jamk.fi

Processes and procedures in test automation

Case: Tailored business support solution

Tarleena Marttila

Bachelor's thesis
May 2021
Information and Communication
Bachelor of Engineering in Information and Communication Technology

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Marttila, Tarleena	Opinnäytetyö, AMK	Toukokuu 2021
	Sivumäärä 32	Julkaisun kieli Englanti
		Verkkojulkaisulupa myönnetty: Kyllä

Työn nimi

Testiautomaation prosessit ja proseduurit

Case: Räätälöity liiketoiminnan tukijärjestelmä

Tutkinto-ohjelma

Insinööri (AMK), Informaatio- ja viestintätekniikka

Työn ohjaaja(t)

Salmikangas, Esa

Toimeksiantaja(t)

Tiivistelmä

Regressiotestauksen merkitys kasvaa ajan myötä pitkäikäisissä ohjelmistoprojekteissa. Saman aikaisesti automatisoidussa testauksessa käytettyjen prosessien ja proseduurien merkitys ja tarkoituksenmukaisuus korostuvat. Tästä johtuen, toimeksiantaja aloitti kehittämistyön, jonka tarkoituksena oli kehittää testiautomaation työtapoja.

Projektin tarkoituksena oli luoda selkeyttä testiautomaation tavoitteisiin ja määritellä näitä tavoitteita tukevat prosessit ja proseduurit. Olennaista tehtävän kannalta oli määritellä olemassa olevat työskentelytavat ja kehittää niitä työskentelytapoja, jotka eivät johtaneet haluttuihin tavoitteisiin.

Testiautomaatiota, regressiotestausta ja erilaisia ohjelmistokehitysmenetelmiä käsittäneen tiedonhankinnan pohjalta määriteltiin prosessit ja proseduurit. Puhuttaessa automaatiosta testauksen yhteydessä on huomionarvoista, että on aina tapauskohtaista, mikä toimii ja mikä ei.

Projektin konkreettisena lopputuloksena syntyi työskentelytapoja kuvaava dokumentaatio wiki-alustalle. Testiautomaatiotiimi otti käyttöön kuvatut prosessit ja proseduurit. Koska testiautomaatio aiheena on monitahoinen, sovellettiin useampia erilaisia kehitysmenetelmiä: kaikki noudattivat ketterää lähestymistapaa.

Projektilla on ollut merkittävä rooli testiautomaatiotiimin kestävien työskentelytapojen kehittämisessä. Parhaiten luodun dokumentaation edut ovat tulleet esiin uusien tiiminjäsenten perehdyttämisessä. Testiautomaation tavoitteet ovat silti edelleen jokseenkin epäselvät. Annettujen resurssien tulisi kohdata asetetut tavoitteet.

Avainsanat (asiasanat)

Testiautomaatio, regressiotestaus, ohjelmistokehitysmenetelmät, Agile

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Marttila, Tarleena	Bachelor's thesis	May 2021
		Language of publication: English
	Number of pages: 32	Permission for web publication: Yes

Title of publication

Processes and procedures in test automation

Case: Tailored business support solution

Degree programme

Engineer, Information and communication technology

Supervisor(s)

Salmikangas, Esa

Assigned by

Abstract

The significance of regression testing increases over time with long-lived software projects. Simultaneously increases the importance of well-considered processes and procedures to apply automated regression testing. With this reasoning, commissioner assigned a development project to improve the ways of working in test automation team.

The targets of the project were to create clarity over the goals of test automation and define processes and procedures to support these goals. The essential tasks were describing existing ways of working and improving those that did not produce wanted result.

The processes and procedures defined during the project are based on knowledge search about test automation, regression testing and different kind of software development models. Also, it must be acknowledged that when it comes to automation in testing, it is always depending on the context what will work and what will not.

A wiki page documentation describing the ways of working took a place as a concrete output of the project. The described processes and procedures were applied by test automation team. Because of the complexity of test automation as a topic, there was more than one development model applied: all has an agile approach.

The project has done remarkable contribution to the development of sustainable working habits for test automation team. The benefit of created documentation has realised at the best with inductions of new team members. However, there is still some unclarity about the goals of test automation. Resourcing must be balanced with set targets.

Keywords/tags (subjects)

Test automation, regression testing, software development models, Agile

Miscellaneous (Confidential information)

Contents

1 Basis of thesis 3

1.1 Driven by need and passion 3

1.2 Objectives 3

1.3 Scope 4

2 Case: tailored business support solution 5

2.1 Commissioner 5

2.2 Program 5

2.3 Test automation team 5

2.4 Product under testing 6

2.5 Automated tests 7

3 Test automation 8

3.1 Complicated topic means challenges 8

3.2 Automating is software engineering 8

3.3 Automation in testing 9

4 Regression testing 10

4.1 System and system integration testing 10

4.2 Regression test approach 10

4.3 Measuring effectiveness is base for improvement 11

5 Software development models 12

5.1 Selection of development models 12

5.2 Challenges of sequential development models 13

5.3 Agile development methods 13

5.3.1 What is Agile? 13

5.3.2 Scrum 14

5.3.3 Lean 15

5.3.4 Kanban 16

5.3.5 Scrumban 16

6 Processes of test automation 17

6.1 Output of thesis 17

6.2 Test development and quality assurance 17

6.3 Looking for suitable development model 19

6.4 Sharing workload and following up progress in meetings 21

6.5 Implementing new features to scope of regression testing 21

6.6 Ensuring quality of release candidate 24

6.7 Improving ways of working and setting targets 25

7 Conclusions 27

7.1 Benefits of thesis 27

7.2 Further improvements 27

References 29

Figures

Figure 1. Activities consisted by the test process by the ISTBQ definition. 18

Figure 2. Test automation in the life cycle of a feature. 20

Figure 3. The ticket template for implementing new features to the scope of regression testing. 22

Figure 4. The sprint of the release testing. 25

1 Basis of thesis

1.1 Driven by need and passion

I did my study related practical training for a software company that is acting as a commissioner of the thesis. After the training period, I was hired by the program and continued to work for it now over five years. Test automation has begun the professional passion of mine. All the time that I have been working on test automation, I have been desiring to find out the best ways of working: improving the processes and procedures. The thesis is a method to produce most efficient and reasonable working habits. It combines the theoretical knowledge collected during the process and the concrete output of improvement work.

In the beginning of the thesis project there was only few people working on test automation project and collaboration habits were unformed. The improvement of processes and procedures is naturally required when project grows and evolves. With a long-lived software project, the meaning and necessity of regression testing increases over time. Automating most of the regression testing will become essential in sense of reasonable resourcing and contribution margin.

1.2 Objectives

The goal of the thesis has been to improve the ways of working on test automation project by creating the clear vision about the targets of test automation and by defining necessary processes and procedures to achieve these targets. The essential task would be describing existing processes and procedures and defining new processes or improving existing ones in case the result has not been like desired.

The concrete output of the thesis shall be the wiki page documentation describing the processes and procedures of test automation in this specific study case. There did not exist any documentation about the processes previously as there barely existed any common ways of working.

1.3 Scope

The thesis is concentrating to automation applied to functional testing. Topics that are not specific to test automation such as test techniques and programming techniques are out of scope. Also unit and integration level testing is out of scope as it is taken care by application developers. Units and modules can only be invoked from code and tests for those are usually considered to be created by developers that are implementing the units and modules (Gijsen 2020, 9). Test automation team is concentrating on the system and system integration level testing. The tests written by test automation team are running separately to the system and interacting with the system via system interfaces and user interfaces.

The application project is applying Agile like development practices and releases new version biweekly. Therefore, only iterative-incremental models are considered when evaluating software development models to find out suitable ways of working for the test automation team. There is no reason to try combine sequential development model to the test automation project while the project exists only to serve the iterative application project.

2 Case: tailored business support solution

2.1 Commissioner

The commissioner provides business support solutions for teleoperators. It has over 20 years of experience producing software on the field of telecommunications. The company offers tools for managing customers, contracts, and products. A cornerstone of the unique full-stack solution is the reliable revenue management system that makes the difference to the competitors.

2.2 Program

As an organization, the company is divided to the *programs*. The target program of the thesis is providing tailored solution for a specific customer. In the program there is about 40 people assigned in different sub teams: software developers, quality assurance, user experience designers, project managers, business analysts and billing managers. Together these people are responsible for the development of new features ordered by the customer and handling the operational issues in the production environment.

The test automation sub team is an independent group of engineers within the quality assurance team with its own processes and habits. Anyhow, ensuring the quality and wrapping the release candidate is a common task so collaboration within the quality assurance team is inevitable.

2.3 Test automation team

The headcount of test automation engineers has been varying from two to four and a half test automation engineers in the program. The test automation engineers are associated with the quality assurance team that includes also testing specialists taking care of manual testing tasks like feature testing. The test automation engineers are concentrating on automated testing: executing the scripts, analysing the results,

and reporting defects. The rest of time is used to build up test automation: designing, implementing, and maintaining test scripts.

The main concern of the test automation team is regression testing. Regardless of would it be automated or manual task, a test automation engineer is evaluating the necessary actions of regression testing. Beside of that the team takes care of security and performance testing as well – in the sense of regression but also with the new interfaces and applications.

2.4 Product under testing

The program is providing tailored solution as a service for a teleoperator which is one of the leading ones on its own market area. The solution includes tools for customer, order and product management, and a revenue management system. There is also provided interfaces for customer and order management, and for reporting system supplied by 3rd parties. The solution also includes many other interfaces for multiple business processes that are used for communication with business partners, like the interfaces for mobile network, logistics partners and accounting agency.

Like most of applications in their early phases, also this solution used to have monolithic architecture when the very first code repository was established in 2012. When a software has been continuously developed for years, it tends to grow until it is uncontrollable big. Slicing up the application to several independent applications and turning to more like micro service architecture has been inevitable progress. Because of this change and continuous improvement work there is new applications and interfaces established all the time even if the main business processes are not changing radically. After almost a decade of development work, the system is very complicated, and the size is huge. All these aspects are composing challenges for automated regression testing but also increasing the value of it.

In the thesis the product under testing is called *application project*.

2.5 Automated tests

All graphical web user interfaces are in the scope of automated regression testing. Most vital application programming interfaces are also covered by automated functional regression testing. Test suites are divided to the test sets per application, except the main regression set which covers testing of all functionalities that used to be included to the old monolithic solution but are now handled by several applications.

The main tool used for test automation is Robot Framework extended with Python. Robot Framework is suitable for functional testing and the greatest benefit of it beside of wide extension library selection is the logs and reports that framework provides. Jenkins is used as a task server for automated executions of test suites. For automated security checkings is used OWASP ZAP that is a ready-made attack tool provided by The Open Web Application Security Project. Performance testing for regression is done with Locust that is a Python based load testing tool.

All together there is about 30 000 rows of code in the test automation project. There is over 2000 test cases included to almost 150 test suites. The test suites form five test sets. As the size of code project is venerable it is essential to have common ways of working and the agreed best practises defined.

3 Test automation

3.1 Complicated topic means challenges

As a result of the knowledge search, it revealed that there is lots of publications about development models used for software development: from internet articles written by professionals to peer-reviewed scientific research reports. There exist several sources also to learn about processes of testing. Many bachelor's theses describe how to set up test automation for web application in practise. But extremely few sources provide an off-the-shelf process for test automation team to follow. Most of the sources are answering to a single question about test automation or introducing a specific tool (Gijsen 2020, i). This might be because work of test automation engineer is usually a combination of both: testing and software development project.

Also, when it comes to automation in testing, it's always depending on the context what will work and what will not. The topic is rather complicated that simple. Because of that it is not easy if not even impossible to provide an approach that would suite for all situations. One should always consider the context and the business needs that the automation is meant to support to be able to choose the best automation strategy for their situation. (Gijsen 2020, ii.) Overall, it has been said that in the field of software engineering, the most important thing is deep understand the matter under question (Haikala 2011, 28).

3.2 Automating is software engineering

It's called automating when a machine is made to execute a task that used to be done by a person. In general automating has nothing to do with testing: nobody would assume that building a weaving machine is done by a tester. Automating includes much more than testing tasks. It is software engineering and includes beside of software testing also requirement engineering, software design and programming. (Gijsen 2020, 17-18.)

It’s an often-forgotten fact that test automation is an independent software project, separate to the application code that is the target of testing done by test automation. The good programming principles that are applied to the application project, work as well to the test automation project. (Kfir 2017.) Why couldn’t then commonly used development models also be applied to the test automation project?

3.3 Automation in testing

Testing includes different kind of tasks and only a few of them can be automated. Many of testing activities requires real intelligence and are therefore unsuitable for automation, for example reviewing requirements or defining good test cases. Only tasks that are algorithmic by nature are convenient to be automated. Nevertheless, it is surely beneficial to automate simple and boring tasks and relieve more time of testers for interesting and most valuable testing tasks that can be executed only by human beings. (Gijsen 2020, 15-16.)

Testing of a functionality that was found to work before an enchantment to the system is called regression testing. Regression testing of functional requirements is often suitable target for automation. Usually, the whole test flow can be automated: from setting the initial state to checking the results. The biggest benefit of an automated checking is high execution speed if compared to the checking performed by a person. (Gijsen 2020, 13-14.)

The suitable testing activities to be automated are not limited to the regression testing. Automation can be used for checking if a new version of code has been successfully deployed to the environment. Or it can be used to create suitable preconditions for testing, for example to create test data. Automation could be also used for scanning other entities than test objects for interesting entries, for example log files. (Gijsen 2020, 15.)

4 Regression testing

4.1 System and system integration testing

The purpose of a system testing phase is to achieve the confidence that the application will be accepted by users and will not cause issues in the live environment that may impact on the business from a functional, financial or image perspective. System integration testing is providing the confidence that the application will interoperate successfully with other systems and may not expose or be exposed to failures when interacting with other systems in the live environment. (Watkins 2001, Chapter Appendix B.)

4.2 Regression test approach

Regression testing is a testing technique which is typically used in the system and system integration testing phases. The system integration testing could be done as a part of acceptance testing when there is a significant requirement of the interoperability with other applications. The purpose of the regression testing is to ensure that the application still works as intended after a modification like adding a new feature. (Watkins 2001, Chapter 11.1.)

The entry criteria for the regression testing is completing the functional testing of a new enchantment. It is necessary to examine requirements specifications and other design documents to be able to find out changes of requirements that might have done existing test cases or scripts obsolete, incorrect, or insufficient. Appropriate verifications must be added to the coverage of test cases to ensure comprehensive regression testing. Those documents are also vital for evaluating which kind of compatibility issues may occur and what test actions are necessary to verify that the key aspects of the application will still perform correctly regardless the changes. Beside of the functional tests, regression testing may also include non-functional testing to study the quality of non-functional requirements such as performance. (Watkins 2001, Chapter 11.2; Watkins 2001, Chapter 11.7.)

When the appropriate test activities have been defined, test team shall execute test cases and scripts towards the new version of the application and record results. They may design and implement necessary test cases and scripts required by a new feature of the application. Additionally, new test data may be created or existing modified to enable testing. (Watkins 2001, Chapter 11.4.)

After conducting the regression tests, the results will be analysed to point out potential errors. Errors may exist in the test cases or scripts as well as in the application software. The regression test procedure should be repeated after the correction of code is provided. When the regression testing does not anymore reveal new defects, the test artifacts such as test cases, scripts and logs should be archived. (Watkins 2001, Chapter 11.8.)

4.3 Measuring effectiveness is base for improvement

It is impossible to make a statement about the effectiveness of the testing process without formal quantitative measurements. By collecting information about how many defects were found during testing and how many defects were found by end users after delivery of the tested application, it could be evaluated if the testing process has improved over time or not. These values could be used to determine the effectiveness of the testing process by comparing them to how much effort was expended in completing the project. Collecting metrics enables the possibility to improve the process by experience. (Watkins 2001, Chapter 12.1.)

There is always a risk that the process of collecting metrics would affect the item being measured, when measuring attributes of the work done by people. One must be aware of this phenomenon called Hawthorne effect when planning the process of collecting metrics. The process should be as unobtrusive as possible. (Watkins 2001, Chapter 12.4.)

5 Software development models

5.1 Selection of development models

There is unlimited amount different kind of approaches for software engineering. The simplest way would be code-and-hack, but many of models are more sophisticated. They see software development as a project with a life cycle and include some project management structures. A project could be defined to be a bunch of tasks that have schedule and resources. The targeted result will be achieved by accomplishing these tasks. Traditionally, software development tasks are divided to following phases: requirement specification, architectural designing, programming, testing, production deployment and maintenance. (Haikala 2011, 29-31.)

Software development models could be divided to two main types based on their understanding of the software life cycle: sequential and iterative-incremental models. In the sequential development, all tasks of the development phase are finalised before tasks of the next phase will take place. When the requirement specification has been finalised, the architectural design will follow, and so on. A typical development cycle of the sequential model could take from 6 moths to 2 years and the entire system is released in the end of the cycle. Sequential models are the traditional ones, for example waterfall and V-model. (ISTQB/ISEB Software Testing Foundation Certificate Course, Chapter 2.1.)

In the iterative development, a cycle is much shorter, typically from 1 week to 1 month. A small part of the system, maybe only one or couple of features, is finalised and released in the end of the cycle. These short increments are iterated, and new features are introduced by every iteration. Agile development methods are the most famous ones of the iterative-incremental development methods. (ISTQB/ISEB Software Testing Foundation Certificate Course, Chapter 2.1.)

5.2 Challenges of sequential development models

There are two main issues with the sequential development models from the business point of view. Requirements for the software are set by the business and they are changing along the changes on the business. A development cycle of the sequential development models tends to be relatively long and often the business and the requirements will change during that period. Traditional development models are not usually responding these changes very well. A software should fulfil the business requirements but in the worst-case the business may end up to outliving the software. Another big issue with the sequential development models is that they do not provide any business value before the very end of the project. (Blankenship 2011, Chapter 1.)

To be able to proceed in a settled timetable, the development model should also take into consideration the technical expertise of the developers who are taking care of the implementation. There should also be a correction mechanism for estimation errors in workload and timetable. It is problematic if the only mechanism to handle the delays that occur in the project is the over-estimation added workloads in the beginning of the project. (Blankenship 2011, Chapter 1.)

5.3 Agile development methods

5.3.1 What is Agile?

Agile has evolved from earlier development methodologies as an answer to the issues introduced in previously. In the very beginning of the 21st century the agile movement was started by Agile Manifesto which defined a new philosophical point of view for software development. The agile methodology appreciates a tight relationship with the inevitably ever-changing business. Improving the processes by learning and redefining them are core features of Agile. (Blankenship 2011, Chapter 1.)

As a software development model Agile provides the maximum return to the business for their investment by mitigating the risks related to opportunities and costs. Compared to the traditional development models, Agile projects deliver business value very rapidly. The iterative-incremental development with short development cycles and the regular delivery of working software provides the return of the investment already in the early stages of the project. This approach also leads to higher user satisfaction and reduces training and maintenance costs. Agile provides a built-in mechanism to better accuracy of the timetable and workload estimates. Better visibility is achieved by breaking down the deliverables of the project into smaller pieces that are delivered via iterative increments. (Blankenship 2011, Chapter 1.)

There are various methodologies of Agile that all have shared interest to fulfil needs of the business and achieve high customer satisfaction. They have more or less similar characteristics but what they all have common is an iterative development model. (Blankenship 2011, Chapter 1.) There will be introduced some of the most popular agile development methodologies in the next paragraphs. These have been considered when looked the working solution for the test automation development in case of tailored business support solution.

5.3.2 Scrum

Scrum is an iterative model that embraces the agile development principles (Blankenship 2011, Chapter 2). Scrum is a way to organise iterations on the implementation phase. It does not include project management methods and it does not provide tools for the requirement specification. The essential parts of Scum are planning activities and review activities for the iterations, usually performed by meetings. (Haikala 2011, 47-54.)

In Scrum methodology, implementation iterations are called sprints. There is held planning, review, and retrospective meetings with every sprint. Also, daily stand-up meetings and backlog organising meetings are taking place regularly. (Bibik 2018, Chapter 2.)

Scrum teams are self-organising and self-governing. A team is composed of a product owner, a scrum master and a bunch of technical experts that varies by a team. The product owner is in charge of the product and the scrum master oversees the process. Technical experts can be developers, quality engineers, architects, and technical writers. A typical team size is ten people. (Bibik 2018, Chapter 3.)

5.3.3 Lean

Lean software development bases on the management culture of Japanese auto industry and it's also called The Toyota Way. This approach was widely adopted to the software engineering on the 1990s. Focus and discipline are the essentials of Lean. As typical for agile development Lean has close relation with the business and its primary focus is the customer value. Lean highly appreciates adding value for the business and heeds the value streams of the business. Implementing something else than the most important features from the business point of view would be waste. (Coplien 2010, Chapter 1.)

Another way to avoid waste is implement only such a thing that is important right now. Lean includes a principle to avoid any storage that would in the case of software development mean for example creating design artifacts or code for future purposes. Something created without an actual demand could turn out waste: something that would never be used. (Coplien 2010, Chapter 1.)

Lean appreciates people and interactions over processes and tools as typical for the agile development models. Usually, the customer is considered as a stakeholder and the most important "people" but Lean emphasises the whole development team and other involved people as well. "All hands on deck" mentality requires all parties to be present at the beginning of a project. The intensity at beginning seems to reduce the overall life cycle cost and improve the product quality regardless there is usually lots of iterations and rework in design during the project. (Coplien 2010, Chapter 1.)

5.3.4 Kanban

Kanban was invented together with Lean in the car factory of Toyota. Kanban can be seen as a method evolved out of Lean principles. Beside of an independent development model, it could be also considered as a method of Lean software development. (Hughes 2016, 14-15.)

Kanban is a visualization of the work process: it does not provide much more than visibility to the progress and bottlenecks via Kanban board. On the contrary to many other agile methodologies, Kanban does not define roles of team members and the development is not necessarily happening in iterations. The essential element of Kanban is that one can have only few tasks under work in parallel. This approach is very simple to adapt, and it is excellent for work with a continuous flow like the maintenance or testing phase. It is also possible to combine Kanban with some features of other methodologies e.g., support Kanban with daily Scrum meetings. (Bibik 2018, Chapter 2; Hughes 2016, 15.)

5.3.5 Scrumban

Scrumban is combination of Scrum methodologies and usage of Kanban. It combines both methodologies by having Scrum like time-boxed iterations with the continuous flow which is typical for Kanban. (Hughes 2016, 15.)

6 Processes of test automation

6.1 Output of thesis

There has been created a wiki page documentation as an output of the thesis. The documentation is describing the processes and procedures of the test automation. The ways of working in the test automation have evolved constantly during the journey.

The chapter describes the processes and procedures originated from the thesis that are applied by test automation team in their daily work.

6.2 Test development and quality assurance

The processes of test automation have been defined based on the fundamental test process introduced on the testing foundation certificate course by International Software Testing Quality Board (ISTQB). Activities consisted by the test process are illustrated in Figure 1: test planning and control, test analysis and design, test implementation and execution, evaluating exit criteria and reporting and finally test closure activities. (ISTQB/ISEB Software Testing Foundation Certificate Course 2018, 1.12-1.16)

The activities of test automation team can be divided roughly to two groups of tasks. The first half of the activities is considered test development activities as they are related to implementation of new features to the scope of regression testing. Development tasks are test planning, test analysis, test design and test implementation. The second half of activities is considered quality assurance activities: test execution, evaluating exit criteria, reporting and test closure activities. This distribution helps sharing work between several people and also eases up prioritising work. Testing related tasks are always with the highest priority as ensuring the quality of release candidate is the main job of test automation. Quality assurance tasks are not just the

most important but also time critical as they are bound to the release cycle of application project. Test development related tasks are unavoidable but usually more adjustable as they mostly do not have a hard deadline.

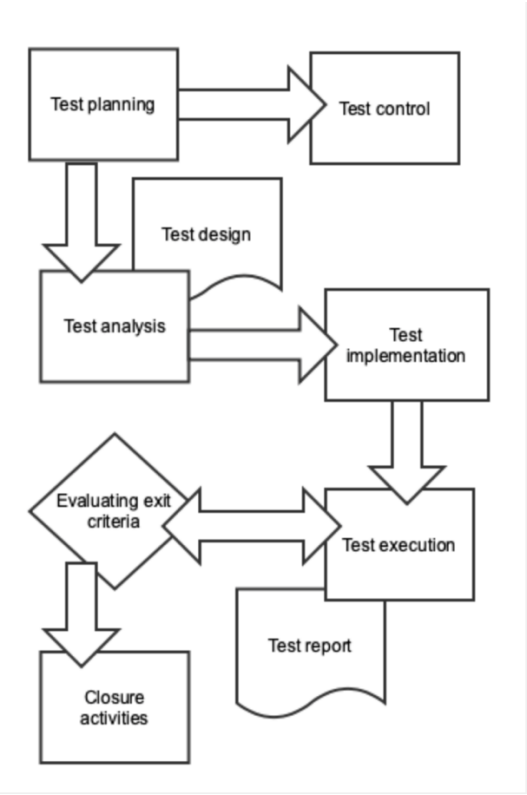

Figure 1. Activities consisted by the test process by the ISTBQ definition (Internal documentation of the commissioner 2021).

The allocation to development and testing activities is not absolute. As these two kinds of tasks are performed by same people, it is natural to combine the tasks from both groups when it gives advantage. Test planning and test analysis could be also considered to be preliminary tasks for quality assurance activities, so they have higher priority than test implementation. Also, they must be done before the feature is included to the release as they include a risk analysis and evaluation about necessity of manual regression testing activities. Bonding development and testing related tasks together forces the stages to be performed also in the case that the new feature does not require any test development work, but it also guarantees that all new features has been evaluated for necessity of manual regression testing activities.

Bonding violates Lean development principles as planning, analyse and design work for test development is done in early phase and it is unsure if actual implementation of tests will ever take place. There is a significant risk that this work will never produce any business value for the customer. Nevertheless, it is still the most efficient way of working as in another case there must be separate planning and analyse work to be done for development purposes.

One of the most time-consuming tasks in planning phase is getting familiar with the new feature. Feature familiarising is the base for all further activities so to avoid repeating this work it is beneficial have analyse and design work to be done simultaneously with planning. In the end, it would talk more about insufficient resourcing than about bad ways of working if planning, analyse and design work for test development will never redeem itself.

Another combined task is analysing the test log. It is usually considered as a quality assurance activity, but it also serves test maintenance which is test development task.

6.3 Looking for suitable development model

It was noticed during the knowledge search that automation in testing is rather a complicated topic than simple and the choice of the development model should be done to serve the business. There is not one solution to the challenge of the thesis. There is several needs to serve so several ways of working should be applied as well.

The test development is software development. It includes all traditional software development phases introduced in chapter 5.1. The test implementation can be done by following any development model. Some of the tasks of test automation could be done as synchronised with the application project. The requirement specification and architectural designing are starting in the beginning of the project that is providing an enchantment for the application. Actions related to these phases may trigger the corresponding actions in the test automation project. In this case the test automation can take benefit out of all processes of the application project that are there to

support the development like solution review meetings and other requirement clarification provided by business analysts. Corresponding activities in different phases in the test automation and the application project are illustrated in Figure 2.

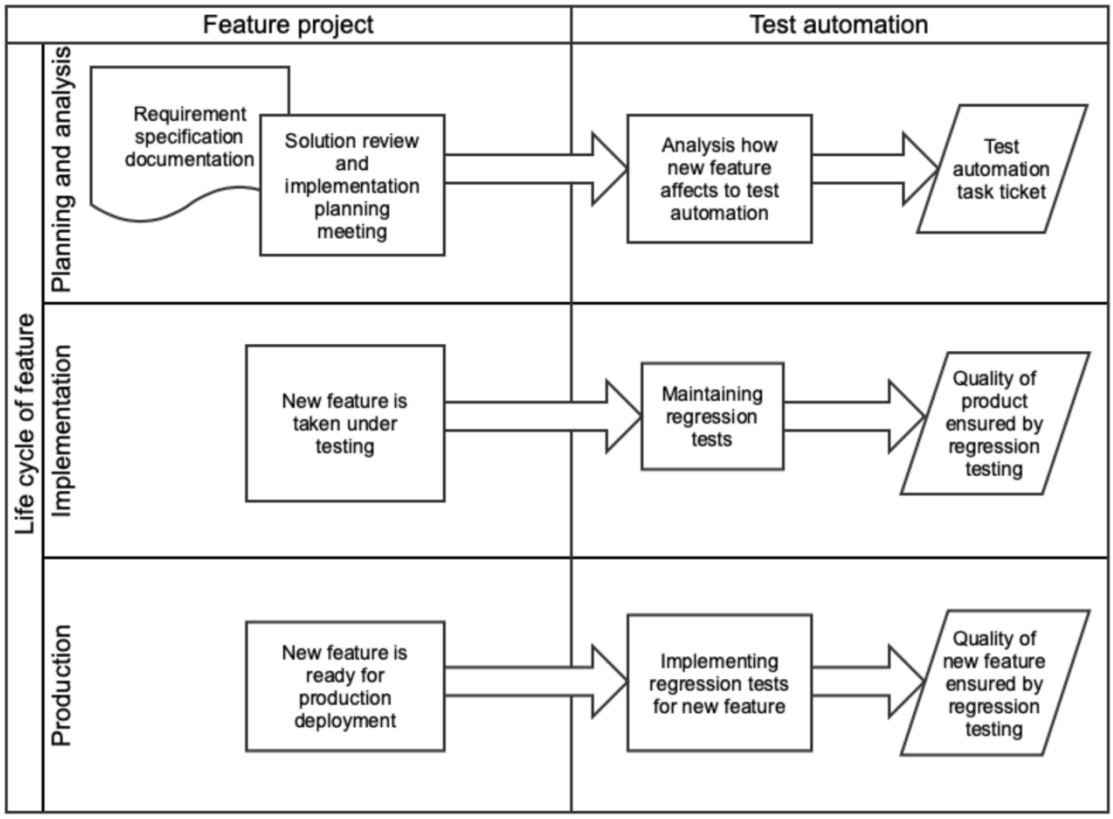

Figure 2. Test automation in the life cycle of a feature (Internal documentation of the commissioner 2021).

If test automation activities are tied to the application project in any way, then the characteristics of application project development must be considered when defining the test automation processes. In the case of agile methodologies, requirement specification and architectural designing phases are not necessarily finished before programming and testing are already taking place. It means that also test automation planning and design must live along with the changes on the application project e.g., schedules, requirements, or design. This means that if the application project follows agile methodologies, the test automation project must follow agile methodologies as well. The test automation project might be bound to the application project's schedules and ways of working also by practicalities. In agile methodologies interactions

between people are appreciated over the documentation. This might lead to a situation that after enchantment has been released to the production, there is no more any reliable source to recover the requirements as the written documentation is not comprehensive enough.

To beat the challenge of tasks with different kind of priority and time affiliation, there is separate processes created for test development and quality assurance activities. Within these processes, different development models are followed, although all of them are following the agile development principles. Scrum is used as a frame for combining different kind of ways of working.

6.4 Sharing workload and following up progress in meetings

Test automation tasks are shared in a bi-weekly sprint planning meeting. To be able to find the balance with limited resources it is useful to assign only tasks with similar time affliction for a person at time. Otherwise, one might end up to neglect tasks without a strict schedule in the favour for more acute tasks. In practise this means that one should not have both test development and quality assurance tasks at time because it is inevitable that a quality assurance task would always override the test development tasks. Test planning and analyse activities are exceptions as they are bound to the schedule of the application project via solution review and implementation planning (SRIP) meetings. Based on this practicality, the tasks are divided to four roles: release testing, new development, SRIP meetings and improvement work.

Quality assurance team has a habit of a daily stand-up meeting. The test automation team is invited to the QA daily as ensuring the quality requires collaboration and having the common meeting supports the shared goal.

6.5 Implementing new features to scope of regression testing

As stated in the beginning of the chapter, test development activities consider test planning, analysis, design, and implementation tasks. Available tools guided the

choice of the development model for test automation development tasks. The program was using Jira by Atlassian as an application to plan, track and manage development projects. Documenting also test automation tasks to Jira was a natural choice. Jira provides Kanban board to view task tickets, so Kanban combined with Scrum, also known as Scrumban, was easy to try out.

This tool-oriented choice consolidated the habit to markdown all test automation related tasks as Jira tickets. The process of test automation is triggered by SRIP meeting invitation when a new feature to the application is introduced. As a pre-task of meeting, test automation engineer studies the requirement specification and creates the test automation task ticket to Jira. See Figure 3 for the ticket template.

```
h1. {color:red}DRAFT{color}

h1. Test Planning

h2. Test Basis
- Provide a short description of feature and a links to specification and to other necessary work documents
- High level architecture:

h2. Scope of testing and risks

h3. The parts of product to be effected
- List all components and other things which could be considered as a test objects with the new feature

h3. Product risks
- List worst case scenarios with high business risk

h2. Test activities and schedule
- Define test activities (creation/update of test cases, manual/automated special regression testing)
- Provide some schedule estimates based on project milestones

h4. Scope of automated regression testing
- Define which test objects are included to the scope of automated regression testing

h4. Scope of manual regression testing
- Define which test objects are included to the scope of manual regression testing

h1. Test Analysis

h2. Test conditions
- Review the test basis (requirements, specifications, risk analysis)
- Identify test conditions (what to test: function, transaction, quality characteristics or structural element)
- Evaluate testability of test basis and test objects
- Create bi-directional traceability between test basis and test cases

h2. Test data
- Identify necessary test data to support test conditions and test cases

h1. Test design
- Design high level test cases for test conditions:
set of input values, execution preconditions, expected results and execution postconditions
- Define if test cases should be added to existing suites (which one?) or new test suites created
```

Figure 3. The ticket template for implementing new features to the scope of regression testing (Internal documentation of the commissioner 2021).

Fulfilling sections about test planning and test analysis can be started already before the meeting and completed during and right after the meeting. Test design is advisable to be done only after SRIP meeting and when all necessary clarification about specification has been done. Creating and fulfilling the ticket and participating to SRIP meeting is responsibility of the person who is in SRIP meeting role at time.

Implementation of the new tests could be done later when the feature is ready for testing. The features still might change radically in GUI mind during the last moments before the production release, so implementation of regression tests is typically done only after feature's production release. Implementing new tests for previously released features is responsibility of the person who is in new development role at the time.

To ensure good code quality with test automation scripts, there is certain ways of working followed in test automation development. The team has advanced version control policies and branching strategies. This allows development of several product versions at time and enables the ability to respond agilely to changing implementation plans on the application project. Code review habits are not only significant for the quality of code but also the way to share knowledge over new features and new tests to the team. Code review is also remarkable way to improve one's programming skills based on feedback.

Beside of SRIP meeting and new development roles there is also one more role that aims to development of test automation scripts: improvement work. Improvement work considers internal development tasks like library upgrades, architectural improvement work and all other kind of technical improvements that are not related to any new feature or included to the scope of any project based on a customer requirement. This kind of technical improvement work is inevitable maintenance work required constantly to sustain the value of test automation. If improvement work is neglect the scripts will became useless before long.

6.6 Ensuring quality of release candidate

The role that is responsible for ensuring the quality of a release candidate is called release testing role. Release testing collects all tasks related to the regression testing of a release candidate, including quality assurance activities like test execution, evaluating exit criteria, reporting and test closure activities. Beside of quality assurance activities, release testing role also includes some test development tasks that are bound to the schedule of the application project: supporting manual testing by providing test data and maintaining existing test scripts i.e., fixing them to fit with the enchantments introduced to the application.

In the release cycle that the program follows, there is a sprint of two weeks. The first week is called the feature testing period and during the week the validation and verification of new features is finalised by manual testing methods. The test automation team is supporting the job by delivering test data generated by test scripts. To be able to do that the maintenance of regression test sets is required to get scripts fit with new enchantments of the application. The maintenance is also precondition for the regression testing.

The second week is dedicated purely for the regression testing. All manual regression checking performed is organised by a test automation engineer. Many of manual regression tests are based on data generated by test automation scripts. Also reporting defects found by test scripts and monitoring errors arising on the system are included to the release testing tasks and executed continuously during the sprint. The release cycle is illustrated in Figure 4.

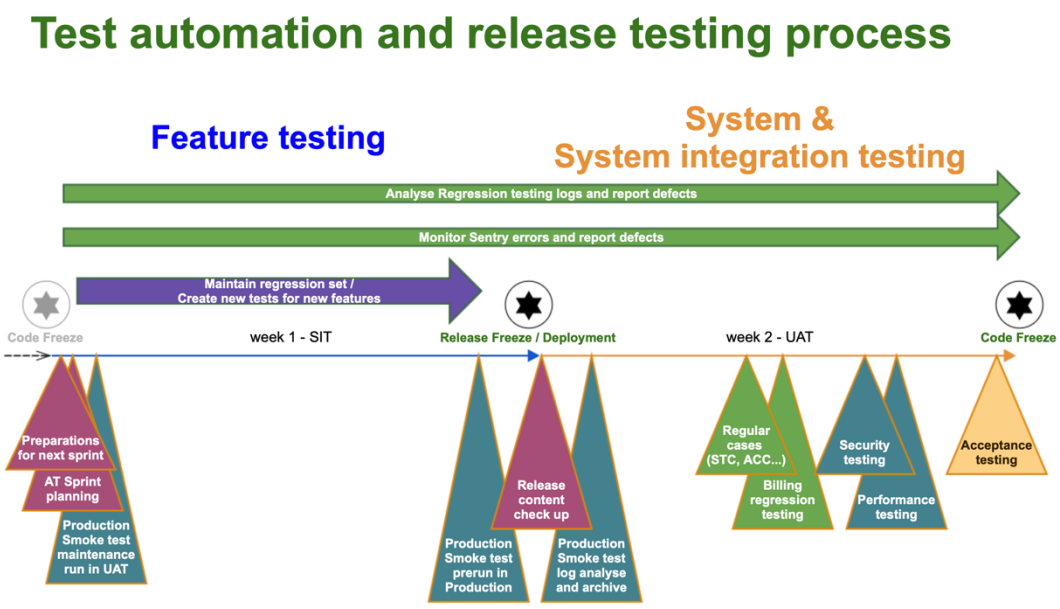

Figure 4. The sprint of the release testing (Internal documentation of the commissioner 2021).

Finally, after all regular checking it is time for acceptance procedures. These are carried out with support of test automation scripts by a test automation engineer. Beside of the functional testing there is also some performance checking included. Based on the acceptance testing analysis done by a test automation engineer the release manager will proclaim the code freeze and wrap the release candidate to wait for the production release.

6.7 Improving ways of working and setting targets

The test automation team gathers to a retrospective meeting to evaluate the past period and to improve the ways of working based on the evaluation. Retrospective meetings are not taking place in the end of every sprint but few times per year as it would cause too much burden for a small team to allocate so many resources for improving the processes. As the organisation is flat and agile, the new ways of working can be also applied without formalities like the retrospective meeting if seen necessary at the moment. In the end, everything is just about agreeing the ways of working between a couple of people.

Improvement projects for the next year are chosen based on the feedback collected during the retrospective meeting. This happens in another meeting, annual and biannual status meeting, where is invited the most important stakeholders. The test manager, the lead of project management, the lead of business analysts and the head of program are gathering to hear about the latest achievements of the test automation and to seal the targets for next year. The targets are guiding the team to ensure the most business critical features but also to achieve reasonable coverage overall the whole system. A self-organising team needs clear targets to be able to act and make decisions independently. These targets are usually considering only the development of test automation – not related to the quality of the product.

Product quality related measurement has been done only for a part of the system, for a special project which has been more interested in the key performance indicators. There are collected the classical indicator values: the execution time and pass percent of tests, and the number of preproduction and production defects per a release. There is no absolute way to separate regression issues and issues related especially to the new features, neither there is a way to divide the quality of product to the quality produced by automated testing and the quality produced by other testing or actions. Therefore, all changes in quality are accomplishments of the quality assurance, regardless of the methods.

7 Conclusions

7.1 Benefits of thesis

The thesis project has done remarkable contribution to the development of sustainable working habits for the test automation team. Writing down the guidelines and best practices has helped the team to create the concordant ways of working leading to higher and constant level of quality with the test automation and the application project.

The most remarkable benefit of created the documentation has been not only for the test automation team members but for new people joining to the team: the documentation gives good expression about what are the main tasks, how things are done and what is expected for the test automation engineer. The induction process is heavily leaning on the documentation.

The documentation also provides necessary information about the test automation for other development activities like feature testing, release management and project management. Based on the described processes, the test automation team can communicate about necessities and inevitable time limits with reasoning to the stakeholders.

What comes to the clear vision about the targets of test automation, there is still work to do. The goal setting has been done, but resourcing is not meeting the requirements of the target. To create the sustainable ways of working, the resources and the targets needs to be balanced.

7.2 Further improvements

The thesis process has taught a lot about the software development, the testing as a part of it and the meaning of automation to the author. The development of processes will naturally continue also afterwards based on the criticism by author. The

most important change would be to include the implementation of regression tests to the new feature's definition of done. After every iteration, the importance of regression testing is increasing (ISTQB/ISEB Software Testing Foundation Certificate Course 2018, chapter 2.1). With modern agile development models, like CI/CD-pipeline, a constantly developed automated regression testing set up is mandatory and implementation of test is tightly tied to the feature development.

Similar persistent reverence of good quality and focus to the automation of quality assurance activities would be welcome also with this software project. Committing to the schedule of the application project with the test implementation would require considerable increase to the resources of test automation. An additional way to relieve the burden is DevTestOps like thinking. Instead of having an army of test automation engineers, the group of developers who are developing the product are also contributing to the test automation project. Business analysts and feature testers would participate to the planning, analysis, and design work of automated testing. Nevertheless, it would still require remarkable increase to the resourcing of test automation tasks and radical change to the ways of working.

It's important to clearly state the objectives of testing (ISTQB/ISEB Software Testing Foundation Certificate Course, Chapter 1.5). As all activities considered to support activities, quality assurance rarely gets too many resources regardless its role as an inevitable part of software development. Since testing everything is an unrealistic plan, choices must be made. Clarifying the targets of regression testing should be done to balance product quality and resources like time and cost. Setting clear goals and usage of enhanced and efficient methodologies and techniques are essential to take maximised advance out of limited resources. The ultimate target of stable processes is after all to create predictability to best support the business goals.

References

Bibik, I. 2018. How to Kill the Scrum Monster: Quick Start to Agile Scrum Methodology and the Scrum Master Role. Electronic book in Skillsoft. Apress. Requires account. Referred 27.1.2021. https://janet.finna.fi/, Skillsoft Books ITPro.

Blankenship, J., Bussa, M. & Millett S. 2011. Pro Agile .NET Development with Scrum. Electronic book in Books24x7 ITPro. Apress. Requires account. Referred 4.11.2020. https://janet.finna.fi/, Skillsoft Books ITPro.

Coplien, J. & Bjørnvig, G. 2010. Lean Architecture for Agile Software Development. Electronic book in Books24x7 ITPro. John Wiley & Sons. Requires account. Referred 6.11.2020. https://janet.finna.fi/, Books24x7 ITPro.

Gijsen, M. 2020. Automation In Testing. Electronic book in EuroSTAR Huddle. Lean Publishing. Requires account. Published on 10.8.2020. Referred 28.10.2020. https://huddle.eurostarsoftwaretesting.com/, EuroSTAR Huddle.

Haikala, I. & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. Hämeenlinna: Talentum Media Oy.

Hughes, R. 2016. Agile data warehousing: a guide for solution architects and project leaders. Electronic book in eBook Collection (EBSCO). Elsevier. Requires account. Referred 28.1.2021. https://janet.finna.fi/, eBook Collection (EBSCO).

Internal documentation of the commissioner. 2021. Unpublished Confluence wiki pages. Referred 13.5.2021.

ISTQB/ISEB Software Testing Foundation Certificate Course. 2018. Course syllabus. Grove Software Testing Ltd. Lessons taken at 29.-31.5.2018.

Kfir, N. 2017. A Pixie Dust Recipe For Automated Tests That Fly. Speech at 8.11.2017 on EuroSTAR Conference in Copenhagen.

Watkins, J. 2001. Testing IT: An Off-the Shelf Software Testing Process. Electronic book in Books24x7 ITPro. Cambridge University Press. Requires account. Referred 16.1.2020. https://janet.finna.fi/, Books24x7 ITPro.