Viktor Kubica

IMPLEMENTATION AND IMPROVEMENT OF SOFTWARE TESTING IN TELCO

Degree Programme in Information Technology
2019

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

IMPLEMENTATION AND IMPROVEMENT OF SOFTWARE TESTING IN TELCO

Kubica, Viktor
Satakunta University of Applied Sciences
Degree Programme in Information Technology
January 2019
Supervisor: Trast, Ismo and Aromaa, Juha
Number of pages: 50
Keywords: testing, test management, telco, test strategy, test plan

---

Software Testing plays a crucial part in the modern software development process. It is usually associated with the terms, ‘validation’ and ‘verification’. A fine line lies between testing too much which is not ideal from the business, cost and time perspective; and not testing enough which negatively affects the quality of the final product, ultimately resulting in higher costs (fixing on production cost more and has a direct impact on customers). Many factors must be taken into consideration when implementing effective test strategy. The purpose of this thesis is to show the application of specific test strategies usable in software testing.

This Master thesis provides an insight into the various phases of software testing including system, system integration, regression, user-acceptance, automation and performance testing. The way how and when the tests are executed depends on the software development type as well.

Table of Contents

Abbreviations

Bug - Flaw, Defect

CRM - Customer Relationship Management

BREQ - Business Requirement

Jmeter - Open Source Tool For Performance Testing

HLD - High Level Design

ISO - International Organization for Standardization

IT – Information Technology

KPI - Key Performance Indicator

LLD – Low Level Design

SIT - System Integration Test

SDLC – Software Development Life Cycle

PPT – Preproduction Environment

TAF - Test Automation Framework

TC – Test Case

TST – Test Environment

UAT - User Acceptance Test

1 INTRODUCTION

The main focus of this thesis is to describe the testing process of software solutions at O2 Slovakia and to provide option of how to improve the testing process or solve situations which have a direct qualitative effect.
Even though the testing process has a standardized structure, it consists and depends on several factors which may influence its execution and therefore – the result. This paper describes what is the main setting for the preservation of the standard software quality as well as internal and external factors which affect the outcome.

1.1 Background

This paper focuses on the company called 'O2 Slovakia s.r.o.' (limited) ("O2 Slovakia"). O2 Slovakia was sold to the Czech financial group PPF, so it is not currently affiliated with O2 International, a Telefonica subsidiary.
The company is one of the three main companies competing in the telecommunications industry in Slovak Republic. Currently, O2 Slovakia has many ongoing projects, such as development of a new CRM system, improving the quality and coverage of the 4G band while implementing new campaigns and features in existing systems. The company currently employs about 700 internal employees. However, many long-term employees are working as contractors or body-leasing.

1.2 Purpose and Objectives

The aim of this thesis is to improve the test strategy which can be applied for projects in ITC environment and review the current testing processes at O2 Slovakia. Consequently, it aims to address the issues which may result in improved effectivity of the test team and its stakeholders.

1.2.1 Importance of the thesis

- Relevance for the company:
    - Process optimization will result in better quality what leads to decreased costs
    - Other projects will benefit from the presented changes
    - Improvement in the performance of testing team and its stakeholders
- Relevance for the researcher:
    - Optimization leads to easier management of the team
    - Knowledge improvement
    - Getting overall feedback of the testing process from different stakeholders

1.3 Structure of the thesis

Chapter 1 contains the introduction of O2 Slovakia and a high-level focus of the thesis.

Chapter 2 presents the methodology of how the research was conducted.

Chapter 3 introduces different types of software development life cycle methods and compares them with the one used by O2. Even though~~,~~ the main focus is on the testing process, the other development methods? are necessary to cover too as they are mutually dependent.

Chapter 4 shows why a test strategy is needed and what are its main elements.

Chapter 5 reviews the elements and process in test strategy and introduces the changes/advices that should be carried out or were have already been implemented.

Chapter 6 summarizes the findings and concludes with the recommendation for the future.

2 RESEARCH METHODOLOGY

2.1 Action research

Action Research is chosen as the main type of the research of this study. Action Research was first mentioned at MIT in 1944 by a professor Kurt Lewin. Kurt Lewin described it as "a comparative research on the conditions and effects of various forms of social action and research leading to social action" (Lewin, 1946) which uses a repetition of steps, of which everyone is composed of a cycle of planning, execution and fact-finding about the results of the action.

As the aim of the thesis is ultimately to improve the already existing processes, the Action Research is best suited for this purpose. It can be viewed as a method of research process as well as methodological approach. The action research process implies that the subjects of research as well as the ’author’ are active participants in the research process. (Reason, 2008).

The Action research process is cyclical and includes continuous steps:

- planning
- action and observation
- evaluation
- reflection

The graphical representation is figure 2.1. The main advantage of the spiral model is that it shows the possibility of analyzing a certain issue in a greater depth each time. This results in higher level of problem understanding. However, as the individual stages may overlap, the initial plan may become obsolete during the process.

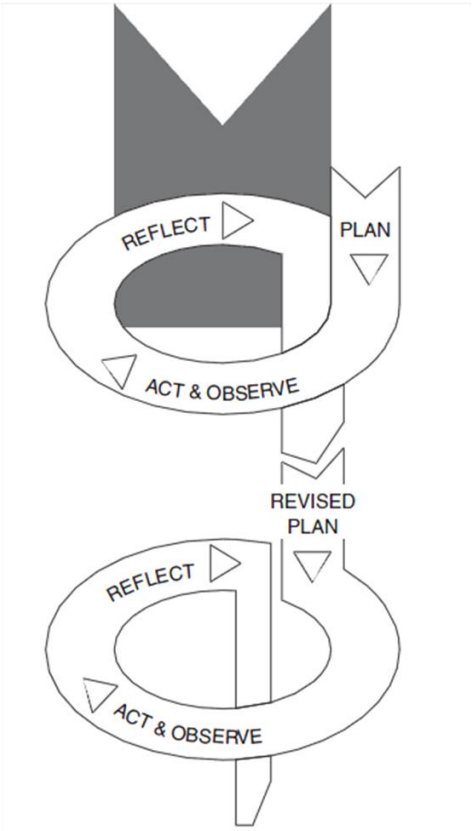

Figure 2.1 (Kemmis, S., & McTaggart, R. (2000))

2.2 Interview and Discussions

As an addition to the Action research, informal interviews and discussions were conducted. The interviewed colleagues were working on the same projects on various positions in development, testing and management. There were not separate questionnaires since the interviews occurred between the members of the same project and discussions had many different topics. A selection of /the most relevant of the gathered information is reflected in this thesis, specifically in the Testing Process chapter, where the proposed/executed changes are described.

3 SOFTWARE DEVELOPMENT MODELS

This section will cover the process of development and testing in following development processes.

3.1 Waterfall

Waterfall model is a software development process first introduced in 1970, published in a paper by Dr. Winston W. Royce. This model delivers the final product in a logical progression of steps of the Software development life cycle. The name of the model refers to cascading steps down an incremental waterfall. (Powell-Morse, 2016)

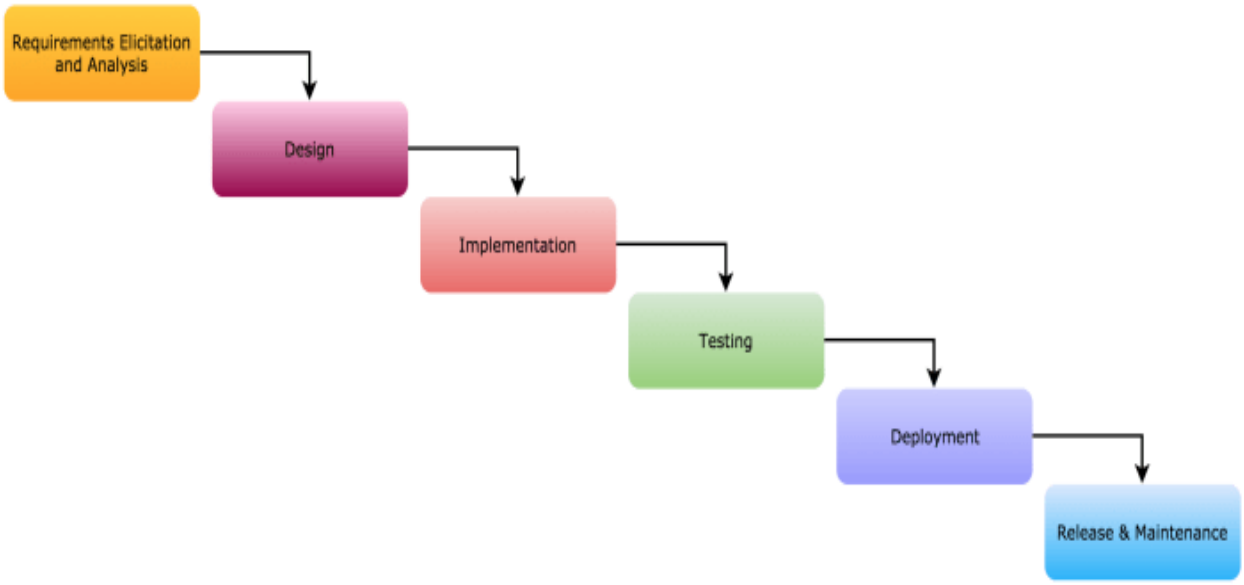

Figure 3.1 The Waterfall

Even though, the popularity among developers aims to agile methodology, the waterfall model remains and has place in the industry.

3.2 Waterfall Model Phases

As was mentioned before, waterfall model is sequential and has 6 phases. This means that the phases of the development process start only if the previous phase was completed and accepted.

3.2.1 Requirement Gathering and analysis

This phase focuses on writing down the business requirements of the final solution. These requirements are analyzed and written in a specification document that serves as a foundation for the next phases. The deliverable of this phase is usually called requirements document or business analysis.
(Software Testing Help, 2018)

In this phase, the testing does not need to be presented, however, in specific cases it may provide additional value. If testing does have knowledge of the desired processes, senior tester may provide valuable information or identify potential flaws. Moreover, test manager will be able to provide gross estimates of the testing phase if the testing is presented in the process.

3.2.2 Design

During this stage, the designers will analyze and design the desired functionality of the application. Technical design defines in what manner should the processes be implemented, the structure of the databases as well as integration agreements between different modules or applications. Deliverable of this phase is design document. The requirements from the previous step (analysis) should be linked to the way of implementation in design.
When the design is finished, the testing team can start to work on test strategy and test plan. First of all, the testers need to understand the design, therefore they need to read it thoroughly. Ideally, the project or test manager sets up a meeting where designers make a hand over of documentation to the test team. After that, the testers may start with test case preparation. Test cases should reflect design and its requirements. Even

though, there is a risk that design may not reflect the requirements from analysis, the testing team is not usually responsible for the correctness of a design. However, this risk should be reported handled by the project manager or customer who review and accept the design.
(Software Testing Help, 2018)

Test cases needs to be matched to the specific requirements, so the team will be able to confirm that the desired functionality will be covered and tested. For this purpose, the team can use traceability matrix. Example of traceability matrix:

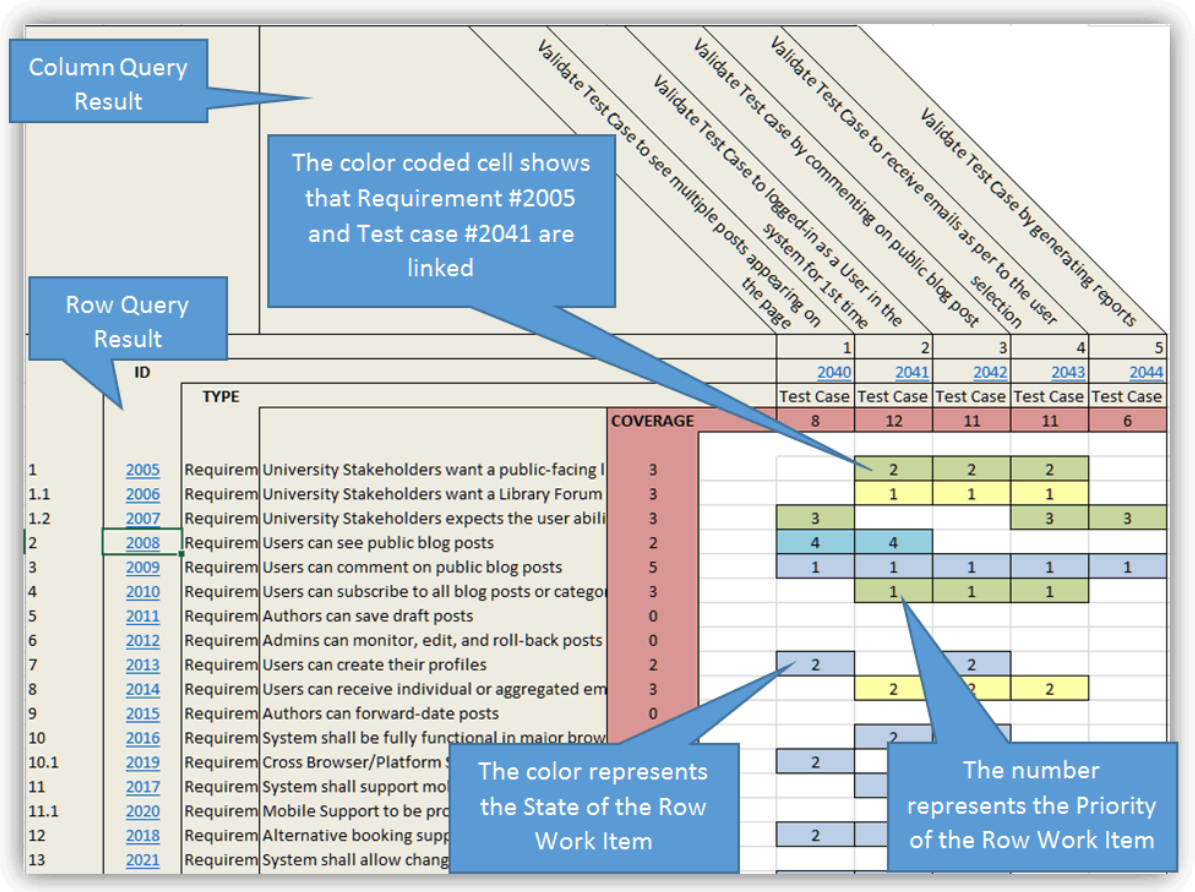

Figure 3.2.1 Example of Traceability Matrix (Modern Requirements, 2019)

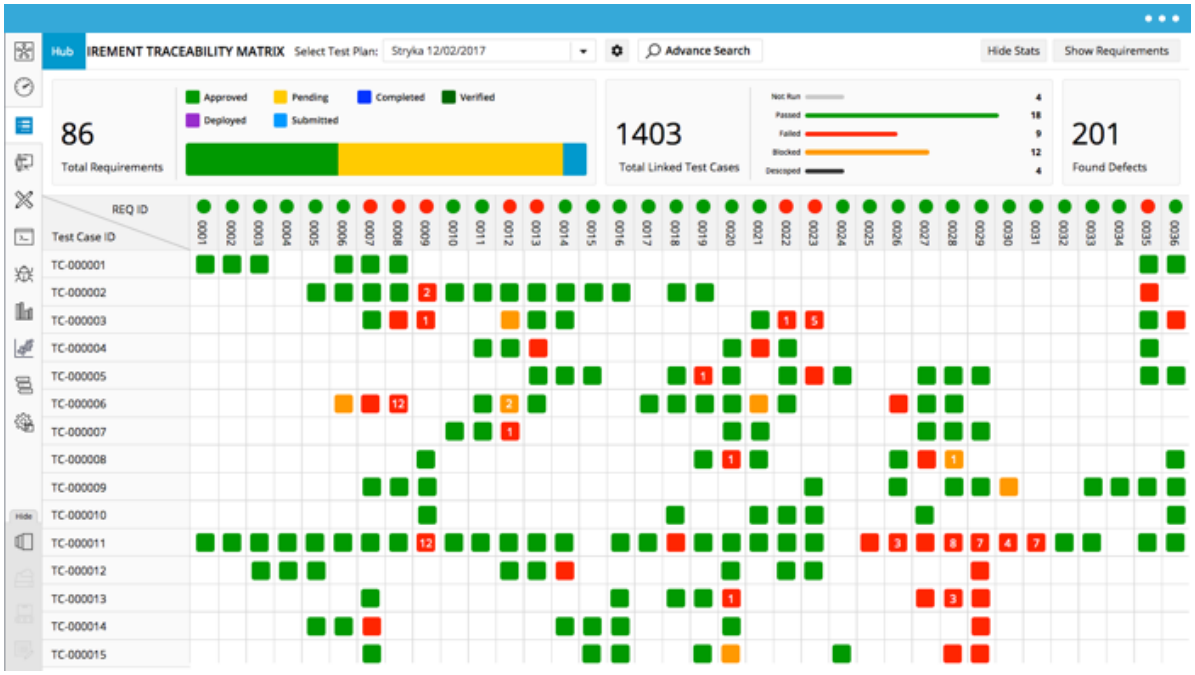

Figure 3.2.2 Example of Traceability Matrix (Remi, 2018)

In specific cases, the testers (or other team members) can start to look for the bugs already in the design. These bugs can be found by design review and their nature may be:

- Not understanding the process from analysis which may result in wrong implementation
- Forget/miss the impacts on the systems integrated with the application
- Incorrect/insufficient interface agreements

Testing should start to define strategy for the testing of no-functional requirements such as security or performance testing. These topics will be covered in the chapter ‘Test Strategy’.

3.2.3 Implementation

This phase takes the deliverables from the design phase. Development team starts to work on the implementation of the new requirements. Usually the application is developed in small programs (also called units) which are not integrated yet. (Sami, 2018)

Testing is not usually implemented in this stage; however, some organization may adopt unit testing. Unit tests are written by developers and is a method that verifies that individual units of code work correctly.

3.2.4 Testing

For the first time, the code is deployed on the test integrated environment where the testing team may start to run test cases defined in test plan. Even though the tests should be performed on integrated environment, the beginning of the tests may include only tests of certain part of systems. This may happen when the delivered system has many bugs that prevent end-to-end testing. (Sami, 2018)

Testing phase can be divided to these types of tests:

- System integration tests:
    - This type of test is performed by testing team, which verifies the whole functionality by performing end-to-end scenarios
    - Testers are reporting bugs which should be repaired according to their severity and priority
    - Low quality of the code may bring the development phase back
- User acceptance tests:
    - Performed by business or end users of application
    - Usually the test coverage is a subset of the system integration tests and covers the most critical processes
- Performance and Security tests
    - Non-functional tests which are performed by specialist

- Scope of these tests may vary accordingly to the scope and size of the project

3.2.5 Deployment

Deployment phase gets the approved code to the production. However, before the code is deployed, usually there is need to implement cut-over strategy. Cut-over strategy should cover at least these scenarios:

- Roll-back plan – is it possible to roll the system to previous version (unless the project is green field)? If the answer is yes, there may be a necessity to test this scenario
- Parallel run – This situation may occur when there is possibility of running old and new system in the same time for the purpose of comparative test
- Smoke tests after deployment – Before the new system will be revealed to the customers the functionality should be tested on the production environment as well.

After the cut-over, there may be necessity to perform few hot-fixes – this is usually referred as hot-run support.
(Sami, 2018)

3.2.6 Maintenance

There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.
(Sami, 2018)

3.2.7 Advantages and Disadvantages of Waterfall

Advantages	Disadvantages
• Scope is fixed, what allows early changes in business requirements and design	• Not very useful for large projects due to high costs of changes presented in later stages
• Simple and easily manageable stages • Documentation is well written	• Very difficult to go back to previous phase/stage
	• Bugs are found late in the process, what increase the costs of the project

(Powell-Morse, 2016)

3.3 V-model

V-model is an extension of the Waterfall model. However, the V-model implements a testing phase for every stage of the Software development life cycle. The reason for the introduction of validation phases is to prevent late discovery of the defects. Therefore, the process focuses on delivering desired functionality in every phase by preparing or executing tests.

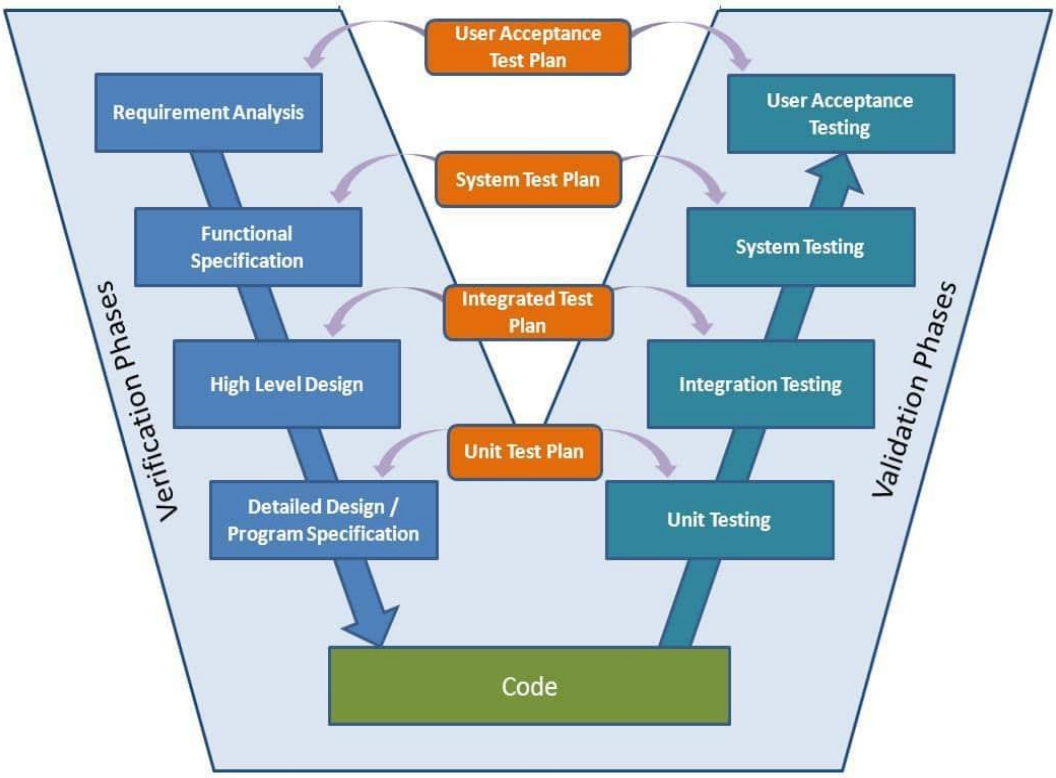

Figure3.3.1 V-Model (Software Testing Studio, 2017)

As we can see from the graph above, the phases of the Software development life cycle are similar to the waterfall, however the granularity of main stages (analysis, design, etc.…) is higher.

V-model is divided into two branches of the 'V' letter. One side of the model stands for the Verification phases and the second one for the Validation phases. These two branches are joined by the coding phase. (Software Testing Help, 2018)

3.4 V-Mode Verification Phases

3.4.1 Business Requirements Analysis

First phase of the development process focuses on the gathering of the customers' expectations. It involves details communication with the customer (internal/external)

so the business analyst would be able to define clear requirements which would meet client's expectations. (Software Testing Help, 2018)

The acceptance tests are written in this phase which usually contains happy-day scenarios.

3.4.2 System Design

The complete system is designed in this phase based on the product requirements. The system design solves and have the details for hardware configuration and communication setup for the developed system.

System Tests and plan are developed during this phase. Performing this activity earlies saves the time later for the execution of the test cases. (Software Testing Help, 2018)

3.4.3 Architectural Design

These phases break down the system design into High Level Design (HLD). The communication, data transfer, and other technicalities between modules are defined.

Testing team start and defines integration tests in this phase. (Software Testing Help, 2018)

3.4.4 Module Design

Detailed internal design of all the system parts is defined. Deliverable of this effort is usually called Low Level Design (LLD). One of the crucial parts of the LLD is compatibility with the other systems - internal as well as external. (Powell-Morse, 2016)

Unit tests can be designed in this stage.

3.4.5 Coding

The requirements are coded based on the design in the coding phase. The optimal programming language is chosen as well as the definition of coding guidelines and standards. (Software Testing Help, 2018)

The code is stored in the versioning system, so it can be deployed on the non-production/production environments in next phases.

3.5 Validation Phases

Right side of the V-model is represented by validation phases, where the product is tested by following techniques:

3.5.1 Unit Testing

In this phase, all of the unit tests are executed. Unit tests are white box technique, what means that the creator of test has understating and visibility of the code. This test consists of a piece of code which runs a method to test if the method (or any part of the code) returns expected value. This testing is usually performed by the development team. (Powell-Morse, 2016)

3.5.2 Integration Testing

Tests written in the Architectural design are executed in this phase. Purpose of this phase is to validate whether the components of the application are working or communicating correctly. (Powell-Morse, 2016)

3.5.3 System Testing

During this phases tester are executing system tests as well as non-functional tests. In other words, the newly delivered application is integrated with the whole system, where the end-to-end tests are validating new as well as existing functionality. The application is tested on the integrated environment which is close to the production one. (Powell-Morse, 2016)

Defects from this phase are logged into bug-tracking system. High defect ratio may result into repetition of different scenarios for several time. Report or output of the tests should be delivered frequently as this phase is usually the most challenging.

3.5.4 Acceptance Testing

Acceptance tests (or user acceptance tests) are executed by business users. Its purpose is to validate if the delivered system is in compliance with the business requirements. The scope of the tests is usually combination of the real-life user end-to-end scenarios with the subset of the system tests. (Powell-Morse, 2016)

3.5.5 Advantages and Disadvantages of V-Model

Advantages	Disadvantages
Development and process is organized and systematic	Not ideal for complex projects
Testing is involved early in the development process	Not suitable if the requirements cannot be completed upfront
Phases are easily manageable with clear criteria	High risk and uncertainty

(Powell-Morse, 2016)

3.6 Agile

Agile methodology was founded on 2001 when several technologists drafted the Agile Manifesto. The Manifesto has four major principles for developing better software:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

There are many variations of how the agile methodology can be applied. Scrum, Lean and Kanban or Extreme Programming are the most common, however, there are many others such as Crystal or Feature Driven Development.

As O2 Slovakia picks and uses only certain parts of agile, the paper will cover only basics of Scrum software development.

Scrum is a framework for delivering iterative and incremental projects of all type. For the last decade, Scrum achieved increased popularity in the agile software development because of its ease of use and productivity. CollabNet (2018)

3.6.1 Key features of Scrum

Scrum supports holistic product development where a team works to reach a common goal while promoting:

- Challenges traditional sequential approach, what means that multiple phases of development can be done simultaneously
- Teams self-organize themselves, while promoting physical co-location (or close online collaboration at least)
- Frequent feedback from the business owner about the product and specification

3.6.2 Scrum Roles

1. Development Team – The team that is responsible for delivering the product. The team consists from multiple team members who execute all of the product development phases including analysis, design, coding, testing, etc.… In scrum team, every team member is associated as ‘developer’ despite the fact that he or she may have different role to the actual coding of the product.

2. Scrum Master is the person who is responsible for proper usage of Scrum framework within the team. Moreover, scrum master takes responsibility for removing impediments which the team encounters while delivering the product goals. However, the scrum master does not have a role of traditional team leader or project manager as he or she acts as a buffer between the team and its impediments.
   Schwaber, K. & Sutherland, J. (2018)
   Core responsibilities:
   - Facilitating the scrum within the team what includes:
     - Organizing team events
     - Education of stakeholders in the Scrum principles
     - Helping the team to remove or avoid potential risks, impediments
   - Helping the product owner with the maintenance of the backlog. Scrum muster ensures that the requirements from the backlog are understood and clear for the development team.
3. Product Owner is a representation of the product’s stakeholders. He or she is accountable for backlog, therefore the product owner creates the product roadmap and prioritize what the delivery team delivers to the business. The product owner defines the product features, in Scrum called ‘user stories’, which are added to the backlog. This role focuses on the business side of the development process, therefore he or she is not responsible for technical implementation. The role of the product owner should not be combined with the scrum master. Schwaber, K. & Sutherland, J. (2018)

Core Responsibilities:

- Demonstration of the delivered solution to the key stakeholders

- Defines the product release and prioritize the user stories

3.6.3 Scrum Framework

Scrum has defined events which should be happening on regular basis. These events should minimize the need of the other meetings. All events are time-boxed, so the event must not take more time that is scheduled for it. The Scrum contains these events:

- The Sprint – it is the core event of the Scrum. The goal of the Sprint is to deliver the product or its usable increment by completing tasks, usually called ‘Stories’. The length can vary from one to four weeks, but the most usual length is two weeks. A new sprint starts immediately after the previous one was finished.

  Sprint consist, besides developers work, of Sprint Planning, Daily Scrum, the Sprint Review and Retrospective.
  During the Sprint:
    - Quality standards do not decrease
    - Scope can be adjusted only after the team discovers new impediments
    - No changes that would endanger Sprint Goal can be made

  Cancellation of the sprint can be performed only by Product Owner under specific circumstances, such as:
    - Spring Goal becomes obsolete
    - Change of the technology

  The cancellation occurs rarely due to the short durations of sprints, its negative effect on the team and causes a waste of resources.

- Sprint Planning is where the work for the next Sprint is defined. The Scrum Master takes care of that the event happens and takes maximum of 8 hours for a month Sprint. The plan is created by the participation of the whole Scrum Team. (Scrum Inc, 2018)

The Sprint Planning focuses on answering the two questions:

- What functionality be delivered in next Sprint?

- How the team can achieve the goal?

Topic One: What functionality be delivered in next Sprint?

The Product Owner and the Development Team discusses the objective of the Sprint or the product itself. The outcome of these discussions should be the Product Backlog, what is a list of desired functionalities. The Development Team forecasts each backlog functionality. However, the team decides which items from backlog are selected for the Sprint as they know which task would accomplish the goal. The Product prioritize what features from business point of view should be selected.

Topic Two: How the team can achieve the goal?

- Sprint Review occurs in the end of the Sprint to inspect the completed Stories and adapt the product backlog if necessary. During the Sprint Review, the Team and the stakeholders collaborate about the way the stories were done in the Sprint. The increment is usually presented so the stakeholders can evaluate if the requirements have been met. (Scrum.org 2018)
    - Sprint review includes:
        - Product Owner what stories from the backlog have been completed and what (if) some issues remained open
        - The Development Team address what went well and what problems emerged and how those problem were solved during the Sprint.
        - Demonstration of completed task by Development Team

    The result of the Review is a revised Backlog that shows how the next Sprint may look like. Moreover, it is a good opportunity to adjust the backlog if the new opportunities emerge or change of scope occurs.
- Daily Scrum is a 15 minutes time-boxed meeting for a development team. The event occurs every day of the Sprint. The main purpose of the meeting is to:
    - Review the work for the past 24 hours
    - Forecast upcoming work

- Discover impediments Scrum.org (2018).

The team can decide how to organize the meeting; however, the usual way is that every member of development team answers these three questions:

- What did I do yesterday that help the Development Team meet the Sprint Goal?
- What will I do today to help the Development Team meet the Sprint Goal?
- Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?

- Sprint Retrospective occurs between the Sprint Review and the Sprint Planning of the next Sprint. This is the ‘meeting’ where every team member can address their issues and talk about them.
  - The Retrospective involves:
    - What went well
    - What the team can improve
    - What and how the team will make improvements in the next Sprint

The ultimate goal of Retrospective is to collect feedback and use it identify possible improvements which can be implemented in next Sprint. (Scrum Inc, 2018)

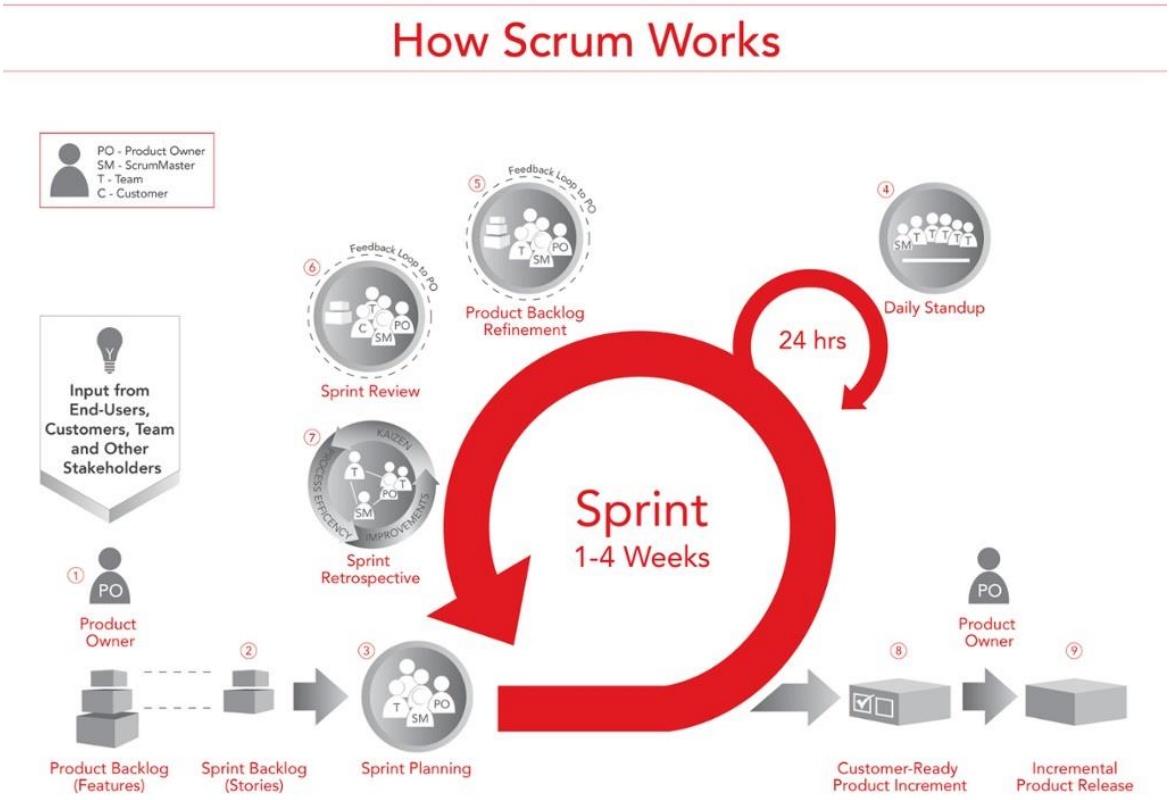

Figure 3.6.1. SCRUM SDLC (Scrum Inc., 2018)

3.6.4 Advantages and Disadvantages of Agile

Advantages	Disadvantages
Fast responding to change	Flexibility can lead to more problems when the process is not followed
Accepting uncertainty	Hard to implement to existing organizations
Greater flexibility	Less predictable what will be delivered at the end

3.7 O2 Software development life cycle

While O2 Slovakia does not strictly follow one software development strategy, it takes advantage of using combination of several strategies. The process is shown on the figure below.

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		**OZO-201804**	**95 days**	**Tue 1.5.18**	**Mon 10.9.1**	
2		**Business Analysis**	**50 days**	**Tue 1.5.18**	**Mon 9.7.18**	
3		S	50 days	Tue 1.5.18	Mon 9.7.18	
4		M	30 days	Tue 1.5.18	Mon 11.6.18	
5		XXL	10 days	Tue 1.5.18	Mon 14.5.18	
6		**Design**	**50 days**	**Tue 15.5.18**	**Mon 23.7.18**	
7		S	10 days	Tue 10.7.18	Mon 23.7.18	3
8		M	10 days	Tue 12.6.18	Mon 25.6.18	4
9		XXL	20 days	Tue 15.5.18	Mon 11.6.18	5
10		**Development**	**26 days**	**Tue 1.5.18**	**Tue 5.6.18**	
11		**Testing**	**65 days**	**Tue 12.6.18**	**Mon 10.9.18**	
12		Develop integration test plans using product specifications	15 days	Tue 12.6.18	Mon 2.7.18	4
13		Integration Testing	12 days	Mon 6.8.18	Tue 21.8.18	
14		UAT Testing	6 days	Fri 17.8.18	Fri 24.8.18	
15		Regression & Performance Testing	2 days	Tue 4.9.18	Wed 5.9.18	14
16		**Deployment**	**2 days**	**Fri 7.9.18**	**Mon 10.9.18**	
17		**OZO-201805**	**110 days?**	**Tue 12.6.18**	**Mon 12.11.1**	
18		**Business Analysis**	**75 days**	**Tue 12.6.18**	**Mon 24.9.18**	
19		S	75 days	Tue 12.6.18	Mon 24.9.18	4
20		M	55 days	Tue 12.6.18	Mon 27.8.18	4
21		XXL	35 days	Tue 12.6.18	Mon 30.7.18	4
22		**Design**	**50 days**	**Tue 31.7.18**	**Mon 8.10.18**	
23		S	10 days	Tue 25.9.18	Mon 8.10.18	19
24		M	10 days	Tue 28.8.18	Mon 10.9.18	20
25		XXL	20 days	Tue 31.7.18	Mon 27.8.18	21
26		**Development**			Fri 28.9.18	
27		**Testing**	**43 days**	**Tue 11.9.18**	**Thu 8.11.18**	
28		Develop integration test pl	10 days	Tue 11.9.18	Mon 24.9.18	24
29		Integration Testing	20 days	Mon 1.10.18	Fri 26.10.18	
30		UAT Testing	6 days	Tue 23.10.18	Tue 30.10.18	
31		Regression & Performance T	7 days	Wed 31.10.1	Thu 8.11.18	
32		**Deployment**	2 days	Fri 9.11.18	Mon 12.11.1	

Figure 3.7 O2 Software development life cycle

As we can see, the O2 has similar process steps as the Waterfall or the V-model. The project plan shows the high-level time plan for two releases: OZO-201804 and OZO-201805. The main difference is that the process phases are not strictly sequential. Some of the phases are running simultaneously.

Every release delivers multiple change requests which need to be delivered in given time frame. Change requests with higher time estimate would be delivered as separate project with their own project plan.

3.7.1 Business Analysis

Business Analysis is divided into three categories – S, M and XXL. Those categories differentiate in the amount of effort needed for the completion of the task. The most difficult change requests require to be finished among the first.

Usually, the analyst is working already on different release that is being tested, however, he or she needs to have a dedicated time frame for the supporting of the current release.
Business analysis is written in ‘BREQs’ what stands for business requirements.

3.7.2 Design

The same rules apply for the design, so it is divided into the same categories as the business analysis. Design transforms BREQs into functional units and groups them into features. Features can be developed independently, so the development does not have to wait

3.7.3 Development

Development starts as soon as some parts of the design are finished. If possible, BREQs are grouped into ‘features’ which represent stand-alone part of the functionality.

3.7.4 Testing

Testing starts as soon as the change request is developed and deployed on test environment. Prior to the testing, the test engineer needs to analyze the change request and prepare test cases.

Testing may overlap with the development. Usually, at least few change requests are planned to finish before the official end of the development.

3.7.5 Advantages and Disadvantages of O2 Slovakia model

Advantages	Disadvantages
Enables to effectively allocate resources	Documentation – as designs are written as a delta compared to the current functionality.
Delivery is swifter comparing to waterfall or v-model	Overlapping of development and testing leads to decrease in quality and instability of test environment
Phases are easily manageable with clear criteria	

4 TEST STRATEGY AND TEST PLAN

4.1 General

The test strategy describes general approach to test methodology used in the organization. It includes the way of how testing is used to mitigate project risks, manage product, and divide testing into different levels and the high-level activities done or associated with testing. (ISTQB, 2012)

There may be multiple test strategies in the organization. It usually happens when there is a change in software delivery, the level or nature of the risk changes or a new project that has much bigger scale than the usual life-cycle occurs. Moreover, there may be different test strategy for the long-term and short-term goals.

Test Strategy should cover:

- Entry and exit criteria for various types of tests
- Types of the tests
- Test Environments
- Test Tools and Reporting
- Defect Management
- Roles and Responsibilities
- Test measurements and metrics
- KPI's

Typical test strategy methodologies include:

- Analytical strategies – what usually includes risk-based testing. The aim is to identify the risk, asses to what degree may occur and consequently create a plan and manage the risk. As an example, we can take requirements-based testing, when the analysis and test cases are derived from the technical design (or other requirements). Those tests are executed, usually by the business priority, and the status is tracked according to their status (passed, failed, blocked, etc...)
- Model-based strategies are strategies where "the test team develops a model (based on actual or anticipated situations) of the environment in which the

system exists, the inputs and conditions to which the system is subjected, and how the system should behave." (ISTQB, 2012) This strategy is usually used for the performance tests, where the model can be developed for hitting specific throughput rates (minimum, maximum...), response times, network traffic and so on. Moreover, the model can be adjusted according to specific hardware or software setting of the environment.

- Methodical strategies are used when the test team works according to predetermined quality standards. This situation is usually associated with the change of specific certificates (ISO), checklists or a specific set of conditions related to the industry or the domain. These tests can range from the simple ones (checklists for verifying the most important processes in the maintenance testing) to the more complex ones (security testing)
- Process strategies describes what steps or phases should be followed by the test team. These processes are either defined by specific methodology, committee or any other authority such as government officials or organizations. Example of the process can by analyzing user stories in Scrum, where the tester identifies what parts of the application needs to test, estimate the effort and execute prepared test scenarios for each user story. Each of the phase should be covered in process strategy – for example estimation have specific formula which can be used generally for every user story.
- Reactive strategy or approach is best shown on the example of exploratory testing where the aim is to find as many defects as possible in predefined period. These tests have a light structure of the test scope. Each tester is assigned with the set of guidelines which helps him with the structure of exploratory session. The results are usually reported directly to the Test Lead or Test Manager who directly manages the process.
- Consultative strategies are used in specific situations where a stakeholder sets the condition for the tests. This strategy usually tests functionality horizontally. For example, web application could be tested on different versions of browser, compatibility of third-party applications (antivirus SW), different operating systems or configuration options. Pairwise testing (verifying all possible combinations) or equivalence partitioning (dividing possible values into logical sets) can be used. (Guru99, 2019)

- Regression-averse testing uses functional and non-functional tests to mitigate the risk of decreasing the quality during regression tests. Automation regression tests should cover stable functionality which is critical for the business in real-life scenarios. Therefore, these tests should be run after every deployment. (ISTQB, 2012)

4.2 Test Plan

O2 Slovakia has one general test strategy that is widely applicable to the ongoing as well as the new projects. The ownership of the test strategy belongs to the Test Manager, however, there are many stakeholders including departments through the whole organization.

As the test strategy is a general document, test plans are created for specific purposes when the needs arise. Test plans covers all the testing work that would be done for a specific project. Test plan should be aligned with the organization's test strategy, however, if there are specific areas where it is not, test plan should explain the exception and its effect. (ISTQB, 2012)
Moreover, test plan should cover at least:

- Test schedule and execution of the tests
- What is in/out of scope testing
- Entry and exit criteria for each test type (including deliverables)
- Responsibilities and escalation matrix
- Risks defined by test team

Test plan should go much deeper than test strategy in specific areas, such as level of the tests. System tests can be shown as an example. Implementation project of the new billing system is divided into several testing periods, where system tests have higher coverage as in usual releases. These tests should be divided into smaller test plans based on the application. System test of new billing system consist of test of Product Catalogue, Rating, Provisioning, Migration and Billing. System Integration Test plan would include combination if systems integrated in the environment.

5 TESTING PROCESS

This chapter will focus on the specific phases of the test strategy. The report covers only the phases which were discussed and identified as the ones that should be reviewed. The research was conducted during the year 2018. Some processes were already able to implement proposed solutions, while other solutions could be only recommended or scheduled for a later review/implementation.

5.1 Test Analysis

Even though in some cases it makes sense to divide test analysis and design into separate phases, O2 have only one phase that includes analysis together with design. During the test analysis, design and business analysis are analyzed by a member of the test team. The aim is to identify testable features, preconditions and ultimately create a set of test cases which need to be executed for the verification of the functionality for the change request. (ISTQB, 2018)

Findings and ideas for improvement:

This process was already working sufficiently for several years; however, the development of the new CRM added ~~a~~ further technological and usability complexity. This complexity resulted in:

- Design became more complex and, in some cases, too technical for reading
- Test Cases needed more time to acquire the same level of quality as with the older CRM
- It was harder for test analysts to identify all scenarios and its criticality
- Design does not provide relation between mutually-affected change requests

We identified several possibilities which would remedy the situation described in the above points.

The first recommendation was to add an official test analysis review with the designer early in the process. In some cases, the business owner is presented as well to provide his or her view of the newly required functionality. This enabled the team

to prioritize the most important features of the change request, which resulted in earlier discovery of defects.

Test Analysis is used in different degree in all of the test strategy methodologies. The most important part is to identify the biggest risks and cover critical scenarios. Analysis with identified problems help the project to mitigate the risks as management can plan and take necessary steps. During testing the risks and issues are monitored, so the result is either fixed issues or a time for preparation of alternative plan or creation of workaround.

5.2 Unit Test

Unit tests are created by developers and executed automatically on each committed change of a unit under test. Additionally, developers will execute unit tests on demand.

The purpose of unit tests is testing specific points in the code for a specific reason. While code coverage of system tests should be as high as reasonable, code coverage of unit tests is irrelevant. Unit tests should be as independent as possible of other components. They must not contain any I/O operations. For server code they should be Java only.

Typical use of unit tests is for a complicated algorithm or for a specific bug to ensure correct propagation of fix through environments.

Unit tests must be side-effect free.

Findings and improvement ideas:

O2 Slovakia does not have set minimum coverage of the code by unit tests, therefore developers are not implementing them thoroughly. We identified that this leads to:

- Higher instability of non-production environments after deployment
- Critical scenarios for newly implemented change requests took longer time to pass

- New developers can cause unwanted changes in the legacy code

The main problem is that without requirement of writing unit tests, only few developers are doing it, when they can. Improvement of the situation does not have a quick and easy solution.
The proposed solution is to divide tasks into short, mid and long-term. Shor-term solutions should be completed in 1-3 months, mid in 3-6 and long in 6-16 months.

Short/Mid/Long Term	**Task**	**Responsibility**
Short	Define the standard for unit test	Tech/Lead of Development
Short	Present and discuss feasibility of adding unit tests in the processes for the new and legacy code	Tech/Lead of Development
Mid	Cover at least 30% of the new code with unit tests	Development team
Long	Cover at least 70% of the new code with unit tests	Development team

5.3 System Test

System tests are executed by testers and developers throughout the development before delivering the functionality to the System Integration Tests. Tests are executed on the integrated test environment.

System tests can be executed on any work packages – those work packages are selected by Test Manager and agreed by Project Manager and Lead Developer.

The purpose of the tests and test activities is to:

- Perform functional and non-functional tests to catch the bugs early in the development
- Measure the quality of the delivery

Entry criteria:

- Deployed functionality on the test environment
- Necessary documentation and design
- Test cases and test analysis does not need to be completely finished

Acceptance criteria:

- Developer is responsible for testing of basic scenarios (happy day) which is confirmed by SW delivery for system integration test
- System Tests are finished when the delivery is deployed to the test environment for the SIT tests.

Findings and improvement ideas:

Unfortunately, due to insufficient time during development, a limited number of tests become executed. Consequence of not running even a basic scenario may result in:

- Extension of the system integration test tests for achieving required quality
- Compromise in quality deployed to the production environment

Mitigation of these problems can be solved by:

- Increasing the quality of delivery
- Extending the time for the development
- Prioritizing passing of basic scenarios as an entry criterion for system integration tests

System tests take most of the value from analytical strategies. As the main purpose of the system test is to verify the basic functionality of the change request. Identifying and executing the critical business scenarios early in the process provides information about the stability of the delivery. Moreover, the tester can verify if the aim of the delivery meets the business requirements.

5.4 System Integration Test (SIT)

The goal of the system integration test is to verify the functionality on the integrated environment to ensure its stability and reliability with other systems. The tests will be executed by technical testers on the integrated legacy TST environment. SIT tests will be aligned with the SIT tests of the legacy releases. Business testers can be presented during the SIT phase.

Role of the Business Testers during the SIT:

- Getting familiar with the functionality
- Adjusting test cases for the UAT
- Executing UAT scenarios already in SIT phases for CRs with huge impact. This process would be allowed only for already tested and passed scenarios with full support of technical testers

The purpose of the tests and test activities:

- Perform functional and non-functional tests to catch the bugs early in the testing
- Verify that the delivered functionality works correctly and is integrated well with other systems

Entry criteria:

- Deployed functionality on the legacy test environment
- Necessary documentation and design
- Closed development tasks which need to be tested are assigned to the Test Manager
- Test analysis and test cases are prepared

Acceptance criteria:

- Execution of all test cases
- No critical bug is open
- Defined Pre-UAT test scenarios successfully passed

Findings and improvement ideas:

Experiences gained from lessons learned from releases 2018 showed that System Integration Tests benefits from the following test strategies:

- Analytical strategy - as SIT are following system tests, they need to incorporate critical scenarios but include their derivation as well. The design provides information about what should be developed, therefore, analytical strategy covers that part.

- Methodical strategy - as telco is regulated industry, there are change requests based on the new law and regulations. For example, European Union pushes roaming regulations which changes the tariffs. These changes are usually consulted with legal department which provide 'checklist' for the IT.

- Regression strategy - change requests often modify the functionality that is not explicitly mentioned in the design, therefore it is beneficial to include test cases which verifies functionality tied with new change request. If possible, use of automation regression scripts is highly recommended.

- Reactive strategy - Exploratory testing is beneficial as it opens and questions new possibilities of how the product is used. This strategy uncovers the situations which could be forgotten during previous phases. Exploratory testing should be time boxed to 90 minutes; however, the session may by extended or reduced by 45 minutes in specific cases. (Guru99, 2019)

5.5 Regression Test

The goal of the regression tests is to verify if the new functionality tested in the SIT is correctly integrated and do not cause bugs in the already existing functionality.

Regressions tests are performed by technical testers.

Entry Criteria:

- UAT is finished
- Release is correctly built and deployed on pre-production integrated environment
- Prepared Test Cases and Test Plan

Findings and improvement ideas:

Regression testing provides best results combining these two strategies:

- Regression-averse strategy focuses in this case on verifying the critical functionality regardless of the changes made in the code. The regression tests are in most cases the best candidates for automation. Processes are usually stable, and the changes are mostly cosmetic. Even though, it would be ideal to automate all tests, the reality is different. Maintaining huge catalog of automation tests takes a lot of time, resources. There are always some specific scenarios which need some sort of manual tweaking, therefore, it is needed to choose optimal ration for automated and regression tests.

- Analytical strategy is beneficial when the regression of the new release happens. The tester must verify the new functionality; therefore, it is needed to add or replace test existing scenarios.

5.6 UAT Test

The goal of the UAT is to establish confidence in the system and should focus mainly on the functionality of the system. UAT are executed according to the prepared test scenarios and their exploratory derivations. Test Cases are defined and written by Business Tester (owner).

Creation of the test cases should be done in two phases:

1. Business Tester should define main test cases (happy path scenarios) in the phase of writing business analysis and design. Test Case should be understandable from its name; however, it may contain some additional information in the summary. Those test cases will be executed in Pre-UAT.

2. Before the start of the UAT, Business Tester should finalize test cases and prepare or request specific test data from technical tester who prepare them.

Entry criteria:

- Deployed functionality on the legacy test environment
- Necessary documentation and design stored in Jira and Confluence
- Created Test Plan with written Test Cases
- Test data ready
- UAT can start upon successfully finished passed SIT and Pre-UAT
- No critical bug is open
- Major and Minor bugs will be reviewed on Acceptance Meeting and evaluated which will be fixed in current version or moved to the next release
- Test cases will be stored in Jira in UAT project

Findings and improvement ideas:
After the evaluation of defects from several releases we found out that some processes had slightly different implementation that the business was expecting. There are several root causes of these problems as miscommunication, different versions of mock-ups, and bugs in assignment or politics. To mitigate this outcome from testing perspective we improved the process:

- Pre-UAT scenarios: Pre-UAT scenarios are defined by the business user and should cover basic functionality of the WP (happy day scenarios).

Business Tester will have the responsibility for the execution of the Pre-UAT test cases. Legacy Checks, test data, review of test cases and consultations will be provided by the responsible tester defined before the UAT. This phase occurs during the SIT phase when the change request is appropriate quality.

- Review of the test cases: In some cases, it does make sense to check SIT test cases with business user.

Optimal strategies for UAT tests:

- Analytical strategy - as SIT are following system tests, they need to incorporate critical scenarios but include their derivation as well. The design

provides information about what should be developed, therefore, analytical strategy covers that part.

- Reactive strategy – Exploratory testing benefits described in test strategies for SIT applies for the UAT as well, however, UAT tester can go deeper because they have business view. In telco industry business testers have experience from the stores, so they have information or experience how the system is used in real-life scenarios.

5.7 Performance Test

Performance tests are done internally by O2 testers. They are executed on preproduction environment as it is the one that has the closest HW a SW setup to the production environment.

KPIs: Defined separately for each process.

Every release contains at least three test runs for verifying the functionality.

Test Cases:
We already have existing set of test cases which covers current functionality, however, editing and creation of the new test cases is done every release according to the new functionality. Test Manager has the full responsibility for Performance tests and preparation of Test plan for these tests. The tests will be executed on Preproduction environment in order to verify performance requirements. Test Manager has the responsibility for preparation of the Preproduction environment. By the beginning of the tests, Preproduction environment must be prepared and fully integrated and all legacy systems must have implemented the changes described by integration agreements. The Preproduction environment must also contain test data. Performance is finished once the acceptance criteria are met. Scripts for performance are stored in version control.

5.7.1 Performance test approach

Initial runs of Performance tests are performed at the end of SIT/beginning of UAT (due to debugging) on TST environment without impact on UAT tests. Other runs are executed during regression tests on PPT environment without impact on regression tests to verify if the requirements are fulfilled and to find out performance and limitations of the solution (e.g. maximum number of concurrent users etc.). Performance test is part of an acceptance criteria.

Performance tests:

- Performance Tests Plan is prepared by Test Manager
- Scripts for performance tests are prepared by Test automation engineer
- Tests are executed in several runs. Found errors should be fixed and new version of the application prepared for the next run. The final run is executed after completion of the regression tests
- Exact timing for these tests is specified per project
- Portlets are loaded within 3s(5 s)

Entry criteria for start of test:

- SIT is finished.
- Installation on Preproduction environment is done.
- Test documentations (Test Strategy, Test Plan and test scripts) are pre-pared.
- Test data are prepared.

No open critical and other faults from SIT, UAT and Migration tests, which blocks execution of tests.

Findings and improvement ideas:
Current coverage, tools and the time of implementing new changes is sufficient, however, we feel that in some cases the performance tests are run too late in the

process. The root cause of the problem is the environment. Currently, we have two integrated environments for testing purposes.

TST environment is integrated environment dedicated for SIT and UAT tests.
PPT environment is integrated environment for regression and performance tests. Hot-fixes and small releases are tested only on preproduction environment.

The performance tests were refactored and currently have relative links so they can be executed on any environment. Measuring and retrieving relevant data on TST environment have two major obstacles. The hardware is not powerful enough and the data and content are not updated regularly. Even though the tests can be executed on TST environment in limited mode, the upgrade of TST environment would be beneficial as the performance tests would provide the relevant results sooner.

5.8 Automation

Basic idea of Test automation framework (TAF) is to be easy to use, well documented and maintainable, supports a consistent approach to automating tests. In order to establish an easy to use and maintainable TAF, the following must be done:

- The test reports should provide information (pass/fail/error/not run/… etc.) about the quality of the tested application
- Reporting should provide the information for the involved testers, test managers, developers, project managers and other stakeholders to obtain an overview of the quality

In addition to the test execution and logging, the TAF must provide an easy way to troubleshoot failing tests.

- Address the test environment appropriately - Test tools are dependent upon consistency in the test environment. Having a dedicated test environment in time is necessary in automated testing. If there is no control of the test environment and test data, the setup for tests may not meet the requirements for test execution and it is likely to produce false execution results.
- Document the automated test cases - The goals for test automation must be clear, e.g., which parts of application are to be tested, to what degree, and which

attributes are to be tested (functional and non-functional). This must be clearly described and documented.

- Trace the automated test - TAF shall support tracing for the test automation engineer to trace individual steps to test cases.
- Easy maintenance: Ideally, the automated test cases should be easily maintained so that maintenance will not consume a significant part of the test automation effort. In addition, the maintenance effort needs to be in proportion to the scale of the changes made to the SUT.

To do this, the cases must be easily to analyze, change and expand.

We divide automation tests into these categories:

1. Smoke tests - A subset of all defined/planned test cases that cover the main functionality of a component or system, to ascertaining that the most crucial functions of a program work, but not bothering with finer details. They should run after every installation of software on test environments (TST/PPT).

2. Regression tests - Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the soft-ware or its environment is changed.

Definition of test cases for Smoke tests and Regression tests will be discussed with business owner of O4R deputies. All test cases will be part of Test plan.

3. Brand Monitoring – Subset of the Smoke tests executed on dedicated computers located in different brand stores around Slovakia. Provides real-time data about availability, response times and functionality.

5.9 Security

Technical tester has the full responsibility for security tests and preparation of Test plan for these tests. The tests will be executed on non-production environment in order to verify the security requirements. By the beginning of the tests, the environment must be prepared and fully integrated and all legacy systems must have implemented the

changes described in design or integration agreements. The environment must also contain test data.

Security test approach

- Security tests will be prepared and executed by Technical Tester
- Tests are executed in several runs
- Exact timing specified in project plan

Entry criteria for start of test:

- SIT is executed
- Installation on Preproduction environment is done
- Test documentations (Test Strategy, Test Plan and Test Analyze with test scenarios) are prepared
- Test data are prepared
- No open critical defects from SIT and Migration tests, which blocks execution of tests

Assumptions Security tests

The Performance and Security tests can be executed on the same environment as SIT or UAT. Test Manager is responsible for preparation of the test environment and test tools. The environment used for performance and security tests should be of the same specification, sizing and configuration as the production environment in order to be the measurement valid.
Tests may be repeated on any party request in order to test new releases for bugs or identify causes of problems that could not be identified in later analysis. Tests will be repeated until the acceptance criteria are met or the agreed deadline is reached. Not meeting the acceptance criteria should be escalated to the steering committee. The steering committee (or relevant alternative) should decide how to proceed. In order to minimize the impact, the security tests may be paused for specific time on request. During the security tests the infrastructure will be monitored by IT Operations.

6 CONCLUSION

I would like to use this section for the conclusion of this master thesis where I wrap up the theory, implemented solutions for the found problems and my recommendations. The aim of the thesis is to provide information about software testing itself in general as well as for specific situations in telco industry. I believe that the thesis provides beneficial information for IT professionals and general public as well.

The conclusions and recommendations presented in the thesis are based on several factors. Many of the research ideas and implementation topics were clear before I started to work on the thesis; however, some of them emerged during the writing or as a feedback from the colleagues or teachers. The study of different use cases helped me to improve the test strategy as well as my own experiences applied in the field. Even though, the software development models as well as testing have some traditional approaches the industry force them to evolve. It is not possibly to pick one test strategy for every situation.

My recommendation is that the organization should have one general test strategy which is focused on the criteria reflecting general view on the organization's quality standards and standard processes used in software testing. It should define what the testing activities are and how should those activities be executed. The strategy should be review regularly and questioned what could be done better for the next project. Projects with specific needs may require more detailed test strategy which will cover the 'missing' parts and changes or deviations from the general test strategy.

Test plan needs to be created for most of the projects. Its main purpose is to show the scope and the planned execution of the test scenarios. Even though, the ownership of the test plan lies in test team I believe that contribution of the project stakeholders, analytics, designers and business representatives is beneficial. Do not be afraid to include and cooperate with people outside of testing team as everybody can contribute either from technical or business side.

I found several areas which can be improved further; however, I believe that the most crucial are:

- Unit and System Tests: Prevention is better than cure, therefore fixing the bug early in the process is cheaper compared to fixing in UAT or even production. Executing unit and systems tests as a standard procedure in every project would help the code stability and accelerate the delivery.
- TST environment: Upgrade and regular maintenance of non-production environment does not bring immediate visible benefit. However, neglecting these requirements will eventually result in lower work efficiency and slower delivery. Upgrading the environment would improve its stability and enable to run full scope performance tests.

The presented improvements can be implemented; however, it will take time, money and the effort of many people. My experience shows that implementing the improvements early in the development process will save time and reduce costs. This approach creates the time buffer and increase the quality because the crucial parts of the projects are assessed first.

7 REFERENCES

CollabNet. (2018). What Is Agile Methodology? Retrieved September 3, 2018 from: https://resources.collab.net/agile-101/agile-methodologies

Drumond, C. (2018). What is Scrum? Retrieved September 12, 2018 from: https://www.atlassian.com/agile/scrum

Guru99 (2019). Boundary Value Analysis & Equivalence Partitioning with Examples. Retrieved January 2, 2019 from: https://www.guru99.com/ equivalence-partitioning-bound ary-value-analysis.html

Guru99 (2019). Boundary Value Analysis & Equivalence Partitioning with Examples. Retrieved January 6, 2019 from: https://www.guru99.com/ exploratory-testing.html

ISTQB (June 2018). Foundation Level Syllabus. Pages -19-21. Retrieved from: https://www.istqb.org/downloads/send/51-ctfl2018/208-ctfl-2018-syllabus.html

ISTQB (October 2012). Advanced Level Syllabus (2012) Test Manager. Pages 32 – 36. Retrieved from: https://www.istqb.org/downloads/send/10-advanced-level-syllabus-2012/54-advanced-level-syllabus-2012-test-manager.html

Kemmis, S., & McTaggart, R. (2000). Participatory action research. In N. Denzin & Y. Lincoln (Eds.), Handbook of qualitative research (2nd ed., pp. 567–605). Thousand Oaks, CA: Sage.

Lewin, K. (1946). Action research and minority problems. Journal of Social Issues, 2(4), 34–46.

Modern Requirements. (2019). Creating an Intersecting Traceability Matrix. What Is A Traceability Matrix? (Easy Explanation). Retrieved January 10,

2019 from: https://www.modernrequirements.com/creating-an-intersecting-traceability-matrix

Powell-Morse, A. (December 2016). Waterfall Model: What Is It and When Should You Use It? Retrieved August 11, 2018 from: https://airbrake.io/blog/sdlc/waterfall-model

Powell-Morse, A. (December 2016). V-Model: What Is It And How Do You Use It? Retrieved August 11, 2018 from: https://airbrake.io/blog/sdlc/v-model

Reason, P. & Bradbury, H. The Sage Handbook of Action Research: Participative Inquiry and Practice. 2nd edition. London: Sage Publications Ltd, 2008.

Remi. (February, 2018). What Is A Traceability Matrix? (Easy Explanation). Retrieved January 10, 2019 from: https://thinkthyme.com/project-management/what-is-a-traceability-matrix

Software Testing Studio. (2017, May). V Model For Software Development Life Cycle | Verification & Validation. Retrieved January 10, 2019 from: http://www.softwaretestingstudio.com/v-model-software-development/

Sami, M. (February 2018). The Waterfall Model, a different perspective. Retrieved August 10, 2018 from: https://melsatar.blog/2018/02/16/the-waterfall-model-a-different-perspective/

Scrum Inc. (2018). Scrum Framework. Retrieved January 6, 2010 from: https://www.scruminc.com/scrum-framework/

Scrum.org (2018). What is a Sprint Review? Retrieved August 1, 2018 from: https://www.scrum.org/resources/what-is-a-sprint-review

Software Testing Help. (August 2018). What is STLC V-Model? Retrieved September 3, 2018 from: https://www.softwaretestinghelp.com/what-is-stlc-v-model/

Software Testing Help. (August 2018). What is SDLC Waterfall Model? Retrieved September 3, 2018 from: https://www.softwaretestinghelp.com/what-is-sdlc-waterfall-model/

Schwaber, K. & Sutherland, J. (2018). The Scrum Guide™. Retrieved August 1, 2018 from: https://www.scrumguides.org/scrum-guide.html#events