in the final software.

Despite these difficulties, prototyping is yet a robust software development methodology. According to [52] and [63], to be successful in the project, both customer and developer must reach consensus that the prototype only serves to help define the requirements despite solving various software development related problems. It should be noted that some parameters are not provided for by existing methodologies.

Spiral Methodology

The spiral methodology was designed to encompass the best practices of both classic life cycle and prototyping. Innovation brough about by this methodology is introduction of a new element - risk analysis. Besides, it was one of the first methodologies to adopt the concept of iteration. Successive iterations gradually shape the most complete solutions of the software [28].

In the first iteration, the objectives, alternatives, defined constraints, and identified risks are analysed. The client evaluates the result of the iteration and the next iteration starts considering the notes made on the previous one. It enables the client and the developer to perceive and react to risks in each of the evolutionary stages. However, the spiral methodology requires considerable experience in assessing risk, which is the key for its success. It envisaged that if a significant risk is undetected, problems will undoubtedly occur.

Agile Methodology

In brief, an agile methodology is a way of development, which adopts early feedback from the customer and development of a prototype version of the operating software as its central principle.

According to [57] and [63], this strategy allows for iterative development with the possibility of adjusting the already running project. Thus, teams achieve a balance between chaos and rigid structuring development.

Agile methodology is one of the most adopted presently in software development process [164], and the author discussed about detail in the next sub-chapters.

1.7 Agile Testing Techniques

According to [39], [40], [42] and [49], agile software development is based on four core values: a) Individuals and interactions for communication; b) Working software creation; c) Close customer collaboration; d) Responding to change quickly in the project. Within agile approach, there are three most frequently used testing techniques used to improve the testing practices – Test-Driven Development (TDD), Behaviour-driven Development (BDD) and Acceptance Test Driven Development (ATTD).

Agile Way of Working

This sub-chapter describes the general agile testing process of software products. Since software products in the company differ from each other, they have different needs, and each product should have its own test strategy [30]. The test strategy and the test processes should follow the guidelines laid out in the test management document. The SQA manager must approve any deviations from the organisational guidelines and the document. The purpose of testing is to document the quality of the systems provided by the company.

The test team should work according to the test guidelines as well as determined and documented processes. The test guidelines/strategies are continuously monitored and improved. Deviations from the organisational guidelines must be approved and documented. The amount of test effort would always depend on risk, how critical a business area is or what the functionality of the test is. Every task that is delivered by a developer should be testable. In some cases, there are exceptions to this rule, but if this is the case, the reason and alternatives should always be discussed with QA and the test leader.

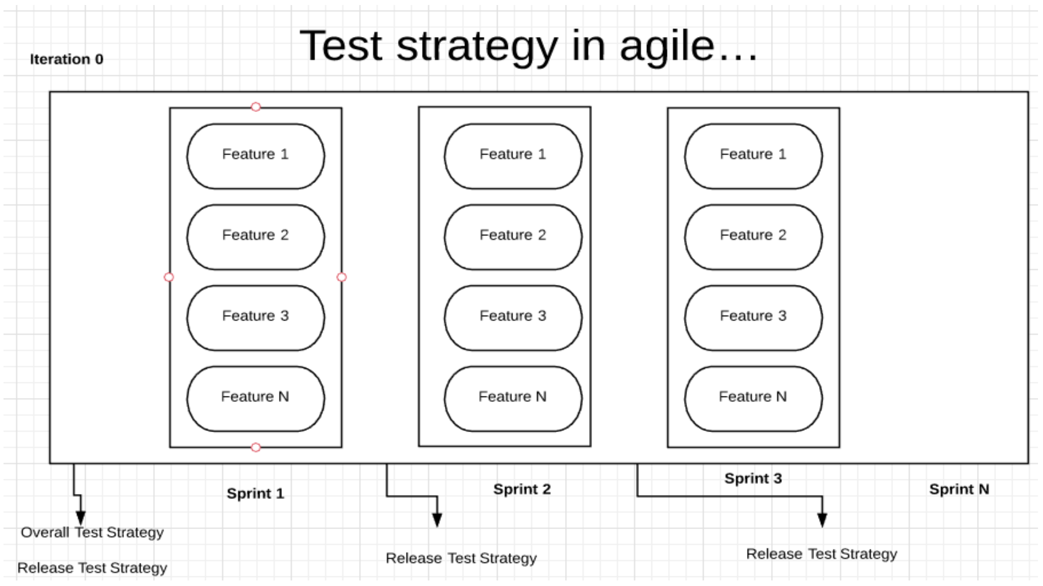


Figure 1.3: Test strategy in agile.

Layton [49] claims that if the development team is working according to the agile project methodology, then an active collaboration between QA/tester and developer should be maintained. In agile, due to short sprint many types of testing need to be implemented (Figure 1.3). Definition of completion of a task should always include functional testing, defect fixing, and re-testing. Each sprint (except for the first sprint) should preferably begin with a reasonable number of days/hours of regression testing. The amount of time depends on the project and is specified in the test plan. Tests should cover the potential regression defects that might have occurred in the previous sprint. The reason for the regression testing at the beginning of the next sprint is that the workload of testers is lowered during the first days of a sprint. According to [50] and [51], each project should always end with a stabilisation phase after completion of the sprints. Stabilisation phase is the time between the code-stop of the last sprint and the release date. The context of the stabilisation phase can slightly vary depending on the project, but should always include the finalising regression testing. The context of the stabilisation period should be described in detail in the test plan.

The long-term aim is to automate as much of the regression testing as possible where suitable, so that regression testing is performed continuously. To improve the test process, it is necessary to have a strong focus on performing root-cause analysis for the issues and defects detected. The test plan provides detailed information on the testing of the specific project and should be a complement to the test strategy document.

Software development according to an agile methodology in the company implies that there should be an active collaboration between QA/tester and developer. During sprints, as much testing as possible should be performed. Definition of completion of a story should include functional testing, defect fixing, and re-testing.

Regression testing should preferably be included in the sprints. Further details on the regression testing are specified in the test strategy and test plan. The team aims to automate as much of the regression testing as possible where it is suitable. In that way, the regression testing is performed continuously. According to Kumar Lal M and Cohn [53], [55], each project should always end with a stabilisation phase after completion of the sprints. Stabilisation phase is the time between the feature freeze of the last sprint and the release date. The content and the length of the stabilisation phase can slightly vary depending on the project but should always include the finalising regression testing. The content of the stabilisation period should be described detail in the test plan.

Working in the industry, the author had an opportunity to get acquainted with the following agile test documents. However, these documents did not apply to all projects.

Agile Test Strategy Document: The test leader is responsible for establishing a test strategy according to the organisational /test guidelines or should ensure that the existing strategy is applicable in test process.

In the previous section, the author explained the agile way of working highlighting the tools, test documentation and resources.

Agile Testing Overview

This sub-chapter presents an overview of the agile testing and test strategy. The author extensively used pictures to explain the principles of agile testing.

Testing Strategy in Agile

Research [65], [67], [80] [106] and [104] have found that test strategy consists of several broad areas. In this sub-chapter, the author uses figures to highlight all important areas.

Two significant agile software product testing strategies include a) risk assessments testing strategy in the software product and b) test automation strategy in the software product. According to Agile [167] and [3], there are four specific agile testing quadrants (Figure 1.4).

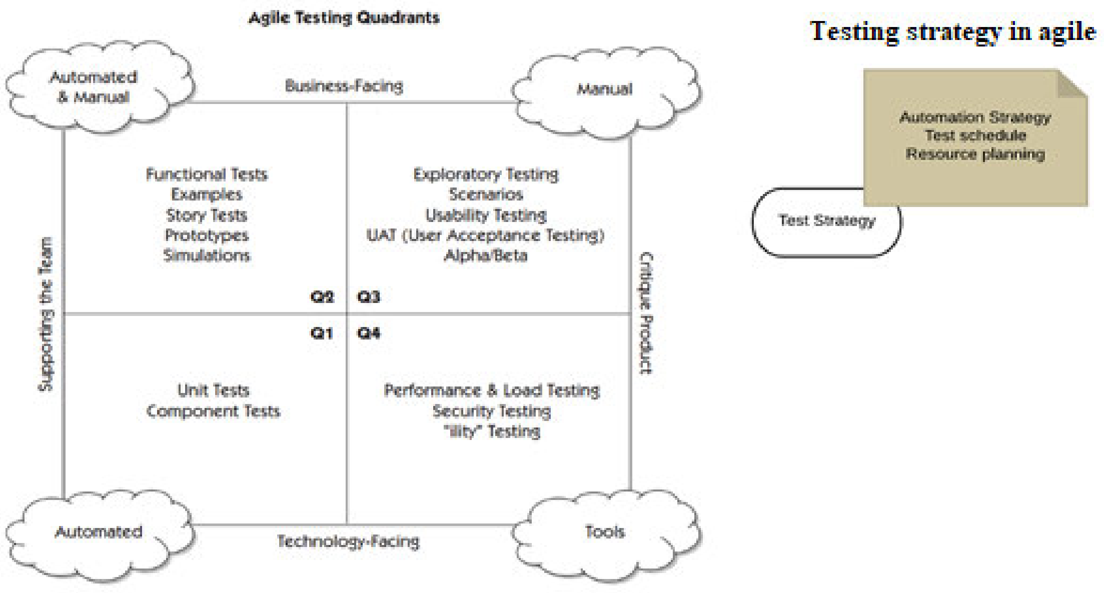


Figure 1.4: Agile testing quadrants and testing strategy in agile [167].

1.8 Software Development Tools and Teams

In this sub chapter software testing technologies, tools and people are described to show how these factors may be adopted in the test strategy.

Software Test and Integration Tools

Test management tools are used in the daily routine of the testing process. According to [96] and [103], they are used for creating and maintaining test cases, managing the traceability and coverage, test execution and reporting, metric collection, issue tracking.

CI/CD tools - these tools are used for continuous integration, deployment and release management in development as well as the testing process. These tools help to advance the level in the test automation process. Popular CI/CD tools include Jenkins, TeamCity, Travis CI, Go CD, Bamboo, and GitLab CI.

Test automation tools - these tools help to automate the software testing process. Generally, these tools provide GUI capabilities where a software tester records and replays testing actions. These tools reduce the overall programming effort in the test process. Such popular test automation tools as Selenium, TestingWhiz, Unified Functional Testing, TestComplete and Tosca Testsuite mat be mentioned.

Software Development Teams

According to the software engineering literature on the software development life cycle [30], [31] and [32], many types of skilled people are involved. In agile, in most cases the team itself is responsible for the software quality.

This sub section describes the skill sets and technical and domain knowledge to comprehension how these factors may be adopted in the test strategy (Table 1.2). Within the software development project, a team hires a tester as a team member to closely examine software product quality, the tester's responsibility depends on the team structure and tester skills.

Team - Human resource in the projects. Finding human resources with the right skills to work on the project is challenging in the information technology services industry and having such people in the team makes the team more confident in delivering the quality of software.

Skill sets - in an organisation, software testers and developers have different skill sets and knowledge. According to Figure 1.10, these skills include implementation of the artificial intelligence (AI) in business processes, software development engineering testing skills, development and coding skills, test strategy and test design skills, data science skills, test data setup expertise and non-functional testing skills (mainly security and performance).

Technical knowledge - In an organisation, software testers have different technical and domain expertise. These skills contribute to high-quality software testing, these abilities are specifically necessary for a software architect to produce relevant software solutions.

Table 1.2

Software Development Terminology and Key Values

Term	Aim	Solution	Examples
Methodology	To describe the type of project	Identifying the type of project methodology in use	Agile, XP, Serum, TDD, Waterfall
Technology	Technologies involved in testing / test automation	Identifying possible testing practices and test automation solution adopted for the project	Java, cucumber, c#, PHP, CI/CD Tools
Team	Team members	Identifying the skills of team members and listing them top down	Analyst, Developer, Development, Architect, Information architect, Senior, testers, Junior tester, QA specialist
Scope	Testing scope – long and short	Identifying the scope of testing	End-to-end testing, end user testing, Functional testing, Load testing
Process	Project flow	Identifying the type of process adopted	Test planning and test estimation
Guide	Project links	Identifying the outgoing links to knowledge base repository	www.domain.com/files
Metrics	Indicators	Quality metrics	Number of defects found in each life cycle
Software module	Software component under test. Gain the domain knowledge.	Identifying and listing the module under testing	“Health benefit” module under test

This chapter provided an overview of agile software testing. In agile software development and testing, test strategy creation is a complex process. While creating any test strategy, the team needs to consider numerous issues, such as different technologies used in the project, people skills, and processes. Other circumstances such as a short development lifecycle with extended documentation, impossibility of complete testing, setting the right process, lack of proper communication, lack of resources, test coverage, test continuously (test automation CI/CD), aimless test strategy, test planning, the need to coordinate the work of a distributed team in different time zones and test process information alignment should also be considered.

1.9 Test Process Management

Test process management is a combination of what to test and how to manage the test process; In this sub chapter two types test process management approaches are examined a) TMap b) ISO-IEC organizational process management analysed.

Considering the TMap is a way of controlling all test activities [27], [28] and [29]. TMap has extra dimensions because of the industry practice. It considers ISO/IEC/IEEE 29119 test process management steps. TMap Structured test process life cycle consist of seven steps a) control b) planning c) preparation d) specification e) execution f) completion g) Setting up and maintaining infrastructure. The adaptation of TMap structured testing is reviewed in four fundamental components: a) business-driven test management; b) structured test process; c) complete toolbox; d) adaptive test method.

Considering the ISO-IEC organizational process management as reference: A diagram in Figure 1.5 ISO-IEC organizational process management.

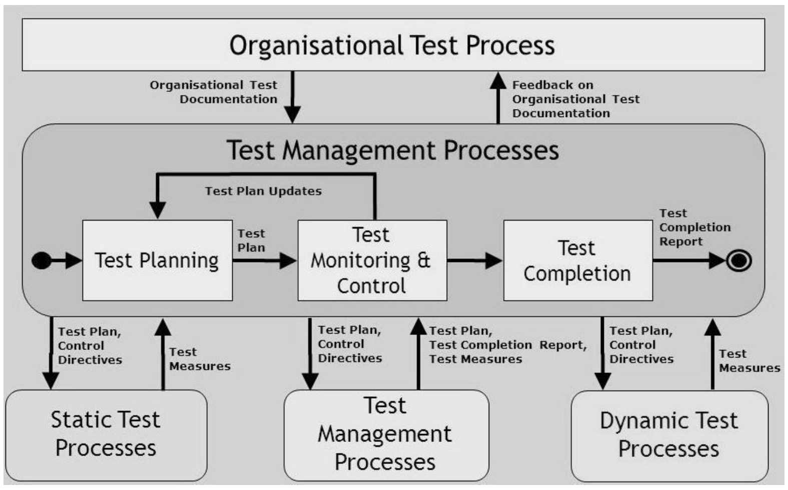

Organisational Test Process
Organisational Test Documentation
Feedback on Organisational Test Documentation
Test Management Processes
Test Plan Updates
Test Planning
Test Plan
Test Monitoring & Control
Test Completion
Test Completion Report
Test Plan, Control Directives
Test Measures
Test Plan, Control Directives
Test Plan, Test Completion Report, Test Measures
Test Plan, Control Directives
Test Measures
Static Test Processes
Test Management Processes
Dynamic Test Processes

Figure 1.5: ISO-IEC organizational process [199].

Test ISO-IEC organizational process management comprises the following main tasks:

Test Planning: Planning testing is an important step in software quality assurance. Test planning can be done at various points, such as before the development of the system, after the completion of a section of code, and during the production of a release. An important part of test planning is selecting what to test and why that will provide value to your product.

Test Monitor and Control: Monitoring and controlling the testing process is a key component of regression testing. The monitoring and control processes can be manual or automated.

Test Completion: Evaluation of the results of a completed test is a critical part of software quality assurance. This documentation is achieved through manual and automated processes.

In the above-presented section, the author provided brief insights into test management. Although test management is a vast topic, the author specifically described ISO/IEC/IEEE 29119 test management procedure. In the next sub-chapter, the author will discuss the role of test documentation in the test management process.

Different Test Documentation

This sub-chapter focuses on different test documents used in the software testing process. The following section covers the entire software testing life cycles. According to [20], [27], [185] and [59], test documentation process can be unique since it should comply with different organizational standards.

A diagram in Figure 1.6 presents an overview of IEEE 829 standard for software test documentation.

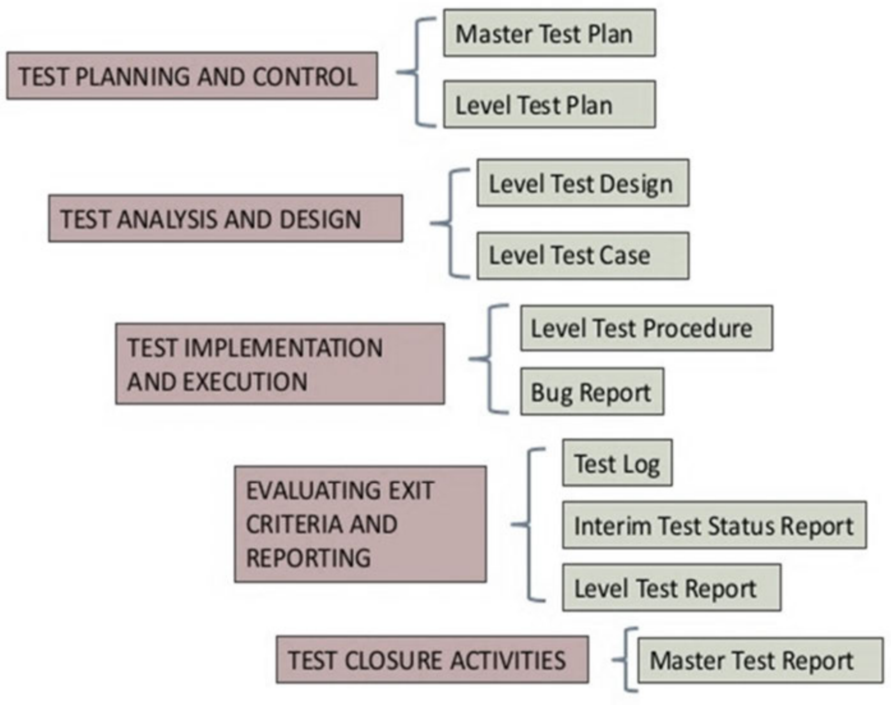


Figure 1.6: Overview of IEEE 829 standard for software test documentation.

Documentation In Software Development Life Cycle

Test documentation has undergone many changes over the years, with each iteration improving on what came before [190] and [192].

The waterfall life cycle was famously superseded by the agile life cycle which, in turn, is being replaced by modern capability development models. Alongside this progression, documentation forms have also changed to keep pace with new technologies existent in the industry [191]. From the paper form to additional advanced electronic formats, test documentation continues to evolve as developers learn how best to write it for their projects.

The use of an automated Tester's Workbench is also another advantage that has enabled documentation to take another evolutionary step [187], [188]. For example, the use of XML has enabled developers to take their colour-coded work to the next level, while easier-to-read electronic file formats mean that testers can access information much faster than before.

Life Cycle Documentation

The Waterfall life cycle, while being frequently criticised for being too linear in its approach to software development, is nevertheless still frequently used in the industry [190]. In this model, each aspect of the development process is followed consecutively from one stage to another before starting anything else.

Documentation During the Waterfall Lifecycle

To start with, documentation is mainly written in a chronological fashion. Each phase of development requires a different type of documentation: Requirements Analysis and Design (describing the business concept and problem), System Design (describing how to build software), Software Development (describing how to build software), Software Testing (describing how to test software). These documents will be used throughout the entire process until the end of the project [191].

This stage is mainly for changing the requirements of the system, not writing detailed documentation. Requirements should be set in this phase and then refined in the Design phase. If the client wishes to change their mind based on what they see in this phase, then that is no problem since it doesn't affect anything else. Also, any other issues regarding development can be fixed before moving on to the next phase which can take care of all other problems; It reduces the resistance or rejection since nothing has been committed yet. The main things that need to be fixed before moving on are any of the business's specification changes. In this phase, the developers also figure out the interface of the software.

In this phase, further details regarding any of the design issues have been fixed and laid out for development to follow. Here, all documentation is written on a detailed level on how something will be built since a lot of decisions have already been made about how it needs to look and function. If any defects are found, they can be fixed before committing future changes to the documentation.

Software Development Documentation

In this phase, the developers will continue creating the system from the design documents laid down during System Design, as well as writing any source code necessary. In this phase, there isn't much documentation to do except for creating the individual objects and methods inside a program [192]. The developers will also have to test these objects and write unit tests specifically designed by developers for each one of them based on how it needs to function.

The unit tests will mainly be for the object, but the developers will also have to write a whole system test in order to check that the whole framework is working correctly. This documentation will not change much. Any changes that must be made to it will be fixed in this phase. The main reason for this is that it isn't done yet, therefore not much can go wrong.

Test Documentation

At test documentation creation point in the process, the developers are done with everything they need to create the system. They also tested it thoroughly to make sure that it will run properly. The last step before moving on is to document how everything works effectively [186]. The documentation for this phase will mainly be written to help others use the system later. The tester should write general acceptance criteria to help software developers, testers and analysts to know how the system should work. This document describes the functionality of all the objects that are created in the system to make sure that they are working properly. The tester will also need to write test plans for each method in order to use them later on.

The main advantage of this type of documentation is that the developers are done with their work, so it doesn't take them much time or effort to wait for this phase. Also, any changes made at this point won't affect the development process since it is done already. The main disadvantage is that it is essentially manual. The testers will have to do a lot of work just to write down everything that needs to be documented.

An automated system could save time, but it isn't as helpful as it could have been since they have to write everything by programming first. Test documentation use has become streamlined for various test automation tools [3], such as PEAR or GenTEST.

Quality Assurance Testing Documentation

In this phase, the testers will still be testing throughout the entire process. However, the testers won't be doing it all by themselves since they will primarily be working together to find and fix bugs and other issues that would otherwise slow down the development. It isn't as tedious as testing documentation since they will primarily test through their own code using unit tests which are easier to write than test plans.

In this phase document the software defects found in their code or functionality which they will report to whoever is responsible for fixing them. The developers can continue working on their code during this phase since it isn't affecting any other part of the development process [193]. Test documentation is important for this phase because it ensures that all the software defects are fixed before moving on to the next phase. After the software defects are fixed, the testers will be responsible for writing documentation on how to fix or avoid them in case they occur again. This documentation is mainly written from a user point of view.

Maintenance Documentation

In this phase, a lot of changes need to be made to the system after it's been released so it can stay updated. If it isn't kept up to date with new features and fixes, then it will fall out of use and become outdated, so software development team won't want to use it process [91] and [92]. Maintenance documentation is necessary because it ensures that the software is being updated and tested. In this process testers will need to confirm that the changes work as expected, as well as if a new feature works correctly and not causing any other issues. They will also have to write any bugs found around the new changes along with how they can be fixed. Maintenance documentation also makes sure that all functions of the system are working properly. The main disadvantage of this type of documentation is that it takes up a lot of time.

Any changes a developer makes to the existing system will have to be documented before being accepted, which takes a lot of time. This may cause delays in the development process. However, if written properly, it could save developers from wasting time making unnecessary changes that won't help much in the long run. It would also mean less problems come up later on as a result of bad design decisions made during the System Design phase.

State Diagram Documentation

In this phase, the developers will be sure to document all of their code for each object in the system. This will often be done by drawing a diagram of what is being created. The process of creating a state machine can also be automated with certain software, which would save them a lot of time. State diagrams are helpful because they show everything needed for another programmer to comprehension how many processes works together [89] and [90]. They are also helpful to ensure that the code is working properly.

The main disadvantage of documentation is that it must be done manually. However, this type of documentation can be very helpful and will ensure that everything works correctly.

Scrum Testing Documentation

Scrum is a useful method to use when working on software development projects, especially to get the most out of the team. It assists ensure that work is visible, and everyone is working towards the same goal [194]. It also assists ensure that the work is completed before moving on to the next step, ensuring that nothing is forgotten. It is important to document the process that occurs during an Agile software development project.

Scrum can be used in several ways, but Scrum Testing Documentation should be used instead of the other approaches since it ensures that all parts of software are working properly. This make sure that all bugs are fixed prior to moving on to any other steps. It also ensures that all steps are documented correctly and is a worthwhile endeavour.

There is a lot of output from this type of documentation, so it should be done after most of the coding has already been completed. This will save time and allow to move on to the next part of the development with confidence.

The most important thing about Scrum Testing Documentation is that it ensures complete integration of all parts of software. This type of documentation is extremely important when developing software in an Agile manner because it ensures the entire project is completed correctly.

User Story Documentation

This type of documentation involves using user stories to explain how the system should work. The views on the story will describe what the user can expect to be done, and what they will experience after actually seeing it performed [195]. This type of documentation explains in detail what needs to be done, making sure nothing is left out.

In the process of user story creation there are different ways to document user stories, but it's best to start with the user view. This will ensure that the users know how they should be feeling while using the software later on. It's also important to create both acceptance criteria and a design overview before moving on to the next step. This will help ensure that not to forget anything while creating software.

The most useful part of this type of documentation is that it assists to create the best possible software. It makes sure that code is exactly what the user needs, which will eliminate many problems during the testing phase. The code isn't working properly with the user's expectations, then it sticks to writing new requirements instead of throwing out money writing new software.

Comparison Of Automated Documentation Systems

There are different automated documentation tools generates the different type of reports. Each type of documentation has its own advantages and disadvantages, which is why it's important to see how they compare [71], [72] and [73]. Some of the most appreciated tools for documentation are UML, Excel spreadsheets, Frameworks such as Crystal Report Server.

UML is tool used for drawing diagrams of software or another project so it is understood by programmers. UML can be used for any type of programming project since it is a universally accepted method of communicating ideas. It's also a great tool for creating user stories or any other documentation.

Excel spreadsheets used to keep track of different projects, specifications, and even budgets. In the case of software development, Excel is often used as an input-output spreadsheet to show how many times each part is used during the development cycle. Excel used to identify and eliminate unnecessary parts of the software. It is can also use it to track budget and make sure it is staying within the desired range. Crystal Report Server is a reporting tool used to create all types of reports, including user stories and detailed designs for software. It assists ensure that all of the information need is available for each project, even if it's not implemented yet.

There are several different types of documentation that can be useful during software development, and which it should choose depends entirely on the type of project working on. According to [41] it is important to make a choice, especially it makes sure work is easy to comprehension by the software developers. The main goal is to basically ensure basically that all steps are documented correctly it improve project documentation quality significantly.

Project-Based Type of Test Documentation

Project test plan: This document lays down many activities, such as analysing the product, designing the test strategy, defining the test objectives, defining the test criteria, resourcing planning, planning of the test environment, timing and estimating, testing deliverables.

Test project completion report: It provides an application overview, identifying the status of testing scope items, such as in scope, out the scope, items not tested. According to [27], it must also include the metrics that cover the primary test case planned vs. executed, and many test cases passed and failed [103], as well as the types of testing performed on the application, lessons learned, recommendations, best practices, and exit criteria and finally, conclusion and sign off.

Different Test Level Documentation

Test plan: This document lays down the objectives, processes, treated audience, it also contains a detailed workflow.

The workflow includes design verification, acceptance test, service or support and regression test. It also focuses on

a) test coverage – states the plan of a test and how and what requirements are verified; b) test methods – shows test coverage with testing; c) test responsibilities – includes the test methods in each product lifecycle.

Test specifications: This document covers the test details for software features, input, and execution strategy.

Test results: Reporting the results at every stage of the project is very important as it covers results matrices. A report is provided to stakeholders as a part of the deliverable.

Anomaly reports: It is an issue report by a tester about defects, it contains the defect summary, description, status, steps to reproduce, severity, priority, data, proof of defects and detailed information about the issue.

Level test status report – LTR: It is a short report that covers the testing activities and provides evaluation feedback and recommendations on the test already executed.

Test environment report: This report covers different test environments and status report of each environment testing results.

Test level completion report: The report presents information on testing at each level with completion stages and detailed information.

Test Policy Documentation

It is a central document that defines the principles the organization has adopted for their testing activities. The test policy is typically a high-level document, but organizations may want to publish it separately. The test policy might also be used both in the development and release phases of an organization's software project life cycle [31].

Test policies can be very general or very specific depending on the organization's software development needs for example, software needs pass the usability test before used by the end users. It defines the path how any organization measures the success of testing; it also specifies when the organization changed and reviewed the document. Some organizations may want their policy to relate to all aspects of their testing efforts while others might focus on only one area such as quality assurance activities. Test policies should cover the purpose of the testing, policies on how the organization plans to execute tests including plans for code releases, any special requirements for testing activities such as security tests and communications with end customers, and the policies on how the organization plans to review test results.

Test Strategy Documentation

It is a company level document that is used by the programmer or project manager. This document focuses on the “Testing Strategy” to achieve the goal. It extracts some of the points from the BRS business requirement specifications.

Test strategy has many subcomponents: The document can be applicable to multiple projects, it specifies the scope of the testing object, project scope and out of scope items, business value, such as budget for the testing and control, roles and responsibilities of the persons in the test project, communication and negotiations with the team, test levels including unit, module, system and integration. Test types, such as functional and non-functional, also the entry-level, exit level, stop and restart criteria for testing, are defined in [18], [19] and [23]. Test strategy document also covers the test environment, possible test design methods, test deliverability to each phase of the project, test strategy (manual and automation), availability of testing tools and scope of automation, testing metrics to measure the testing.

The documents should also address defect management in a life cycle, considering the negotiation skills for testing and development teams and defect classification. It also needs to specify how testers can address risk and what training is necessary for the testing team members to upgrade their skills.

1.10 Software Testing Complexity

In this sub-chapter, the author briefly discusses software testing challenges from different viewpoints: human resources, technology, method, tools (Figure 1.7). It formulates the research problem by specifying the research objectives, examines the environment, the context of the research problem, investigates the nature of the problem, defines the variable relationships.

The context of the research problem is as follows: software testing becomes very important as well as challenging for any software development company in an attempt to deliver defect-free software to the end customers. The traditional strategy in testing software applications was adopted from long-established companies. According to [18], [25], [141] and [107], many research articles and industry surveys prove that software testing needs to be improved and that it is necessary to rethink testing strategy, test management, tools and process adaptation.

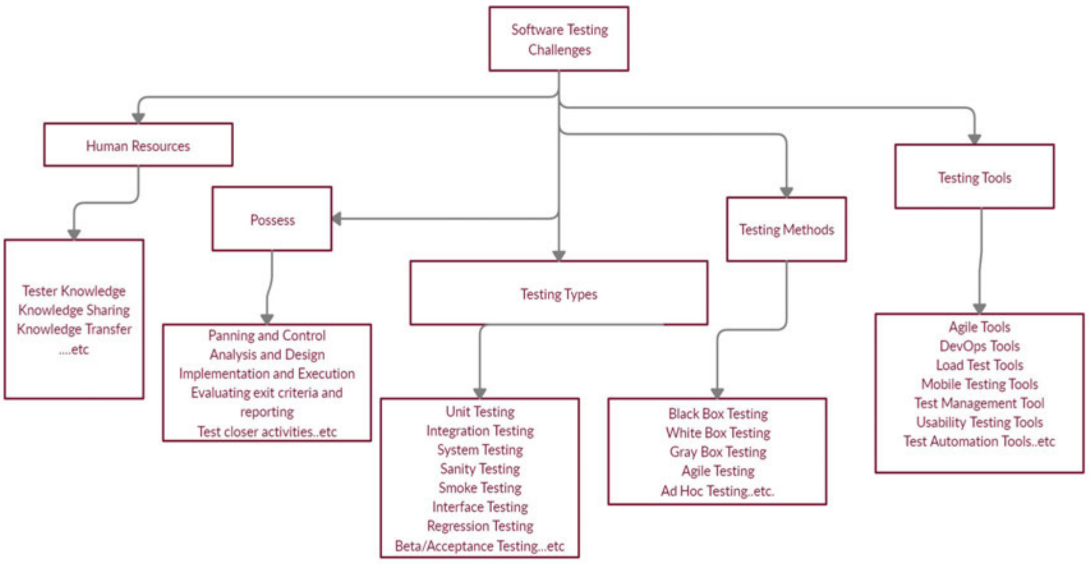


Figure 1.7: Software testing challenges from different viewpoints: human resources, technology, method, tools.

- The present software development point of view: Responding to the rapid technology changes, fulfilling customer demands and handling the project time limitations. Managing the limited infrastructure/resources and reducing the conflicts with software testing teams.
- The software development team point of view: Creating the test strategy, creating and managing the testing documents, test cases, sharing knowledge with projects and requirement managing testing point of view. Within the project, a team member, a test manager, a team leader and the tester need to be aware of and align with the current working project test strategy.
- The composite point of view: Multiple teams are situated in different locations and keep all team members aligned with the project knowledge, tools, and people. The defect-free software is delivered to the end customer.

In the above presented figure, the main blocks of the first level include human resources, processes, methods and tools. The human resource block contains other sub-variables, such as training, knowledge transfer and knowledge training. Similarly, process, methods and tools also include many sub-variables. In the software testing process, all these variables and sub-variables need to be acknowledged during software development and testing process.

Figure 1.8 is a diagram that explains how software development is organised in a company. As its minimal requirements, it contains process, people, methodologies and technologies.

- Process: It contain different process and it is applied to software development projects. The process can also be applied independently only for testing.
- Methodology: Each software development project follows one of the methodologies.
- Technologies: Technologies are variable and may change from time to time.
- People: People are variable and may change from time to time.

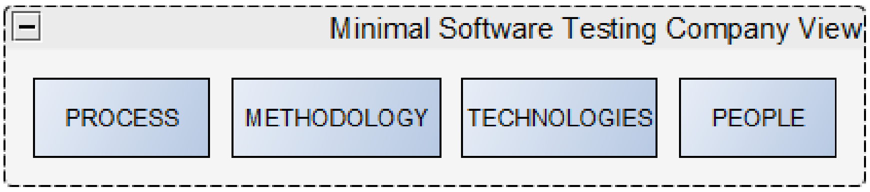


Figure 1.8: Test process model at a software development company.

A diagram in Figure 1.9 showcases the complexity of the testing process as an example of a software test process flow diagram.

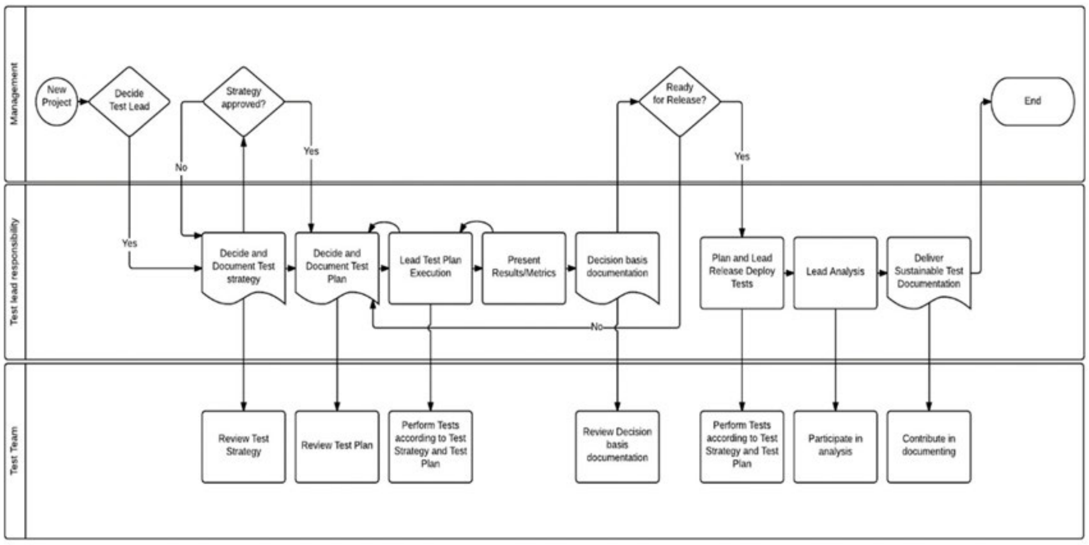

Management
Test lead responsibility
Test Team
New Project
Decide Test Lead
Strategy approved?
Ready for Release?
End
Yes
No
Yes
Yes
Decide and Document Test strategy
Decide and Document Test Plan
Lead Test Plan Execution
Present Results/Metrics
Decision basis documentation
No
Plan and Lead Release Deploy Tests
Lead Analysis
Deliver Sustainable Test Documentation
Review Test Strategy
Review Test Plan
Perform Tests according to Test Strategy and Test Plan
Review Decision basis documentation
Perform Tests according to Test Strategy and Test Plan
Participate in analysis
Contribute in documenting

Figure 1.9: Software testing process flow diagram.

Moreover, the annual World Quality Report [164], [165] makes an emphasis on the test strategy creation (Fig. 1.10 and Fig. 1.11). In the software testing industry, ability to devise a "test strategy" is one of the vital skills needed for the employee.

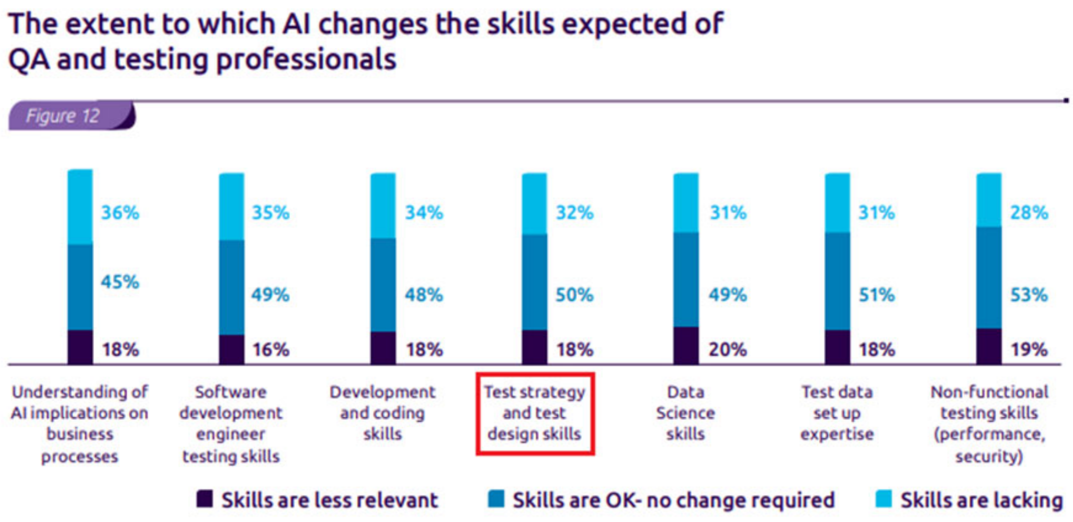


Figure 1.10: Artificial intelligence in software testing and the required tester skills.

According to [164] and [165], the World Quality Report 2017- 2018, the key recommendations are: a) ensuring user satisfaction is at the top of their QA and testing strategy; b) the organisation needs to define a test platform strategy on an enterprise-level; c) the QA analysis strategy on an enterprise-level should be defined.

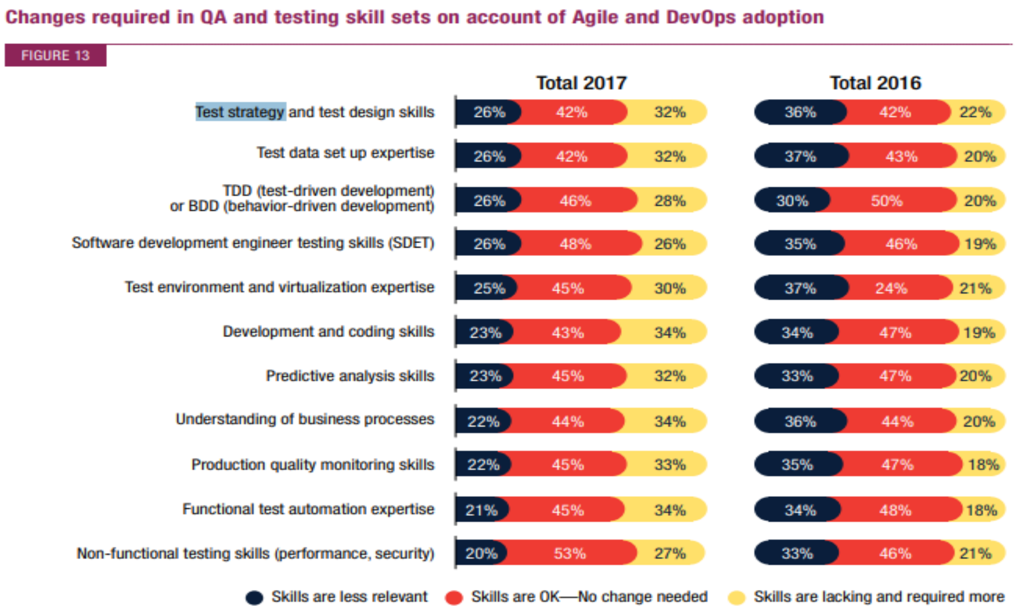


Figure 1.11: Skills to be possessed by quality assurance professionals for Agile and DevOps.

In conclusion, each software development project is unique since different development methodology is adopted for every project and testing challenges are also unique. The author intends to discuss complexity of test strategy creation in the software development projects.

1.11 Visualised Test Strategy Hypothesis

The test strategy describes the software test approach in the software development life cycle.

In IEEE 829-2008 standard for software and system test documentation, test strategy is a part of Level Test Plan (LTP) as test approach [80].

In IEEE 29119 Software test documentation (STD), test strategy is a part of In Organizational Test Process Documentation as an Organizational Test Strategy and in Test Management Process Documentation as a Test Plan (Including a Test Strategy) [81].

In the general practice in Latvia based software companies, test strategy document describes team member roles and responsibilities, testing environment requirements, lists testing tools used in the project, possible risks and their mitigation options, testing schedule, regression tests, test groups in terms of components and sub-components, testing priorities, testing status collections and reporting, testing record maintenance, and requirement traceability. Some documents contain testing process design.

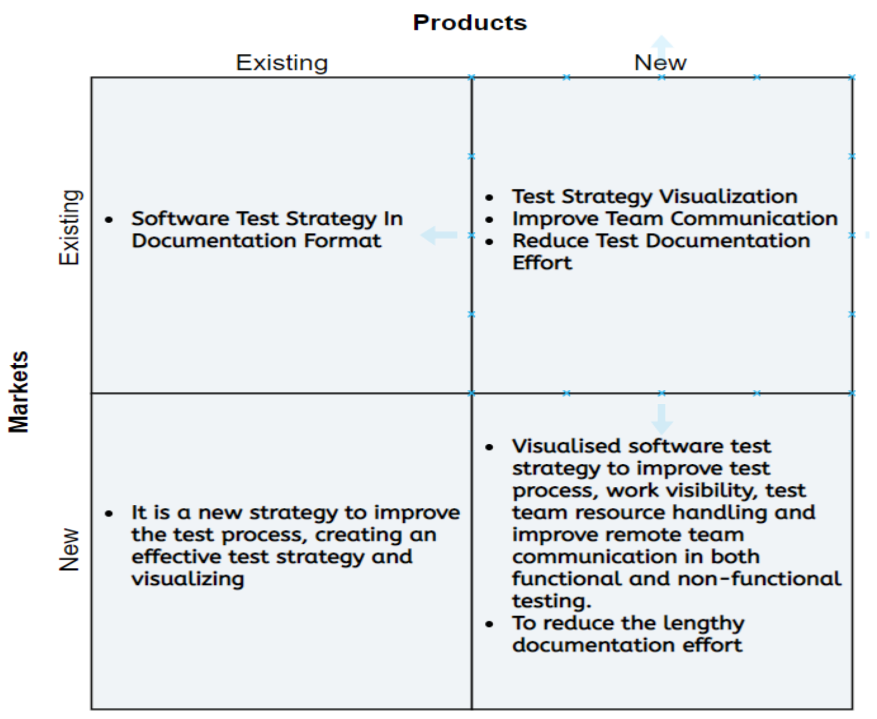
Black and Mitchell [8] claim that test strategy is created at different testing levels. Three preliminary testing levels are unit testing, integration testing and system testing. In the project, most of the time developers are accountable for unit testing to individual testers or testing teams are accountable for writing the test. In agile testing, everyone is responsible for the software quality. The agile team decides who writes a certain kind of a test for the software.

According to [135], so far, all development strategies and types of testing have demonstrated an lack of the visualized test strategy. To avoid stockpiling of the existing test strategy documentation, software testers need some user-friendly tools for drawing up a strategy.

Considering the above presented viewpoints, research [137], [134], [138] and [139] have found that software testing needs a new strategy to improve the testing process and help create an adequate test strategy. According to [164], agile is the most frequently adopted methodology; it emphasizes the involvement of the testing team in each early phase or sprint. The testing team may identify, which type, and area need to be focused on in the testing process, a single page knowledge board may be used for the cross-functional teams. Fewer agile test experts are needed, and software development team members gain testing process overview quickly. Team communication (both local and remote teams), key performance indicator descriptors and essential documents links are available on one platform.

Considering the team’s point of view, test teams start adopting visualized testing strategy for the projects where agile methodology has been selected. It reduces complexity of the testing process as compared with the traditional testing process and its documentation and assist in creating an efficient testing strategy.

Acknowledging the above-mentioned details, the test strategy hypothesis is presented using the strategic planning tool Ansoff Matrix as a framework to comprehension opportunities for future growth (Figure 1.11).



Products
Existing
New
Markets
Existing
New
Software Test Strategy In Documentation Format
Test Strategy Visualization
Improve Team Communication
Reduce Test Documentation Effort
It is a new strategy to improve the test process, creating an effective test strategy and visualizing
Visualised software test strategy to improve test process, work visibility, test team resource handling and improve remote team communication in both functional and non-functional testing.
To reduce the lengthy documentation effort

Figure 1.12: Hypothesis for the visualised test strategy.

Refining the above presented hypothesis, the author has concluded that there is a need for visualising software test strategy. In order to test this hypothesis, it is necessary to develop a visualised software test strategy in different software development life cycles and types of software testing.

In view of the above-presented early-stage findings, the next chapter will discuss a remarkable visualized solution for the software development life cycle.

1.12 Summary

Thus, while studying and identifying the points in this chapter, it can be concluded that the advantages of the test strategy are that it a) provides a framework for decision-making; b) supports comprehension the project strategy; c) provides a team perspective; d) enables measurement of progress. Disadvantages are a) it can be expensive and can involve a lengthy documentation effort; b) the future does not unfold as anticipated; c) one type of strategy is not applicable to all projects; d) a strategy that might not be flexible is implemented.

The following statements may be presented in favour of the further research:

- Software testing is used to show the presence of the issues, but never their absence. Software testing processes of evaluating a system or a component verify that it meets the requirements specified for the software or identifies all possible differences between the expected and the obtained results;
- There is no standard procedure for development of a software test strategy [140] and the existing test strategies do not fulfil possible software testing criteria within the trending agile methodologies;
- Each software development project is unique, since different development methodologies are adopted for the project, and each of them has different testing requirements and possibilities;
- In the current industry, there is a potential skills gap for test strategy and design;
- In the current industry practices, different test documents used in the software testing process;
- The author has found out that not much research has been conducted on the visualization of the test strategy.
- The author discussed the hypothesis of the problem and demonstrated new possibilities, as well as highlighted user experience design and user interface design journey.
- The author has demonstrated that visualising information is a possible solution to the research problem.

2. ASPECTS OF TEST STRATEGY VISUALISATION

In the previous sub-chapters, the author described different aspects of software testing. This chapter focuses on the research analysis, test strategy visualization possibilities and detailed analysis of lean canvas design adoption considering user experience.

In this chapter investigated the existing diagram possibilities for strategy visualization, such as fishbone design, map design, empathy map design, business process model, notation design and lean canvas design. In this chapter, the user’s experience in the design strategy in order to develop a lean canvas design. It considers the software tester and personas points of view on any problem. In this research, the tester, test manager, team member, test lead primely are focused on. In the chapter dedicated to lean canvas design and evaluation technique, the author discusses the user needs, user goals, constraints and assumptions.

2.1 Strategy Visualization

According to [173], [174] and [175], software development projects have a prompt access to project relevant information, which means that they have the power to improve programming and testing speed, scrutinizing of the business domain, seizing opportunities and gaining a competitive advantage in the marketplace. But until that point, information comes in the form of documentation which, alone and without proper analysis, may not mean much. In practice, depending on the amount of documentation, the interpretation itself can become quite time-consuming or provide an extremely complicated result.

The need for strategy visualization is straightforward: A informative diagram can communicate information immediately in a memorable way [83]. Visualization strategies improve scrutinizing of test process overview, help capture significant moments in the text and combine as many tangible details as feasible. Visualising a diagram, users are invited to express their sentiments, visualize the text and make it their own, communicate decisions in an effective way. As a result, actions are facilitated using visualization, which allows users recognize the relationships between software testing and results, interact with visualized lean canvas board and generate new ideas and discussion [141] and [143].

The visualization of strategy can be used in the following way. First, the project manager view overview of test strategy and contributes inputs to it. Test team lead can monitor and advise on the effectiveness of test team members activities, all test team members get overview about test scenarios, test risk, test priority information.

The benefits of visualizing with long test document reading and updating. For example, testers can read and update less larger documentation.

Team members: Team members understand visually the test strategy of the project. For example, they can examine the current situation, they can communicate further effectively.

Test Manager: The test manager manages the testing process. For example, the manager team and simplifies the communication structure on a technical level.

Team lead: Team leaders guide and de-bureaucratize activities. For example, Identify and assign the correct technical task to a skilled team member.

In this condition team members, test manager, team lead, and the tester read a single page document and scrutinize information about the under-investigation product or the project, tools, process and strategy. Visualizing the test strategy within any project, the overall testing process is simplified.

Different Design Possibilities for Strategy Visualization

In this research, the author analyses development of different possible existing strategy visualization designs. At this stage, the author has found several disadvantages and limitations diagrams may demonstrate, when testing the strategy point of view. The author discusses these issues in the next paragraphs.

Fishbone Design

A fishbone design is a cause-and-effect finding tool [30]. The fishbone design assist workers, entrepreneurs, people in business or students to solve problems that may arise in their working life, this approach may also be applied in the social and daily life. Its application allows making the most accurate decisions after making a preliminary study of the potential hurdles. This approach is also known as the Ishikawa design and Fishbone diagram because of its shape. Fishbone designs is a problem analysis tool that may be used in search for solutions in any company [177].

The fishbone design is a graphic representation, which, as it has been pointed out, is shaped like a fishbone. At the head there is a PROBLEM and, in each spine, there are its CAUSES. Each spine can, in turn, have other smaller spines, thus dividing the causes and sub-causes of a particular problem.

Application of the fishbone design starts with a blank paper to make a brief introduction to the problem that is going to be analysed at the company. Then a causal design shaped as a fish skeleton is drawn. Next, the leading causes of this problem are identified and placed at the thorns as categories. For example, Person, Material, Machinery and Method, the ground for emergence of these causes is established. In order to highlight the possible sub-causes of the problem presented in the diagram, the branches of minor spines are drawn. Thus, the primary and secondary causes are identified, which allows correcting some of them immediately. The causes/sub-causes that are out of control of a particular person may be passed to a person who can solve them.

Using Fishbone design to prototype design in the organization is to prevent overlapping of functions within the organization at any level. This assist in organizing all functions at a given level with greater efficiency and effectiveness. This system assists workers to identify the most important tasks they need to perform. Fishbone design provides a systematic way of identifying the causes, the main causes, and the sub-causes of problems. It is applicable to any problem that can be solved by some resolution. A Fishbone design assess the trade-offs involved with any solution it considering, highlighting what can be done (the informative things) and what needs to be done (the bad things) to address problem [30].

Using fishbone designs for strategy creation in the organization is an effective way to develop a complex strategy by using "what-if" type of questions. It helps to manage identifying different alternatives and then choosing a substitute. It is useful in drawing out the strengths and weaknesses of each solution and identify the key issues that have been overlooked. Fishbone design to create a strategy for any difficult situation or problems faced in any business.

Limitations of the Fishbone designs in the testing strategy visualization:

• Team discussion in testing provides unnecessary potential causes along with important ones, occurring in a time;

- Technical discussion regarding project is frequently based on view on reality and evidence. In testing, test criteria are clearly defined, otherwise testers need to ask for them;
- Very ample space for controlling out the design is required for complex obstacles with many branching bones and "why"-bones;
- The complicated interrelationships of various software testing factors are challenging to show on a fishbone.

Strategy Map Design

Accordingly, to [10] and [11] every strategic process consists of the three specific phases (formulation; implementation; and monitoring and control) to achieve its optimization, and it relies on three different methodological tools [178] and [179]: strategic maps (formulation of the strategy); action plans (strategy implementation) and dashboards (monitoring and control of the strategy).

The strategy balances conflicting forces: a significant goal of any company is to achieve long-term sustainable development, for this reason, it is essential to coordinate proposals for cost decline (measured as short-term effects) with investment in indefinite assets (long-term objective). The strategy is based on a differentiated value proposition for the client. A clear value proposition for the customer is essential since it is a crucial part of the strategy. If this proposal is fulfilled and satisfies the wishes of the customers it is addressed to, it is possible to complete the financial intentions. The internal perspective facilitates to interpret, which suggestions for improvement of internal processes allow producing a value proposition appropriate to the clients' specifications.

The strategic maps help the companies to achieve their goal since they allow to identify and filter the opportunities and threats emerging from the market (environment), the industry (competition) and company (analysing complementary competences, future investments and sustainable development).

According to [178] and [11], strategic maps are useful for decision making but they can also be used as a communication tool.

The information should be presented in an easy to acknowledgeable way.

Limitations of the Strategy Map design in the testing strategy visualization.

- It shows the connection between the strategy and goal. In software testing, a point of one more connection line needs to be drawn.
- Breakdown of the large organization goals into smaller and measurable goals.

Empathy Map Design

The empathy map is a collaborative visualization tool [180] and [24]. Empathy map design is a design technique that translates consumer pain points, frustrations, and desires into design features. This process focuse on how to create products for specific target markets adopting insights or taking advantage of unexplored opportunities. The key to empathy map design is understanding the consumer's problems and balancing these insights with business objectives.

The basics of empathy map design are simple: First, identify your target audience. Second, understand their problems. Then, map out where their real pain points are and how they can be solved. In the final stage, find ways to turn these insights into business objectives. Empathy map design is often used in scenario planning to help companies make predictions about what customers will need in 10 years' time, which can help companies make strategic decisions about future product developments.

Limitations of the Empathy Map design in the testing strategy:

- It makes customer segments and elaborates on the user personas. In the test strategy, developers clearly know who their strategy user is.
- It captures behaviours and interviews a customer. It tests the strategy more with respect to how to work instead of capturing behaviour.

Business Process Model and Notation Design

The business process model and notation (BPMN) is a standard form of graphic representation of business processes. This notation graphically depicts the design of the processes and their implementation in practice [181] and [182].

BPMN Objectives are to provide a notation that is easily understood by all users, e.g., business analysts, the technical developer and the businesspeople. It is necessary to create a standardized bridge for the gap between the design of the business process and its implementation and to ensure that languages for the execution of business processes are visualized with a conventional notation (standard) and XML-based languages to describe processes with a simplified graphic representation. There are three levels in process modelling. According to [13], [15] and [14] BPM notation supports all three levels. Process Maps: These are the graphs that represent the flow of activities carried out to execute the process. Process Description: These are process maps accompanied by additional information, but they are not sufficient to define the process completely. Process modelling: These are process maps accompanied by sufficient information so that these processes are analysed, simulated and executed.

Business process model used for documentation is a process model adapted to a particular problem in a particular context, from which it is clear that this has been used to establish enterprise integration architectures [182]. The model provides a typical description of the typical elements of the model. For example, in order to achieve the desired result, we can mark all these elements in an appropriate way. The choice of the elements, their descriptions and their relations is left entirely to the designer. The model does not contain information about how these elements are implemented or exactly what each element is supposed to do.

Limitations of the Business process model and notation design in the testing strategy:

- During the software test, strategy phase test engineers and team need to know how to use the BPMN.

Ontology-Based Visualization for Business Model Design

In the recent research [184] it was mentioned that ontology-based visualization for business model design offers new ideas. The team of researchers developed a tool AOAME4BMC. It consists of the front end and the back end of the graphical user, the interface web-based representation of an enhanced business model canvas, a web service for information exchange with the backend, and a specific ontology for the machine-interpretable representation of a business model.

AOAME4BMC tool bridges the gap between human intelligible modelling language and a machine-interpretable language [45], [46] and [47]. In this process, humans have adopted graphical or textual models and computer formal language to interpret models. As a solution, ontology is used instead of UML as metamodeling language. In this situation, ontology-based metamodeling creates an automatic linking of the graphic score to the semantics in the ontology, initiates models which are both machine-interpretable and human acknowledgeable.

According to [184] ontology-based visualization for strategy creation is use business model canvas as a core and design. For example, it generates the number of different possible strategy at same time.

Limitations of the Ontology-based visualization for business model design in the testing strategy:

- AOAME4BMC tool fits for the big enterprise where domain knowledge is frequently changing at business level.
- Later stage expert advice needed to select the best strategy generated by machine.

Lean Canvas Design

The lean canvas design was created for Start-ups those operate in extreme uncertainty. That is where the lean start-up methodology makes sense, and that is why the lean canvas of a business model for entrepreneurs was born. According to [48] and [56] user is part of an entrepreneurial team, the user must reflect everything in this business model. If, on the other hand, the user is a sole entrepreneur, s/he explains an idea to friends, acquaintances or colleagues (it is recommended that they have an innovative and entrepreneurial profile), puts them on the scene and works with them, they will help the user obtain a diverse point of illustration that can give the user many keys for future business.

According to [110], the canvas model is not meant to be static. It is designed to carry out all the required modifications without changing the complete business plan. The advantage of using this system allows the user to find faults and guarantees a competitive strategy. It allows the user to view the process on a single page and make modifications whenever he needs.

The lean canvas has its value in creating a strategy. It can be used to draw a business model for entrepreneurs or businesses to visualize the needs and the objectives of the project. Designed in such a way to allow each user to work individually, it provides traceability and control for an entrepreneur during the implementation phase.

The lean canvas is meant for situational-analysis and providing traceability; therefore, it does not permit implementation immediately, but is intended for providing assumptions and ideas that will gradually feed into the concept of the company and later into implementation. The author discusses the lean canvas in sub-chapter 2.2 in detail.

Limitations of the Lean canvas design in the testing strategy:

- Large amount of information does not fit in the blocks.

Investigating the existing visualization design strategy creation possibilities, the author identified additional benefits of adopting lean canvas as a base model for further test strategy visualisation research [134]. In the next sub-chapters, the author discusses in detail adaptation of the lean canvas design to visualise the test strategy.

2.2 Lean Canvas Design

The common features of all canvases are as follows: they are a graphic representation of the business model where the hypotheses are reflected and are modified in the course of communication with potential clients until a viable business is achieved, and this all is done iteratively, agilely and collaboratively with the entrepreneurial team. A canvas is nothing more than a language that allows describing the business model and making modifications quickly, thus constituting a working methodology.

According to [109], [113], [114] and [110], Lean Canvas is an adaptation of the Business Model Canvas by Alexander Osterwalder made in 2008, the model was created by Ash Maurya in the Lean Start-up. It is used for strategic management, lean start-up templates, and documenting the existing business models. Figure 2.1 provides an example of the standard lean canvas template. It is a single-page document that visualises different parts of the product or business, such as value proposition, infrastructure, customers and finance section.

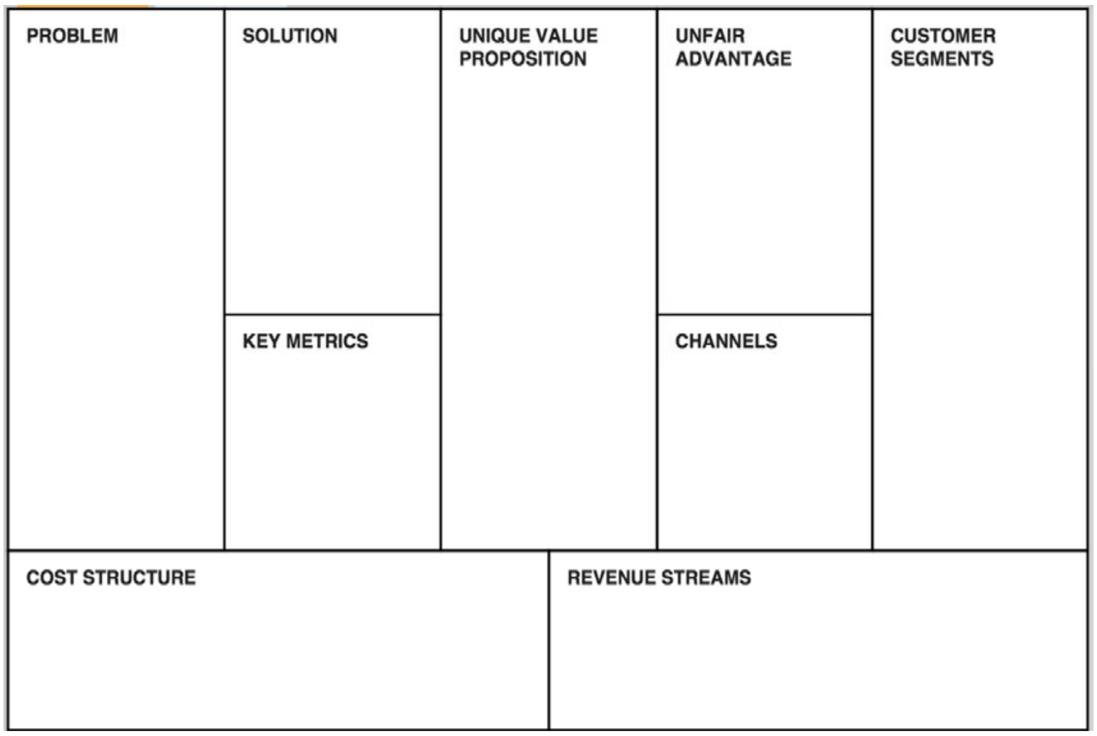


Figure 2.1: Example of a standard lean canvas template.

Adaptation of lean canvas for business strategy creation addresses many issues such as acknowledge the problem, scrutinize the product and services the team needs to focus on, identification of key metrics, competitive advantage factors, unfair advantage, customer relationship and key partnership.

Problem to solve the dilemma: The reason of failure of most start-ups is not creating what they wanted to create. It is a waste of time, capital, and effort in making the wrong product. It happens because not enough effort to " scrutinize the problem" is made from the beginning. There is a separate problem in the box depicted separately.

Product / Service solution: Once the business owner examines the Problem, the business owner is in an excellent position to define solutions that can solve it. In this situation, a small box is deliberately assigned to the enthusiastic 'solution'. Often, the business owner gets stuck in an untested solution if it comes to his mind for the first time. Keeping the solution box small is in line with the minimum viable product (MVP) concept.

Key metrics: Start-ups often fall into the sea of numbers. It is because teams are trying to bring order to the chaos, which is uncertain. However, at any time, there are only few key activities (or key indicators) that matter. Failure to accurately define critical indicators can be a disaster. They are useless activities, such as optimising early, depleting resources to achieve the wrong goal, and so on. If the user focuses on the core indicator from the beginning, it becomes a gross core engine later.

Unfair advantage of competitive advantage factors: It is a competitive advantage or barriers to business planning. Of course, the user knows that there is a tiny start-up that has a real competitive advantage from day one. In other words, this box is blank. This box is not meant to make the user take courage and stop pursuing one’s vision. It is always about thinking about unfair advantages and the encouragement to find and make it. Once the start-up has achieved some initial success, competitors and copycats are coming into the market. The start-up cannot defend against these fast followers, it is in danger of becoming extinct, and this is a real risk.

Customer relationship: Business owners begin with a direct relationship with the customer, no matter what task the business proprietor carries out. The business owner interviews the customers. Then it is necessary to figure out the way to the customers that suits the solution and customer segment (appropriate path to customers). It is covered in the existing channel block.

Key partnership: It is hard to get rid of this box, and it causes much controversy for the business owner. In order to produce certain products indeed it is necessary to find the right partners first to be able to predict success. For example, attempting to create a global photovoltaic grid platform, investors need to have access to considerable capital resources and have to establish relationships with regulators, so a critical partnership is needed first. "Business owner proven product-free start-up, trying to get a partnership from scratch can be a waste of time." Of course, as time goes, partners become increasingly crucial in optimising their business models. However, absence of partners is not a risk. Instead, it depends on the cost of the structure and channel. Each section of the document focuses on the core challenge and solutions in one single document.

Applications of the lean canvas are: identifying the customer segments, a group of people can do structured discussions, fast to gather ideas and possibilities, visual and intuitive, empower teams to create and think big and keep people on the same page. Considering above all investigation on lean canvas design, it increases the favourable condition for software test strategy creation. Use of lean canvas design on different software development projects provide further insights for testing process. In broad view, it is also possible to use lean canvas [134] in the software testing process to simplify the testing process and create a test strategy.

This sub-chapter has demonstrated that lean canvas is used only for solving the business and start-up challenges. In the broad view, it is also possible to use lean canvas in the software testing [137], [141]. Lean canvas model may be adapted for the scrum software testing process that may directly impact the improvement of the software quality [134]. The author wanted to investigate whether the existing lean canvas model with nine blocks is sufficient for test strategy visualization or further changes are needed. The chapter 2.4 describes user experience design, persona's point of view, in our case, "Software tester" is a user.

2.3 Lean User Experience Design

The author aims to develop scrutinize of the test strategy creator interaction patterns as an end-user of lean canvas. The author also intends to find out whether there is any need for change in the existing lean canvas design for strategy design and visualisation. In this sub-chapter, the author highlights user experience design and considerations related thereto.

In terms of the user, experience design is not about interface behaviour. It is about human behaviour and adapting technology accordingly. In the first step it should be uncovered what a designer is. A designer is an architect, a theoretical builder, who creates fictional artefacts, which later become real products.

Marsh [125] claims that the team makes sure that the digital product is constructed according to the needs of the user and requests of the stakeholder. User Experience Design (UXD) is the designing process the objective of which is to design digital systems, interfaces and applications to offer the most efficient and straightforward user-friendliness. Thus, user experience design embraces theories of many disciplines, such as user interface design, usability, accessibility, information architecture and human-computer interaction (Figure 2.2).

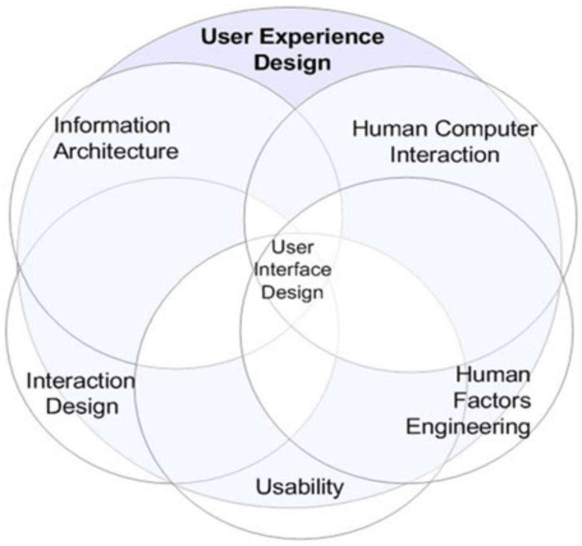


Figure 2.2: User-experience design considerations [125].

The above-mentioned disciplines are included in the user-experience design. The process often starts with studying who the users are, what information they need to use to succeed in creating products and services according to their expectations and needs through different methods. The products/services are continuously validated through real user feedback and later iterated to ensure that the product is as satisfactory as possible.

User Experience Design Process

According to [126], the user experience design process denotes how the end user gets the most out of a product/service. The process is conducted through different methods depending on the project and issue at hand. Some main project intentions are to improve the existing services, as others are to develop new ones.

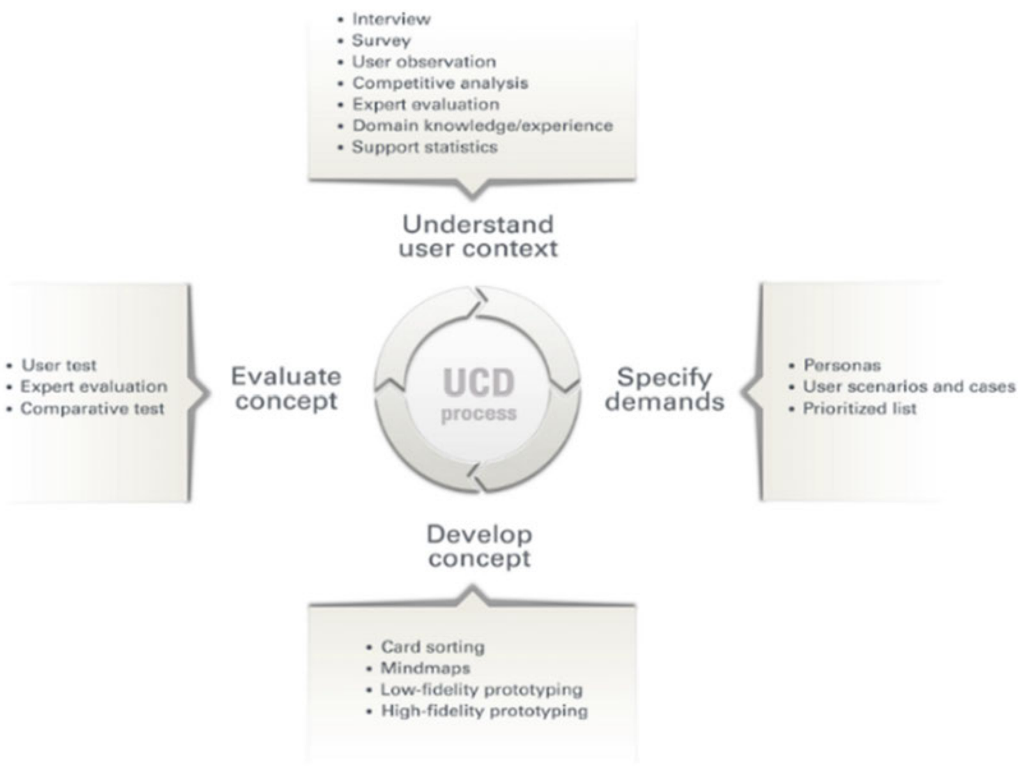


Figure 2.3: User-centred design model considerations [126].

According to the user-centred design model considerations (Figure 2.3), in the context of the present research, the software tester is viewed upon as a user and the tester starts evaluating design considerations.

1. Examining user context (Research) - An essential step of the process is to examine the user context. In this process, it is not known who the actual users are nor whether they understand the environment they use the application in. It is impossible to develop a user-friendly product. This step is used to gather facts about how the users think and behave, rather than rely on the opinion and speculation.

2. Specifying demands and developing the concept (Design) - Specifying demands is necessary to make the findings made in the previous process step concrete. It is the phase where software tester documents the issues; summarises, analyses and compares the results. At this point, software testers set the foundation for the next conceptual step to develop concepts. It is also an inventive step where the tester gets the chance to explore new ideas. It is an exciting part of the process where the software tester's ideas come alive.

3. Evaluating concepts (User Testing) - To ensure quality, the tester must evaluate the concepts against the testing process and the project itself. According to [127], [128] and [130], the idea is to find areas of improvement when evaluating and going back to develop and refine the concept further. It is a natural, effective, and cheap way to make sure the tester concept is quality assured before implementing a test strategy.

4. Iteration - Repeating the process to maximise the result. User experience design is a way to clarify, make something that appears complicated easily understandable in a way that it is efficient, satisfying, and effective. There are many reasons why testers shall work with user experience and user-centred design. With good user experience and a user-centred design process, the tester can reduce costs on development, maintenance, redesign, support, and training. The tester can also increase product quality, improve the testing process; overall, it increases the trust in the software and the user's satisfaction.

In order to study further adoption of user-experience and lean canvas for visualisation the author continued investigation of design personas and discussed that issue in the next chapter.

2.4 Design Personas Point of View

Personas are fictional characters that have general characteristics of the persons who create the test strategy researched by the author. Before proceeding to design of the lean canvas test strategy directly, it is necessary to identify the needs of personas. In our case, a software tester, test manager, team member, and test lead are focused on first and foremost. The author used the agile approach to create personas; an agile user experience practitioner creates personas collaboratively. Grant and Mittal [126], [131] discuss that the persona is a typical or standard user (the famous archetype), a fictional representation of possible target users, and it is used to set priorities and guide interface design decisions.

Personas are strategically an essential component of the product vision. Naturally, it is a target who will use the product, so it is important to establish the needs of this target. The persona allows the agile team to build and deploy a shared vision of service or product users in a highly engaging format. Personas help to focus on and examining what users want, their behaviours, their needs and expectations. The other benefit of personas for agility is that they can be easily related to user stories, which are very popular among agile teams.

To sum up, personas are an excellent way to integrate the user experience throughout their application in the development projects. The role of the agile user experience (UX) practitioner is that of a promoter as well as initiator of team collaboration with the product owner and the team members [9].

Creating Personas with Agile Teams

The author intends to create personas to examining the tester who draws up the test strategy and represents persona’s face while visualizing it on lean canvas. In this process, the author used an agile team to create personas to get early feedback strategy visualization. According to [126], the process of building personas must start at the beginning of the project, before the first sprinting start. Sprint 0 or short exploration period is a particular moment, which is very suitable for creating personas. However, this creative process requires mobilisation of everyone (product owner and team) and is resolutely collaborative. It is facilitated by the activity of the agile user experience design practitioner.

According to [125] and [126], the following steps should be made to account for the persona’s point of view:

1. **Initial preparation**

It is necessary to organise one or two workshops with the software tester and various stakeholders to align objectives, identify the data sources, and determine the first categories of people to interview (marketing segments or roles in a business application are relatively frequent starting points); to sensitise the team on personas, techniques, and its benefits; to collect data from various sources previously identified, including going to the field and interviewing users.

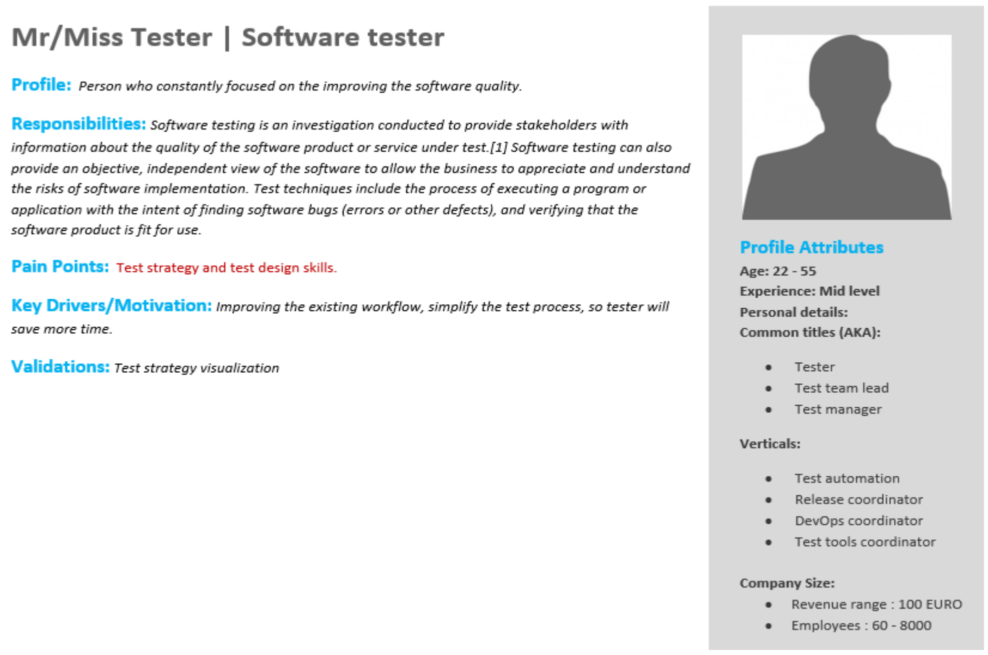
2. **Building together**

It is necessary to analyse data (facts) collaboratively in workshops and identify variables and patterns; to create persona skeletons; to personify and give birth to tester personas by working in the elements of context, storytelling according to a pre-established template (formal or not) and validate the results with primary stakeholders (or even quantitatively).

3. Communication and use

The personas should be placed on the information radiator in the team’s work environment. It is necessary to link personas to user stories; to use them as a tool to prioritise product backlog and user story design (as a conversation element of them). If necessary, a communication plan for marketing, sales, or training should be set up.

A diagram in Figure 2.4 showcases the software tester persona template. The personas method is an agile user experience that mainly reduces the distance between the end-users (represented by the personas) and the designers of the service or product.



Mr/Miss Tester | Software tester

Profile: *Person who constantly focused on the improving the software quality.*

Responsibilities: *Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test.[1] Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use.*

Pain Points: Test strategy and test design skills.

Key Drivers/Motivation: *Improving the existing workflow, simplify the test process, so tester will save more time.*

Validations: *Test strategy visualization*

Profile Attributes
Age: 22 - 55
Experience: Mid level
Personal details:
Common titles (AKA):

- Tester
- Test team lead
- Test manager

Verticals:

- Test automation
- Release coordinator
- DevOps coordinator
- Test tools coordinator

Company Size:

- Revenue range : 100 EURO
- Employees : 60 - 8000

Figure 2.4: Software tester persona template.

Utilization Of User Experience To Collecting Feedback

To analyse the user experience and feedbacks with lean canvas test strategy adopted project, following steps are carried out.

• In the first step, created the tester persona template (To analyse the difficulties faced by the software tester while creating the test strategy). Described in 2.4 Design Personas Point of View.

• In the second step, organised the interview session for software development team members to ask questions to fill in the persona template. (To introduce the lean canvas as a base model and how to use it for strategy creation). Described in the 3.5.1 Survey and Feedback.

• In the third step, given an empty 9 sub-block lean canvas test strategy template (electronic or paper printed) to use in the ongoing project.

• In the fourth step, collected the feedback to scrutinize the adoption (type of projects, resources) and utilisation (sub-blocks and design concerns) of visualised test strategy. Described in the 3.5.1 Survey and Feedback.

The final step was to come up with a visualise test strategy based on these findings. Described in the 3.8 Strengths and Limitations of the New Strategy

Within the research, the author interviewed different software development team members who used lean canvas as described in the 3.5.1 Survey and Feedback.

. As described above the fourth step received and main two feedbacks:

1. The nine-block lean canvas might not fit for different kinds of test projects
2. Large information text not fit in the block

In next step, the author further investigated visual management and lean canvas design and evaluation technique.

This section has analysed the testing process from the personas' point of view. The key player is the "software tester" who is mainly focused on software quality. Personas allow developing clear examining who the target of a visualised test strategy is, who will use it in the software development project. These concepts related to visual management, which are used to design the lean canvas test strategy boards, are discussed in sub-chapter 2.5.

2.5 Lean Canvas Design and Evaluation Technique

In this sub-chapter, the author demonstrates adoption of the lean canvas design framework and its evaluation. The lean whiteboard design is essential to design a whiteboard that covers the problem domain, it can be further effectively used by agile product teams, user experience design and user interface design teams and other project associates.

According to [130] and [131], the information radiator in the best way embodies the principles of Visual Management. It is an integral part of any lean and agile approach. The idea is to gather essential information about the product under development in one place, making it accessible and visible to all, and present in real-time its leading progress indicators. This whiteboard should cover all possible problems and points of view in the purest form, later it can be visualised on one page. The author suggests to note down all possible parameters and create a structure around it for a better design. The use a structured framework improves the work in the iterative way and facilitate to build strong relations between the design and problem domain.

According to design literature [125], [126] and [187], the design whiteboard should be divided into two parts: a) quadrants; b) experience. While designing the solution, there are no restrictions on how to follow the steps, and this can help a design team to work on the same page linearly.

According to [187], in the first step four quadrants are drawn on the whiteboard: user needs, assumptions, user goals, and constraints are used for the lean whiteboard design (Figure 2.5).

User needs: Considering the problem domain, it is necessary to start asking questions to software test professionals about their problems and reviewing other previous techniques by other researchers about the main design problem. Possible user needs should be listed. The goal of defining user needs is to produce a design that is user-friendly. In the first step, it is necessary to define the needs, describing user goals and expectations. The list of needs that move towards the goal of the user should be drawn up.

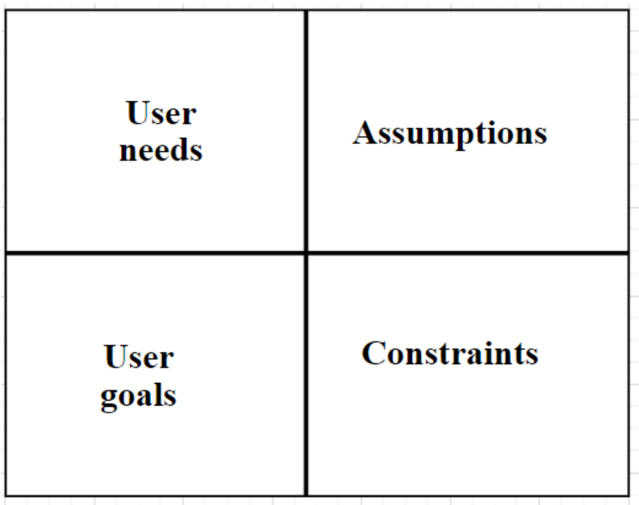


Figure 2.5: Lean whiteboard design [187].

In the first step, needs should be defined: There should be a description of user goals and expectations. It is recommended to make the list of needs that move towards the goal of the user. In the second step, scenarios should be generated: Once the user needs are defined, in the later steps, it is necessary to define scenarios that meet all needs. In the third step, an initial concept should be developed: The designer considers ambiguous issues and together with the team brainstorms the initial concept of the lean whiteboard design. In the fourth step, features should be defined: The designer prioritises the features based on the user needs at each stage of the design.

User goals: It is essential to know what the end-user goal is to start learning further about the users, start noticing the patterns and an overarching goal. In design, a developer should try to address the user goal. To achieve user goals, it is required to define the goals, generate scenarios, develop an initial concept and define features.

Constraints: It is necessary to learn further about the user needs in design and start noticing more constraints in the process. In this process, the business goals of the solution should also be checked.

Assumptions: In the design process, it is essential to find out what kind of design stands out in the problem solution. It is essential to list all assumptions on board to know the perfect goal. The assumption can be validated whenever needed by means if the interviewer, and sometimes few assumptions cannot remain unverified, and it is still acceptable.

According to [187], in the second step, user experience journey information should be collected.

The user experience journey: It is focused on interaction design and whether the anticipated user behaviour corresponds to the actual user behaviour.

Current journey/User-flow: Mittal [125] claims that the primary goal is what user flow needs to achieve. Once a user starts the journey, he/she needs to reach the goal as quickly as possible. There is always a chance the user does not arrive at the final goal because of the break in the real journey. Whenever the user journey breaks, brainstorming needs to be done to fill in the gap.

Sketches/New experience: It is essential to draw and play around the ideas on the whiteboard considering a high-level user experience first. It is recommended to start with the best ideal experience considering constraints and assumptions.

In some cases, it is necessary to design further details considering endpoints. Here some time is needed to do brainstorming to address the endpoints. Asking questions and drawing the design in parallel with the solutions should be done.

2.6 Summary

In this study, the author concluded collecting 33 individual software testers/developers and early adopters' feedback, that the existing nine sub-blocks are not sufficient for strategy visualisation as mark out in table 3.5 questions and answers used in the survey. To scrutinize the problem, the author collected the personas point of view. In the next step, the author re-evaluated the entire solution with lean canvas design and evaluation technique. Thus, re-evaluating the ideas from the personas point of view, learning the basics of user experience design and user interface design allows formulating suggestions and guidelines to develop a better-visualized test strategy of lean canvas.

3. DEVELOPMENT OF VISUALIZATION APPROACH

The previous chapter presented formulation of the research problems and provided introduction to lean canvas. It also discussed the appropriation of lean canvas in the testing process, utilising the user experience design and user interface design principles.

This chapter focuses on finding the TABB (Test activities building block) for the lean canvas and experimental pilot project used to develop and evaluate it. The author discussed possible identification and classification of software development life cycle waste in the test process. In the next sub-chapter, the author discussed the system design and end-user utilization of the designed system. In the sub-chapter dedicated to the role and limitations of the system design, the author investigated design limitations and collected feedback. In the concluding sub-chapter, the author used the data from different sources get detailed insights. The sub-chapter explained lean model transformation and subsequently collected the data by means of quantitative statistical analysis: experiment planning and data collection were performed.

Test Process Waste Classification

According [133] and [134] in the software development life cycle, the team may encounter different types of waste, this waste comes from different sources. If the team does not address waste, the project might take more time than planned. Waste may have direct or indirect impact on the development and test life cycle, in this situation, software development life cycle waste dentification gains momentum (Figure 3.1). The author suggests thinking critically to move towards additional lean thinking with development and software testing before adopting visualization approach itself.

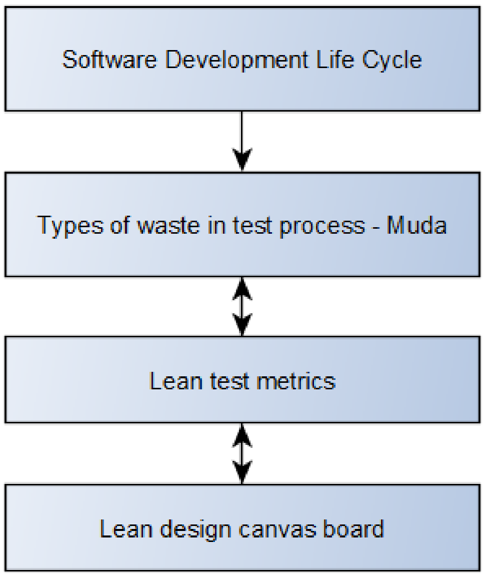

Software Development Life Cycle
Types of waste in test process - Muda
Lean test metrics
Lean design canvas board

Figure 3.1: Software development life cycle waste identification.

According to [118], [119], [115] and [117], further investigation into how to determine and eliminate the existing eight types of waste is significant in lean manufacturing. Recognition of the solutions may also be useful for those interested in this issue in software development. Such a situation motivated this investigation, which aims at identifying diverse types of possible waste and analysing whether the similarities may be integrated in the agile software test process. Eight waste defects, namely, overproduction, waiting, inventory, motion, transportation, over-processing, and skills, formerly came from the lean manufacturing. Such concepts as Seven Muda and Six Sigma also mention the waste of skills. These eight wastes were analysed within software test process to offer solutions to eliminate or minimise undesirable issues in the test process (Table 3.1). All possible inputs as described in the examples below table are further used in the creation of the visualised test strategy.

Table 3.1

Different Waste Situations in Software Development and Test Process

Different waste	**Example situations in software development and test process**
Defects	In the software test process, when a defect occurs, a software developer spends much time trying to fix the defect, then it takes additional time for the test engineer to re-test the defect again.

Different waste	**Example situations in software development and test process**
Overproduction	In the software testing process, before the tester starts creating a test document, ask tester self-questions or ask a tester team member or test lead who read it and what value it brings. If there is no answer or the answer is ambiguous, it is best not to create it to avoid overproduction.
Waiting	In software development project, the testing team often has to wait for the build to be delivered from the development team for testing.
Inventory	In software test process number of redundant test documents or many software builds at the same time (due to waiting) to be processed result in creation of inventory.
Transportation and Motion	In many of the software testing projects, teams are geographically distributed and networked in a complex way, multi-tiered security leads to the fact that building, installing, testing, obtaining the test results and identifying defects require a lot of time and effort.
Over-Processing	In the some of the small software testing projects adopt the standards such as capability maturity model to manage the project leads to overprocessing.
Skills	It is associated with under-utilizing skills and assigning tasks with incomplete training to software test team members and believing team members do the task.

In the above table, all possible Muda were analysed from the viewpoint of software testing. Although new types of Muda are added in the Kaizen Event Implementation Manual from time to time, currently, all wastes are universally accepted in the industry.

Before proceeding, the author proposes a new strategy. The research discusses eight wastes in software testing based on eight types of waste in lean production, but this does not mean that there might be only eight types of waste in software testing. It should be considered that project activities and testing activities should be done every day, and it is important to analyse whether the test team wastes anything. According to [116] and [117], waste is not necessarily harmful to the project, but defining and eliminating waste is one of the critical activities in process improvement aimed at working effectively and economically.

As described above seven types of possible waste author applied in visualising test activities building blocks. To achieve the lean visualising test activities building blocks following steps need to be considered.

Step 1: Keep lean principles in the analysis in software testing process (above seven types of possible waste, corelated to testing process), in this process we find the fine-tuned primary key meta titles.

Step 2: Apply lean principles in creating TABB. (In this process we do not add every meat title information).

Step 3: TABB creation. (Visualises, update according to product life cycle)

TABB is a distinctive approach that visualises the key areas that are critical for achieving the best performance, hence minimising waste.

In the sub-chapter 3.2 TABB creation discussed in detail.

In this sub-chapter, classification of seven types of waste was discussed, which allowed finding the primary key meta title for the lean canvas board.

3.1 Developing Ontology

This sub-chapter describes adoption of ontology in this research work. The author intends to collect information on TABB within ongoing software development and testing project. In this situation, the author used a mind map as an ontology.

Software testing is an essential activity in the lifecycle development, it produces a large amount of knowledge. Software testers need to collect domain information to be able to manage a software testing activity successfully. Lack of access to proper information base within its individual context in software testing circumstances makes software testers deal with insufficient knowledge or have to consult peer software testers, which considerably influences their decision-making process. The ontology is supposed to represent a heuristic execution task of software testing and fine-tuning domain. No one can adequately represent the domain in ontology because, as shown in [169], an ontology is not able to consider in its models and rules all possibilities of a specific domain. Consequently, the ontology became dependent on the concepts used by the heuristics, when the ontology is applied until it is extended to include the other rules and concepts of new heuristics.

If the metadata are used for structuring the content, both thesauri and ontologies make semantics possible to build them. Ontology is a specification of a conceptualisation, that is, a common framework or a systematised and consensus conceptual structure not only to store information, but also to be able to search and retrieve it. Ontology describes the fundamental terms and relationships for compression of an area of knowledge, as well as the rules to combine the terms to define the extensions of this type of controlled vocabulary.

It is about converting information into knowledge through formalised knowledge structures (ontologies) that reference the data, utilising metadata, under a standardised scheme on some domain of knowledge. The metadata will not only specify the data scheme that should appear in each instance but may also contain additional information on how to make deductions about them, that is, how to establish axioms that may, in turn, be applied in different domains that deal with the stored knowledge. In this way, search engines can obtain information by sharing the same web annotation schemes and software agents. Not only will they find the precise information, they can also make inferences automatically looking for information related to what is located on the web pages and with the requirements of the queries made by the users. Besides, producers of web pages and services are able to exchange their data following these collective consensus schemes and may even reuse them.

The benefits of using ontology can be summarised as follows: They provide a way to interpret and share knowledge utilising a common vocabulary, allow using a knowledge exchange format, provide a specific communication protocol and allow reuse of knowledge.

The term ontology has been used for many centuries in the field of philosophy and knowledge management, and several decades ago it gained particular relevance in the field of library science and documentation. In the recent years it has experienced a new impulse due to the development of the Semantic Web, where the idea of transforming the network not only into an information space but also into a knowledge space prevails.

The development of testing activity builds a block strategy on the lean canvas for testing strategy generation involving different types of testing, mind maps tools are used for this purpose. Although ontology mind maps and use cases are built in parallel, they determine precise basic requirements before starting creation of the use case. Considering the suggestions in [170] and [171], the author starts with the ontology mind map for core software testing (Figure 3.2).

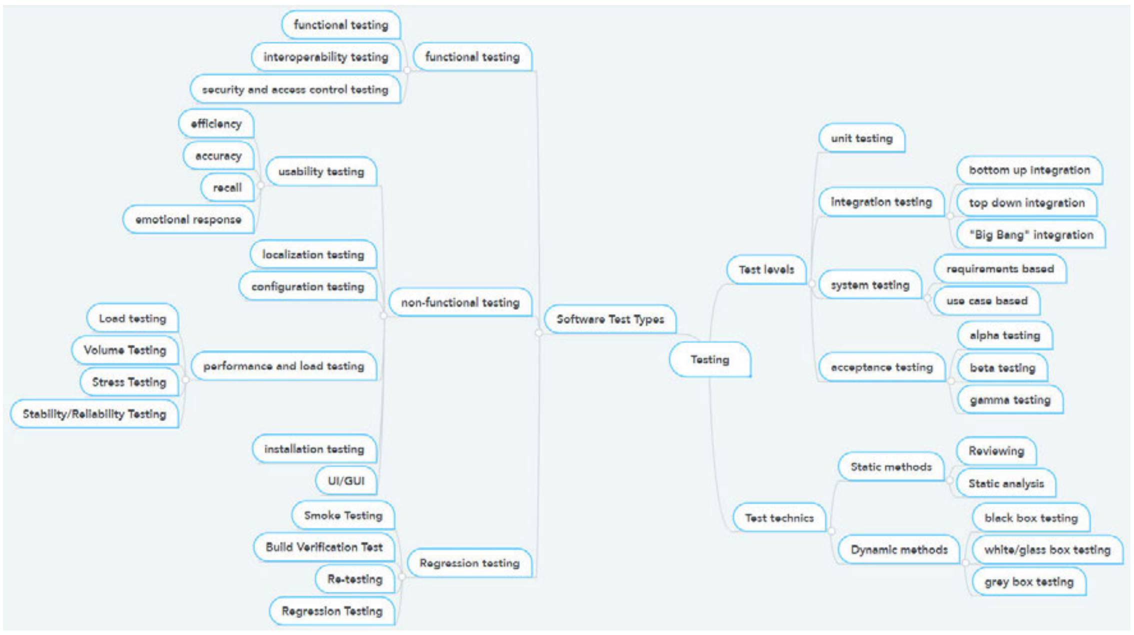


Figure 3.2: Ontology as a mind map for core software testing (example).

Benefits of the ontology mind maps are intuitive visualization, guidance for authoring process, online collaboration, offline operation and use case generation. The author suggests viewing ontology as a mind map to represent the knowledge management system for software testing, in the next step, the author wanted to use collected mind map meta tags TABB. In the next sub-chapter, the author in detail discusses how to design lean canvas test activities building block and further a visualized lean canvas test strategy in three steps.

3.2 Test Activities Building Block

It is important to have appropriate titles of blocks of the canvas, each block focusing on a different factor. The titles guide users to get meaningful information quickly from the observation. In this sub-chapter, creation of blocks, sub-blocks, titling and subtitling process are explained.

Three main steps to design lean canvas test activities building block include:

In step one the primary key meta title for the board is identified. A user can use the existing test strategy document or brainstorm with team members to scrutinize the critical meta title for the board (e.g.: Testing deliverables).

In step two, key meta title is split into smaller ones. Using the lean principles, the critical meta title is found and added on the lean canvas board. Later the meta description is added. In step three, the meta title or meta description is modified. The meta title or meta description is modified according to the software development life cycle.

Figure 3.3 demonstrates the starting point of blocks and sub-blocks, proposing the TABB for the lean canvas blocks and sub- blocks.

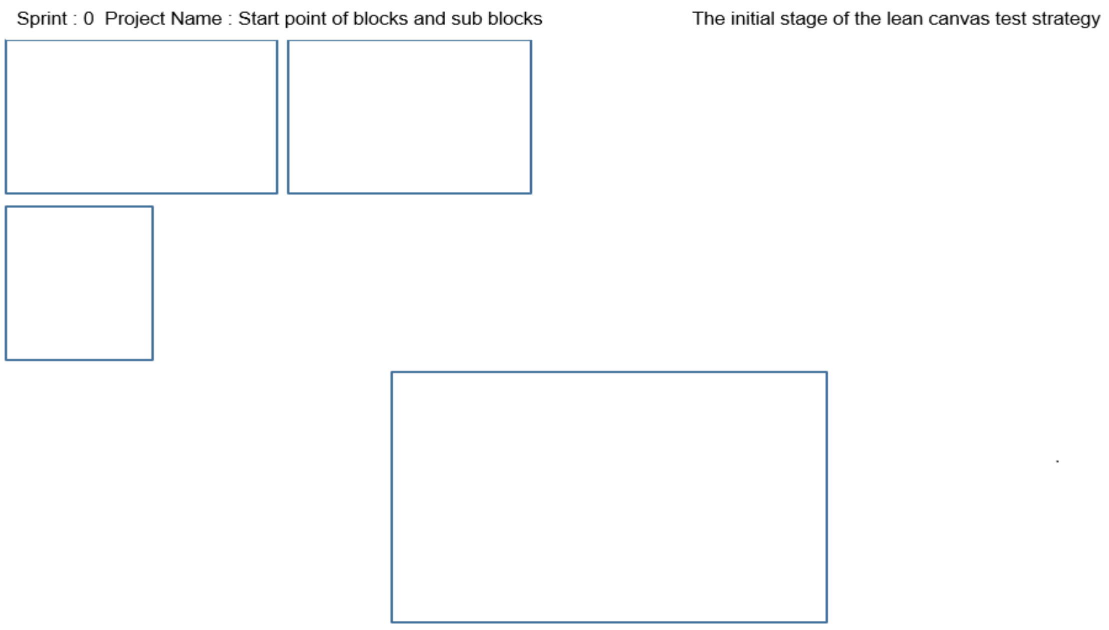

Sprint : 0 Project Name : Start point of blocks and sub blocks
The initial stage of the lean canvas test strategy

Figure 3.3: Starting point of blocks and sub-blocks.

Considering the lean concept, analysis of the possible waste in the software development process and test process provides a better insight into finding of a better meta title or meta description for blocks. In the next step, additional empty blocks and sub-blocks are created (Figure 3.4). The author described each step-in detail in visualized lean canvas test strategy design.

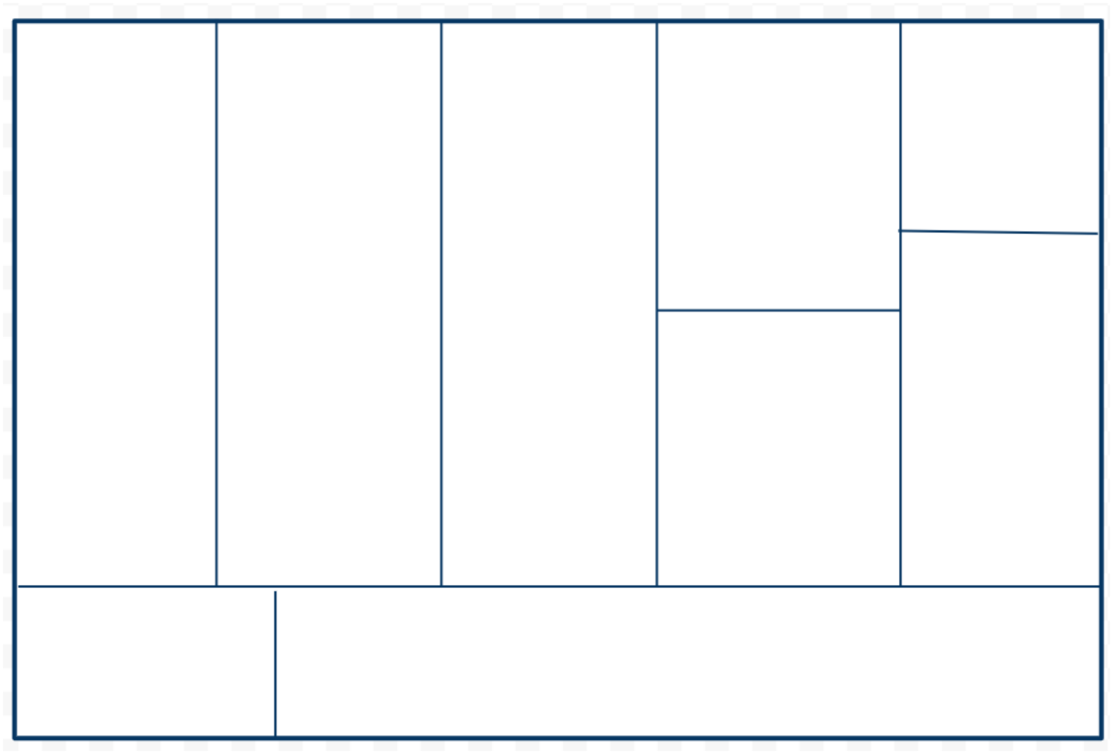


Figure 3.4: Empty blocks and sub-blocks.

Visualized lean canvas test strategy design in steps in detail:

Step 1: Identify the current type of software development process.

In this step, the software development process is identified (e.g.: Agile, Waterfall, Lean);

Step 2: Identify the type of software testing.

In this step, the type of software testing is identified (e.g.: regression test, performance test);

Step 3: Use any existing test strategy or test documents.

In this step, investigate and use any existing test strategy or test documents OR

Run a brainstorm session with the team in order to create one (e.g.: test plan document, test strategy document);

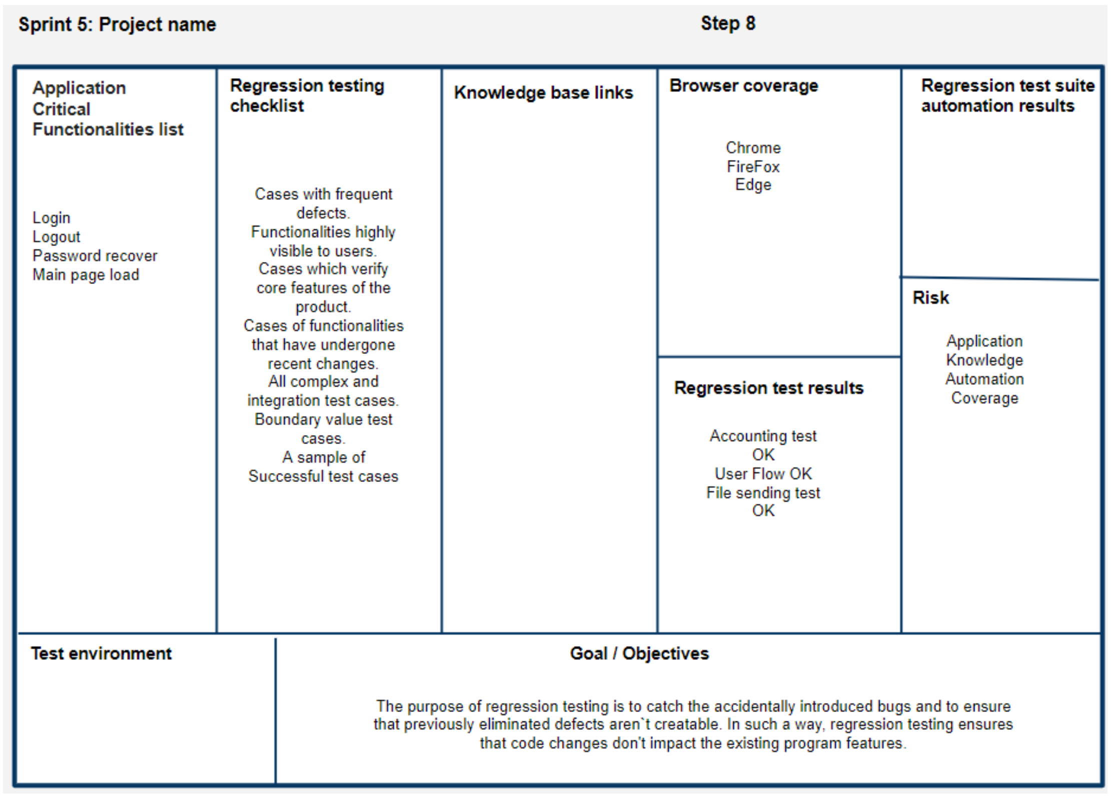
Step 4: Identify the primary key meta title for the board and use empty canvas with the board and start adding sub-blocks (e.g.: test levels, testing scope, test deliverables, risk);

Step 5: Split key meta title into smaller ones and name the appropriate key meta title names for blocks and sub-blocks (e.g.: manual, automation);

Step 6: Start filling the short meta description in sub-blocks (Figure 3.5), (e.g.: testing scope in sub-blocks as a scope of testing, out of scope);

Step 7: Visualized test strategy is ready to be used.

Step 8: In case it is needed after each sprint or after development phase, re-review the existing test strategy and modify a meta title or meta description.



Sprint 5: Project name **Step 8**

Application Critical Functionalities list

Login
Logout
Password recover
Main page load

Regression testing checklist

Cases with frequent defects.
Functionalities highly visible to users.
Cases which verify core features of the product.
Cases of functionalities that have undergone recent changes.
All complex and integration test cases.
Boundary value test cases.
A sample of Successful test cases

Knowledge base links

Browser coverage

Chrome
FireFox
Edge

Regression test results

Accounting test OK
User Flow OK
File sending test OK

Regression test suite automation results

Risk

Application
Knowledge
Automation
Coverage

Test environment

Goal / Objectives

The purpose of regression testing is to catch the accidentally introduced bugs and to ensure that previously eliminated defects aren`t creatable. In such a way, regression testing ensures that code changes don't impact the existing program features.

Figure 3.5: Writing titles and filling description.

Considering the above-presented figure, the titles of the first block feed the transformation process and output feedback is transformed into the input. This concept is used to identify the right title for a lean canvas box.

The author emphasises the lean approach at each stage, as discussed in chapter 3 Test Process Waste Classification for TABB (Test activities building block) for lean canvas board (Figure 3.6).

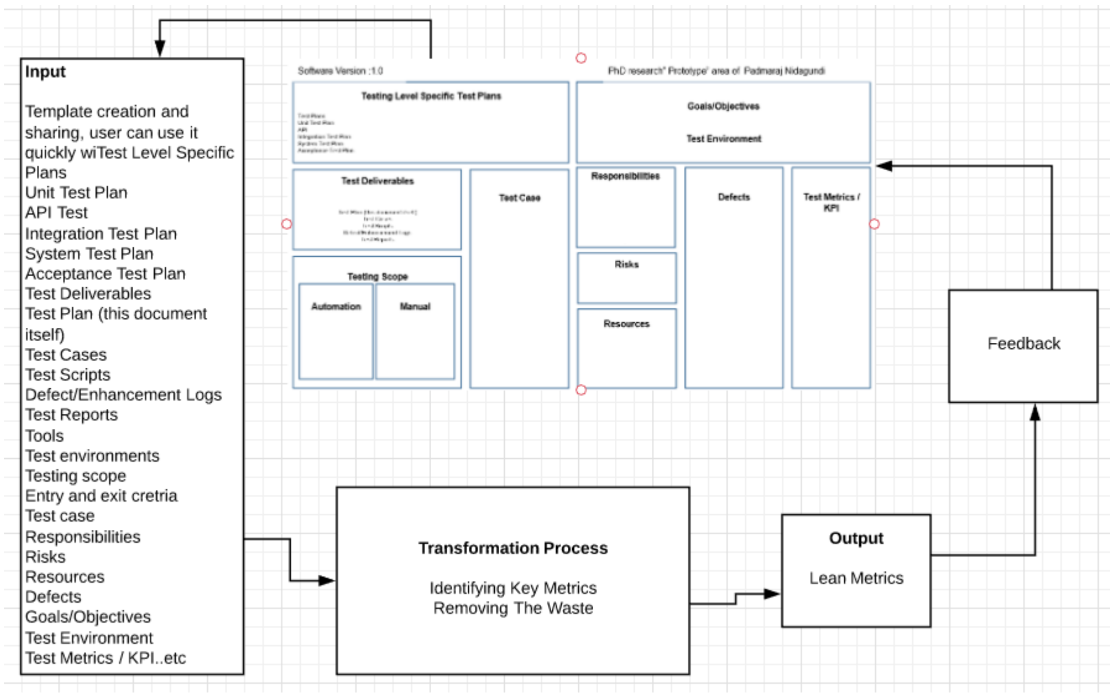


Figure 3.6: Test activities building block (TABB) strategy for the lean canvas blocks and sub-blocks.

Input: Collection of critical terms used in the existing test strategy or test process, if there is no historical evidence available for test strategy, then the team can get these from the brainstorm session.

Transformation process: In this phase, adopting the lean principles, undesired meta title or meta description needs to be removed. The aim is to keep important titles only for the board [135]. While using the transformation process, users need to keep in mind 8 lean principles. Also, the user can receive feedback from the team on what they view as an essential title that needs to be visualised further.

Output: Collected key metrics is considered as a title, sub-titles or short description for the visualised lean canvas board.

Feedback: It is a collected key metrics proposed, further these key metrics titles are reused in the lean canvas board.

The preceding section presented the description of the TABB (Test activities building block) for the lean canvas blocks and sub-blocks lifecycle used to explain and visualise the lean canvas test strategy design in eight steps.

Formal Specification of TABB

The author bridges the gap between informal models and a formal model using the Unified Modelling Language (UML) class diagram (Figure 3.5).

In the first step, an empty board is created, and the user adds an appropriates meta title based on the strategy document or brainstorming with team (e.g., testing deliverables). In the later stage, the user can add additional meta titles or remove them. The user can add description to the meta model, as already explained in the visualized lean canvas test strategy design in steps:

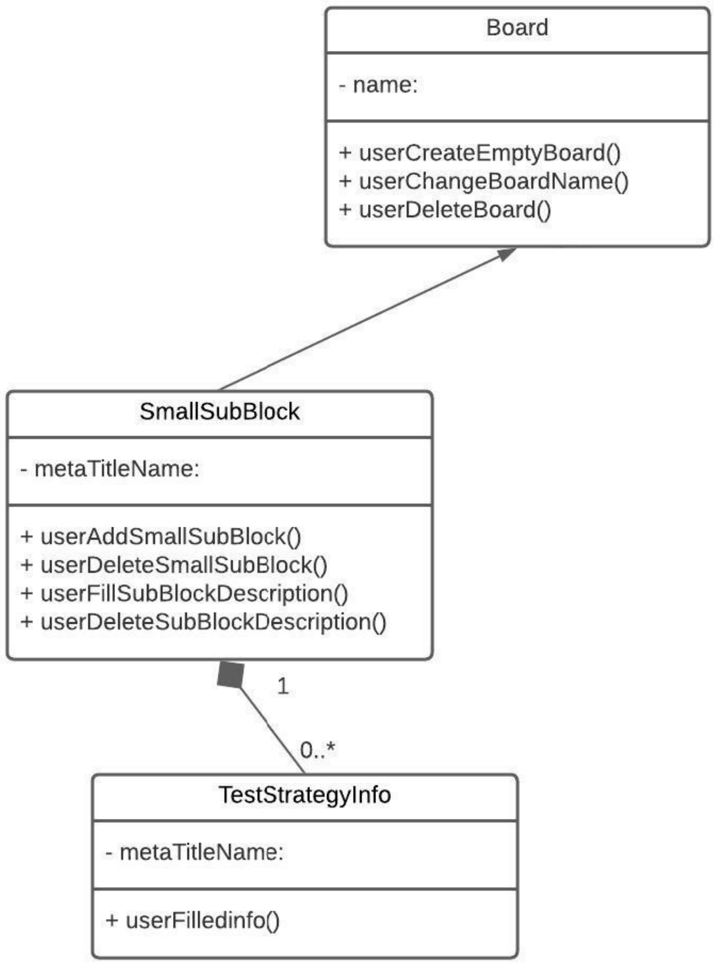


Figure 3.7: Formal approach to TABB using UML class diagram.

As a result of investigation, it is further possible to develop as a conceptual solution (Figure 3.8). Formal methods have proven efficient in the development of an explanatory system.

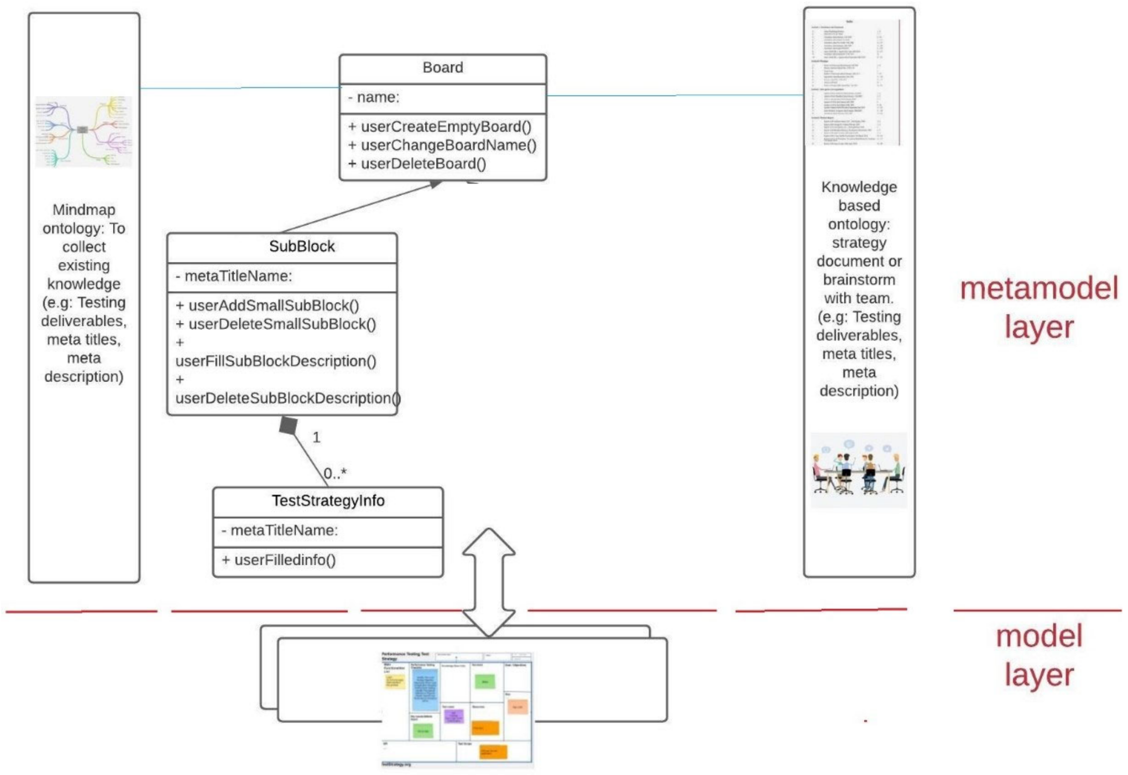

Board
- name:
+ userCreateEmptyBoard()
+ userChangeBoardName()
+ userDeleteBoard()
SubBlock
- metaTitleName:
+ userAddSmallSubBlock()
+ userDeleteSmallSubBlock()
+ userFillSubBlockDescription()
+ userDeleteSubBlockDescription()
1
0..*
TestStrategyInfo
- metaTitleName:
+ userFilledinfo()
Mindmap ontology: To collect existing knowledge (e.g: Testing deliverables, meta titles, meta description)
Knowledge based ontology: strategy document or brainstorm with team. (e.g: Testing deliverables, meta titles, meta description)
metamodel layer
model layer

Figure 3.8: Conceptual solution for a graphical representation of TABB generation.

As demonstrated above, knowledge comes in meta format from different sources, it depends on the user and at a later stage information is put into the TABB model format. Considering all inputs discussed above, the author explains the system design in the next sub-chapter.

3.3 System Design

In this sub-chapter, the author demonstrates the new system design technology requirements. The idea is to develop a working online application where ready-to-use TABB templates are integrated. However, there are different possibilities to design this system using different technologies. The author suggests to use opensource technologies to design test activity building blocks.

From the previous sub-chapter dedicated to test activities building blocks, system design requirements are analysed in term of technology, usage and possible impact on test activities building block application design (Table 3.2). To make system design, all received plans are mapped onto technology, usage and impact in the application listed in the table below.

Table 3.2

Technology Selected for the Pilot Project

Technology	**Usage**	**Impact on application**
PHP	Core programming of the application	Software pattern design
HTML5	Web programming of the application	Web interface design
MySQL	Application storage process	Storage and retrieval of information
CSS	Application view	UI / UX
Java Script	Application rendering	Server and client interaction
HTTPS	Secure SSL certification Protocol	Secure client and server communication
WordPress CMS	Content management	Easy to upgrade and add the contents Back up creation
Google documents	Google docs	Template creation and sharing with user

Turning to the experimental evidence on the pilot project, the study was conducted. Identifying the appropriate technologies and building the test strategy, a web portal with ready-to-use strategy template is applied for the testing process. The number of visualised test strategy templates is available at www.teststrategy.org

3.3.1 System Design Details

Within the pilot project, a number of use cases generated to design the system are demonstrated in Table 3.3. Utilising the information gathered about the actors and use cases, the pilot study projects are created in the domain www.teststrategy.org.

In the pilot project system, the first 13 use cases and 4 actor tables are identified. Each name, description and participants are described in Table 3.3.

Table 3.3

Actors and Use Cases

Name	**Description**	**Participates in**
Admin (A-2)	Admin is a person who collects people's ideas and makes them cerate test strategy templates for functional and non-functional testing. Admin also maintains the TestStrategy.org	UC-5 Maintaining the web portal UC-6 Creating and adding new test strategy templates
GDrive Test strategy template (A-4)	GDrive Test strategy template is a storage area to create and store test strategy templates	UC-10 Creating test strategy templates UC-11 Storing test strategy templates UC-12 Updating test strategy templates UC-13 Deleting test strategy templates
User (A-1)	User is a person who uses the system to generate the test strategy, this user downloads the test strategy template on Google drive and integrates into the ongoing project	UC-1 Using test strategy template UC-2 Suggesting new ideas for test strategy template UC-3 User downloads the functional testing types test strategy templates UC-4 User downloads the non-functional testing types test strategy templates
Web Mail System (A-3)	Web Mail System is a communication channel between admin and the system user	UC-7 Sending emails to user UC-8 Receiving emails from user and non-users UC-9 Sending bulk emails to user

The pilot project was created in four steps:

Step 1: Provides essential explanations of pilot project possibilities. In this step, considering the hypothesis test strategy, visualization possibilities are investigated, as discussed in the sub-chapter 3.3 System Design.

Step 2: Pilot project activities and possible solutions are developed. In this step, a number of use cases are identified. The pilot project user flows are developed at www.teststrategy.org for test strategy visualization in the sub-chapter 3.3.2 Use Case Overview.

Step 3: Advantages and loopholes of the pilot project are identified. In this step, strengths and limitations of the visualized software test strategy are identified after implementation of activities discussed in the sub-chapter 3.7 Importance of Strategy and Limitations.

Step 4: Provides a brief analysis, summing-up, and tips to move forward. In this step, research data are collected from the number of case studies for examination in the sub-chapter 3.5 Proposed Analysis.

3.3.2 Use Case Overview

This use case design is illustrated according to Table 3.3 Actors and Use Case Inputs. The next step is based on the number of use cases within the pilot study project at www.teststrategy.org. Figure 3.9 showcases the overview of connection between use cases and actors. In the final step, all possible use cases are implemented at teststrategy.org.

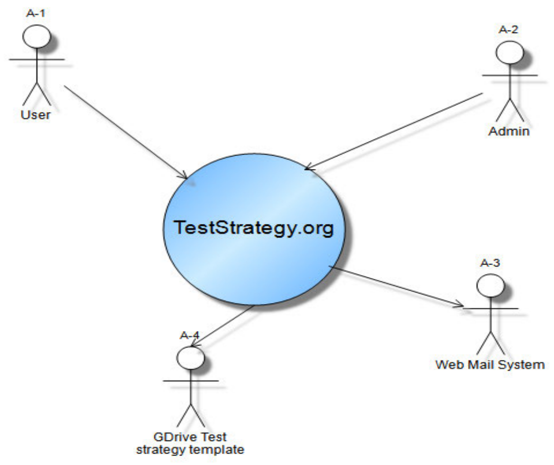


Figure 3.9: Use case main actors.

The goal of the use case is to design the system overview and identify the actors' connection overview (Figure 3.10). Using the use cases developed, the working software as a service application is implemented with the following considerations.

- Interpreting the goals of the system-user interactions: Collecting all significant user interactions and interpreting them to the system goal where users can create the visualised test activity building block using the lean canvas for test strategy.
- Representing and integrating functional requirements in a system: Identifying and developing the functionality of the system for the end-user. For this system development, the author used minimal most needed functionality for the system.
- Defining the circumstances and requirements of a system: Defining the system needs and creating the environment where the end-user can utilise the system. The author's focus was to ensure that the end-user may use the ready-to-use visualised template in an easy way.
- Illustrating the basic flow of transactions in a use case: Implementing the use case in the web application. The author recognises the basic flow of transactions with the group of two additional tester/system users from the workplace who are interested in giving early feedback in this process.

The above points, as the author described, are implemented into the system to use the visualised test strategy. Using the developed system, the end-user who wishes to use the test strategy in their project can download and use it directly. Ready-to-use templates are flexible to modify and re-use. At the moment, the author provided only a few numbers of the ready-to-use template to experiment and collect the feedback. The author intends to collect feedback in an incremental way and constantly improve the template based on end-user suggestions.

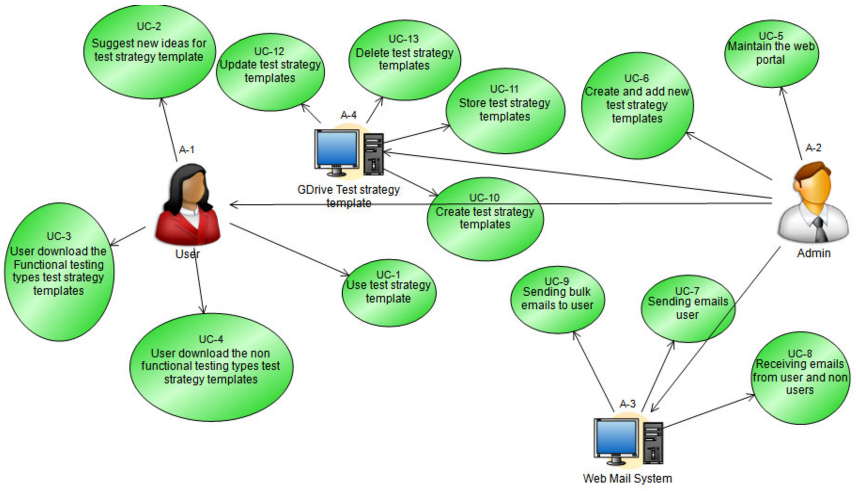

UC-2 Suggest new ideas for test strategy template
UC-12 Update test strategy templates
UC-13 Delete test strategy templates
UC-11 Store test strategy templates
UC-6 Create and add new test strategy templates
UC-5 Maintain the web portal
A-4
A-1
A-2
GDrive Test strategy template
UC-10 Create test strategy templates
User
Admin
UC-3 User download the Functional testing types test strategy templates
UC-1 Use test strategy template
UC-9 Sending bulk emails to user
UC-7 Sending emails user
UC-4 User download the non functional testing types test strategy templates
UC-8 Receiving emails from user and non users
A-3
Web Mail System

Figure 3.10: Overview of connection between the use case and actors.

In the next sub-chapters, the author discusses the limitations of the implemented system and TABB providing additional insights for further improvements.

3.4 Study on Prototype System

The author identified the system design requirements building TABB. In the process of building the first system prototype, the author received several feedbacks. In the later stage, these feedbacks were used for improvement of the system and TABB. In this process, some limitations were also identified. The triangulation method demonstrated that a set of questions should be focused on the users, such as testers, developers and test managers, to get feedback on the visualised test strategy. These questions are asked by individual users and adopters of the www.teststrategy.org. This chapter concludes with the limitations of the study.

3.4.1 Survey and Feedback

The process of exploring the strengths and limitations of the visualised test activity building block strategy required application of the exploratory research method to develop better scrutinize of a relatively new research field. This method basically involves two steps: a) analysis of the comments and feedback from online users; b) multiple-case study done within live projects. The author used different methods of data collection and used triangulation as a technique to analyse results (Figure 3.11).

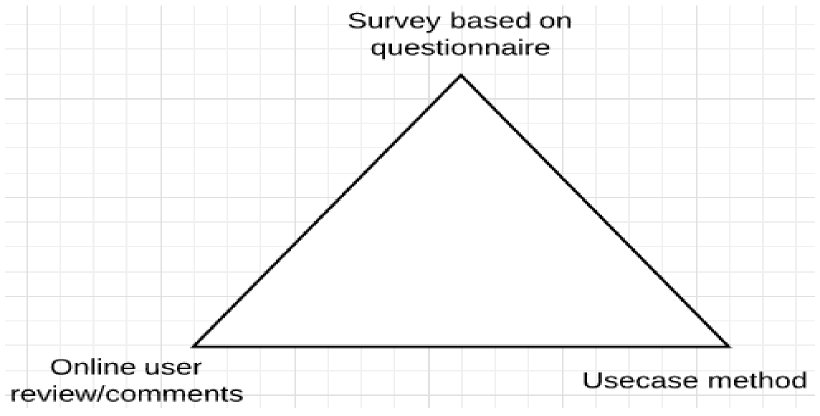


Figure 3.11: Quantitative and qualitative triangulation of the visualized test strategy.

In the first step, comments and feedback given by online users are collected and analysed. Most users are software testers; these users are early adopters of www.teststrategy.org. Table 3.4 provides the overview of quantitative and qualitative effort.

Table 3.4

Quantitative and Qualitative Triangulation of the Visualized Test Strategy

Quantitative and qualitative triangulation of the visualized test strategy			
Survey based on questionnaire	Quantitative research for prototyping and implementing	Selected member group discussion at the initial stage	2 groups participated
		Selected member group discussion after visualised lean canvas has been implemented	4 groups participated
		Table 3.5 Questions and answers of the survey	33 individuals participated
Online user review and comments	Qualitative research for design improvement	Table 3.6 Different feedback received at www.teststrategy.org.	15 independent feedback considered for design
Use case implementation	System design and improvement considerations	A number of users used ready TABB templates developed and stored at Gdrive www.teststrategy.org	Many designs implemented at www.teststrategy.org

In the second step, a multiple-case study done with live projects was used to analyse possible use cases and the experience of the interviewees with the visualised test activity building block strategy. In the overall process, the triangulation method was used to collect the results.

Survey Based on Questionnaire

Table 3.5 shows the questions and answers used in the survey. This method demonstrates a set of questions focused on the users such as testers, developers and test managers to get feedback on the visualised test strategy. 33 individual software testers/developers and early adopters of teststrategy.org were asked these eight questions.

Table 3.5

Questions and Answers Used in the Survey

Number	Questions	Answers
1	What type of testing you do use more frequently?	• Functional • Non-functional
2	What development methodology does your team use?	• Waterfall • Agile • XP • DevOps
3	Where is your team located?	• Local • Remote • Mixed (Local & Remote)
4	Do you work for a product-based company or service-based company?	• Product-based company • Service-based company • Mixed
5	Do you write detailed test documents?	• Yes • No
6	Would you like to use the visualised test strategy in testing process?	• Yes • No
7	Who is responsible in your team for creating a test strategy?	• Test lead • Test Manager • Tester • We have corporate level test strategy • We don't have any test strategy • Team
8	Would you like to use a visualised test strategy on a single page instead of a written document?	• Yes – I wish to visualise the test strategy; visualisation helps in analyse the information • No – I wish to document the test strategy

Online User Reviews and Comments

Table 3.6 presents the feedback received on www.teststrategy.org. In this process, 15 feedback/comments were considered.

Table 3.6

Feedback Received at www.teststrategy.org.

No	Type of testing	Feedback
1	Smoke test strategy	It is useful for my project and helped me in the visualization of test process and test strategy
2	Smoke test strategy	The new strategy is helpful, and in my ongoing project its adaptation reduces the documentation effort
3	Smoke test strategy	It is difficult to analyse for is tests for complex functionality. I think I need user guide / steps information for adaptation for me and my team.
4	Smoke test strategy	This visualization strategy fits into the small projects precisely. Small sub-blocks give additional information about the testing precise information.
5	Smoke test strategy	An excellent and easy method
6	Smoke test strategy	Test team manager or test team lead needs to know its adoption process.
7	System test strategy	The number of documentation pages could be effectively reduced adopting this test activity building block strategy
8	System test strategy	I will suggest this template and strategy to my team
9	System test strategy	I am now using this and will give you feedback next month
10	System test strategy	Well, it is just an easy way to reduce test process
11	System test strategy	One more new strategy, it maybe does not fit complex and substantial projects.
12	Security testing	Open-source foundation for application security area needs to be covered in additional detailed, it’s really complex to visualise each detail in the canvas size template
13	Performance testing	I think I can reuse this strategy for different performance testing
14	Performance testing	Performance testing now visualized. Most important, I think, is that multiple teams now use the same strategy for the ongoing project. This visualisation improved communication between the teams.
15	Performance testing	It's useful for high-level documentation only

On the website, a number of ready-to-use test activity building block test strategy templates were demonstrated. The website user could download test strategy templates and use them in the test process. The user could give feedback on each test strategy page as a comment or could send a message directly to the author from ‘Contact us’ page. These reviews were analysed and further considered in order to improve the existing visualized test strategy templates.

Use Case Implementation

In this step, use cases recognised in Table 3.3 Actors and Use Cases were implemented in the domain www.teststrategy.org.

3.5 Proposed Analysis

In the previous sub-chapters, the hypothesis was set forth, it highlighted the importance of the need for the visualised test strategy. The test strategy can be developed using a test activity building block strategy where information is divided into the sub-blocks. Considering the software development life cycle waste identification, the TABB (Test activities building block) for the lean canvas blocks and sub-blocks attests reusability of the strategy template.

Renaming the block titles according to the need of the testing context allows building a clearer test strategy. In agile software development life cycles, many aspects are constantly changing. The reusability of the test strategy template allows solving agile software testing issues.

TABB for the lean canvas blocks and sub-blocks addresses four key points of the manifesto for agile software development: The individuals, interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; responding to change over following a plan.

Considering IEEE 29119, the strategy is a sub-item of a test plan. In industry practise there are different strategies, such as analytical, model-based, methodical, process, dynamic, consultative or directed, regression-averse for test strategy; they are described in the text format in the documents. Every organisation has its internal standards and processes to maintain test documents. Some organisations include strategy as a part of the test plan. Test documents take additional time to create and update.

The idea of the test strategy is to define guidelines for testing. These are followed to accomplish the test objectives and ensure execution of different tests defined in the main testing plan. It focuses on the test objective, testing strategy, different test environments, automation strategy, testing tools and risk analysis with a disaster recovery plan.

The competence team was formed for practical testing activities for the visualised strategy. The competence group members included test managers, test team lead, testers, QA, programmers and agile team members, and TestStrategy.org. Their daily work was related primary or indirectly to software testing.

3.5.1 Research Data

This section lays down the research data collection methods and presents a short discussion of the research. In this process, two types of data: a) primary research data and b) secondary research data were analysed. The primary research data were collected directly from the first-hand experience by means of a set of questionnaires used to collect user feedback on the adoption of the visualized test strategy. These data were not changed or altered.

The secondary data were collected from [164] and [165] World Quality Reports published in 2018-2019 and from the portal www.teststrategy.org, where the number of visualized test strategies were embedded according to the types of testing and any user could use them having accessed the web portal. The users were also able to participate in the set of questionnaires and provide feedback.

Finally, the collected data were analysed using the methods of quantitative research, using surveys, questionnaires, and experiment.

3.5.2 Primary Research Data

In order to collect the primary research data, a number of questions and answers were analysed using the method of survey. Test strategy visualisation is a new strategy; in this context, the author briefly described what the aim of the survey was to help respondents get a clear idea about the survey. In sub-chapter 3.4.1, Table 3.5 lists the questions and answers used in the survey. Answers to the questions were collected online surveying software testing groups from February

2018 to September 2019. Google survey forms were used for collecting data. The survey comprised a number of questions and answers. The user could select the appropriate answer based on his/her practice.

Questions were generated based on discussions with the field experts and a group of testers to make sure appropriate questions were raised to get sample data to use in the research. Testers, test team leads, test managers, managers and other users who know about software testing domain participated in the online survey.

In the process of data collection, the answers to the questions presented in Table 3.6 were collected directly from the web and early users of the visualized test strategy (Figure 3.12).

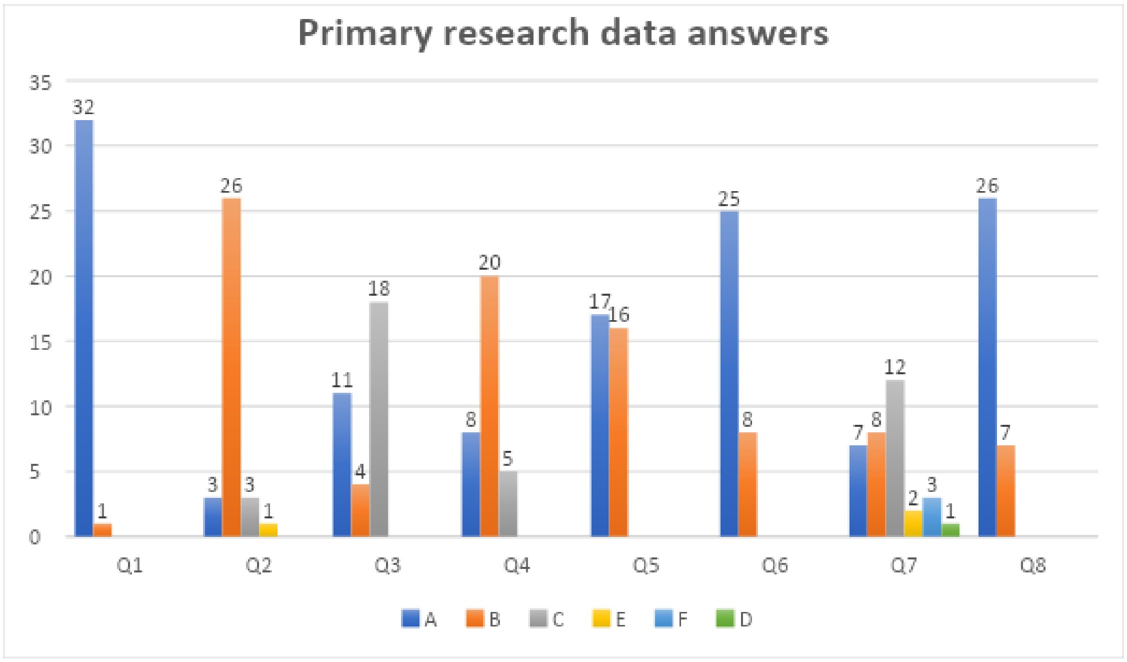


Figure 3.12: Answer and question samples (Table 3.5).

The research data also show a strong need for test strategy in the software development process recognizing its importance.

3.5.3 Secondary Research Data

These data were collected from the World Quality Report [164] and [165] conducted every year by Capgemini, Microfocus, and Sogeti. This research involved a extensive set of user groups

- 1,700 companies, 10 sectors, 32 countries, in a sense 53% of an average country and these data are relevant to 1 million companies. These data sampled 2/10th of 1%; different types of questions were asked to the people who work with software quality around the world. A recent report demonstrates the need for research on test strategy and test design skills (Figure 3.13).

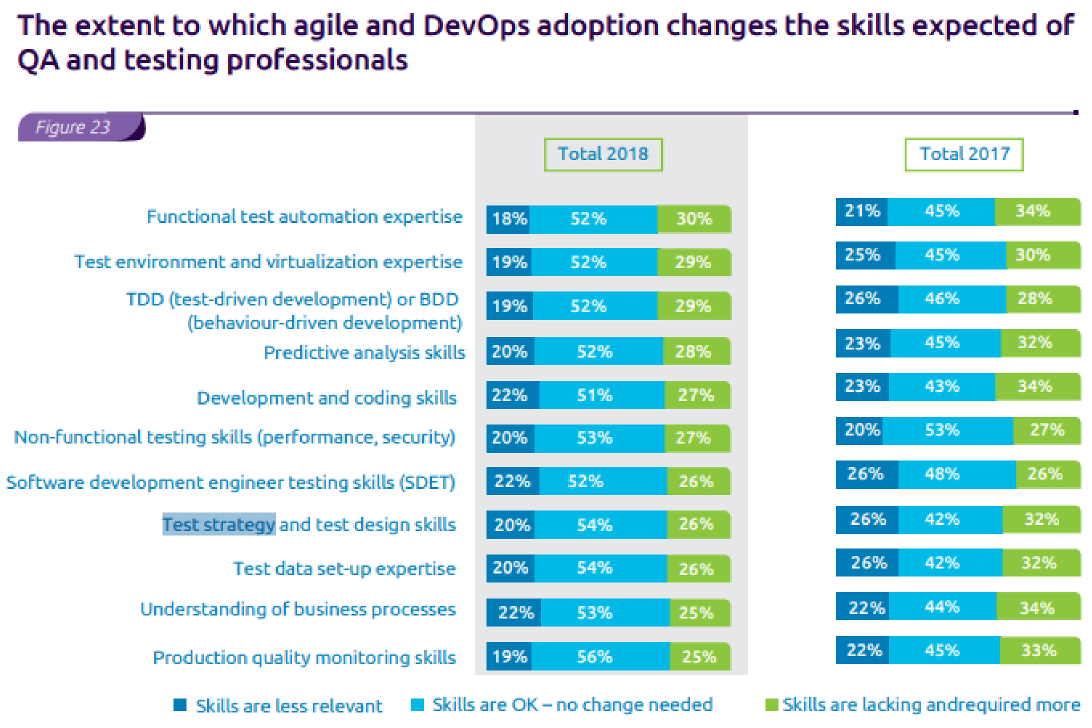


Figure 3.13: DevOps adoption and the required skills.

Recently, Artificial Intelligence (AI) and Machine Learning (ML) started to be adopted slowly by numerous organisations. In this regard, the tester needs to start developing quality assurance and testing strategy for AI and ML solutions. The growing Internet of Things (IoT) needs additional capable testing and better test strategies. At the moment, teams do not have any specific test strategies. According to [138], adoption of DevOps is growing and changes in the skill sets are required in the area of quality assurance and testing. An extensive scope of test automation created new opportunities for focusing on development of a proper test strategy for the project.

The number of survey questionnaires was created by the experts in quality assurance and testing in Capgemini, Microfocus, and Sogeti. 41 questions covered the issue of software testing. Qualitative data were obtained from in-depth interviews, although a supplemental in-depth quantitative study is needed to process these survey questions.

3.6 Adoption of a New Strategy

According to [172], successful adoption of a new strategy depends on the purpose, design/methodology, findings, practical implications, originality/value, and key values. In this research, the author focuses on all of them. The author suggests using the proposed steps as guidelines for starting new or running the existing software development or test projects. A user can adopt sample TABB ready-to-use templates proposed by the author available at the portal teststrategy.org (Figure 3.14).

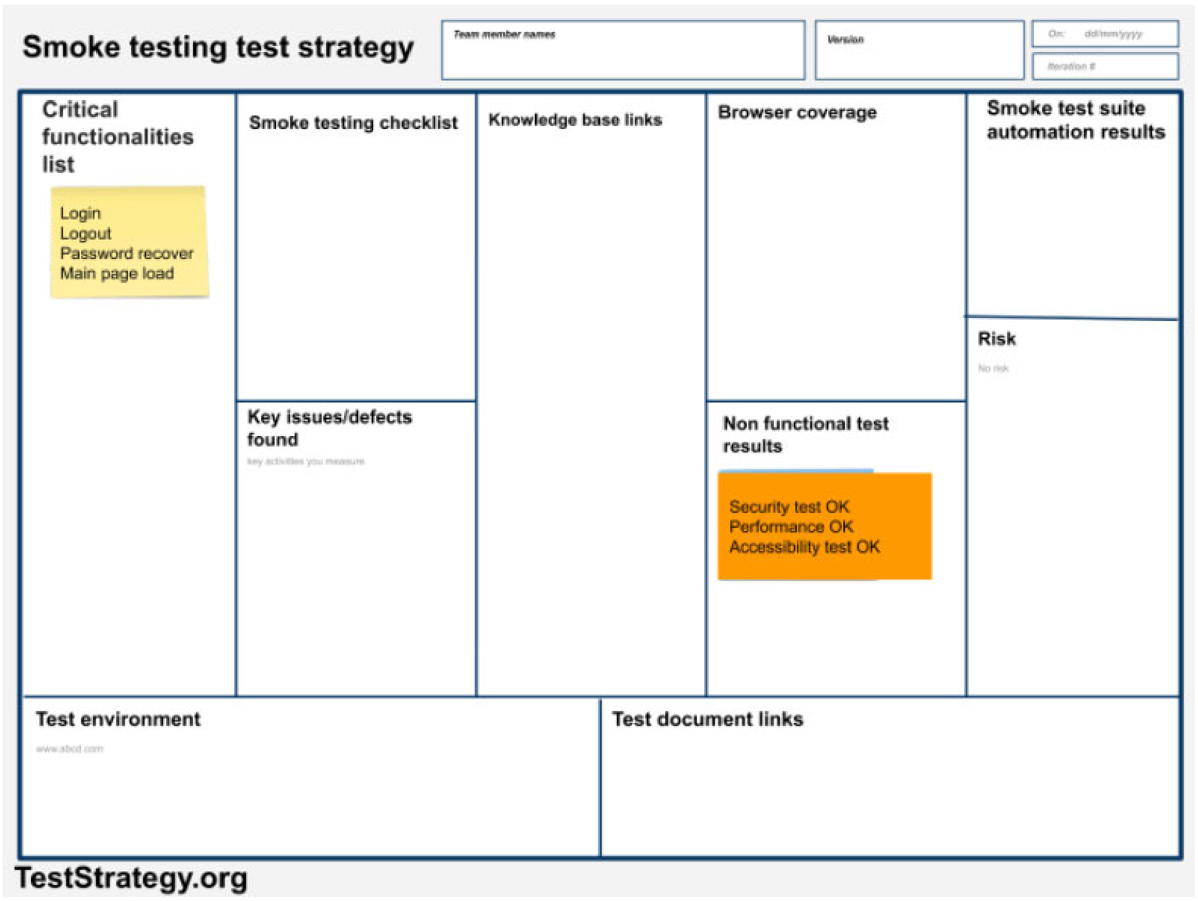


Figure 3.14: Sample template from teststrategy.org.

Step 1: Agree or propose your team to visualise the test strategy using the lean one-page template;

Step 2: Investigate the existing test strategy document or test process document to get an overview, if the team does not have any, then ask questions or brainstorm with the team members for clarification;

Step 3: Use the ready-to-use template from teststrategy.org or GitHub resource;

Step 4: Rename the block and subblock meta title or meta description according to your project and the type of testing;

Step 5: Share "Test Activity Building Block Strategy Using Lean Canvas - Visualised Test Strategy" with the team to update it when the project lifecycle changes.

3.7 Strengths and Limitations of the New Strategy

In this research, the author wishes to describe the new proposed test activities building block strategy profoundly to further finetune it. Thus, Chapter 3 concludes with the analysis of the following strengths and limitations. Table 3.7 presents the strengths and limitations of the visualized software test strategy.

Table 3.7

Strengths and Limitations of the Visualized Software Test Strategy

Strengths:	**Limitations:**
An apparent strength is recognizing the centrality of value. The purpose of constructing a visualized test strategy model is to capture and deliver value to the software test process.	A clear consensus was identified in the absence of external factors, such as software test process maturity, regulatory compliance for special software projects. The need for a skilled team, team and work relationship and detailed estimation of test effort in the visualized test strategy.
During the online user survey and processing respondent comments, the visual representation, usefulness of the template in the test process, usability and simplicity of the test strategy have been mentioned as strengths.	Individual building blocks were investigated, and the repeating limitation was identified. All interviewees mentioned the confusion about the building blocks, which are not sufficient to add a larger set of information

Strengths:	Limitations:
The strength of the visualised test strategy is that it functions as a communication tool between the teams, test managers, individual testers, remote team members and partners. Because of the visualised format, the structure and simplicity of the test strategy contribute to better communication between different disciplines.	The last limitation is focused on the interaction of teams and the value of the creator of the visualized test strategy. It should be taken into account that in agile testing, all team members should be involved in the visualized test strategy design process.
The visual representation of the test strategy is well focused on internal factors.	Disregard of external factors, such as process maturity, skilled team, team and work relationship and estimation of test effort.
The coverage of different dimensions of a visualized test strategy, such as risks, goals/objectives, testing level specific /test plans, test metrics, key performance indicator	The visualized test strategy is based on building blocks of different levels of abstraction. This results in the emphasis solely on building blocks, such as test case, main functionality list, long text descriptions

3.8 Summary

This chapter has demonstrated the basic test activity building block strategy using lean canvas generation hypothesis. Using user experience design and user interface design, the development principles and the visualised lean canvas board itself have been developed. It uses the TABB (Test activities building block) to find the meta title or meta description names on the lean canvas test strategy board. System design is demonstrated in the pilot project study created at the web portal www.teststrategy.org.

The author used industry feedback to collect the data. In this process, data were collected from various sources. a) Survey based on questionnaire; b) online user review and comments; c) use case implementation were used for the survey and feedback.

The Research data comprise a) the primary research data - questions and answers used in the survey and b) the secondary research data - testing strategy skill gap.

The author has demonstrated how to adopt a new test activity building block strategy step by step in any new or ongoing software development project.

In conclusion, significance and limitations of the strategy have been discussed.

Also, in the sub-section System Design Details, use cases, system design importance, importance of strategy and limitations, and survey feedback have been discussed. Eventually, the proposed analysis and research data have been evaluated.

4. APPROBATION AND TEST ACTIVITY BUILDING BLOCK STRATEGY

This chapter is dedicated to discussion of the experimental evidence; it provides a case study of applying the test activity building block strategy on the lean canvas for testing strategy in different software development projects. Different case studies were done in two different companies with short term and long-term projects. The scope of these projects differs in terms of project management, people and technology. In this section, the author briefly highlights the testing process.

In this research, the author has worked closely with two companies - a medium and a large size company. The author has found a platform to implement TABB withing the ongoing software development projects. Company Eptron SIA is a medium-size company which deals with the development of software products for the end customer. At the company, a number of software development projects were developed using different tools and different development methodologies. The author proposed an idea for testing strategy design. In this way, the author collected results and valuable feedback, which were documented in the sub-chapter. The author also got the opportunity to work in Visma Labs. It is one of Europe's leading cloud software development product-based companies. Products development life cycles are in agile only. The company itself is very big, the author found a legacy project among new product development projects where the author proposed an idea, which was accepted, and received feedback.

The author provides an overview of the test process implemented in Latvia based two software companies in the appendices.

4.1 Domain

The test activity building block strategy on the lean canvas for test strategy has been evaluated in two Latvia based companies adopted the agile development process. The author provided an overview of the approached vs adopted test activity building block practical implemented project (Figure 4.1). In this process 22 projects are evaluated and, in the thesis documented 6 projects outcome. Each company used different technologies, tools, testing process, and unique business domain software development projects. Considering all these factors, project evaluation outcome looks as presented below.

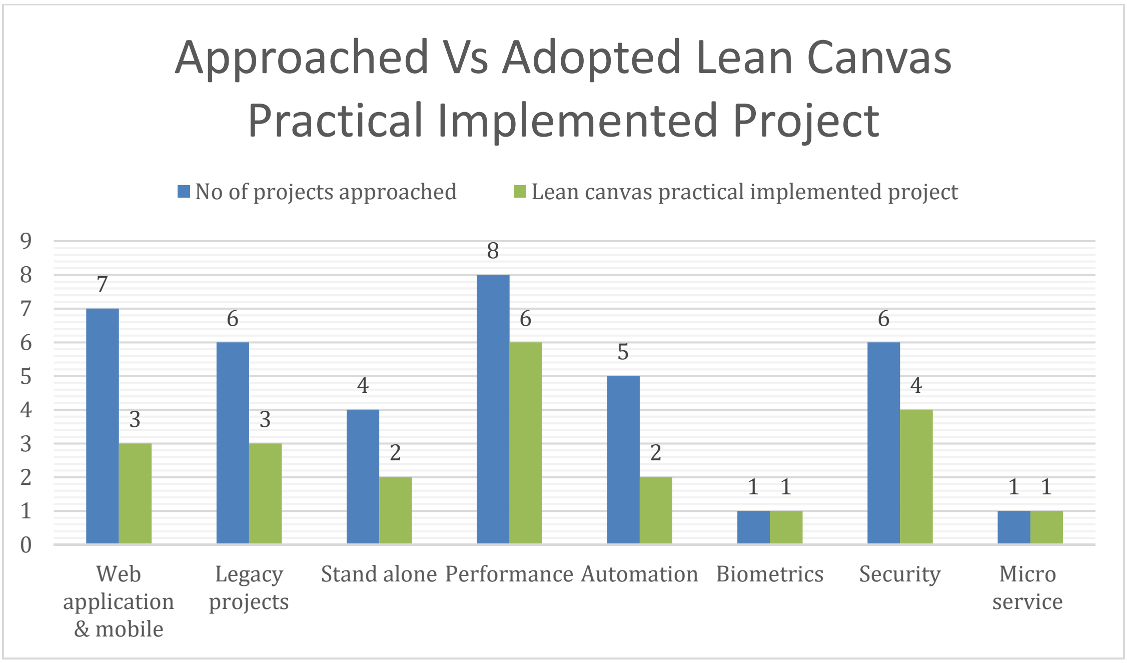


Figure 4.1: Approached vs adopted test activity building block practical implemented project.

The author made a proposal to adopt TABB in the test process in front of the software development teams. In the next step, the team discussed and decided to adopt TABB or not. It also depended on the time, budget, resources and priorities. In such a situation, the approached vs adopted lean canvas practical implemented project should have been considered. A graph in Figure 4.2 showcases the number of the team members in the project connected to Figure 4.1.

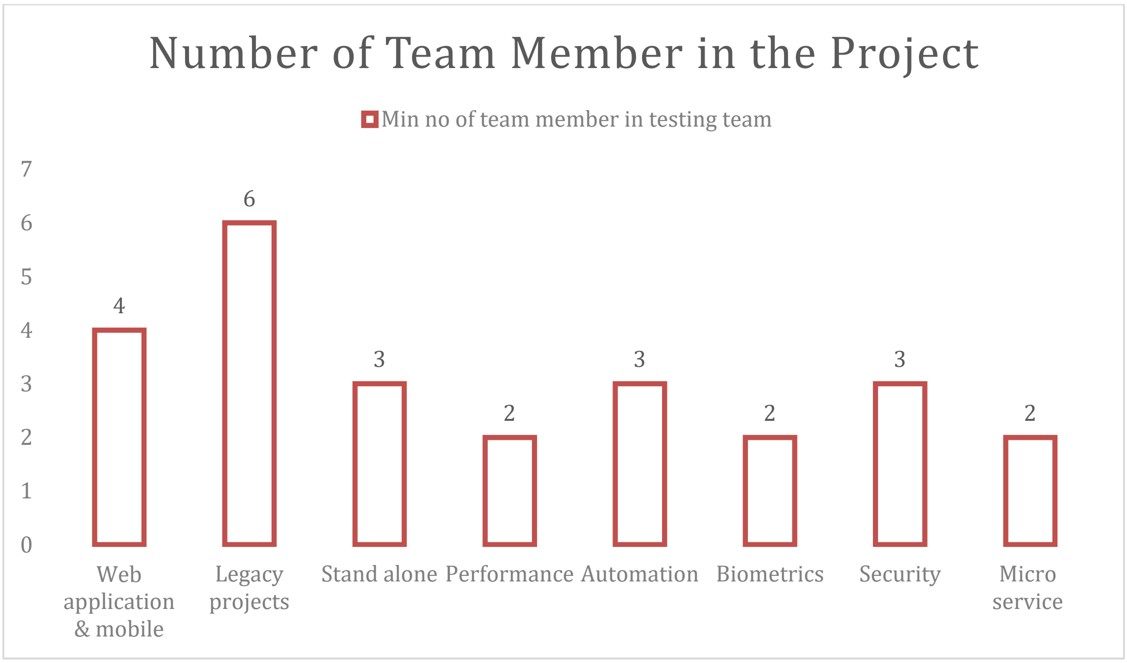


Figure 4.2: Number of team members in the project.

Each team and project are unique in terms of the business domain, technology, and tools (Figure 4.3).

Test activity building block strategy

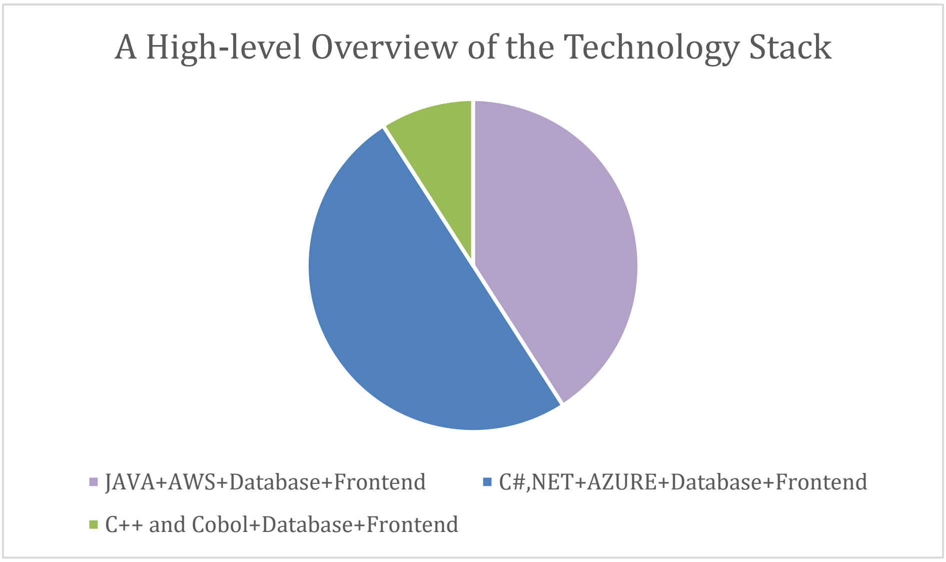


Figure 4.3: Project technology encountered in the adopted test activity building block strategy on the lean canvas within practically implemented project.

In the first software development company, many projects migrated from waterfall to agile and experiments were done within the agile-based project only. The agile team adopted scrum methodology and sprint duration was two weeks. The team was partially following the scrum guidelines [137], and it impacted the test process.

In the second software development company, a number of legacy software projects ran under the agile development life cycle. The team worked in 2 to 3 weeks sprints and incrementally new software features were delivered to the end customers. In this process, documenting was not done extensively, due to the short delivery cycle the team needed to make sure they always delivered the quality of the working software to end customers.

4.2 Empirical Test Strategy

In the first software development company, each project contains its scope and requirements, in this situation to define the testing strategy the team adopted **WWWWWH** questionnaire survey. The teams created high-level test strategy documentation, although this approach is unique and not adopted in all projects. The author explained in detail WWWWWH (Empirical Study) questionnaire survey documentation process for testing strategy creation in the appendices.

In the second software development company, each project contains its unique scope for software requirements and testing process. In a common internal test, the team developed its testing strategy. This testing strategy is called **STD (Story, Development, Testing)**. The author explained STD in detail in the appendices.

4.3 Developing the Case Study

In order to develop the case study and test strategy models in the both software development company, it is necessary to have comprehensive analysis of the current situation within the project (Table 4.1). In this context, all high-level data have been collected and grouped under different attributes.

Table 4.1

High-level Data Collected in the Form of Questions and Answers

Questions	**Answers**
Type of SDLC	Waterfall, AGILE (Scrum) AGILE (Kanban)
Test team members	Tester, Senior tester, Test lead, Quality assurance, Analyst
Development team members	Developer, Senior developer, Lead developer, Software architect
Test tools	Test tarantula, Hiptest, Atlassian jira, atlassian page, MS word documents, Excel spreadsheets, Jmeter, Wireshark Burpsuite, DLLInjector Ranorex, Testcomplete, Selenium, Specflow, Cucumber, Mabi, Mindmaps
Types of testing	Functional testing (Regression, Exploratory testing) Non-functional testing (Performance, Security, Accessibility)
Who is responsible for software quality	Test team lead in waterfall projects Team in agile processes
Do they have a test strategy for testing	No – Most of the project do not have Yes – Few projects have high-level WWWWWH test strategy
Do they have test documents	Yes. Long documented and unstructured
Do they use any defect reporting tools	Test tarantula, hiptest, JIRA
What kind of challenges does the test team face	Short regression time Lack in creating test strategy Remotely distributed teams and time zone is different Test case updating Tracing changing requirements and document them Frequently changing project priority and human resources
How big is DEV/QA team	4 -9 team members

Considering the number of criteria, this research work aims to develop the visualised testing strategy using the test activity building block strategy and analysing the results in ongoing projects. The next sub-chapters provide additional empirical information and a case study.

4.4 Scope of the Case Study

This sub-chapter presents a detailed case study where canvas adoption is done for the project. Each case study is unique because of its development scope and requirements. The author documented each instance of case observation, questions and project details and implementation of the visualised test automation strategy.

The scope of the case study covers the testing activity and adoption of building block strategy on lean canvas in both software development company. Although the scope of the project included different business domains, different types of software applications and their development of life cycles, the study focuses on different possible domains to cover different types of testing done in the software development life cycle (Table 4.2).

Table 4.2

Type of Project, Business Domain, Type of Test Strategy Created and Type of Development Methodology

No	Type of project	Business domain	Type of test strategy created	Type of development	Documentation size (up to) in pages (WWWWWH test strategy)	Documentation effort (up to) in each software release in hours
1	Microservice	Procurement	Test automation	Agile - Kanban	14	4
2	Web application	Entertainment	Performance testing	Agile - Scrum	12	4
3	Web application	Entertainment	Security testing	Agile - Scrum	7	4
4	Web application	Ecommerce	E2E AI testing	Agile - Scrum	9	4
5	Web application	TAX declaration	Legacy testing	Waterfall	22	8
6	Web application	Ecommerce	DevOps testing	DevOps testing	14	4

From above table titles

- Documentation size (up to) in pages (WWWWWH test strategy) - page size go higher and lower depend on description written inside the document.
- Documentation effort (up to) in each software release in hours - documentation effort in hours go higher and lower depend on time allocated in the project planning session.

In the next sub-section, the author describes test automation, performance testing, security testing, end-to-end testing with an artificial intelligence tool, legacy software application testing and DevOps types of testing. In the experimental part, the visualised test strategy is developed and explained.

4.4.1 Test Automation

This sub-section highlights adoption of the TABB strategy in one of the test automated project. This project contains the microservice architecture, following the agile Kanban model. In this project, the test activity building block strategy on lean canvas is created for test automation.

In sub-chapter 3.2, the author describes three-step design test activity building block strategy on lean canvas test activities building block. These steps include:

1. Identify the primary key meta title for the board;

2. Split key meta title into smaller ones;

3. Modify a meta title or meta description.

Asking different sets of questions to the team members, collecting feedback and analysing the presented documents allowed simplifying the created test automation strategy.

Considering the earlier investigation on a project that adopted test automation, several questions were asked to the team members in a group. The answers were recorded for further test strategy development (Table 4.3).

Table 4.3

Test Automation Observations, Questions and Project Details

Project details	Project name: P-roce Microservice Team size: 5 Type of SDLC: Agile Type of testing: Unit, Integration, UI test automation Team members: Analyst, Developer, QA, Architect		
Questions	Q1: What kind of testing is done? **Functional testing, Unit, Integration, UI, UAT**	Q2: What kind of challenges does the test team face? **Continuous deliver, with test automation**	Q3: Who is responsible for software quality? **Team**
	Q4: What kind of documentation does your team draw up? **Test document in Atlassian Jira confluence**	Q5: Does the project have a test strategy in the document or visualized form? **No**	Q6: The team would like to have a visualized one-page test strategy. **Yes**
Observations	P-roce is procurement software, it is basically a legacy code. The team is developing a small component as a microservice. Project technologies: JAVA, AngularJs, MariaDB, AWS cloud, Bitbucket, AppDynamics,TeamCity Test management: Maximum test automation and minimum manual testing Test strategy: Release checklist and continuous delivery		

Considering all the above inputs, analyzing the existing document and adopting the TABB for the lean canvas blocks and sub-blocks, the visualized test strategy is shown in Figure 4.5.

As the author demonstrated in sub-chapter 3.2, test activities follow the visualized lean canvas test strategy steps.

Step 1: Identify the current type of the software development process.

Agile

Step 2: Identify the type of software testing.

Unit, Integration, UI test automation

Step 3: Use any existing testing strategy or test documents.

Test document in Atlassian Jira confluence. Analysing the documents, one gets to know additional about the existing test management information.

Step 4: Identify the primary key meta title for the board and use empty canvas with border and start adding sub-blocks.

Test case, Risks, Resources, External links, Testing scope, KPI, test deliveries, document links, goal, objectives and different environment.

Step 5: Split key meta title into smaller ones and give the appropriate key meta title names to blocks and sub-blocks.

Testing scope – Automation, Manual and out of scope

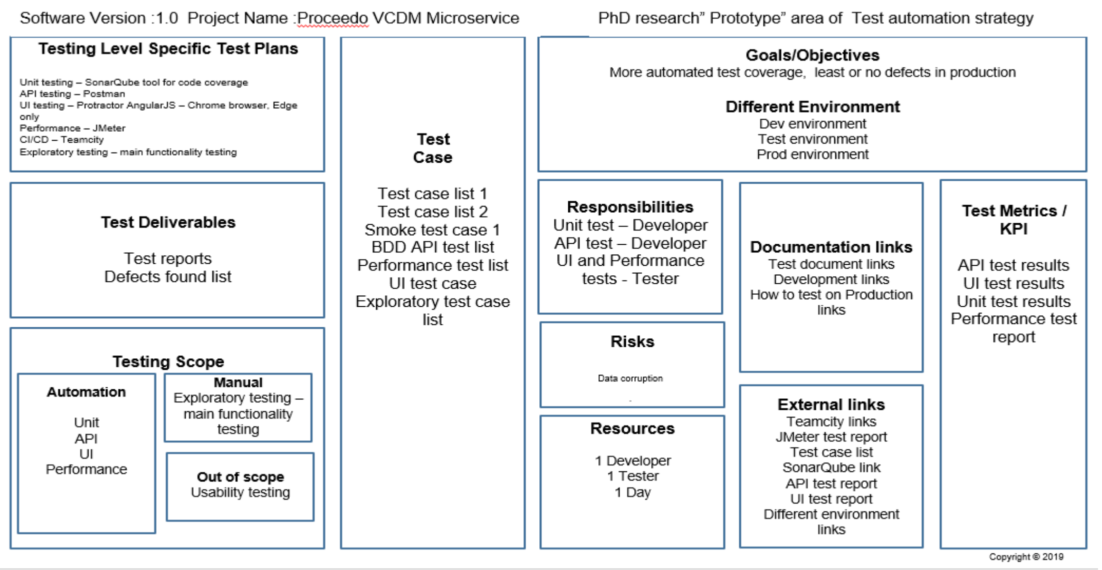
Step 6: Start filling the short meta description in sub-blocks (e.g.: Testing scope in sub-blocks as a Scope of testing, Out of scope)

Step 7: Visualized test strategy is ready to use.

The user can create a new one adding all collected information in one test activity building block strategy template, or the user can apply the existing ready-to-use template available at www.teststrategy.org

Step 8: In case it is needed after each sprint or after development phase, re-review the existing test strategy and modify a meta title or meta description.

Software version 1.0



Software Version :1.0 Project Name :Proceedo VCDM Microservice

PhD research" Prototype" area of Test automation strategy

Testing Level Specific Test Plans

Unit testing – SonarQube tool for code coverage
API testing – Postman
UI testing – Protractor AngularJS – Chrome browser, Edge only
Performance – JMeter
CI/CD – Teamcity
Exploratory testing – main functionality testing

Test Deliverables

Test reports
Defects found list

Testing Scope

Automation
Unit
API
UI
Performance

Manual
Exploratory testing – main functionality testing

Out of scope
Usability testing

Test Case

Test case list 1
Test case list 2
Smoke test case 1
BDD API test list
Performance test list
UI test case
Exploratory test case list

Goals/Objectives
More automated test coverage, least or no defects in production

Different Environment
Dev environment
Test environment
Prod environment

Responsibilities
Unit test – Developer
API test – Developer
UI and Performance tests - Tester

Risks
Data corruption

Resources
1 Developer
1 Tester
1 Day

Documentation links
Test document links
Development links
How to test on Production links

External links
Teamcity links
JMeter test report
Test case list
SonarQube link
API test report
UI test report
Different environment links

Test Metrics / KPI
API test results
UI test results
Unit test results
Performance test report

Copyright © 2019

Figure 4.5: Visualised test automation strategy.

Visualised test automation test activity building block strategy solves many issues.

- Different test level is identified for test automation;
- Test deliverable in each stage is identified;
- The testing scope is recognised without information on the testing scope. Recognition of the scope of testing facilitate the team to scope in-depth for UI, API, and Unit test;
- Test case is mapped on lean and visualised lean design; additional text cases are hyperlinked out of the project;
- Goal and objectives are written at the top of the block, so the team focuses on the goal;
- Team responsibilities are outlined, giving clear guidelines to the team what can and what cannot be done;
- All possible risks are listed in terms of test automation;
- Large documents and project-related important documents are linked via hyperlinks;
- KPIs and key metrics for testing are mapped to analyze the progress;
- A related internal team can use test automation test activity building block strategy;
- A non-technical team member can also use the test activity building block strategy for communication or gathering the results;
- In this project, lightweight documentation and visualised strategy fit in the project due to microservice architecture and this project is part of another bigger project.

4.4.2 Performance Testing

This sub-section discusses adoption of the TABB strategy in one of the projects where performance tests are carried out. This project is a web-based application following the agile Scrum model. In this project, the visualised test activity building block strategy is formulated for performance testing.

Considering the earlier investigation of the project with adopted performance testing, several questions were asked to team members in a group. The answers were recorded for further test strategy development (Table 4.4).

Table 4.4

Performance Testing Observations, Questions and Project Details

Project details	Project's name: Every-Hair Team size: 10 Type of SDLC: Agile Type of testing: Manual testing, UAT and performance testing Team members: Analyst, Developer, QA, Architect		
Questions	Q1: What kind of testing is done? **Function testing, UAT**	Q2: What kind of challenges does the test team face? **Web application performance** ● Capacity Testing ● Load Testing ● Spike Testing ● Stress Testing	Q3: Who is responsible for software quality? **Testers and partially team**
	Q4: What kind of documentation does your team draw up? **Test document**	Q5: Does the project have a test strategy in the document or visualized form? **No**	Q6: The team would like to have a visualized one-page test strategy. **Yes**
Observations	Every-Hair is user engagement web solution for salon owners and salon visitors. Project technologies: Azure, C#, Progressive web, AngularJs, SQL Test management: High level test document, Manual testing Test strategy: Test document and release checklist		

Considering all the above inputs, analysing the existing document and adopting the TABB for the test activity blocks and sub-blocks allow creating the visualized performance testing strategy (Figure 4.6). To create the visualized test activity building block strategy for the user, it is necessary to follow the steps the author described in sub-chapter 3.2: Three steps to design lean canvas test activities building block and visualized lean canvas testing strategy design.

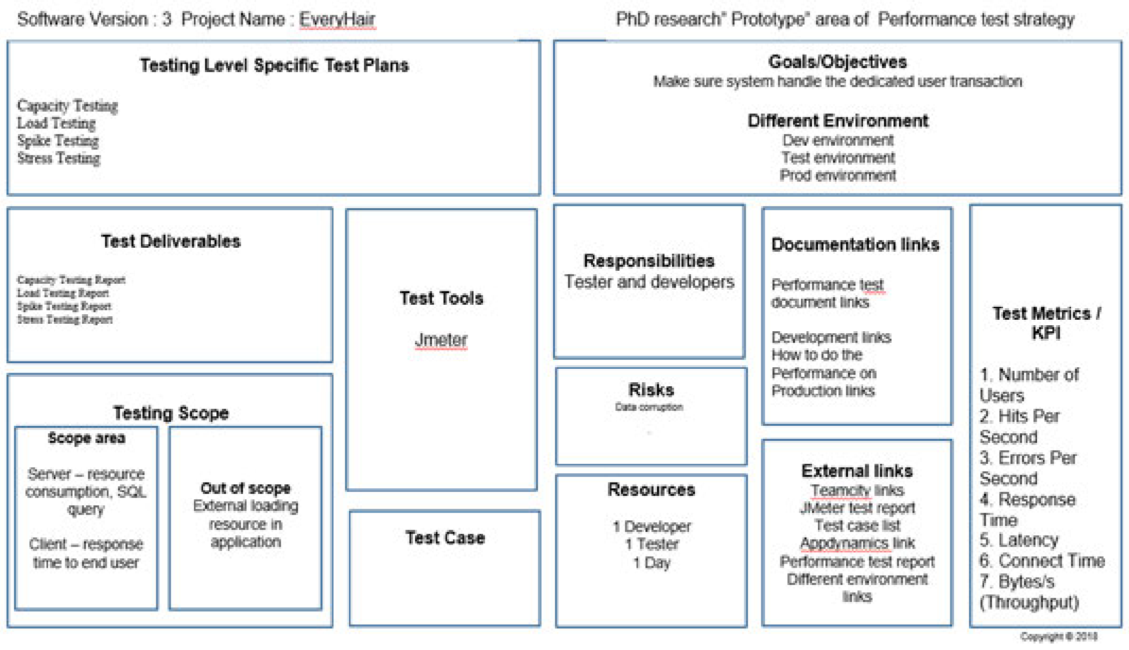


Figure 4.6: Visualised performance testing strategy.

A visualised test activity building block strategy for performance testing solves such issues as:

- Different test levels and types of performance are identified;
- Test deliverable reports are identified;
- The testing scope is recognised very clearly and is identified and out of scope issues are addressed;
- Appropriate performance test tools are identified for the project;
- A test case is recognised and organised according to the priorities;
- The goal of test automation, different testing environment and human resources needed for performance of the task are identified;
- Relevant documentation links and external links are mapped;
- KPIs are used to monitor the progress and consequences of the testing;
- Related internal different teams can also use performance testing test activity building block strategy;
- A non-technical team member can also use the testing activity building block strategy for communication or gathering the results;
- Lightweight documentation strategy fits in this project.

4.4.3 Security Testing

This sub-section highlights adoption of the TABB strategy in one of the projects where security tests are carried out. This project is a web-based application following the agile Scrum model. In this project, the visualised test activity building block strategy is formulated for the web-application security testing.

Considering the earlier investigation on a project that adopted security testing, several questions were asked to the team members in a group. The answers were recorded for further test strategy development (Table 4.5).

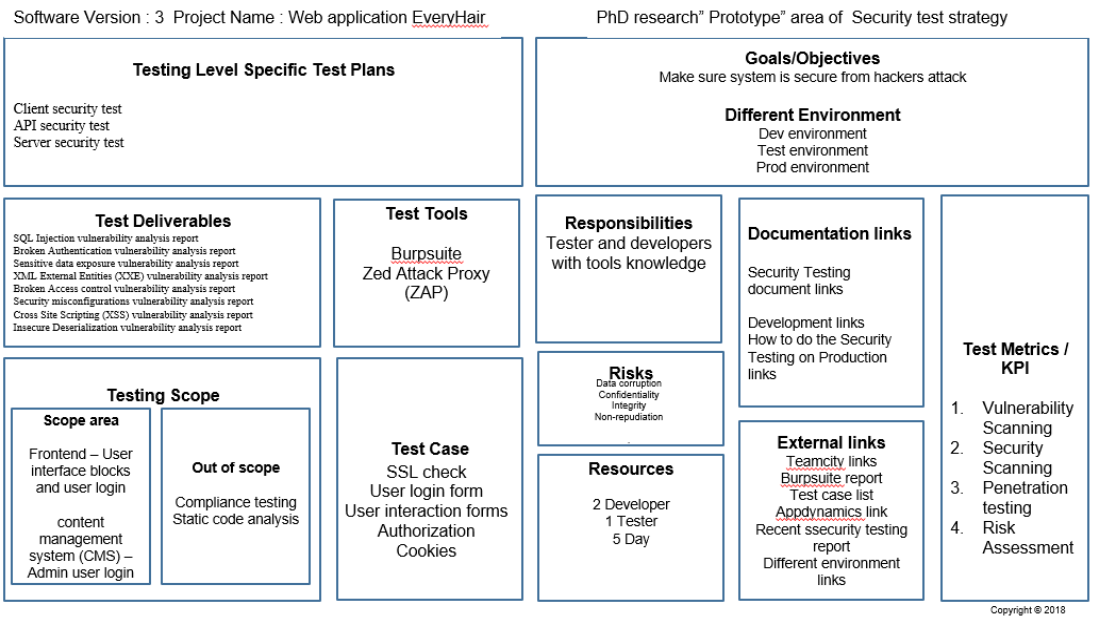
Table 4.5

Security Testing Observations, Questions and Project Details

Project details	Project name: Every-Hair Team size: 10 Type of SDLC: Agile - Lean Type of testing: Security Testing Team members: Analyst, Developer, QA, Architect		
Questions	Q1: What kind of testing is done? **Function testing, UAT**	Q2: What kind of challenges for web application security does the test team face? **Injection** **Broken authentication** **Sensitive data exposure** **XML External Entities (XXE)** **Broken Access control** **Security misconfigurations** **Cross Site Scripting (XSS)** **Insecure Deserialization**	Q3: Who is responsible for software quality? **Testers and partially development team**
	Q4: What kind of documentation does your team draw up? **Test document** **Security test report**	Q5: Does the project have a test strategy in the document or visualized form? **No**	Q6: The team would like to have a visualized one-page test strategy. **Yes**

Observations	Every-Hair is a user engagement solution for salon users. Project technologies: Azure, C#, Progressive web, AngularJs, SQL, Git Test management: High level test document, Manual testing Test strategy: Test document and release checklist

Considering all the above inputs, analysing the existing document and adopting the TABB allow creating the visualised security test strategy (Figure 4.7). The visualized test activity building block strategy is created for the user, it should be followed in the way the author described in sub-chapter 3.2 Three steps to design lean canvas testing activities building block and visualized lean canvas testing strategy design.



Software Version : 3 Project Name : Web application EveryHair
PhD research" Prototype" area of Security test strategy
Testing Level Specific Test Plans
Client security test
API security test
Server security test
Goals/Objectives
Make sure system is secure from hackers attack
Different Environment
Dev environment
Test environment
Prod environment
Test Deliverables
SQL Injection vulnerability analysis report
Broken Authentication vulnerability analysis report
Sensitive data exposure vulnerability analysis report
XML External Entities (XXE) vulnerability analysis report
Broken Access control vulnerability analysis report
Security misconfigurations vulnerability analysis report
Cross Site Scripting (XSS) vulnerability analysis report
Insecure Deserialization vulnerability analysis report
Test Tools
Burpsuite
Zed Attack Proxy (ZAP)
Responsibilities
Tester and developers with tools knowledge
Documentation links
Security Testing document links
Development links
How to do the Security Testing on Production links
Test Metrics / KPI
1. Vulnerability Scanning
2. Security Scanning
3. Penetration testing
4. Risk Assessment
Risks
Data corruption
Confidentiality
Integrity
Non-repudiation
Testing Scope
Scope area
Frontend – User interface blocks and user login
content management system (CMS) – Admin user login
Out of scope
Compliance testing
Static code analysis
Test Case
SSL check
User login form
User interaction forms
Authorization
Cookies
Resources
2 Developer
1 Tester
5 Day
External links
Teamcity links
Burpsuite report
Test case list
Appdynamics link
Recent ssecurity testing report
Different environment links
Copyright © 2018

Figure 4.7: Visualised security test strategy.

A visualised test activity building block strategy for security testing solves many issues:

- Appropriate titles for the sub-blocks make them easy to understand for all internal team members;
- Different types of a security test are identified;
- Test deliverable reports and task are identified;
- The testing scope is recognised very clearly and out of scope issues are addressed;

- Security test tools for testing coverage are identified for the project;
- A test case is recognised and organised according to the priorities;
- The goal of the security test, different testing environment and human resources needed are identified;
- Relevant documentation links and external links are mapped;
- KPI is used to monitor the progress and consequences of testing;
- Lightweight documentation strategy fits in this project;
- Related internal teams can also use security testing test activity building block strategy;
- A non-technical team member can also use the test activity building block strategy for communication or gathering the results.

4.4.4 E2E Testing with Artificial Intelligence Tool

This sub-section discusses adoption of the TABB strategy in one of the projects where E2E tests are carried out using an artificial intelligence tool. This project is a web-based application, following the agile Scrum model. In this project, the visualised test activity building block strategy is formulated for web-application of the end-to-end testing utilising the AI-enabled paid tool.

Considering the earlier investigation on E2E testing within the project adopting AI, several questions were asked to the team members in a group. The answers were recorded for further test strategy development (Table 4.6).

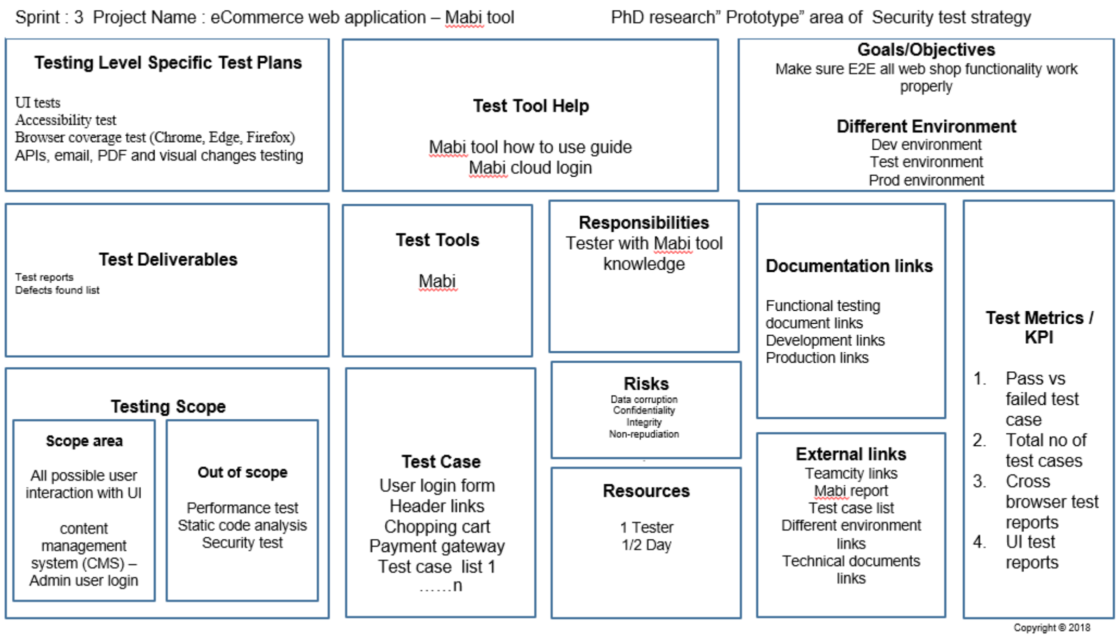
Table 4.6

End-to-end Testing Observations, Questions and Project Details

Project details	Project name: e-commerce website solutions Team size: 7 Type of SDLC: Agile Type of testing: Test automation using mabi tool Team members: Analyst, Developer, QA, Architect, Product owner

Questions	Q1: What kind of testing is done? **UI functional test automation** APIs, email, PDF and visual changes testing	Q2: What kind of challenges does the test team face? **Short regression time** **Localization testing** **Multiple browsers coverage**	Q3: Who is responsible for software quality? **Test team lead for UI test automation**
	Q4: What kind of documentation does your team draw up? **Test documents and test case**	Q5: Does the project have a test strategy in the document or visualized form? **We have a test document in MS word**	Q6: The team would like to have a visualized one-page test strategy. **Yes**
Observations	E-commerce website where more than 1000 product are displayed in 5 languages. A user is able to buy products and pay the amount using a credit card. Once the user purchases, an email confirmation is sent to the user email. The user is able to use coupon codes on the website to get discounts. Project technologies: Java, SQL, AngularJs, Aspx, XML, Azure DevOps, Git Test management: High level test document, UI automation using tool Test strategy: Test main website functionalities and user flows.		

Considering all the above inputs, analyzing the existing documents and adopting the TABB for the test activity blocks and sub-blocks allow creating a visualized end-to-end test strategy (Figure 4.8). To create the visualized test activity building block strategy for the user it is necessary to follow the steps the author described in sub-chapter 3.2 Three steps to design lean canvas test activities building block and visualized lean canvas test strategy design.


Sprint : 3 Project Name : eCommerce web application – Mabi tool

PhD research" Prototype" area of Security test strategy

Testing Level Specific Test Plans
UI tests
Accessibility test
Browser coverage test (Chrome, Edge, Firefox)
APIs, email, PDF and visual changes testing

Test Tool Help
Mabi tool how to use guide
Mabi cloud login

Goals/Objectives
Make sure E2E all web shop functionality work properly

Different Environment
Dev environment
Test environment
Prod environment

Test Deliverables
Test reports
Defects found list

Test Tools
Mabi

Responsibilities
Tester with Mabi tool knowledge

Documentation links
Functional testing document links
Development links
Production links

Test Metrics / KPI
1. Pass vs failed test case
2. Total no of test cases
3. Cross browser test reports
4. UI test reports

Risks
Data corruption
Confidentiality
Integrity
Non-repudiation

Testing Scope

Scope area
All possible user interaction with UI

content management system (CMS) – Admin user login

Out of scope
Performance test
Static code analysis
Security test

Test Case
User login form
Header links
Chopping cart
Payment gateway
Test case list 1
……n

Resources
1 Tester
1/2 Day

External links
Teamcity links
Mabi report
Test case list
Different environment links
Technical documents links

Copyright © 2018

Figure 4.8: Visualized end-to-end testing strategy.

A visualised test activity building block strategy for an end-to-end testing using an AI tool solves many issues:

- Appropriate titles for the sub-blocks make them easy to understand to all internal and external team members;
- Different types of end-to-end test are identified;
- Test deliverable reports and task are identified;
- The testing scope is recognised very clearly and out of scope issues are marked;
- AI test tool for testing coverage is identified for the project;
- A test case is recognised and organised according to the priorities;
- The goal of the end-to-end test, different testing environment and human resources needed are identified;
- Relevant test tool documentation, project links and external links are mapped;
- KPI is used to monitor the progress and outcome of the testing;
- Lightweight visualised documentation strategy fits in this project;
- Related internal teams can also use the visualised test strategy to understand what the test strategy in the project is;

- A non-technical team member can also use the visualised test activity building block strategy for communication or gathering the results.

4.4.5 Legacy Software Application Testing

This sub-section highlights adoption of the TABB strategy in one of the legacy projects. This project is a web-based application following the Waterfall model.

Considering the earlier investigation on a legacy project, several questions were asked to the team members in a group. The answers were recorded for further test activity building block strategy development (Table 4.7).

Table 4.7

Legacy Project Testing Observations, Questions and Project Details

Project details	Project name: TAX dispatch Team size: 10 Type of SDLC: Waterfall Type of testing: Manual testing, UAT Team members: Analyst, Developer, QA, Architect, Product owner		
Questions	Q1: What kind of testing is done? **Manual function testing, UAT, Checklist**	Q2: What kind of challenges does the test team face? **No test automation at any level, maximum dependency on manual testing**	Q3: Who is responsible for software quality? **Testers**
	Q4: What kind of documentation does your team draw up? **Test document**	Q5: Does the project have a test strategy in the document or visualized form? **No**	Q6: The team would like to have a visualized one-page test strategy. **Yes**
Observations	TAX dispatch used by the state and private organization. Project technologies: C#, AngularJs, Aspx, XML, Azure DevOps, Git, Oracel Test management: Manual testing Test strategy: Test document, test case and release checklist		

Considering all the above inputs, analyzing the existing document and adopting the TABB for the test activity blocks and sub-blocks allow creating the visualised legacy project test strategy (Figure 4.9). To create the visualized test activity building block strategy for the user, it is necessary to follow the steps the author described in sub-chapter 3.2 Three steps to design lean canvas test activities building block and visualized lean canvas test strategy design.

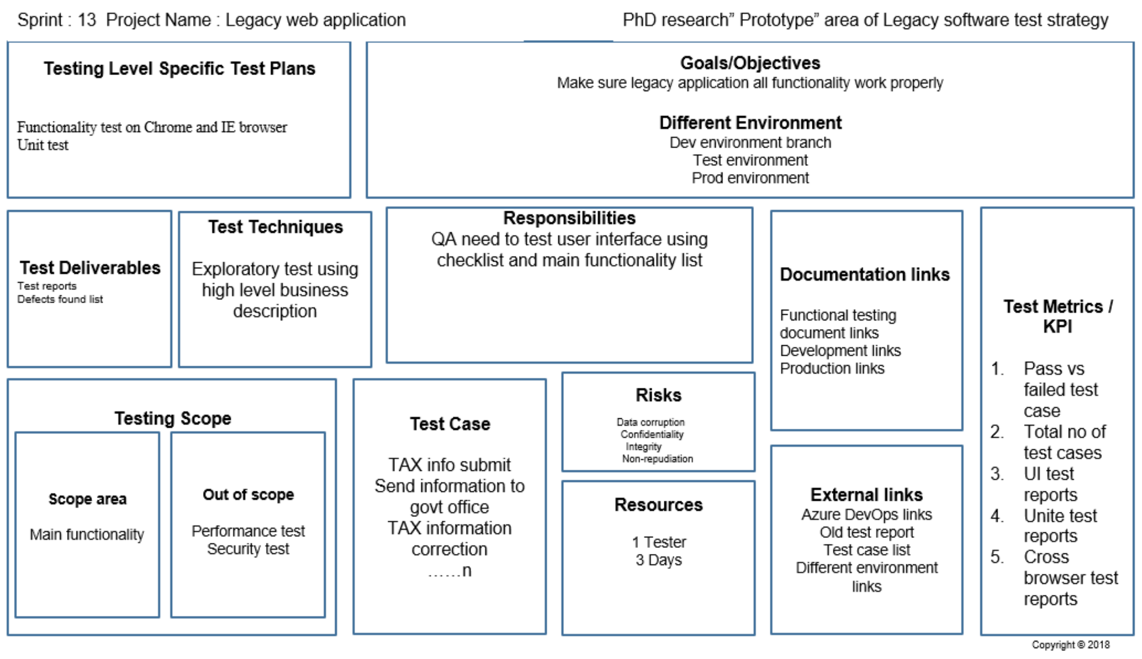

Sprint : 13 Project Name : Legacy web application
PhD research" Prototype" area of Legacy software test strategy
Testing Level Specific Test Plans
Functionality test on Chrome and IE browser
Unit test
Goals/Objectives
Make sure legacy application all functionality work properly
Different Environment
Dev environment branch
Test environment
Prod environment
Test Deliverables
Test reports
Defects found list
Test Techniques
Exploratory test using high level business description
Responsibilities
QA need to test user interface using checklist and main functionality list
Documentation links
Functional testing document links
Development links
Production links
Test Metrics / KPI
1. Pass vs failed test case
2. Total no of test cases
3. UI test reports
4. Unite test reports
5. Cross browser test reports
Testing Scope
Scope area
Main functionality
Out of scope
Performance test
Security test
Test Case
TAX info submit
Send information to govt office
TAX information correction
……n
Risks
Data corruption
Confidentiality
Integrity
Non-repudiation
Resources
1 Tester
3 Days
External links
Azure DevOps links
Old test report
Test case list
Different environment links
Copyright © 2018

Figure 4.9: Visualised legacy project test strategy.

A legacy application visualized test activity building block strategy addresses the many issues:

- Different functional test levels are identified;
- Test deliverable in each stage is identified;
- Legacy application test techniques and appropriate practices are identified;
- The testing scope is recognized and out of scope testing is recognized;
- Team responsibilities are mapped to overcome obstacles;
- Test case is mapped on the visualized lean design and additional text cases are hyperlinked out of the project;

- Goal and objectives are formulated at the top of the block, so the team focuses on the purpose;
- Team responsibilities are outlined to make the team understand what should and should not be done. The boundaries and limitations are set to avoid work duplication and ensure optimization of the existing workflow;
- All possible risks are listed for the legacy application;
- Large documents and project-related important documents are linked via hyperlinks;
- KPIs and key metrics for testing are outlined to monitor progress;
- A non-technical team member can also use the test activity building block strategy for communication or gathering the results.

4.4.6 DevOps Testing

This sub-section highlights adoption of the TABB strategy in one of the DevOps testing projects. This project is a web-based e-commerce application following the DevOps model. In this project, the visualised test activity building block strategy is formulated for web-application.

Considering the earlier investigation on the project adopting DevOps testing approach, several questions were asked to the team members in a group. The answers were recorded for further test strategy development (Table 4.8).

Table 4.8

DevOps Testing Observations, Questions and Project Details

Project details	Project name: AppCenter Team size: 5 Type of SDLC: DevOps Type of testing: Manual testing, UAT, Unit Test Team members: Product owner, Team lead, Cloud architect, Site reliability engineer, DevOps system administrator

Questions	Q1: What kind of testing is done? **Exploratory testing**	Q2: What kind of challenges does the test team face? **Handling the test automation** **Test management**	Q3: Who is responsible for software quality? **Team**
	Q4: What kind of documentation does your team draw up? **Business description**	Q5: Does the project have a test strategy in the document or visualized form? **No**	Q6: The team would like to have a visualized one-page test strategy. **Yes**
Observations	Appscenter solution is developed for interaction with screens. This solution can also be used in entertainment and gaming. Project technologies: Java, JavaScript, Terraform, Jenkins, Spinnaker, Git, Docker, AWS, BDD Test management: Test automation, Exploratory testing Test strategy: Extreme test automation, continuous integration and continuous delivery. Same test automation needs to work on all environments		

In order to create the visualized test activity building block strategy for the user, it is necessary to follow the steps the author described in sub-chapter 3.2 Three steps to design lean canvas test activities building block and visualized lean canvas test strategy design. This visualised test strategy is the beginning stage of the project, as author demonstrated in the Visualized DevOps Project Test Strategy - Sprint 1 (Figure 4.10). In this phase, the test strategy has smaller amount of information, and over time it grows.

In this case, the DevOps project has just started. A creator of the test strategy should consider the following steps:

- Establish clear communication paths;
- Define best practices and rules;
- Create a comprehensive definition of ‘done’;
- Select a suitable continuous integration system;
- Select tools and applications;
- Use version control systems wisely;
- Avoid having multiple test document management systems;

- Define the environments required for the solution;
- Set up development environment, user acceptance testing environment, staging and production environments;
- Prepare the codebase and project structure;
- Create a document for requirement and test management.

Observing Sprint 1, it may noticed that very few meta descriptions and blocks were added.

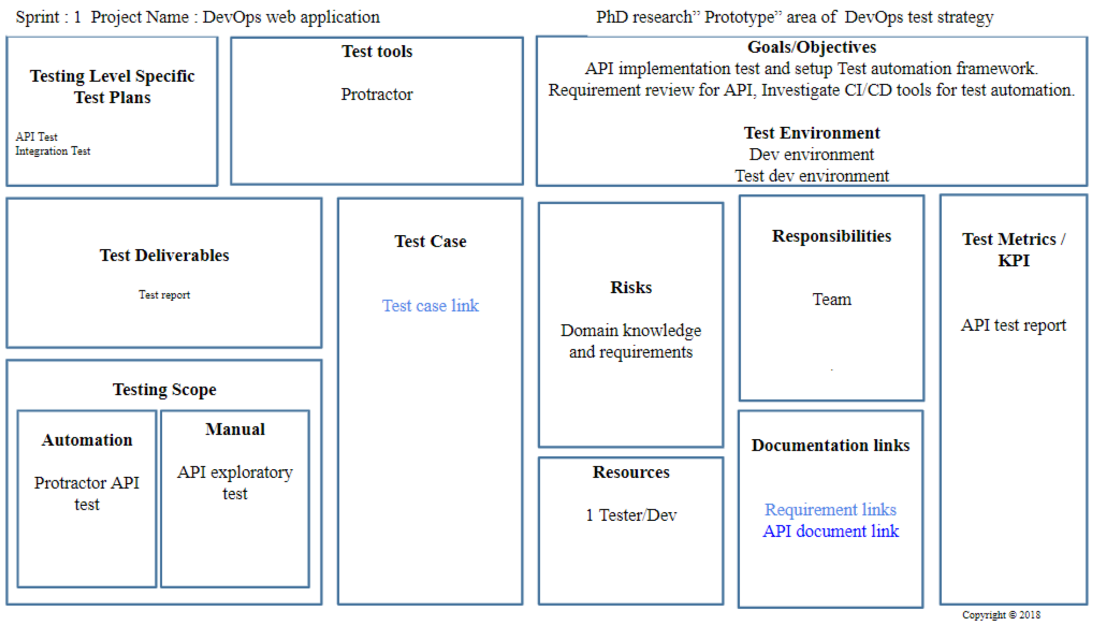


Figure 4.10: Visualised DevOps project test strategy - Sprint 1.

Considering all the above inputs, analyzing the existing document and adopting the TABB for test activity blocks and sub-blocks allow devising a test strategy. In this phase, the project has grown up and new sub-blocks have been added, as well as additional meta description. Observing Sprint 9, it may be noticed that now it has additional meta description, sub-blocks added, as the author demonstrated in the Visualized DevOps Project Test Strategy - Sprint 9 (Figure 4.11).

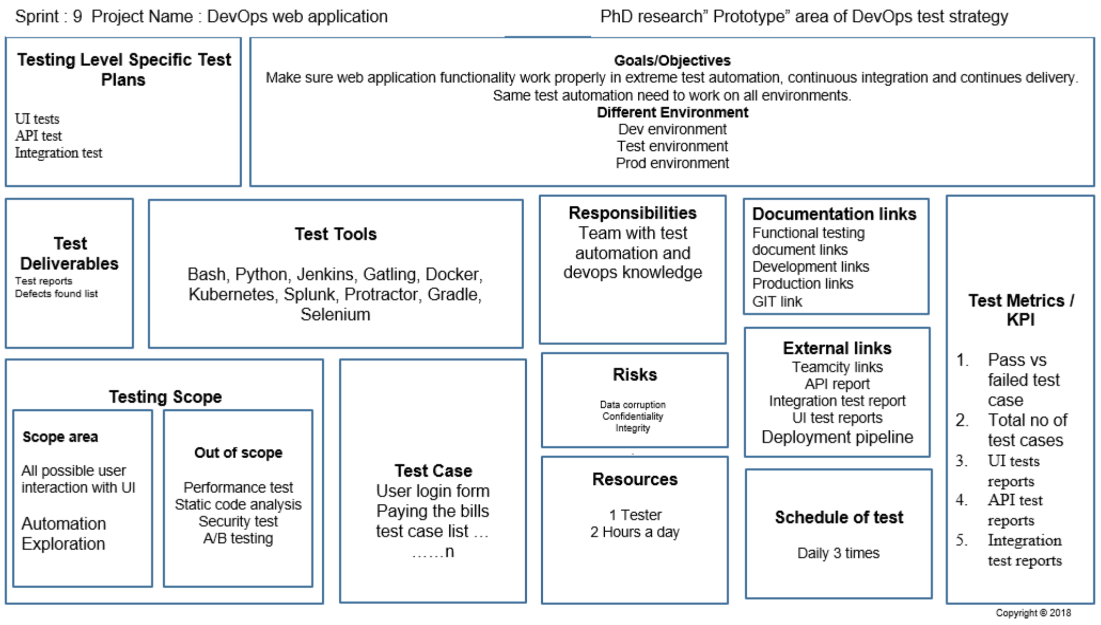


Figure 4.11: Visualised DevOps project test strategy - Sprint 9.

A visualized test activity building block strategy for DevOps strategy web application may help solve many issues:

- Different test levels in API, UI, unit and functional tests are identified;
- Test deliverable in each stage is identified;
- All test and related integration and CI tools are mapped;
- A clear testing scope is recognized and out of scope testing is recognized;
- Team responsibilities are mapped to overcome obstacles;
- Test case are mapped on the visualized lean design and more text cases are hyperlinked out of the project;
- Goal and aims are formulated at the top of the block, so the team focuses on the purpose;
- Team responsibilities are outlined to make the team understand their roles and tasks;
- All possible testing risks are listed for the web application;
- Extensive test and development document and project-related important documents are linked via hyperlinks;
- Different KPIs and critical metrics for testing are outlined to monitor progress;

- A non-technical team member can also use the test activity building block strategy for communication or gathering the results.
- In the observation software defects are reduced due to scope of testing is precisely defined in each sprint.

In this sub-chapter, the author has demonstrated Sprint 1 and Sprint 9 of the visualised DevOps project test strategy. The author would like to emphasise that in case it is needed after each sprint or after development phase the existing test strategy may be re-reviewed and a meta title or meta description may be modified.

4.5 Analysis of Case Study Results

This sub-chapter presents six illustrative case designs to show how to adapt and apply the visualised test activity building block strategy. The cases cover different types of software projects, business domains, technologies and types of software testing. In the initial phase, several questions were asked to the team before starting the test activity building block strategy. In parallel, the existing test and development documents were reviewed to collect the project information. In addition, the visualised test strategy is created and feedback is collected and discussed in this sub-chapter. Table 4.9 presents analysis of the results of the empirical study considering the essential comparison of the type of testing, documentation size [186], communication and visualisation.

Table 4.9

Analysis of Results of Empirical Study

No	**Type of project**	**Documentation size (up to) in pages (WWWWWH test strategy)**	**Documentation size (up to) in pages after adopting TABB**	**Reduced documentation pages in percentage**	**Delta pages**	**Documentation effort (up to) in each software release in hours**	**Documentation size (up to) in pages effort after adopting TABB**
1	Microservice [4.4.1]	14	5	64%	9	4	2

No	**Type of project**	**Documentation size (up to) in pages (WWWWWH test strategy)**	**Documentation size (up to) in pages after adopting TABB**	**Reduced documentation pages in percentage**	**Delta pages**	**Documentation effort (up to) in each software release in hours**	**Documentation size (up to) in pages effort after adopting TABB**
2	Web application [4.4.2]	12	7	42%	5	4	2
3	Web application [4.4.3]	7	3	57%	4	4	2
4	Web application [4.4.4]	9	4	56%	5	4	2
5	Web application [4.4.5]	22	13	41%	9	8	4
6	Web application [4.4.6]	14	6	57%	8	4	2

The adoption of TABB implies the improvement, expressed according to variance in 5.0 pages and 0.6 hours. The reduced documentation pages average 53% and percentage variance 0.8.

From above table titles

- Documentation size (up to) in pages after adopting TABB - page size go higher and lower depend on description written inside the document. From above collected data noticed that minimum pages are reduced comparing to the WWWWWH test strategy pages.
- Documentation effort (up to) in each software release in hours – Documentation effort in hours go higher and lower depend on time allocated in the project planning session. From above collected data noticed that documentation effort minimum hours are reduced comparing to the WWWWWH test strategy.

The author's research into creation of a one-page visualised test strategy (TABB) does not lead to the commonly noticed consequences, such test strategy and test management issues as:

- Quality documentation: Visualisation test strategy dose not impacted on document artifact, coordination, control, delivery, or support of an item required for quality assurance purposes.
- Communication: A one-page test strategy simplifies the inter-team communication and communication within a geographically distributed team.
- Documentation effort: Documentation time and effort shortened.

In the process of case study analysis, a visualised test strategy was created, and feedback was collected in a short format (Table 4.10). The author intended to receive detailed feedback from the users and the team which used it.

Table 4.10

Created Visualised Test Strategy and Feedback

Case study number	**Type of development**	**Type of testing**	**Problem statement**	**Case analysis**	**Received feedback**
4.4.1	Agile - Kanban	Test automation	Continuously changing priorities and software test strategy design	No prescribed roles and continuous delivery. Work pulled in a single piece. Changes to product backlog items made at any time. Responsiveness to change.	Test activity building block strategy is easy to adopt. Test strategy needs to be updated from time to time. Shorter cycle times can help test software and deliver features faster. In Kanban, priorities change very frequently, in a team, QA or Tester need to update the test activity building block strategy from time to time. It covers most of the test risks.
4.4.2	Agile - Scrum	Performance testing	The definition of “done” in Agile project should include completion of “performance testing strategy design.”	Defined roles in the team for Team, Developer, Tester, Scrum Master and Product Owner. Set of timeboxed sprints. In the mid-sprint, nothing changes. In the project product backlog, sprint backlog and product increment are artefacts.	Tester is able to start making test strategy in sprint-planning sessions. The tester can use test activity building block strategy lean canvas design for daily stand-ups. Well-focused test throughout the sprint possible. Test activity building block strategy adoption in the retrospectives.

Case study number	**Type of development**	**Type of testing**	**Problem statement**	**Case analysis**	**Received feedback**
4.4.3	Agile - Lean	Security testing	The aim of lean testing is reduction of waste (rework, defects, corrosion) by holistically creating testing strategy design for the system	Focused on the principles of lean manufacturing and product development methods in software development. Focused on improving the complete development and test process.	Test strategy focused on eliminating waste. Test process is continuously simplified to fit in the scope of testing. Measuring and adopting incrementally for improving software quality. All possible security threats to application are addressed. For longer documentation, hyperlinks work perfectly.
4.4.4	Agile - Scrum	E2E AI testing	Aim to test quickly the flow of software application, which performs as designed from start to finish, using paid tools and artificial intelligence.	Focused on script less cross-browser testing, auto-healing tests to test end-to-end application.	Test strategy focused on end-to-end test of the application. Tester/quality assurance team need tool knowledge. Risk mitigation is very high due to self-healing tests. Test metrics and testing deliverables are clearly achievable.

Case study number	Type of development	Type of testing	Problem statement	Case analysis	Received feedback
4.4.5	Waterfall	Legacy testing	Aim of testing is to make sure software is defect-free when it is delivered to end customers.	Each stage has definite deliverables and a review process. Works properly for additional inadequate projects where requirements are very well understood. Ensured that each unit or segment and developed components work as anticipated.	Test techniques and documentation related to the projects are easy to navigate from the test activity building block strategy lean canvas board. High risk and less flexible issues can be managed from the test activity building block strategy board. Test processes within complex and object-oriented projects simplified utilising the test activity building block strategy board.
4.4.6	DevOps	DevOps testing	Aim is to test continuous software development and continuous delivery.	Focused on Development; QA & Operations work together. Regression test cases are a crucial point in ensuring software quality.	QA and Development communicate using test activity building block strategy lean canvas board. Deployments and regression test automated report could be found on the test strategy page. Schedule of tests, project goal, different test environment and test tools mapped. Supports continuous development and delivery test strategy

Goal analysis is done considering the above table and objectives of the thesis, observations and feedback are collected through this research (Figure 4.12).

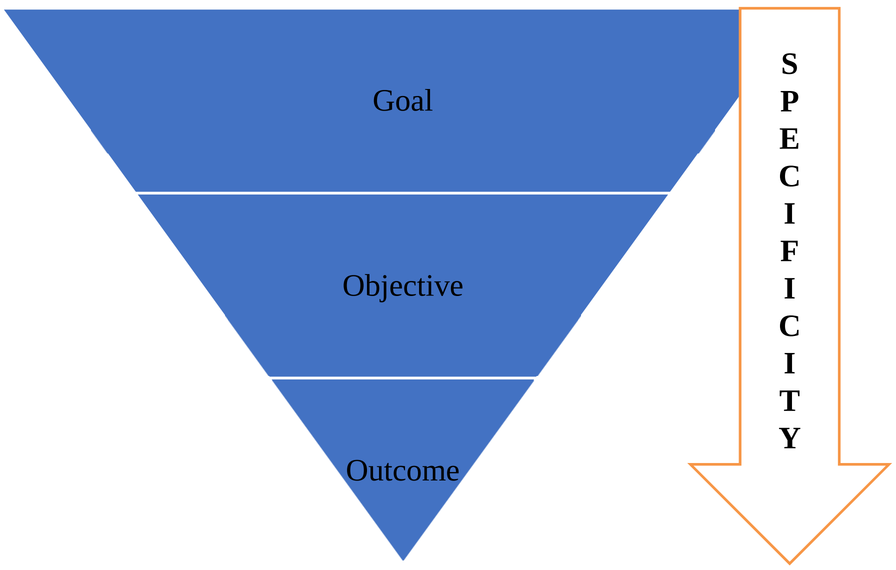


Figure 4.12: Goal analysis Visualised test strategy.

Overview of the goals, objectives and important outcomes is outlined in Table 4.11.

Table 4.11

Overview of the Goals, Objectives and Outcome

Goals	• To explore the existing software test process and test strategy for creating a visualised test strategy and its implementation recognising the main problems and software industry trends; • To identify the main advantages and disadvantages of the most recently used software strategy that meets modern trends; • To develop a strategy, identify different existing strategies and models; • To develop a working strategy based on different real-life projects and software requirements; • To develop criteria to evaluate the proposed strategy; • To implement the visualised software test strategy in the software development projects and develop general criteria to assess advantages and disadvantages of the developed software projects; • Based on the practical results, to investigate and document the benefits of the strategy, implementation risks, and limitations, accepting recommendations from other project and team members; • To define the possible further development scope of the proposed strategy.
Objective	• The object of the research is to investigate how to visualize software test strategy; • Development of the visualising software test strategy in different software development life cycles and types of software testing, collecting feedback and evaluating the results.

Outcome	• Visualising and simplifying software test strategy by reducing the volume of documentation. Provision of improved remote team communication, test team resource handling, work visibility; • Reduction of the lengthy test plan documentation and creation of a test strategy; • In software development projects, distributed remote teams use the visualised software test strategy effectively in both manual and test automation process; • Visualised test strategy adaptable in different development methodologies and different types of software testing; • Visualised test strategy appropriate for agile methodology projects; • Navigation between important external links allows their easy collection and visiting.

4.6 Summary

This case study was performed based on the project implemented in the two Latvia based software development company where software development and testing done for various domains. Both companies work mainly in Europe, some part of business operates in other continents. The TABB visualised test strategy was adopted and analysed in Test Automation, Performance Testing, Security Testing, E2E Testing, Legacy Software Development and DevOps testing.

Table 4.12 showcases the empirical study, literature study and the test strategy proposed by the author. The table demonstrates the scope of support to the key elements of the main test strategy.

- Incomplete support: It is just part of the whole, or partial;
- Support: Support never needs input from anything.

Table 4.12

Comparison of the Proposed Test Strategy with Other Test Strategies

	WWWWWH (Empirical study)	**Visualised test strategy (Proposed test strategy)**
Types of testing	Incomplete support	Support
Scope and objectives	Support	Support

	WWWWWH (Empirical study)	**Visualised test strategy (Proposed test strategy)**
Roles and responsibilities	Support	Support
Communication and status reporting	Support	Support
Test deliverables	Incomplete support	Support
Industry standards to follow	Internal use only	Incomplete support
Testing measurements and metrices	Incomplete support	Support
Test automation and tools	Support	Support
Risks and mitigation	Incomplete support	Support
Easy to adopt in projects	Incomplete support	Support

The author explained in detail documentation of the WWWWWH (Empirical Study) questionnaire survey test process for test strategy creation in the appendices. A list of most common sub-block titles was drawn up within the case study done in two companies (Table 4.13). These primary key meta titles could be used as an input to generate a stable test activity building block strategy design board.

Table 4.13

Common Sub-blocks and Titles Collected from the Board

No	**Title**	**Type of testing**	**Purpose**
1	Goals/Objectives	All types of testing	The testing environment is a setup where testers execute the test case. The test is executed manually or automated.
2	Test Environment	All types of testing	The testing environment is a setup where testers execute the test case. The test is executed manually or automated.
3	Risks	All types of testing	From the testing point of view, technical risks are identified and highlighted. Risk mitigation is the core use of this sub-block.
4	Testing scope Scope area Out of scope	All types of testing	Scope gives testers what to consider and what not to consider for testing.

No	Title	Type of testing	Purpose
			For a few projects, the scope might be big, and writing a few crucial points helps the tester to use them. Long documents are hyperlinked from the same place.
5	Testing level Specific test plans	All types of testing	Software testing levels give a clear view of where the test is executed.
6	External links	All types of testing	These are the hyperlinks that point to extended documentation. It provides a better descriptive reference.
7	Resources	All types of testing	This sub-block gives information about what human resources or time should be allocated for completing the testing.

Although all test case studies were analysed at a high level, one of the pieces of the existing test strategy documentation was also discussed. This chapter has demonstrated wide possibilities offered by adoption of the visualised test strategy in different types of software development projects. Most popular sub-blocks and titles from different case studies were identified. These subblocks and meta titles or meta descriptions were used to create the possibility to redesign the visualised test activity building block strategy on the lean canvas in future. Moreover, the use case for test activity building block strategy models was created and case study analysis was performed.

CONCLUSION AND SCOPE OF FUTURE RESEARCH

The aim of the Doctoral Thesis is to develop a visualised software testing strategy utilizing lean canvas model.

On the way to achieving this aim, the following results have been obtained:

- Test process waste classification has been performed to create better meta titles (Chapter 3);
- In Ontology, mind maps have been analysed for knowledge management system. In the next step, mind map extraction has been used for creation of meta titles (Chapter 3);
- For systems design, use cases have been collected and online web-based application has been developed, online web-based limitations have been considered (Chapter 3);
- To get better insights, both primary and secondary research data have been analysed for TABB (Chapter 3);
- Adoption of TABB steps has been outlined (Chapter 3);
- The existing test strategies WWWWWH and STD (Story, Development, Testing) adopted by the industry have been documented (Chapter 4);
- The overall results of approbation allow stating that TABB visualised test strategy has been successfully adopted and investigated in functional and non-functional projects (Chapter 4);
- The TABB test strategy proposed by the author has been compared with other testing strategy (Chapter 4);
- Analysis of different case studies has been conducted for test automation, performance testing, security testing, E2E testing, legacy software development and DevOps projects, the results have been evaluated and documented (Chapter 4);
- The developed methodology and methods, combining experiments, case studies and expert interviews, have been comprehensively assessed (Chapter 4).

Within the Doctoral Thesis, the following conclusions have been made:

- The adoption of TABB approach reduced documentation pages to average 53% and time to develop documentation approximately 50%;

- The main building blocks of security testing are goals/objectives, test environment, risks, testing scope, testing level-specific test plans, external links and resources;
- TABB adaptable in test automation, performance testing, security testing. E2E AI testing, legacy testing and DevOps testing. It is adaptable in both functional and non-functional testing;
- TABB visualised test strategy is adaptable in Agile (Kanban, Scrum), Waterfall, DevOps and other methodology projects;
- TABB adaptation has been analysed and evaluated, it does not lead to the consequences related to commonly noticed test strategy and test management issues such as communication, visualisation, internal team conflicts, software defects, documentation effort.

The results of the Doctoral Thesis provide an opportunity for conducting future research in the following areas:

- Extension of reusability of TABB, including the evaluation different software development project, such as embedded software, firmware etc;
- Development of tools or plug-ins to integrate TABB in test reporting tools;
- Development of a additional simplified online solution at teststrategy.org and advancement of the visualised test strategy creation using machine learning and artificial intelligence. In this process, the user submits necessary project information and gets back several ready-to-use visualised test strategy models;
- Creating awareness about the "visualised test strategy" model in the software testing community to increase its adaptation. Collecting constant feedback from the community and applying this feedback to better and new model creation.

Bibliography

[1] Spillner, A., Linz, T., Schaefer, H.: Software Testing Foundations. 2014, pp. 23.

[2] Black, R.: Agile Testing Foundations: An ISTQB Foundation Level Agile Tester guide. 2017, pp. 12.

[3] Graham, D., Fewster, M.: Experiences of Test Automation. 2012, pp. 32.

[4] Watters, A., Pinkster, I., Janssen, D.: Integrated Test Design and Automation. 2000, pp. 34-76.

[5] Black, R., Veenendaal, V., Graham, D.: Foundations of Software Testing: ISTQB Certification (3e). 2012, pp. 49.

[6] N. Limam and R. Boutaba.: "Assessing Software Service Quality and Trustworthiness at Selection Time," in IEEE Transactions on Software Engineering, vol. 36, no. 4, pp. 559-574, July-Aug. 2010.

[7] R. Hackbarth, A. Mockus, J. Palframan and R. Sethi.: "Improving Software Quality as Customers Perceive It," in IEEE Software, vol. 33, no. 4, pp. 40-45, July-Aug. 2016.

[8] Black, R., Mitchell, J.: Advanced Software Testing - Vol. 3: Guide to the ISTQB® Advanced Certification as an Advanced Technical Test Analyst. 2011, pp. 21-78.

[9] C. Unger-Windeler, J. Klünder and K. Schneider.: "A Mapping Study on Product Owners in Industry: Identifying Future Research Directions," 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP), 2019, pp. 135-144.

[10] I. M. S. Ardana and Suharjito.: "Software development evaluation process using CMMI-Dev on limited resources company," 2017 3rd International Conference on Science in Information Technology (ICSITech), 2017, pp. 319-324.

[11] C. Giannoulis., M. Petit., J. Zdravkovic.: "Modeling business strategy: A meta-model of strategy maps and balanced scorecards," 2011 Fifth International Conference On Research Challenges In Information Science, 2011, PP. 1-6.

[12] Bosch, J: Continuous Software Engineering, pp. 22. 2014.

[13] W. Lee, S. Ma, S. Lee, P. Law, T. Chang.: "Extending the Multiple Domain Matrix by Using Business Process Model and Notation for Business Process Analysis," 2015 IEEE 12th International Conference on e-Business Engineering, 2015, pp. 311-318.

[14] J. Teixeira, M. Y. Santos, R. J. Machado.: "Business Process Modeling Languages and their Data Representation Capabilities," 2018 International Conference on Intelligent Systems (IS), 2018, pp. 685-691.

[15] N. Debnath, C. A. Martinez, F. Zorzan, D. Riesco, G. Montejano.: "Transformation of business process models BPMN 2.0 into components of the Java business platform," IEEE 10th International Conference on Industrial Informatics, 2012, pp. 1035-1040.

[16] L. Yin, S. Zhi-An and Jiang-Ting-Ting.: "A Software Reliability Test Suite Generating Approach Based on Hybrid Model for Complex Equipment System," 2017 International Conference on Dependable Systems and Their Applications (DSA), 2017, pp. 144-154.

[17] Koomen, T., Pol, M.: Test Process Improvement: A Practical Step-by-step Guide to Structured Testing, 1999.

[18] Test Process Improvement. - Internet / https://www.qualitestgroup.com/assets/test-process-improvement.pdf.

[19] Black, R.: Pragmatic Software Testing: Becoming an Effective and Efficient Test Professional, 2016.

[20] Drabick, R.: Best Practices for the Formal Software Testing Process: A Menu of Testing Tasks, 2013.

[21] L. Hongxin and W. Yi.: "Improvement of software testing process of enterprise project," 2011 International Conference on Electronics, Communications and Control (ICECC), 2011, pp. 3768-3771.

[22] Fujita, H., Mejri, M.: New Trends in Software Methodologies, Tools and Techniques. Proceedings of the fifth SoMeT_06, Volume 147, 2006.

[23] Hass. A M.: Guide to Advanced Software Testing, Second Edition, 2014.

[24] B. Ferreira, T. Conte, S. Diniz Junqueira Barbosa.: "Eliciting Requirements Using Personas and Empathy Map to Enhance the User Experience," 2015 29th Brazilian Symposium on Software Engineering, 2015, pp. 80-89.

[25] Robinson, J., Nanda, V.: Six Sigma Software Quality Improvement. 2011.

[26] Strohmeier, A., J P Rosen.: Reliable Software Technologies - Ada-Europe 2003: 8th Ada-Europe International Conference on Reliable Software Technologies, 2003, pp. 16-20.

[27] Pinkster, I., Burgt, B., Janssen, D.: Successful Test Management: An Integral Approach, 2013.

[28] Spillner, A., Linz, T., Rossner, T.: Winter M. Software Testing Practice: Test Management: A Study Guide for the Certified Tester Exam ISTQB Advanced Level, 2007.

[29] Sistowicz, J., Arell, R.: Change-Based Test Management: Improving the Software Validation Process, 2003.

[30] A. Y. Yurin, A. F. Berman, N. O. Dorodnykh, O. A. Nikolaychuk N. Y. Pavlov.: "Fishbone diagrams for the development of knowledge bases," 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018, pp. 0967-0972.

[31] V. Darmaillacq.: "Security policy testing using vulnerability exploit chaining," 2008 IEEE International Conference on Software Testing Verification and Validation Workshop, 2008, pp. 260-261.

[32] Stephens, R., Beginning Software Engineering, 2015.

[33] Sommerville, I.: Software Engineering (10th Edition), 2015.

[34] Software Test Strategy. - Internet / https://cania-consulting.com/2019/11/01/software-test-strategy/

[35] Quality Assurance: key factors affecting software testing strategies. - Internet / https://www.exposit.com/blog/quality-assurance-key-factors-affecting-software-testing-strategies/

[36] Proactive Vs Reactive: Which Approach is Better to Attain Quality. - Internet / https://www.qualityze.com/proactive-vs-reactive-approach-better-attain-quality/

[37] H. Zhong, L. Zhang, S. Khurshid.: "TestSage: Regression Test Selection for Large-Scale Web Service Testing," 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST), 2019, pp. 430-440.

[38] Dooley, J.: Software Development and Professional Practice, 2011.

[39] Ahuja, L., Sharma, A., Jain, P.: The Impact of Agile Software Development Process on the Quality of Software Product. 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2018.

[40] Fontana, R M., Reinehr, S., Malucelli, A.: Agile Compass: A Tool for Identifying Maturity in Agile Software-Development Teams, Volume 32, Issue 6, 2015.

[41] N. Wattanagul and Y. Limpiyakorn.: "Automated Documentation for Rapid Prototyping," 2016 International Conference on Industrial Engineering, Management Science and Application (ICIMSA), 2016, pp. 1-4.

[42] Hunt, J A.: PMI-ACP Project Management Institute Agile Certified Practitioner Exam Study Guide, 2018.

[43] Sutherland, J., Sutherland, J.: Scrum: The Art of Doing Twice the Work in Half the Time, 2014.

[44] Rubin K S.: Essential. Scrum: A Practical Guide to the Most Popular Agile Process (Addison-Wesley Signature): A Practical Guide To The Most Popular Agile Process (Addison-Wesley Signature Series (Cohn)), 2012.

[45] K. Bereta, G. Stamoulis, M. Koubarakis.: "Ontology-Based Data Access and Visualization of Big Vector and Raster Data," IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, 2018, pp. 407-410.

[46] Y. Zhang, X. Sun.: "The Research on Ontology-based Knowledge Visualization in E-Learning Resources Management," 2008 Fourth International Conference on Semantics, Knowledge and Grid, 2008, pp. 495-496.

[47] R. Damiano, A. Lieto, V. Lombardo.: "Ontology-Based Visualisation of Cultural Heritage," 2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems, 2014, pp. 558-563.

[48] B. H. Yousefi, H. Mirkhezri.: "Lean Gamification Canvas: A New Tool for Innovative Gamification Design Process," 2020 International Serious Games Symposium (ISGS), 2020, pp. 1-9.

[49] Layton.: Agile Project Management For Dummies, 2nd Edition (For Dummies (Computer/Tech)), 2017.

[50] Flewelling, P.: The Agile Developer's Handbook: Get more value from your software development: get the best out of the Agile methodology, 2018.

[51] Todaro, D.: The Epic Guide to Agile: More Business Value on a Predictable Schedule with Scrum, 2019.

[52] Thoren, P.: Agile People: A Radical Approach for HR & Managers (That Leads to Motivated Employees), 2017.

[53] Kumar Lal M.: Knowledge Driven Development: Bridging Waterfall and Agile Methodologies (Cambridge IISc Series), 2018.

[54] Park, J.: Essence and Art of Agile Development: A Practical Roadmap to Agile Enterprises in the Digital Age.

[55] Cohn, J.: Scrum Mastery + Agile Leadership: The Essential and Definitive Guide to Scrum and Agile Project Management, 2019.

[56] M. Ide, Y. Amagai, M. Aoyama, Y. Kikushima.: "A Lean Design Methodology for Business Models and Its Application to IoT Business Model Development," 2015 Agile Conference, 2015, pp. 107-111.

[57] Rasmusson, J.: The Agile Samurai: How Agile Masters Deliver Great Software (Pragmatic Programmers), 2010.

[58] Hanssen, G., Stålhane, T. et al.: SafeScrum – Agile Development of Safety-Critical Software, 2018.

[59] Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.

[60] Bishop, D.: Metagility: Managing Agile Development for Competitive Advantage, 2018.

[61] Reinertsen D G.: The Principles of Product Development Flow: Second Generation Lean Product Development, 2009.

[62] Rothman, J.: Create Your Successful Agile Project: Collaborate, Measure, Estimate, Deliver, 2017.

[63] Cohn, J.: Scrum Fundamentals: A Beginner’s Guide to Mastery of The Scrum Project Management Methodology (Scrum Mastery), 2019.

[64] Jones, S.: Agile Project Management: QuickStart Guide - The Complete Beginners Guide To Mastering Agile Project Management! (Scrum, Project Management, Agile Development), 2016.

[65] Young, M., Gruver, G., et al: A Practical Approach to Large-Scale Agile Development: How HP Transformed LaserJet Future Smart Firmware (Agile Software Development Series).

[66] David, H.: Harned Hands-On Agile Software Development with JIRA: Design and manage software projects using the Agile methodology, 2018.

[67] Economy, P., Patton, J.: User Story Mapping: Discover the Whole Story, Build the Right Product, 2014.

[68] Liang, P, Waseem, M,: Microservices Architecture in DevOps. 24th Asia-Pacific Software Engineering Conference Workshops (APSECW), 2017.

[69] Kersten, M.: A Cambrian Explosion of DevOps Tools. Volume 35. Issue 2. IEEE Software, 2018.

[70] Soni, M.: End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery. IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), 2015.

[71] D. Spinellis.: "Code Documentation," in IEEE Software, vol. 27, no. 4, pp. 18-19, July-Aug. 2010.

[72] N. Wattanagul and Y. Limpiyakorn.: "Automated Documentation for Rapid Prototyping," 2016 International Conference on Industrial Engineering, Management Science and Application (ICIMSA), 2016, pp. 1-4.

[73] M. Magyar.: "Automating software documentation: a case study," 18th Annual Conference on Computer Documentation. ipcc sigdoc 2000. Technology and Teamwork. Proceedings. IEEE Professional Communication Society International Professional Communication Conference an, 2000, pp. 549-558.

[74] Kim, G., K Behr, et al.: The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win.

[75] Vehent, J.: Securing DevOps: Security in the Cloud. 2018.

[76] Daniels, R., Davis, J.: Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale, 2016.

[77] Freeman, E.: DevOps For Dummies (For Dummies (Computer/Tech)), 2019.

[78] Loukides, M.: What is DevOps.

[79] Axelrod, A.: Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects, 2018.

[80] "IEEE Standard for Software and System Test Documentation,".: in IEEE Std 829-2008, vol., no., pp.1-150, 18 July 2008.

[81] S. Ali and T. Yue.: "Formalizing the ISO/IEC/IEEE 29119 Software Testing Standard," 2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), 2015, pp. 396-405.

[82] K. J. Valle-Gómez, P. Delgado-Pérez, I. Medina-Bulo and J. Magallanes-Fernández.: "Software Testing: Cost Reduction in Industry 4.0," 2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST), 2019, pp. 69-70.

[83] A. Figueiras.: "How to Tell Stories Using Visualization," 2014 18th International Conference on Information Visualisation, 2014, pp. 18-18.

[84] Herath, P., Chandrasekara, C.: Hands-On Functional Test Automation: With Visual Studio 2017 and Selenium, 2019.

[85] Mittal, V., Garg, N.: Test Automation using Microsoft Coded UI with C#: Step by Step Guide, 2016.

[86] Malle, G., Sneha, K.: Research on software testing techniques and software automation testing tools. International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), 2017.

[87] Biffl, S., Hametner, R.: Winkler D. Automation component aspects for efficient unit testing. IEEE Conference on Emerging Technologies & Factory Automation, 2009.

[88] Sultanía A K.: Developing software product and test automation software using Agile methodology. Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT), 2015.

[89] S. Jayaraman, K. K. D and B. Jayaraman.: "Towards program execution summarization: Deriving state diagrams from sequence diagrams," 2014 Seventh International Conference on Contemporary Computing (IC3), 2014, pp. 299-305.

[90] Li Wen-Jian, Sun Yue-Lu, Bai Jing and Gao Yan-Hong.: "The state diagram programming on sequential logic design," 2012 International Symposium on Information Technologies in Medicine and Education, 2012, pp. 319-322.

[91] M. Leotta, F. Ricca, G. Antoniol, V. Garousi, J. Zhi and G. Ruhe.: "A Pilot Experiment to Quantify the Effect of Documentation Accuracy on Maintenance Tasks," 2013 IEEE International Conference on Software Maintenance, 2013, pp. 428-431.

[92] F. Ebert.: "From Transient Information to Persistent Documentation: Enhancing Software Documentation," 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2020, pp. 849-853.

[93] Ramler, R., Klammer, C.: A Journey from Manual Testing to Automated Test Generation in an Industry Project. IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2017.

[94] Kumari K S, Yalamanchili, S.: Comparison of manual and automatic testing using genetic algorithm for information handling system. International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), 2016.

[95] Lusser, M., Ramler, R., Ernstbrunner, C., Buchgeher, G.: Towards Tool-Support for Test Case Selection in Manual Regression Testing. IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops, 2013.

[96] Schlogel C, et al.: Regression Test Selection of Manual System Tests in Practice. 15th European Conference on Software Maintenance and Reengineering, 2011.

[97] Olsson, H., Feldt, R., Alégroth, E.: Transitioning Manual System Test Suites to Automated Testing: An Industrial Case Study. IEEE Sixth International Conference on Software Testing, Verification and Validation, 2013.

[98] Surapaneni R T., Varma, H., Joshi H.: Enhanced Test Case Design mechanism for regression & impact testing. International Conference on Computing and Communication Technologies, 2014.

[99] Granda M F: An experiment design for validating a test case generation strategy from requirements models. IEEE 4th International Workshop on Empirical Requirements Engineering (EmpiRE), 2014.

[100] Pendela, S., Route, S.: Combinatorial Test Design – A Smarter Way to Connect with the Business. IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2018.

[101] Tsuda, K., Matsuodani, T., Yumoto, T.: Analysing Test Basis and Deriving Test Cases Based on Data Design Documents. IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2017.

[102] Lawanna A.: Test case design-based technique for the improvement of test case selection in software maintenance. 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), 2016.

[103] Chen T Y.: Fundamentals of test case selection: Diversity, diversity, diversity. The 2nd International Conference on Software Engineering and Data Mining, 2010.

[104] Kasurinen J.: Elaborating Software Test Processes and Strategies. Third International Conference on Software Testing, Verification and Validation, 2010.

[105] Júnior L C.: Operational Profile and Software Testing: Aligning User Interest and Test Strategy. 12th IEEE Conference on Software Testing, Validation and Verification (ICST), 2019.

[106] Barhate S S.: Effective test strategy for testing automotive software. International Conference on Industrial Instrumentation and Control (ICIC), 2015.

[107] Fleurey, F., Haugen, Q., Johansen M F.: A Survey of Empirics of Strategies for Software Product Line Testing. IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, Berlin, Germany, 21-25 March 2011.

[108] Mathrani, A.: Quality assurance strategy for distributed software development using managed test lab model. IEEE International Technology Management Conference, 2014.

[109] Maurya, A.: Running Lean: Iterate from Plan A to a Plan That Works (Lean Series), 2012.

[110] IPC Group.: Eat. Sleep. Create. Entrepreneurs notebook Lean Canvas Business Ideas Journal, 2018.

[111] Olsen D.: The Lean Product Playbook: How to Innovate with Minimum Viable Products and Rapid Customer Feedback, 2015.

[112] 50minutes.: The Business Model Canvas: Let your business thrive with this simple model, 2017.

[113] Lin W.: Lean Startup Canvas Notebook, 2018.

[114] Mirabelli V.: The 1 Page Project: A Practical Guide to Using the Lean Project Canvas to Focus Your Projects on What Matters, 2019.

[115] Poppendieck, M., Poppendieck, T.: Implementing Lean Software Development: From Concept to Cash, 2006.

[116] Jeremy., Bond, G.: Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C# (2nd Edition), 2017.

[117] Razzak M A.: An Empirical Study on Lean and Agile Methods in Global Software Development. 2016 IEEE 11th International Conference on Global Software Engineering Workshops (ICGSEW), 2016.

[118] Iamratanakul S.: The impact of lean practices on NPD performance, IEEE International Conference on Management of Innovation and Technology (ICMIT), 2016.

[119] Paulmani G.: Lean Research Culture: Measurement, Analysis, and Revitalization of Processes and Outcomes of Contemporary Research Practices. IEEE 17th International Conference on Advanced Learning Technologies (ICALT), 2017.

[120] Stefan, H., Mani V S., M S Roopa.: An Approach for Enabling Effective and Systematic Software Reuse: In a Globally Distributed Software Engineering Team That Uses a Lean Development Methodology. IEEE 11th International Conference on Global Software Engineering (ICGSE), 2016.

[121] Reiner, M.: Leadership to lean, IEEE Technology and Engineering Management Conference (TEMSCON), 2018.

[122] Waheed, U., Raza, S.: Managing Change in Agile Software Development a Comparative Study. IEEE 21st International Multi-Topic Conference (INMIC), 2018.

[123] Waheed, U., Shafiq, M.: Documentation in Agile Development A Comparative Analysis. IEEE 21st International Multi-Topic Conference (INMIC), 2018.

[124] Péraire, C., Ralph, P., Sedano, T.: Software Development Waste, 2017.

[125] Marsh J.: UX for Beginners: A Crash Course in 100 Short Lessons, 2016.

[126] Grant W.: 101 UX Principles: A definitive design guide, 2018.

[127] MacDonald D.: Practical UI Patterns for Design Systems: Fast-Track Interaction Design for a Seamless User Experience, 2019.

[128] Levy J.: UX Strategy: How to Devise Innovative Digital Products that People Want, 2015.

[129] Noviyanti A A., Kikitamara S.: A Conceptual Model of User Experience in Scrum Practice. 10th International Conference on Information Technology and Electrical Engineering (ICITEE), 2018.

[130] Schwarz D.: Jump Start Sketch: Master the Tool Made for UI Designers, 2016.

[131] Mittal N M.: Software Design Patterns 2.0: UI and UX Design Principles and Tips.

[132] Notebooks D.: Dot Grid UI and UX Notebook: Planning Sketchbook For Developers, 2019.

[133] Nidagundi P.: Introduction to Investigation and Utilizing Lean Test Metrics In Agile Software Testing Methodologies. Int. Journal of Engineering Research and Applications. ISSN: 2248-9622, Vol. 6, Issue 4, (Part - 1) April 2016, pp.13-16.

[134] Nidagundi, P., Novickis, L.: Possibilities about the design lean canvas model and its adaptation in the agile testing. Symposium for Young Scientists in Technology, Engineering and Mathematics, Volume 1853, SYSTEM 2017, Kaunas, Lithuania, 28 April, pp.20-23.

[135] Nidagundi, P., Novickis, L.: Introduction to Lean Canvas Transformation Models and Metrics in Software Testing, APPLIED COMPUTER SYSTEMS, Year 2016, pp 30-36.

[136] Nidagundi, P., Lukjanska, M.: Introduction to adoption of lean canvas in software test architecture design. Computational Methods in Social Sciences, Volume 4, Number 2, 2017, pp. 23-31(9).

[137] Nidagundi, P., Novickis, L.: Introducing Lean Canvas Model Adaptation in the Scrum Software Testing, Procedia Computer Science. Volume 104, 2017, pp 97-103.

[138] Nidagundi, P., Novickis, L.: Towards Utilization of Lean Canvas in the DevOps Software. Environment. Technology. Resources. Proceedings of the International Scientific and Practical Conference, ISSN 2256-070X, June 2017, pp.107-111.

[139] Nidagundi, P., Novickis, L.: Towards Utilization of a Lean Canvas in the Testing Extra-Functional Properties, Software Engineering Trends and Techniques in Intelligent Systems June 2017, pp 349-354.

[140] Nidagundi, P., Stepanova, V.: Survey on Software Test Strategy. Proceedings of the IRES International Conference, Austria, Vienna, November 2017, pp 26-27.

[141] Nidagundi, P., Novickis, L.: New method for mobile application testing using lean canvas to improving the test strategy, XII-th International Scientific and Technical Conference “Computer Science and Information Technologies”, Lviv, Ukraine, September 05-08, 2017.

[142] Nidagundi, P., Novickis, L.: Towards Utilization of a Lean Canvas in the Biometric Software Testing. A JOURNAL OF MULTIDISCIPLINARY SCIENCE AND TECHNOLOGY, IIOABJ, 2017, Vol.8 (Supple 3), pp 32-36.

[143] Nidagundi, P., Uhanova, M.: Software application security test strategy with lean canvas design, International Conference on Information Technologies. April 2018, pp 50-53.

[144] Bayo E.: Performance Testing with JMeter 3 - Third Edition.

[145] Molyneaux I.: Art of Application Performance Testing: From Strategy to Tools.

[146] Gerardus B.: Performance Testing a Clear and Concise Reference.

[147] Meir.: Performance Testing Guidance for Web Applications -Patterns and Practices.

[148] Peter Z.: J2EE Performance Testing with BEA WebLogic Server.

[149] Milan S.: Web Systems Security Testing.

[150] Arvind D.: Security Testing Handbook for Banking Applications.

[151] Someshwar G.: Security Testing for Web Applications in SDLC.

[152] Brian H.: Web Security Testing Cookbook.

[153] Wysopal C.: Art of software security testing.

[154] Paul M J.: Modernizing Legacy Applications in PHP.

[155] Umar A.: Application Reengineering - Building Web-Based Applications and Dealing with Legacies.

[156] Pallab C.: Legacy Data: A Structured Methodology for Device Migration in DSM Technology.

[157] Ian W.: The Renaissance of Legacy Systems.

[158] Heuvel,W., Brodie M.: Aligning Modern Business Processes and Legacy Systems: A Component-Based Perspective (Cooperative Information Systems) (Cooperative Information Systems Series).

[159] Last, M., Kandel, A., Bunke, H.: Artificial Intelligence Methods In Software Testing.

[160] Freedie R.: Artificial Intelligence and the Future of Testing.

[161] Singh N.: An Artificial Intelligence Approach to Test Generation.

[162] Rich, E., Knight, K., Nair, S.: Artificial Intelligence (Sie).

[163] Vinod, C., Hareendran, S.: Artificial Intelligence and Machine Learning

[164] World Quality Report - Sogeti. - Internet / https://www.sogeti.com/explore/reports/world-quality-report-2017-2018/.

[165] WQR – Sogeti. - Internet / https://www.sogeti.com/globalassets/global/wqr-201819/wqr-2018-19_secured.pdf.

[166] Lean Canvas Template. - Internet / http://nicchimo.info/wp-content/uploads/2018/12/lean-canvas-template-word-inspirational-business-plan-the-mission-model-free.jpg.

[167] Agile Testing Quadrants. - Internet / https://labs.bawi.io/the-purpose-of-testing-30152064c21e.

[168] Gautam, S.; Study of Software Development Process and Their Impact on Software Quality, Year 2017.

[169] Guarino, N.: Formal Ontology and Information Systems, Formal Ontology in Information Systems. Proceedings of FOIS’98, Trento, Italy, Year 1998. Amsterdam, IOS Press, pp. 3-15.

[170] Ontology mapping - The mind-mapping.org Blog. - Internet / https://www.mind-mapping.org/blog/tag/ontology-mapping/.

[171] Mind maps & UX design - UX Collective. - Internet / https://uxdesign.cc/mind-maps-ux-design-96a2d1333d7c.

[172] Graner, M., Mißler-Behr., M: 22. Key determinants of the successful adoption of new product development methods, European Journal of Innovation Management 16(3), Year 2013.

[173] Visualizing the Complex Software Development Process. - Internet / https://www.cmcrossroads.com/article/visualizing-complex-software-development-process.

[174] Visualize Strategy. - Internet / http://udleditions.cast.org/strategy_visualize.html.

[175] Tan, K., Platts, K.: Strategy visualisation: knowing, understanding, and formulating.

[176] Sawant, A., Bari, P., Chawan, P.: Software Testing Techniques and Strategies.

[177] What are some major advantages and disadvantages of fishbone diagrams. - Internet / https://www.enotes.com/homework-help/what-advantages-disadvantages-fishbone-diagrams-299036.

[178] What is a Strategy Map. - Internet / - https://balancedscorecards.com/strategy-map/#strategy-map-showing-performance.

[179] How to Develop a Strategy Map. - Internet / https://www.cgma.org/content/dam/cgma/resources/tools/downloadabledocuments/cgma-strategy-mapping-tool-final.pdf.

[180] Empathy Mapping: The First Step in Design Thinking. - Internet / https://www.nngroup.com/articles/empathy-mapping/.

[181] BPMN Tutorial: Quick-Start Guide to Business Process Model and Notation. - Internet / https://www.process.st/bpmn-tutorial/.

[182] Business Process Model and Notation. - Internet / https://www.goodelearning.com/courses/business-process/bpmn-training/what-is-bpmn.

[183] Chen, N.: std 829-2008 and Agile Process – Can They Work Together?, Year 2013.

[184] Peter M., Montecchiari D., Hinkelmann K., Gatziu Grivas S. (2020) Ontology-Based Visualization for Business Model Design. In: Grabis J., Bork D. (eds) The Practice of Enterprise Modeling. PoEM 2020. Lecture Notes in Business Information Processing, vol 400. Springer, Cham, Year 2020.

[185] Test Strategy. - Internet / - https://strongqa.com/qa-portal/testing-docs-templates/test-strategy.

[186] Penella S.: Overcoming Challenges to Good Test Documentation - Internet / https://www.stickyminds.com/article/overcoming-challenges-good-test-documentation.

[187] Whiteboard Design Challenge Framework - Internet - https://uxdesign.cc/designchallenge-cd7bdb589855.

[188] I. Burnstein, T. Suwanassart, R. Carlson.: "Developing a Testing Maturity Model for software test process evaluation and improvement," Proceedings International Test Conference 1996. Test and Design Validity, 1996, pp. 581-589.

[189] A. K. Talukder et al.: "Security-aware Software Development Life Cycle (SaSDLC) - Processes and tools," 2009 IFIP International Conference on Wireless and Optical Communications Networks, 2009, pp. 1-5.

[190] "ISO/IEC/IEEE International Standard for Systems and software engineering -- Content management for product life-cycle, user, and service management documentation," in ISO/IEC/IEEE 26531:2015 (E), vol., no., pp.1-60, 15 May 2015.

[191] B. V. Thummadi, O. Shiv and K. Lyytinen.: "Enacted Routines in Agile and Waterfall Processes," 2011 Agile Conference, 2011, pp. 67-76.

[192] "ISO/IEC/IEEE International Standard - Systems and software engineering -- Requirements for managers of user documentation," in ISO/IEC/IEEE 26511 First edition 2011-12-01; Corrected version 2012-03-15, vol., no., pp.1-54, 15 March 2012.

[193] J. Bach, P. Aiken.: "Document management as a prerequisite to quality assurance," 1990 IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings, 1990, pp. 739-741.

[194] V. T. Faniran, A. Badru, N. Ajayi.: "Adopting Scrum as an Agile approach in distributed software development: A review of literature," 2017 1st International Conference on Next Generation Computing Applications (NextComp), 2017, pp. 36-40.

[195] H. F. Soares, N. S. R. Alves, T. S. Mendes, M. Mendonça, R. O. Spínola.: "Investigating the Link between User Stories and Documentation Debt on Software Projects," 2015 12th International Conference on Information Technology - New Generations, 2015, pp. 385-390.

[196] A. Mathrani.: "Quality assurance strategy for distributed software development using managed test lab model," 2014 IEEE International Technology Management Conference, 2014, pp. 1-4.

[197] R. Pröll., B. Bauer.: "A Model-Based Test Case Management Approach for Integrated Sets of Domain-Specific Models," 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2018, pp. 175-184.

[198] R. Ramler., C. Klammer.: "Enhancing Acceptance Test-Driven Development with Model-Based Test Generation," 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2019, pp. 503-504.

[199] Testing Methods, Tools, and Techniques. - Internet / https://itabok.iasaglobal.org/itabok3_0/testing-methods-tools-and-techniques/

Appendices

1. List of abbreviations

No	Abbreviations	Description
1.	MTBF	Mean Time Between Failures
2.	MTTF	Mean Time To Failure
3.	MTTR	Mean Time To Repair
4.	API	Application Program Interface
5.	UI/UX	User Interface/User Experience
6.	GUI	Graphical User Interface
7.	UI	User Interface
8.	TMap	Test Process Management
9.	BDD	Behaviour-Driver Development
10.	TDD	Test Driven Development
11.	ATTD	Acceptance Test-Driven Development
12.	BRS	Business Requirement Specifications
13.	LTR	Level Test Status Report
14.	CTTD	Independent Verification and Validation
15.	TCoE	Testing Centres of Excellence
16.	ISO	International Organization for Standardization
17.	IEC	International Electrotechnical Commission
18.	IEEE	Institute of Electrical and Electronics Engineers
19.	KPI	Key Performance Indicator
20.	CI	Continuous Integration
21.	CD	Continuous Delivery
22.	ISTQB	International Software Testing Qualifications Board
23.	CI/CD	Continuous Integration/ Continuous Delivery
24.	XP	Extreme Programming
25.	QA	Quality Assurance

26.	TQA	Technical Quality Assurance
27.	DevOps	Software Development and Information-Technology Operations
28.	MVP	Minimum Viable Product
29.	SAAS	Software As A Service
30.	UXD	User Experience Design
31.	UCD	User Centred Design
32.	PO	Product Owner
33.	IEEE	Institute of Electrical and Electronics Engineers
34.	QC	Quality Control
35.	HTML	Hypertext Markup Language
36.	CSS	Cascading Style Sheets
37.	HTTP	Hypertext Transfer Protocol Secure
38.	CMS	Content Management System
39.	LISTS	Lean Canvas Software Test Strategy
40.	AI	Artificial Intelligence
41.	ML	Machine Learning
42.	IoT	Internet of Things
43.	ABB	Test activities building block
44.	SDLC	Software Development Life Cycle
45.	PBI	Product Backlog Item
46.	UAT	User Acceptance Testing
47.	SLAs	Service Level Agreements
48.	OS	Operating System
49.	SQL	Structured Query Language
50.	CSS	Cross-Site Scripting
51.	CSRF	Cross-Site Request Forgery
52.	SNS	Social Networking Service
53.	XML	Extensible Mark-up Language

54.	E2E	End to End
55.	IDE	Integrated Development Environment
56.	NIP	Neuro-Linguistic Programming
57.	POC	Proof of Concept
58.	AWS	Amazon Web Services
59.	INT	Internal Environment
60.	FAT	Test Environment
61.	ACC	Acceptance Environment
62.	IT	Information Technology
63.	UC	Use Case
64.	DB	Data Base
65.	PFD	Process Flow Diagram
66.	DOM	Document Object Model
67.	WWWWWH	Why, Who, What, When, Where, How.
68.	STD	Story, Development, Testing
69.	TD	Technical Design
70.	SD	System Developer
71.	PME	Performance Measurement and Evaluation
72.	CMMI	Capability Maturity Model Integration
73.	ISO	International Organization for Standardization
74.	IEC	International Electrotechnical Commission
75.	ISTQB	International Software Testing Qualifications Board
76.	PFD	Process flow diagram
77.	UC	Use case
78.	DEV	Developer
79.	UML	Unified Modelling Language
80.	TABB	Test activities building block

2. List of figures

3. List of tables

4.9	4	Analysis of Results of Empirical Study
4.10	4	Created Visualised Test Strategy and Feedback
4.11	4	Overview of the Goals, Objectives and Outcome
4.12	4	Comparison of the Proposed Test Strategy with Other Test Strategies
4.13	4	Common Sub-blocks and Titles Collected from the Board

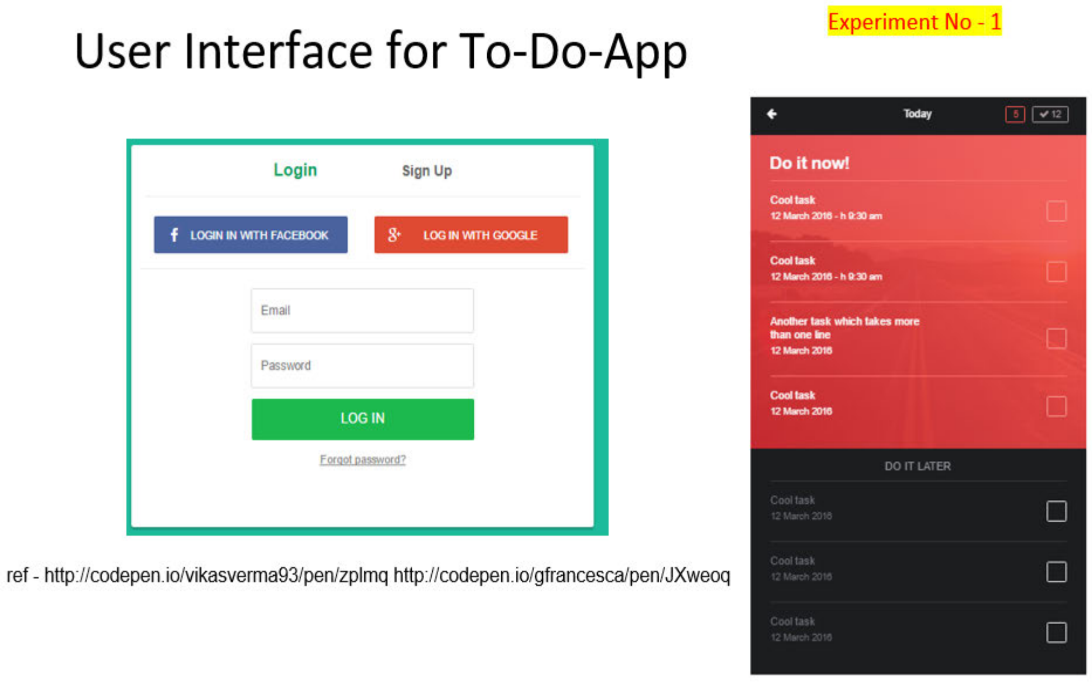
4. Classroom experiment work by students of Riga Technical University (RTU)

Aim: To find out how university students use lean canvas test strategy to test a mobile application.

Task: To assess student software testing skills, utilization of lean canvas providing knowledge about lean canvas in problem-solving and introducing students to strategy creation in real life. In the final step, to use lean canvas in test strategy creation.

Process: A student gets a) mobile application screenshot printout and b) lean canvas test strategy template printout.

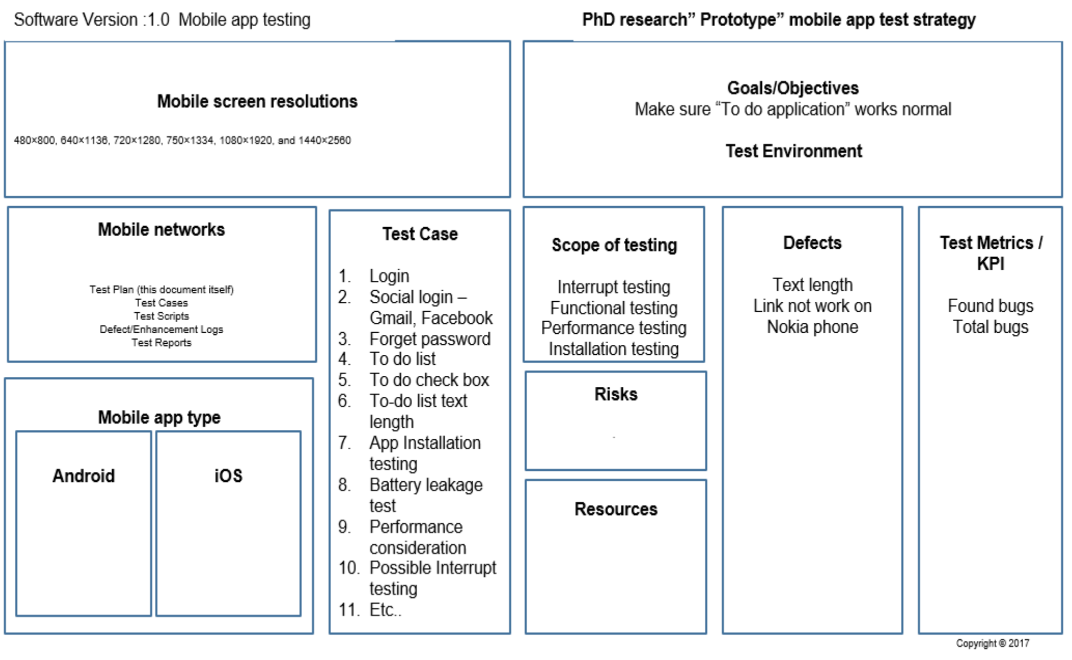
In the next step, students start filling in the canvas test strategy template in terms of testing mobile application screenshot provided.


User Interface for To-Do-App

Experiment No - 1

ref - http://codepen.io/vikasverma93/pen/zplmq http://codepen.io/gfrancesca/pen/JXweoq

Figure A.1: Screenshot of To-Do-App experiment.


Software Version :1.0 Mobile app testing

PhD research" Prototype" mobile app test strategy

Mobile screen resolutions

480×800, 640×1136, 720×1280, 750×1334, 1080×1920, and 1440×2560

Goals/Objectives
Make sure "To do application" works normal

Test Environment

Mobile networks

Test Plan (this document itself)
Test Cases
Test Scripts
Defect/Enhancement Logs
Test Reports

Mobile app type

Android **iOS**

Test Case

1. Login
2. Social login – Gmail, Facebook
3. Forget password
4. To do list
5. To do check box
6. To-do list text length
7. App Installation testing
8. Battery leakage test
9. Performance consideration
10. Possible Interrupt testing
11. Etc..

Scope of testing

Interrupt testing
Functional testing
Performance testing
Installation testing

Risks

Resources

Defects

Text length
Link not work on Nokia phone

Test Metrics / KPI

Found bugs
Total bugs

Figure A.2: Lean canvas tests strategy for To-Do-App experiment.

Student feedback:

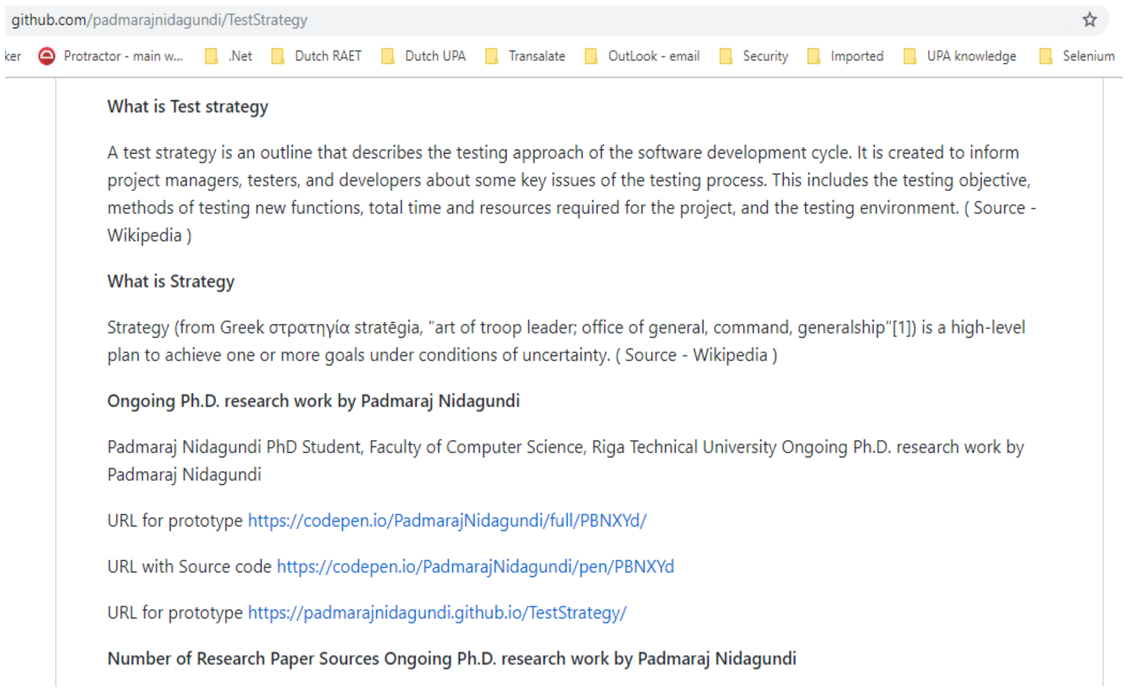
- It is easy to learn what I can fill in.
- It is like a whiteboard to fill in the information.
- Easy to make a testing strategy using this template.
- I understood now! what I need to test.

In other observations it was also noticed that few students lack practical software testing knowledge; for such students, this practical experiment was less appealing.

5. Creation of a generic test activity building block template in GitHub

Aim: To make students aware how to use a test strategy using the lean canvas via creating a publicly accessible codebase.

Task: To create a skeleton of lean canvas in github.com and codepin.io using open-source web technologies.



What is Test strategy

A test strategy is an outline that describes the testing approach of the software development cycle. It is created to inform project managers, testers, and developers about some key issues of the testing process. This includes the testing objective, methods of testing new functions, total time and resources required for the project, and the testing environment. (Source - Wikipedia)

What is Strategy

Strategy (from Greek στρατηγία stratēgia, "art of troop leader; office of general, command, generalship"[1]) is a high-level plan to achieve one or more goals under conditions of uncertainty. (Source - Wikipedia)

Ongoing Ph.D. research work by Padmaraj Nidagundi

Padmaraj Nidagundi PhD Student, Faculty of Computer Science, Riga Technical University Ongoing Ph.D. research work by Padmaraj Nidagundi

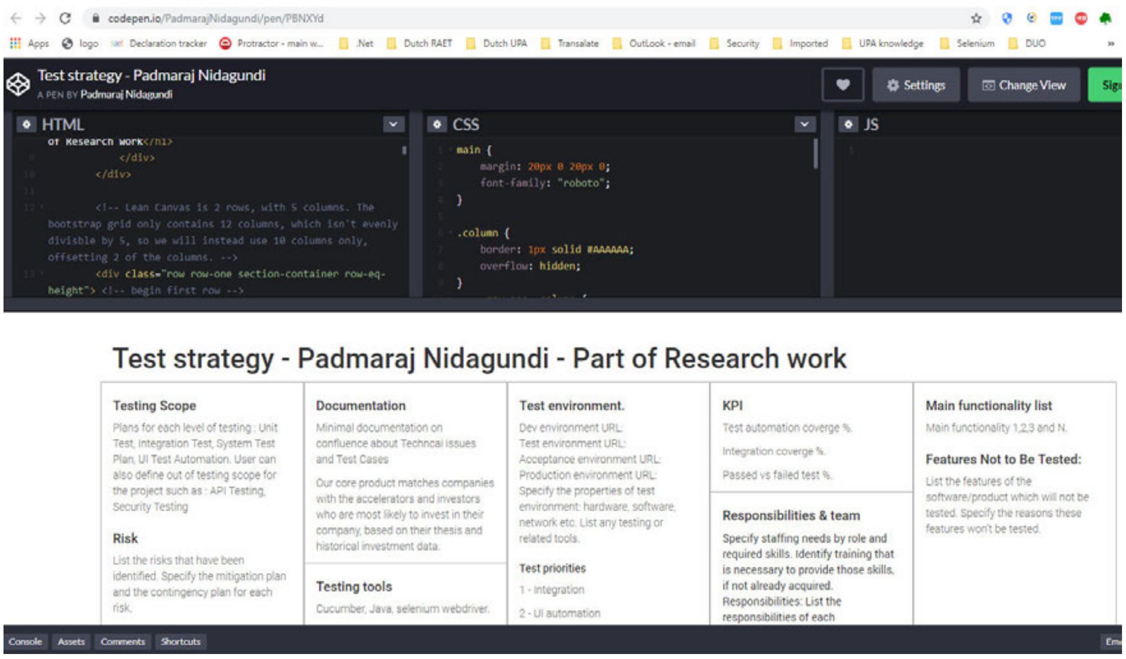
URL for prototype https://codepen.io/PadmarajNidagundi/full/PBNXYd/

URL with Source code https://codepen.io/PadmarajNidagundi/pen/PBNXYd

URL for prototype https://padmarajnidagundi.github.io/TestStrategy/

Number of Research Paper Sources Ongoing Ph.D. research work by Padmaraj Nidagundi

Figure A.3: Lean canvas test strategy on github.com resource page.



```html
of Research work</h1>
                </div>
            </div>

            <!-- Lean Canvas is 2 rows, with 5 columns. The bootstrap grid only contains 12 columns, which isn't evenly divisible by 5, so we will instead use 10 columns only, offsetting 2 of the columns. -->
            <div class="row row-one section-container row-eq-height"> <!-- begin first row -->
```

```css
main {
    margin: 20px 0 20px 0;
    font-family: "roboto";
}

.column {
    border: 1px solid #AAAAAA;
    overflow: hidden;
}
```

Test strategy - Padmaraj Nidagundi - Part of Research work

Testing Scope
Plans for each level of testing : Unit Test, Integration Test, System Test Plan, UI Test Automation. User can also define out of testing scope for the project such as : API Testing, Security Testing

Risk
List the risks that have been identified. Specify the mitigation plan and the contingency plan for each risk.

Documentation
Minimal documentation on confluence about Techncai issues and Test Cases

Our core product matches companies with the accelerators and investors who are most likely to invest in their company, based on their thesis and historical investment data.

Testing tools
Cucumber, Java, selenium webdriver.

Test environment.
Dev environment URL:
Test environment URL:
Acceptance environment URL:
Production environment URL:
Specify the properties of test environment: hardware, software, network etc. List any testing or related tools.

Test priorities
1 - Integration
2 - UI automation

KPI
Test automation coverge %.
Integration coverge %.
Passed vs failed test %.

Responsibilities & team
Specify staffing needs by role and required skills. Identify training that is necessary to provide those skills, if not already acquired.
Responsibilities: List the responsibilities of each

Main functionality list
Main functionality 1,2,3 and N.

Features Not to Be Tested:
List the features of the software/product which will not be tested. Specify the reasons these features won't be tested.

Figure A.4: Lean canvas test strategy template in codepen.io with its source code.

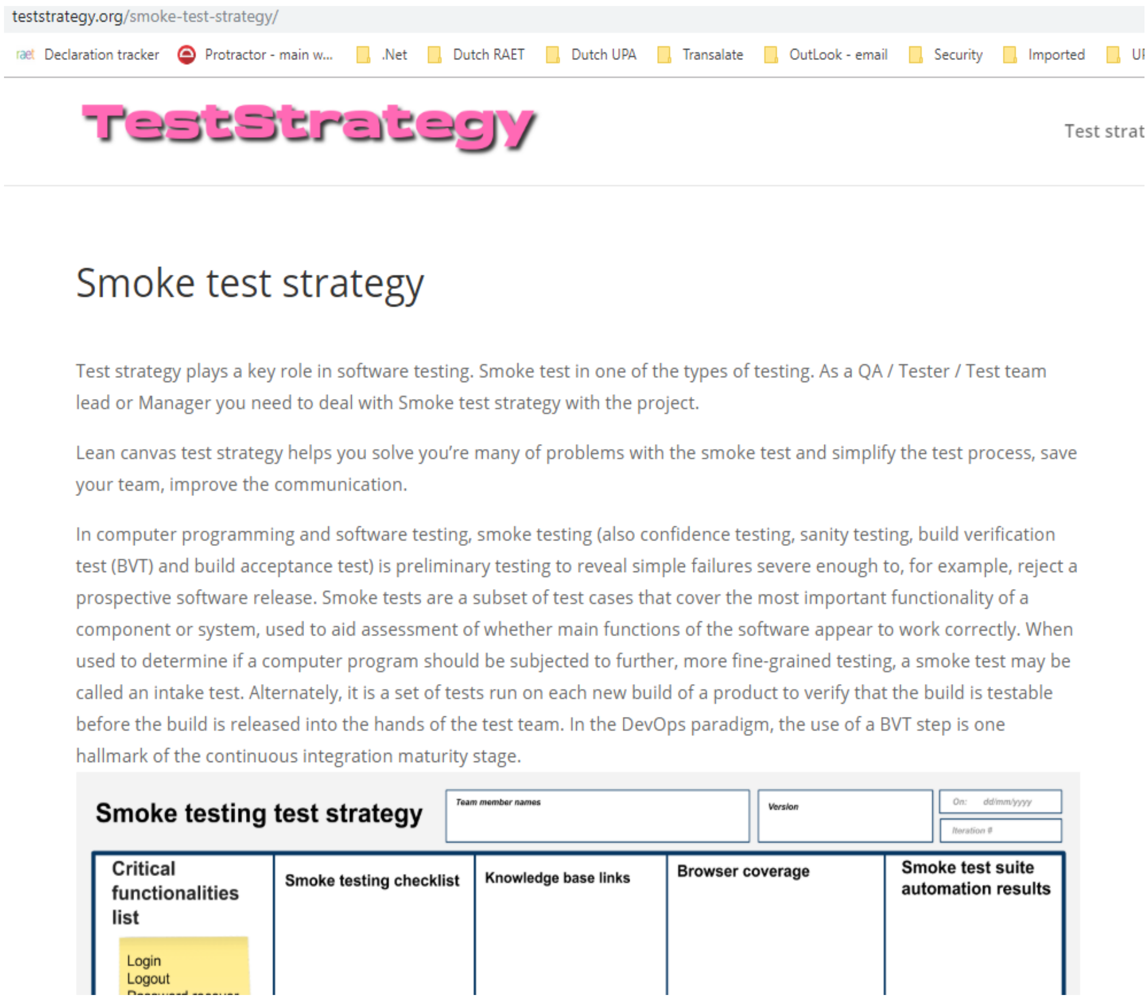
- ●Resource link: https://codepen.io/PadmarajNidagundi/full/PBNXYd/
- ●Resource link with codepen.io source code https://codepen.io/PadmarajNidagundi/pen/PBNXYd
- ●Resource link: https://padmarajnidagundi.github.io/TestStrategy/

6. Creation of a test strategy resource website for software testing community

Aim: To create a website with ready-to-use templates for the testing community around the world.

Task: Software test professional use the lean canvas for test strategy creation. A user can use Google templates by downloading them to their docs, optional sticky notes are also provided.

Resource link: https://teststrategy.org



Smoke test strategy

Test strategy plays a key role in software testing. Smoke test in one of the types of testing. As a QA / Tester / Test team lead or Manager you need to deal with Smoke test strategy with the project.

Lean canvas test strategy helps you solve you're many of problems with the smoke test and simplify the test process, save your team, improve the communication.

In computer programming and software testing, smoke testing (also confidence testing, sanity testing, build verification test (BVT) and build acceptance test) is preliminary testing to reveal simple failures severe enough to, for example, reject a prospective software release. Smoke tests are a subset of test cases that cover the most important functionality of a component or system, used to aid assessment of whether main functions of the software appear to work correctly. When used to determine if a computer program should be subjected to further, more fine-grained testing, a smoke test may be called an intake test. Alternately, it is a set of tests run on each new build of a product to verify that the build is testable before the build is released into the hands of the test team. In the DevOps paradigm, the use of a BVT step is one hallmark of the continuous integration maturity stage.

Smoke testing test strategy

Team member names	Version	On: dd/mm/yyyy
		Iteration #

Critical functionalities list	Smoke testing checklist	Knowledge base links	Browser coverage	Smoke test suite automation results
Login Logout				

Figure A.5: Lean canvas test strategy template at teststrategy.org with Google docs templates.

7. WWWWWH questionnaire survey documentation for test strategy

In the second comapny, **WWWWWH** questionnaire survey approach was adopted for test strategy generation. Teams create high-level test strategy documentation using these questionaries.

The team needs to start discussion with asking WWWWWH questions and answering them.

Why?

What defines 'success' for our TA efforts?

How does TA fit into our overall testing process? (and does it fit at all in the first place?)

What tests are we going to automate?

What?

What tests will be part of our deployment pipeline?

Are we going to focus on regression testing first? or cover new features?

What knowledge / experience is present in our team?

What do we lack in this area?

Who?

Are we going to set up a dedicated TA team?

Who will benefit from the results of our TA? Who do we need to inform?

How do we make TA part of our Definition of Done?

When?

Are we going to run our tests daily? Hourly? Post commit?

How do we make sure we don't get overwhelmed by

'urgent' matters and don't have enough time for TA?

What language are we going to write our tests in?

Where?

What environment is test run for?

Test run on multiple environments at the same time or one after another

How?

What seams in our application can we leverage?

How do we monitor the results of our TA efforts?

As a result of the above discussion different level test possibilities can be considered.

Test Strategy at Different Levels

Example set of WWWWWH questions and answers:

- **Unit Testing**

WHY: To ensure the code is developed correctly

WHO: Developers / Technical Architects

WHAT: All-new code + refactoring of legacy code as well as JavaScript unit testing

WHEN: As soon as the new code is written

WHERE: Local Dev

HOW: Automated

- **API Service Testing**

WHY: To ensure communication whether one or many components are working

WHO: Developers / Technical Architects / Test automation engineers

WHAT: New web services, components, controllers, etc.

WHEN: As soon as the new API is developed and ready

WHERE: INT / FAT / ACC environments

HOW: Automated, Soap UI, Rest Client, Postman, Spec Flow

- **UI Test automation**

WHY: To ensure user E2E UI actions are working

WHO: Test automation engineers

WHAT: User interface and main user flows

WHEN: After product release or when interface gets stable

WHERE: INT / FAT / ACC / PROD environments

HOW: C#, Spec Flow

- **Integration Testing**

WHY: To make sure the complete system works when integrated with the new functionality delivered

WHO: Manual, automated QA and Product Owner

WHAT: Basic scenario testing, user flows and typical User Journeys

WHEN: Once the build package is deployed in INT environment

WHERE: INT environment

HOW: Automated, Soap UI, Rest Client, Postman, Spec Flow

- **FAT Testing (Systems Test)**

WHY: To make sure the complete system works when integrated

WHO: Manual & Automated QA / CAT team / UAT team

WHAT: Scenario testing, user flows and typical User Journeys, performance and security testing

WHEN: Once the build package is deployed in the FAT environment

WHERE: FAT environment

HOW: Manual / Automated

- **User Acceptance Testing (UAT)**

WHY: To ensure customer’s expectations are met

WHO: UAT team / End users / PD Teams

WHAT: Verifying acceptance tests on the stories, verification of features

WHEN: When the feature is ready

WHERE: FAT / Acceptance Environment

HOW: Manual / Automated

- **Performance Testing**

WHY: To ensure customer’s expectations are met

WHO: Performance team

WHAT: Verifying the performance tests

WHEN: When the feature is ready

WHERE: FAT Environment

HOW: Automated

- **Security Testing**

WHY: To ensure customer’s expectations are met

WHO: Security team

WHAT: Verifying the security tests

WHEN: When the feature is ready

WHERE: FAT & ACC Environment / PROD

HOW: Automated

- **Acceptance Verification Test**

WHY: To ensure customer’s expectations are met

WHO: Manual & Automated QA / CAT team / UAT / End Users

WHAT: Verifying acceptance tests on the stories, verification of features

WHEN: When a feature is ready

WHERE: Acceptance Environment

HOW: Manual / Automated

- **Production Verification Test**

WHY: To ensure customer’s expectations are met

WHO: Manual & Automated QA / CAT team / end users

WHAT: Verifying acceptance tests on the stories, verification of features

WHEN: When the feature is ready

WHERE: Production Environment

HOW: Manual / Automated

Running the above-presented discussion of the test strategy at different levels leads to generating the overview of test strategy.

Step #1– Scope and Overview:

Project overview along with information on who should use this document. Also include such details as who will review and approve this document. Define testing activities and phases to be carried out with timelines with respect to overall project timelines defined in the test plan.

Step #2– Test Approach:

Define testing process, level of testing, roles, and responsibilities of every team member. For every test type defined in the test plan (e.g.: unit, integration, system, regression, installation/uninstallation, usability, load, performance, and security testing), describe why it should be conducted along with such details as when to start, test owner, responsibilities, testing approach and details of automation strategy and tool if applicable. In test execution there are various activities like adding new defects, defect triage, defect assignments, re-testing, regression testing and finally test sign-off. You must define exact steps to be followed for each activity. You can follow the same process which worked for you in your previous test cycles. A Visio presentation of all these activities including a number of testers and who will work on what activity is very helpful to quickly understand the roles and responsibilities in the team.

Example: defect management cycle – mention the process to log the new defect: Where to log in, how to log new defects, what should be the defect status, who should do defect triage, whom to assign defects after triage, etc.

Also, define the change in the management process. This includes defining change request submission, template to be used, and process to handle the request.

Step #3– Test Environment:

Test environment setup should outline information about a number of environments and required setup for each environment. Example: one test environment for functional test team and another for UAT team. Define the number of users supported in each environment, access roles for each user, software and hardware requirements like operating system, memory, free disk space, number of systems, etc.

Defining test data requirements is equally important. Provide clear instruction on how to create test data (either generate data or use production data by masking fields for privacy).

Define test data backup and restore strategy. Test environment database may run into problems due to unhandled conditions in the code. I remember the problems we faced in one of the projects when there was no database backup strategy defined and we lost the whole data due to code issues. Backup and restore process should define who will take backups, when to take a backup, what to include in the backup, when to restore the database, who will restore it and data masking steps to be followed if the database is restored.

Step #4– Testing Tools:

Define test management and automation tools required for test execution. For performance, load and security testing describes the test approach and tools required. Mention whether it is open source or commercial tool and how many users are supported on it and plan accordingly.

Step #5 – Release Control:

As mentioned in our last UAT article, unplanned release cycle could result in different software versions in test and UAT environments. Release management plan with proper version history will ensure test execution of all modifications in that release.

Example: set build management process which will answer where new build should be made available, where it should be deployed, when to get the new build, from where to get the production build, who will give the go, the no-go signal for production release, etc.

Step #6 – Risk Analysis:

List all risks that you envision. Provide a clear plan to mitigate these risks and also a contingency plan in case you see these risks in reality.

Step #7 – Review and Approvals:

When all these activities are defined in the test strategy plan, it needs to be reviewed for sign-off by all entities involved in project management, business team, development team, and system administration (or environment management) team.

Summary of review changes should be tracked at the beginning of the document along with the approver's name, date and comment. Also, it is a living document meaning it should be continuously reviewed and updated with the testing process enhancements.

Tips to Write Test Strategy Document

1) Include product background in the test strategy document. In the first paragraph of your test strategy document answer the question – Why stakeholders want to develop this project? This will help to understand and prioritize issues quickly.

2) List all important features you are going to test. If you think some features are not part of this release, then mention those features under "Features not to be tested" label.

3) Write down the test approach for your project. Clearly, mention what types of testing you are going to conduct, i.e., Functional testing, UI testing, Integration testing, Load/Stress testing, Security testing, etc.

4) Answer such questions as how you are going to perform functional testing? Is it manual or automation testing? Are you going to execute all test cases from your test management tool?

5) Which bug tracking tool are you going to use? What will be the process when you will find a new bug?

6) What are your test entry and exit criteria?

7) How will you track your testing progress? What metrics are you going to use for tracking test completion?

8) Task distribution – Define roles and responsibilities of each team member.

9) What documents will you produce during and after the testing phase?

10) What are all risks you envision on the way to test completion?

In order to have the possibility to deliver one or more of the following work items, the project manager and teams are obliged to invite all testers, or the person responsible for the test, to relevant meetings, daily stand-up and provide them with all necessary information that they need to deliver valuable deliverables.

A sample table presents a test strategy developed considering all aspects discussed above.

What	**Who**	**When**	**Where**	**Why**	**What**	**How**
Master Test Plan / Test Plan	QAR	When release plan is establis hed	publish on confluence	Documentati on	Short test plan overview	Manual
Test Lists	QA	When require ments are approve d	Zephyr	Documentati on	Test list overview	Manual
Test Cases	QA respon sible for requir ement s	When require ments are approve d	Zephyr	Documentati on	Test cases	Manual
Acceptanc e Criteria	QA respon sible	When writing	Confluenc e	Documentati on	How it needs to work	Automated

for each User Story	for requirements	requirements				
Regression Test List	QA		Zephyr	Documentation	Test everything	Automated
System Test Report	QAR	After system test	publish on confluence	Documentation	Test report	Automated

8. STD (story, development, testing) test strategy

I the first company, software quality was implemented from the very start of the project – the team must build quality and inform on the start of the development process. Testing begins immediately, including the exploration of the requirements and what the client wants.

By refining the user stories from various perspectives, both productive and efficient tests are created. The team must concentrate on the automated tests over manual tests to assure quick test results.

- Quick, immediate test feedback is expected;
- Defects should be identified quickly;
- The objective is to achieve better quality and shorter lead times with the least overhead.

When a user story is ready for grooming:

- It includes the role, need and benefit;

Example: As a buyer, I want to know the specific shipping cost of my order, so that the entire process is more predictable.

- It includes high level acceptance criteria;

Example: When a user enters a zip code, the system should automatically fill out the identical city, so that the checkout process is more straightforward.

Example: The system must feel responsive to the user while displaying 500 items in the shopping cart.

- It includes concrete examples (if necessary, use real data);
- It combines mock-ups/flow charts (if necessary);
- It takes no more than 3 days to implement.

When a user story is ready for development:

- The team has a shared perception of the story;
- Potential risks have been addressed and decreased;
- It needs to be clear how the story is tested, and by whom (QA, Developer, Product Owner);

  How the story is tested: automated, manual, exploratory, code review, regression test or a smoke test.

  Test type: Functional, performance, security, accessibility.

  Test level: unit, integration, web service, GUI.

  Test support: test data, logging, test functionality.
- The work is necessarily estimated.

Definition of done: A user story is done:

- It is fully implemented, tested and approved of by the QA, UX designer, PO, according to the plan (acceptance criteria, automated testing, manual testing, code review, documentation).
- How many approvers were involved depends on the story. Who should be included is determined in the grooming by the team.

Table for Outline of Test Activities for the Team

Test type	How	Type	Owner
Define Story Acceptance Tests (Use Example)		Manual	TEAM (PO/DEV/QA)
Unit Tests		Automated	DEV
Static Tests	Review	Manual/Automated	DEV
Automated API Tests	RESTSHARP	Automated	DEV/QA
Automated GUI	Selenium	Automated	DEV/QA
Functional - Regression		Automated/Manual	QA
Exploratory		Manual	QA/PO
Performance		Automated	DEV/QA
Usability		Manual	TEAM (PO/DEV/QA)
Security		Automated	DEV/QA
Integration Messages		Automated/Manual	DEV/QA
System Tests		Automated	DEV/QA

Although this strategy is internally developed, it contains many supporting sub-documents to make it comprehensions. This testing strategy is not applicable to all internal projects.

9. Company test process overview

First Company Test Process Overview

The following section provides an overview of software development and testing process in company. The software development life cycle is a standardised method for developing software application used for many projects for a long time. Software testers were working on waterfall projects; it is like a long-term step-by-step process from point A to point B. Projects are a significant kind of Monolithic systems that are difficult to backtrack once they have started.

First, getting started as a tester and getting a new role in a company, a new tester is not familiar with the company adopted testing process and tools. In such a situation the first thing the tester wants to do is to check what type of systems development life cycle (SDLC) is used, is it waterfall or agile. It is necessary to analyse how the software development life cycle looks like, how the software is created and when it is delivered to the end customer. In general, the tester needs to investigate the company's working process and its tools. Another methodology used in the company is agile. It is a much quicker methodology of creating software which can be done monthly on a weekly basis. The software development steps are all sensitive and the same regardless of the type of the software developer necessary steps. The tester responsibilities make sure that the user requirements are fulfilled in the software. The user wants to get the expected return on data.

The testing plan is at least 20-page document with chapters and subchapters, and it is necessarily what the tester is going to test as it is describes the testing plan, the process, and it contains details inside of the system features to test, how to do the testing, the overall strategy, the documentation of the testing process.

The test cases, which are designed to verify the software function, are used to consider expected results, what users expect from the system, and what the tester gets as an actual result. If any software functionality does not match the test case, then potentially it is needed to report the defect. Defect reporting investigates how it happened, what exactly happened before. Creating the defect reporting in detail the tester can communicate with the developer because the defect must be fixed. It meets the user’s expectations. The test case is written on the spreadsheet, and then the manually executed test records the results on the spreadsheet. If the team has written a test automation script, then it examines it on a day-to-day basis.

Software defect reports need to be generated by a test group. This phase uncovers software defects, and each defect is noted in the script and it can be found in testing. The moment detail defect report is created as well, and developers use defect reports. It is imperative when testers report a defect, a tester must make sure that the tester has the recreation steps that are used by developers more precisely. In other words, the tester should be able to test the scenario, in this process the tester should be able to go to the screen, the tester finds the defect on a click, adds a picture or a video and lists the steps in detail.

Each project has management software to deal with the project. At this stage, the following items appear: specific deliverables and who is responsible. Specific team members are assigned to various tasks. They are designers, developers and a tester.

In the first company, there is no testing strategy, and most of the time it is necessary to deal with reading long documents and updating them. Mainly the testing team members create and update a short description of the testing to have the business application and test documents as a further reference. In the testing process, the test team used tools such as test tarantula, hiptest, Atlassian JIRA, Atlassian page, Microsoft word documents, excel spreadsheets, JMeter, Wireshark, Burp suite, DLLInjector. Most of the time, the software was tested manually, and a few tests were automated for Selenium and custom automated tools for biometric software validation and verification. With non-functional testing, only performance test was done in the project.

Second Company Test Process Overview

The following section provides an overview of software development and testing process in company. It is a corporate-level company and the number of projects include legacy or microservices individual teamwork. In the company, each project is developed applying a different technology. The team needs to follow some standard process, regulations, tools for software development and testing according to internally developed guidelines. In the organisation, the testing strategy is specified in documents on a very high level. In some cases, the author noticed that the testing strategy was in Excel document. In some projects, there was no testing strategy mentioned in the testing process or in the documents.

In the software testing process and testing strategy investigation process, several similar test management processes in the testing were identified. All standard testing process steps implemented in the companies are described in the following sub-sections.

Standard Test Process Steps

Test planning overview: Several sources maintain that executing the functional tests is the most significant part of the work a tester should be doing. It is partly true, as in many projects, quality assurance is a low priority task until problems start to arise, in which case the tester is already very late. In these kinds of projects, there is a sudden influx in employees focused on quality assurance at the end of the project. However, in a healthy quality assurance environment, it is recommended to spend more time on designing test cases, verifying product backlog items on testability and finally conducting the actual functional testing.

Start of the sprint around planning: In the agile scrum adopted projects, as functionalities are described in product backlog items nearing the end of the 'sprint', the quality assurance team (preferably lead) should verify that the product backlog items are testable during the 'sprint'. If requested, the quality assurance employee should be able to help the product owner in defining the product backlog items in such a way that the functionality is testable during the 'sprint'.

Designing test cases: At this point, the tester should know which product backlog items have been committed for this 'sprint' and thus have had the planning. The tester should already have some insight into the contents of different product backlog items.

The tester designs the test cases for different product backlog items, at this point, the tester might find out that some of the product backlog items are dependant (for testability) on other product backlog items. If this is the case, the tester should report this to the tester product owner and discuss how to continue the work.

Functional testing and defect: Once the tester finishes designing test cases for the coming 'sprint', the team will most likely be starting to finish up the first product backlog items, which the tester manually tests by hand. The test cases, which the tester has designed for each product backlog item beforehand, are now going to be used to verify the delivered functionality as expected. Of course, if there are defects carried over from the previous 'sprint', the team should prioritize finishing those first (unless otherwise instructed by the product owner). Retesting these defects, priority is ascribed by exploratory testing and by verifying the actual functionality with the requested in the defect.

If there are any defects with the delivered functionality of this sprint, they should be registered in the corresponding product backlog item on the team foundation server and mentioned in the next stand-up. A good practice is to talk with the team about the defect beforehand and say something about it at the daily stand-up meetings if the testing team practises such meetings.

According to [93] and [94], the tester either creates defect reports or manually tests new functionality. If the tester has some time left, it is always a good idea to do some exploratory testing to see if the tester has missed any small defects.

Test automation: The tester with the technical and programming knowledge knows how to program test automation (or has a developer who can help), the tester starts updating regression tests at this point. The tester has probably already considered which tests the tester would like to add to the regression set and which not. Now the tester definitely tests the regression, taking into account how much time the tester has for testing, and then expands the regression set.

In the test automation, the tester can automate tests in multiple teams or multiple projects implemented by the same team. Creating some automated scripts for one team alone costs more in comparison to having one person creating them for more teams as a tester. In this situation, the tester spends less time on duplicate code and trains many people in test automation.

Finishing the sprint: Nearing the end of the 'sprint', most of the product backlog items get finished and most of the work is still to be done refactoring and fixing defects. The tester does a lot of retesting and exploratory testing to ensure all defects are fixed and implemented correctly, and the tester has not missed any defects during the 'sprint'. If the tester is dedicated enough to have someone (or self-test) who has created some automated scripts, the tester runs them.

Defect registration: defect registration can be handled in quite a few different ways, each of which can be viable for the current testing situation. In the current situation, the team foundation server, Microsoft test manager or JIRA for this project, the tester or the team register the defects in Microsoft test manager using a simple naming convention for clarity.

This section has analysed how the quality assurance or a team member uses the same conventions and the same administration tools to communicate within the project.

10. Mathematics formulas

1) To find the variance.

$$s^2 = \frac{\sum (X - \overline{X})^2}{N - 1}$$

$$s = \sqrt{s^2} = \sqrt{\frac{\sum (X - \overline{X})^2}{N - 1}}$$

ACKNOWLEDGMENTS

I want to thank my advisor Professor, Dr.sc.ing. Marina Uhanova, who advised me on the challenging and vital topic of software test strategy creation in the software development process. It has helped me review my previous work and has motivated me to write the PhD Thesis on " Software Testing Strategy Utilizing Lean Canvas Model."

I am also thankful to Late Professor, Dr.habil.sc.ing. Leonīds Novickis, who introduced and motivated me to consider this topic.

I would like to thank Professor, Dr.habil.sc.ing. Jānis Grundspeņķis, Associate Professor, Dr. Aleksejs Jurenoks and Dr.habil.sc.ing. Jānis Grabis for overall academic and moral support provided within the PhD program in Computer Systems at Riga Technical University, Riga, Latvia. I am also thankful to all members of the academic university staff at my four-year PhD program who shared their knowledge with me to improve my competences.

I am obliged to my parents for having taught the importance of education since my childhood, especially my mother, who motivated me to always aim for more. Thanks to my wife and her parents, who have always backed me in difficult times.

Thanks to my sister and brother for eternally being there in critical moments, motivating to reach goals and showing sincere sympathy. I am thankful to my late father for creating the circumstances where my passion for information technologies could increase, while reasonable strictness continued to motivate me to investigate beyond computer games.

Finally, I like to thank the scientific community, contributors to my topic, reviewers, criticizers of my topic, office colleges, relatives, and friends.