OULU UNIVERSITY OF
APPLIED SCIENCES

Tiina Matala

OBSERVATIONS ON BUILT-IN QUALITY

Diary Thesis

OBSERVATIONS ON BUILT-IN QUALITY

Diary Thesis

Tiina Matala
Bachelor's Thesis
Spring 2021
Degree Programme in Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Tiina Matala
Title of the bachelor's thesis: Observations on Built-in Quality
Supervisor: Veijo Väisänen
Term and year of completion: Spring 2021 Number of pages: 35

The objective of this thesis was to make observations on the current level of built-in quality at Enerim Oy, whom the author works for. Observations were planned to be gathered as improvement suggestions and ideas on how to carry on this work in the future to increase the overall built-in quality at Enerim Oy.

The thesis was written in a diary form consisting of five and a half observation weeks of which three weeks included author's participation in release testing. The theory covers built-in quality of SAFe model and agile testing. Shift left testing and quality assurance have also been examined on quality aspects.

The results of this thesis are that Enerim Oy is or is going to be implementing solutions to achieve built-in quality, such as early involvement of testers in planning and gradually increasing the level of test automation. However, there were many factors discovered, which could still be improved or are currently missing. Shift left testing was not happening in TDD, BDD and HDD. Quality standards and controls were lacking or being communicated inadequately.

To benefit the most of test automation, there should be individual plans made with a thought for each team since the needs vary. Shift left testing is not widely implemented in different organizations regardless of its benefits. There are several reasons for it, but one major reason is that shifting left requires additional human resources to testing and development. Otherwise, it may double the workload in the beginning.

Keywords: Built-in Quality, Shift Left Testing, Quality Assurance, SAFe

PREFACE

This thesis work was commissioned by Enerim Oy in the spring 2021. The supervisor of the thesis was lecturer Veijo Väisänen from Oulu University of Applied Sciences and instructor from Enerim's side was test specialist Juha-Pekka Puhakka. I want to thank you both for guidance and feedback.

I want to thank Minna Pallari for giving me the idea for the thesis topic and my whole team for giving me the possibility to concentrate on the thesis work full time.

Last, I want to thank Melina and Markus for your love and support during this thesis process. You are my dearests.

Oulu, spring 2021
Tiina Matala

CONTENTS

VOCABULARY

ATDD = Acceptance Test Driven Development, where team member's take different perspectives and collaborate in writing acceptance tests before implementing the feature

BDD = Behaviour-Driven Development, when used in story specifications, the following format is being used: Given – When – Then

Feature = unit of functionality which fills a requirement

Increment = small portion of functionality which is added in an iteration

Jira = Project management software

NFR = Non-Functional Requirements, such as security, reliability, usability etc.

PO = Product owner. PO's task is to forward insights of the product to development team and maintain project backlog

QA = Quality Assurance, practices and activities used to ensure software quality as a part of development process

Regression tests = testing if any new updates or modifications affect to the overall functionality of the software

Release testing = testing a specific release of a system

SAFe® = Scaled Agile Framework® is a collection of agile and lean methods, which have been bundled under own terminology. Its idea is to produce as much value as possible in shortest possible lead time. SAFe is based on agile development, lean development, DevOps, and system thinking.

SDLC = Software Development Life Cycle includes phases: plan, create, develop, test, and deploy

Story = general explanation of new software feature, written in end user's perspective

TDD = Test-Driven Development, coding includes three combined activities: coding, testing, and refactoring

1 INTRODUCTION

Using different agile methods, such as scrum and Kanban, is common in Information Technology companies. There exist several frameworks, which help organizations on their way to agile, such as Lean Agile (LA), Agile Software Solution Framework (ASSF) and Scaled Agile Framework (hereafter referred as SAFe) (Nader-Rezvani 2018, referred to Mar 9, 2021). This thesis will focus on SAFe and especially built-in quality of SAFe.

There are several theses made of other aspects of SAFe than built-in quality. The literature over the topic is currently modest. For this reason, this thesis will cover some related quality assurance theory.

The quality assurance aspects will be reflected as the author participates in release testing and makes observations on built-in quality while testing. The thesis will also question if shift left testing could shorten the gap between development and testing and improve giving feedback at Enerim Oy. Thesis will also handle the question that if some principles of shift left testing already exist, how they could be further improved.

As a software tester, the author is interested in quality assurance and how well the theory meets the practice in real world. It will be beneficial to study the theory of SAFe and its built-in quality, since Enerim Oy is implementing SAFe and customers value well-tested, working software, which is of good quality.

2 DESCRIPTION OF THE CURRENT STATE

2.1 Analysis of the current work

Enerim Oy is a software company, which is providing information systems and is concentrated on energy sector. Enerim's customers consist of Finnish energy sector companies and currently Enerim Oy is the market leader in its field of business in Finland. Enerim's software does everything from making an electricity delivery and selling contract to data exchange and invoicing. In addition to electricity, Enerim's software also supports gas, heat, and water. Enerim has around 230 employees on 10 different locations in Finland, Oulu office being the biggest. Enerim separated from Empower group in July 2020 and started as own independent company.

The author has worked as a software test engineer at Enerim from last May, first as a summer intern and then making company-oriented product development projects. After that it was natural to continue with writing thesis for Enerim.

The author's tasks consist of manual testing one of Enerim's software, called EnerimCIS. Day-to-day job activities include verifying fixed bugs and writing test cases and testing new stories, which provide new functionalities. At regular intervals there is service pack testing, where testers verify that currently made bug fixes still work before the service pack is released to customers. After a quarter of a year, when increment ends, there is release testing. In release testing, all stories, which have been completed during the increment, are tested before the release is published to the customers. The author also participates in daily, weekly, and monthly activities of her scrum team.

The technical knowhow, which is required in author's work, includes using different tools. The most essential tools are Jira, Postman, mRemoteNG, Logviewer, DBeaver, and SoapUI. The author must also know how the software should function to know which kind of behaviour is not of the right kind.

Considering professional growth, the author is still in the beginning phase of her journey. The school has created a broad basis for the work life, though many subjects were covered quite superficially. It is easy now to study more about the required matters, when having basic knowledge on them. The know-how deepens every week by working and therefore it is good that the end-part of the studies is completed by working in a company. A lack of experience can be seen in the lack of work routine, but as the time passes it will be easier to see things in a bigger scale. It would be also beneficial to be able to make more accurate estimations on how much time each story or bug verification takes.

2.2 Interest groups

Internal interest groups which affect author's work consist of author's team members, meaning the other manual tester of the team, test automation engineer, developers, PO, and Scrum Master. There is also interaction with test engineers from other development teams. Communication with DevOps team is necessary when for example new builds are needed. Other interest groups, which the author is not directly in contact, are test managers, upper management, and business units.

External interest groups consist of customers, who are the electricity retail companies that are using our software. The author is not in direct contact with the customers.

2.3 Interactions at the workplace

Every morning starts with a daily scrum meeting led by the Scrum Master. Every team member keeps a short briefing on what they have been working on the previous day, what they are doing on that day and is there any issues, which they need assistance with.

After each sprint, the Scrum Master holds sprint review, retrospective, and planning meeting as one longer meeting. The review part consists of demonstrations of finished and tested stories. These demos are recorded, so others can watch them later. In retrospective part, the whole team discusses on the accomplishments and development points of the team during the previous

sprint. The aim is constant improvement. The rest of the meeting is used to plan the upcoming sprint, where each story, manual and test automation case are estimated in story points. A Story point means the ideal number of days used for completing the story. A big story may be five story points, meaning five days, and a smaller one for example three. Usually, there are around ten stories per sprint. On top of that there are bugs, test automation tasks and study tasks.

PO hostess backlog refinement meeting on the previous week before sprint review, retrospective, and planning meeting. PO checks how team members are proceeding with their work and introduces the stories and bugs coming to the following sprint.

These three meetings are held between the development team. On top of that there are other meetings, which the author attends such as testers' meeting, Digital platforms unit meeting, and Wednesday meeting for Product Development personnel. Once a month there is also meeting held by the CEO and the leadership team.

All the meetings are held remotely via Teams due to prevailing Covid-19 regulations. The author has now worked remotely almost a year at Enerim, meaning the whole time she has worked there.

3 PURPOSE AND OBJECTIVES

This thesis discusses different aspects of built-in quality and how well they are currently being implemented at Enerim Oy. It also will question if shift left testing could provide an answer for quality related issues at Enerim Oy. Enerim has recently started a new quality project and the author hopes that this thesis will be beneficial for the project as well.

This thesis is carried out during March and April 2021. The reporting format has been chosen to be a diary thesis and it consists of five and a half observation weeks. This schedule has been chosen because the author wants to graduate in June. Reporting will be done on a weekly basis. This is because in most weeks, the work consists of multiple different tasks, which would make the text inconsistent. The actual working phase consists of three-week release testing of increment 21.1. The whole theme of this thesis has been built around quality issues in testing and, more specifically, in release testing to make the text more unite.

The author’s personal learning goals are to get familiarized with agile testing and built-in quality, increase knowledge on SAFe model and enhance own level of quality in release testing, and in testing in general. Release testing, which will be done during this thesis process, is the second one, which the author has ever participated. Other goal is to benchmark good test case writing practices and utilize them later in own work.

The objective of this thesis is to make observations on the current level of SAFe model’s built-in quality at Enerim Oy. The observations are gathered as improvement suggestions and ideas on how to carry on this work in the future to increase the overall built-in quality at Enerim Oy.

4 OBSERVATION WEEKS

The observations were agreed to be done on a weekly basis.

4.1 Week 1, March 3 – 5

Working on the thesis began in the middle of the week with preliminary preparatory tasks. During these couple of days, the author wrote a project plan and the first version of the table of contents. The thesis kick-off meeting was held together with the instructing teacher and supervisor from the employer's side. Otherwise, time was used to gather data and sources and reading thesis instructions.

Concerning the sources, it was relatively easy to find e-books from Oula-Finna library database with the search instructions received from the school library.

The following week is the winter holiday week, and the thesis work continues the week after that. Only one day of the holiday will be used for reading the thesis literature and making notes.

4.2 Week 2, March 15 – 19

The main subject of the week was to study the core values of SAFe and especially the five dimensions of SAFe built-in quality. The objective was to combine the theory with the practice, which the author had seen on her agile team.

It is said that values are the basis, which guide what we do and how we work. Organizations also have values, which direct how projects are managed and ensure that decisions are made based on values. The four core values of SAFe are built-in quality, alignment, program execution and transparency. These are the essential forces, which drive the effectiveness of SAFe. These values should guide everyone in portfolio SAFe in their daily actions and behaviour. (Scaled Agile, Inc. 2021a. Referred to Mar 16, 2021.) This thesis will focus on the built-in quality.

The built-in quality means pursuing high quality right from the start. The built-in quality must be reflected in every increment of the development lifecycle; it cannot be added to the end product. In every increment quality standard are taken into consideration, resulting high quality in every phase of development. If built-in quality is discarded, it will cause slower velocities and rework. (Scaled Agile, Inc. 2021a. Referred to Mar 16, 2021.)

Building high quality creates the following benefits: high customer satisfaction, better system performance, improved velocity and delivery predictability, and improved ability to scale, innovate and meet compliance requirements. It may require continuous trainings and commitments, but these can be seen as business investments. (Scaled Agile, Inc. 2021b. Referred to Mar 17, 2021.)

There are five dimensions in built-in quality presented in the figure below.

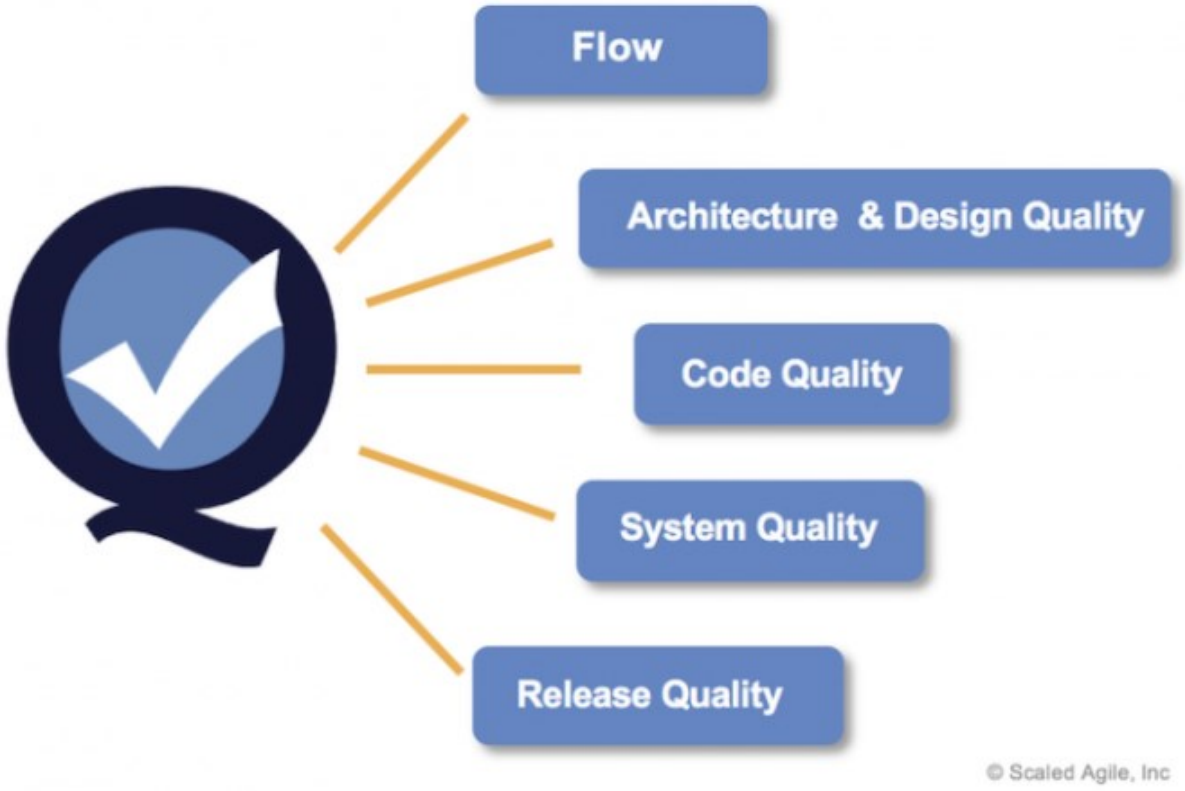

FIGURE 1. Five dimensions of Built-In Quality (Scaled Agile, Inc. 2021, referred to Mar 17, 2021)

Flow is an all along proceeding Continuous Delivery pipeline, where stories, features and code are being tested all the time to achieve development goals. This is called a test-first model and it is essential in agile, as testing is done as early as possible, many times and in multiple levels. This differs from the traditional waterfall model, where testing is done at the end, after all development has been finished. The purpose of this extensive testing is to prevent introducing new defects while making recurrent changes in agile development and to enable reliable and fast delivery. Testing can be seen as the most important factor of built-in quality. (Scaled Agile, Inc. 2021b. Referred to Mar 17, 2021.) The author has seen this in how developers do unit tests, and testers test first the individual stories and after that the whole feature, which consists of the verified stories.

Ideally agile teams create tests for features, stories, and code before or at the same time when the item is made to find defects and react to them immediately. Test-first model creates shift left testing, where the traditional V-model collapses as illustrated in figure 2. (Scaled Agile, Inc. 2021b. Referred to Mar 17, 2021.) Shift left testing will be covered in this thesis more in depth later.

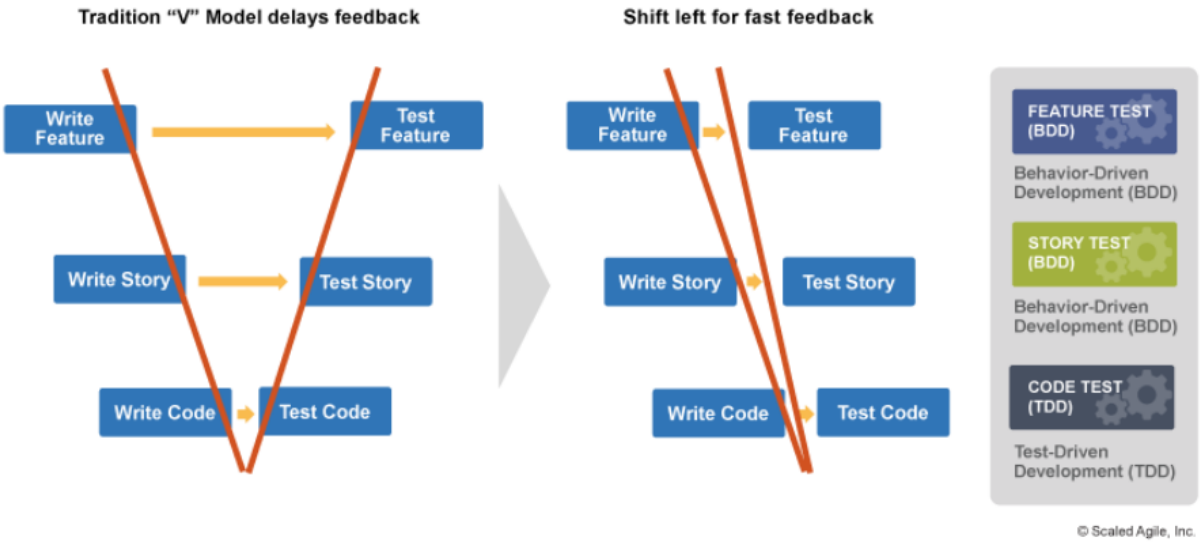

FIGURE 2. BDD and TDD shift testing left (Scaled Agile, Inc. 2021, referred to Mar 17, 2021)

Testing manually extensively takes time, which delays feedback. Therefore, automating tests are seen as a necessity and a way to save money. (Scaled Agile, Inc. 2021b. Referred to Mar 17, 2021.) The author has some minor experience of test automation but has already discovered how effectively it can find defects and detect changes in code. Manual test cases are used as basis for test automation scenarios. Currently test automation is lacking human resources at Enerim Oy, but there are plans of increasing the number of automated tests. The management supports this with changes, which will take place after summer.

Architecture and system quality define how well a system can support current and future business needs. When this dimension is of good quality, it makes testing the system easier, helps satisfying NFRs and requirements easier to implement. As markets and system requirements evolve, it is important that architecture and system quality evolve at same pace. It is impossible to predict future demands when building the system. In SAFe this is solved by modelling, previous experience, simulation, and trial. (Scaled Agile, Inc. 2021b. Referred to Mar 17, 2021.) Currently Enerim is going through architecture and system quality changes to meet the future needs. These include microservices and platform technology updates.

How the code works results in if the system works or not. Good quality code can be distinguished from others on how it can be modified reliably and quickly. Code quality can be ensured in several ways. In pair work two developers works on the same code, one is coding, and the other is giving feedback, and roles are being switched recurrently. Pairing creates and maintains quality as information, best practices and perspectives are being shared. Mutual learning supports the work of the whole team. (Scaled Agile, Inc. 2021b. Referred to Mar 17, 2021.) A practice in author’s team is to call for code review, where two developers comment and learn from each other.

Unit tests affect to code quality, too. In unit testing parts of the code are automatized to test them automatically when code changes. This ensures that new code does not break the existing code, so new features work together with the old ones. (Scaled Agile, Inc. 2021b. Referred to Mar 17, 2021.) In author's team, Scrum Master takes care that every developer has sufficient unit test coverage with the newly created code and notifies if unit tests are inadequate or missing completely in daily scrum meetings. At Enerim, SonarQube tool is also utilized to automatically detect bugs and vulnerabilities based on pre-set rules. SonarQube is checked in every daily scrum. It must be emphasized that developers test their own code extensively in localhost before freezing the code.

System quality means ensuring that the system is working expectedly, and development is doing the right things. PO agrees with the team members on the precise behaviour for a story or a feature. When making quick changes to the system, integrating them and testing, it is important to utilize automation, when possible, to stay in schedule. (Scaled Agile, Inc. 2021b. Referred to Mar 17, 2021.) In author's team, PO hostess backlog refinement at the end of each sprint. After going through the situation of the current sprint, PO introduces the stories of the future sprint and discussion on the behaviour starts there. If necessary, the discussion continues in sprint planning meeting.

Release quality is a matter of how to ensure that wanted feature has been implemented, tested and it meets the requirements. As a metric tool, definition of done is used. Definition of done means that the right thing has been done at the right time. Every story and feature have their own definition of done. Documentation must be ready and there cannot be any must-fix defects. Regression tests should be performed and there should be plans made when minor fixes are done and who will test them. Only after that a decision on going to production can be done. (Scaled Agile, Inc. 2021b. Referred to Mar 17, 2021.) The release testing of current increment will start next week, and it is scheduled to last four weeks. The author will be doing release testing at least three weeks and report her observations in the following chapters.

4.3 Week 3, March 22 – 26

Monday was used to study theory related to built-in quality from agile testing perspective. Release testing started on Tuesday with test environment issues. The new build did not include the process which was meant to be tested and server had multiple configurations concerning it and its feature locks. Communicating and solving the situation took half of the working day. This was one example on how working remotely affects negatively to the effectiveness. If everyone had been working from the office, direct communication between DevOps team and testers would have been quicker. However, testing was started after these difficulties and the author managed to verify two test cases. Altogether author's team had 38 test cases and four regression test cases to verify during the release testing.

On Wednesday and Thursday, increment planning meetings for the following increment were held. At Enerim, the working year is divided into four increments, and each increment is divided into several two- to three-week sprints. In the beginning of an increment, the content of the increment is introduced by the release manager. After the introduction, development teams hold meetings, where PO explains all the new features related to the team's own field and team decides what features they can commit on doing. In the summary meeting, POs introduce the commitments of their teams. This time the author paid special attention to testing perspective, which was mentioned several times in different contexts during the summary meeting. At Enerim, POs and Scrum Masters take testing well into consideration, but testers themselves also bring testing aspects actively to everyone's knowledge.

During those two days the release testing also went a bit forward with four verified test cases, because the author did not take part in all the meetings. One bug was found and reported to Jira.

The importance of testing and its impact on quality in agile cannot be overemphasized. Testers help preventing defects in code by asking questions. As testers are brought along in early phase of planning a new feature, they can ask questions on how to test the new feature or story. That is how built-in

quality is brought to the product. (Crispin & Gregory 2017a. Referred to Mar 15, 2021.)

According to a query performed by Crispin et al (2017b. Referred to Mar 22, 2021), there are seven key success factors in agile teams which help to achieve built-in quality. The most central success factor is to involve the whole team in testing. When a new feature is introduced, the team should consider how to test it and share the responsibility. There should be continuous willingness to improve, learn from each other and not to be afraid of making mistakes. In author's team, the pursuit of continuous improvement is handled in every sprint retrospective meeting, where the whole team reflects the finished sprint and what could be done better. However, the whole team participation could be improved from the current level, as most of the times it seems that only Scrum Master and PO are discussing about the new features.

The second success factor was adopting an agile testing mind set, which means in tester's perspective ability to ask questions, collaborate, and communicate the risks. Being curious, ready to learn continuously and taking responsibility for gaining more competence were considered essential skills as well. (Crispin et al 2017b. Referred to Mar 22, 2021.)

Black (2017, referred to Mar 22, 2021) agrees with Crispin et al when he considers that besides excellent testing skills, the key skills for an agile tester are interaction and effective communication abilities in team collaboration. Agile tester should actively keep in contact with business representatives and work as a part of a whole team responding frequently and early to customer feedback.

Test automation is important to be run daily to be aware of regression. Automation is not a bug finder; it is a change detector, and it gives feedback quickly. Well-functioning test automation will save time and find the real issues. The feedback received from regression test automation can be used to improve the whole team experience. In retrospectives, testing issues should be covered and the whole team should address it. Crispin et al (2017b. Referred to Mar 22, 2021) emphasize that testing should be made the whole team's issue.

Giving and receiving feedback is difficult because people have different ways of expressing themselves, some are direct, others indirect or somewhere in between. Therefore, succeeding in this can be seen as one remarkable success factor. Especially now in remote work, giving and receiving feedback has turned out to be even harder. It is good if, for example exploratory testing, findings can be presented in a visible form, because it creates transparency among the team. (Crispin et al 2017b. Referred to Mar 22, 2021.)

Understanding core agile practices is the fifth success factor when pursuing built-in quality. Understanding core agile practices is essential in agile. These include releasing in small increments often, cooperating between testing and coding, and implementing test automation to name a few. Agile testing mind set means improving constantly. (Crispin et al 2017b. Referred to Mar 22, 2021.)

Understanding what the customers want and collaborating with them is crucial in every business. If the feature is implemented incorrectly or does not meet the needs of the customer, then it can be considered as a failure. Testing can help to identify risks, which will result as trust. Testers can ask questions on what the worst scenario is and what will then happen to customers and to the reputation of IT Company. (Crispin et al 2017b. Referred to Mar 22, 2021.)

Remembering the big picture is a strength, which testers bring to the agile teams. It requires that testers should not have only a narrow view, instead they should also make business-facing tests. Testers should consider how the feature needs end users and customers' needs. If the feature cannot be easily used by everyone, then it is not creating the expected value. (Crispin et al 2017b. Referred to Mar 22, 2021.)

4.4 Week 4, March 29 – April 1

On Monday morning the author got familiarised with shift left testing. On the afternoon release testing continued and the author completed her part of the test cases on Wednesday afternoon. On the following week there will still be some test cases made by the other manual tester of the team that will require verifying in release testing.

There were some issues also in this week's release testing. Setting up the suitable test data for one test case took half a day, because synthetic data was missing from the test environment. Another issue was that other test case did not work according to specifications anymore and solving the situation took half a working day. Finally, it was found out that a bug was reported of the case earlier on the same day and test case was put as 'fail'. This again showed the downsides of remote work because PO was the only one who knew about this newly reported bug. If the team had sat in the same premises, the information about this would have most likely spread and had saved time from the author.

Thursday morning started with general meeting, where the CEO of Enerim introduced a project concerning quality issues at Enerim. The areas which the project covers are related to this thesis. The author decided to contact the project manager and ask if this thesis could cover still some field which would benefit Enerim.

Cost savings are a thing that any organization is interested in, and which built-in quality can affect. Especially this was high-lighted in theory concerning shift left testing.

Shift left testing can be seen as precondition of other agile practices, such as DevOps, CI, DBB and TDD, to become reality. Testing should start earlier in the SDLC, even before any code has been written as illustrated in Figure 3. The idea is to shift the mind set of testing from QA to QE, which stands for Quality Engineering. The foundation to shift left testing is laid with automation, machine learning and AI. These together free the time of testers from traditional manual testing to start more strategic planning of their work. (Palamarchuk 2018, referred to Mar 29, 2021.)

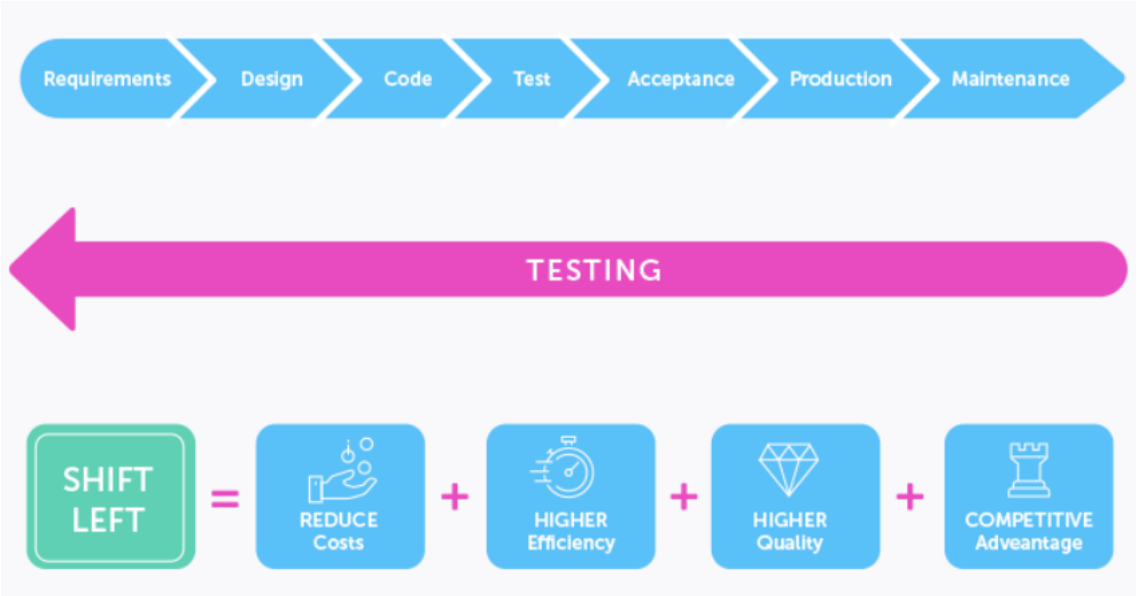

FIGURE 3. The idea and benefits of shift left testing (Palamarchuk 2018, referred to Mar 29, 2021)

Shifting left is not a new idea. Already in the 1950's developers noticed they should test earlier on, while writing the code. There were no dedicated testers at the time. According to SmartBear (2021, referred to Mar 29, 2021), it was unfortunate misunderstanding of the waterfall model that there became separate, independent testing teams and a gap between development and testing.

It is a known fact that the sooner a bug is discovered, the cheaper it is to fix. Shift left testing makes it possible to detect bugs in real time due to testing with each build, especially unit tests. Testing in frequent portions facilitates locating small bugs early with low costs. (Palamarchuk 2018. Referred to Mar 29, 2021.)

This is also what Nader-Rezvani (2018, referred to Mar 9, 2021) is explaining in Figure 4. When defects are discovered in unit tests, the cost is minimal compared to cost, if the defect is discovered in end to end or acceptance testing.

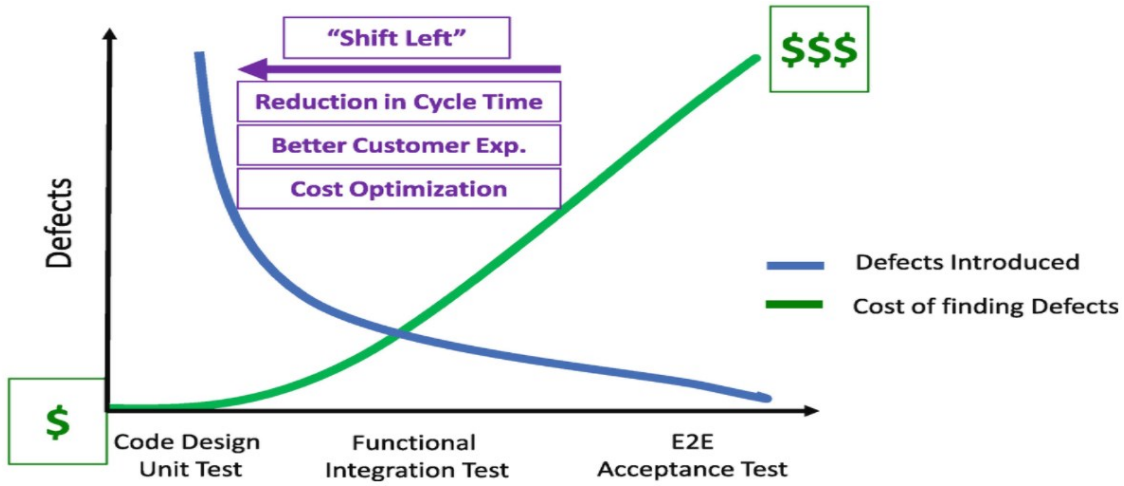

FIGURE 4. Shifting defect detection left in SDLC (Nader-Rezvani 2018, referred to Mar 9, 2021)

Shoham (2018, referred to Mar 29, 2021) uses an example to demonstrate the snowball effect of the bug fixing. When designing a requirement, fixing a bug costs one dollar. Same bug fixed by developer costs 100 dollars and if it is found by tester, the cost is already 1,000 dollars. If the bug gets to production phase, the cost of fixing it rises to 10,000 dollars. This is the reason why shift left testing is important.

Agile teams also benefit from shift left testing and the use of automation with increased quality and efficiency. The test coverage increases with reduced human error rate, built-in security and code quality are easier to check, and reducing number of bugs which the end user may encounter in production. This also results to built-in quality as bugs are found early rather than in the end when there is limited time to fix them. (Palamarchuk 2018, referred to Mar 29, 2021.)

When an IT company is shifting left with their testing, it may also attract skilled workforce compared to competitors (Palamarchuk 2018, referred to Mar 29, 2021).

According to the article, the cons of shift left testing are risk of bottlenecks, resistance to change and difficulty to adapt to new way of working. Even though

shift left testing should prevent bottlenecks, those may occur due to complexity of environments and composite applications. Utilizing service virtualization such as Jenkins, may help to reduce these bottlenecks. (Palamarchuk 2018, referred to Mar 29, 2021.)

Switching to shift left testing may require drastic changes from the team and the whole team must commit to the change. Team members are used to their own roles and responsibilities in the team, whereas shifting left requires discussion on changes on for example processes, roles, tooling, and skills. (Palamarchuk 2018, referred to Mar 29, 2021.)

4.5 Week 5, April 6 – 9

The quality project manager had answered to the e-mail sent last week and he suggested a brief meeting concerning the thesis topic. Sadly, by the end of the week, he had not had time for the meeting nor replied to e-mails.

During Tuesday morning, the author participated release testing and verified one test case with SoapUI. This required modifying the Soap request in a new way and there was no-one available to ask. However, the issue was solved, and the case verified. The rest of the day was used to study about shift left testing. It was surprising to discover how few sources were academic and how many commercial.

On Wednesday release testing continued with one verified test case and one discovered bug. The author had to verify from three other teams' testers was the bug caused by wrong use or was it an actual bug. This together with bug reporting took time.

Thursday was the last day of release testing and finally everything worked fluently with test environment, test data, and test cases were accurate too. Two test cases were verified and there was still spare time to investigate what should be done for the thesis on the following day. Friday was used mainly for finding good sources and writing notes. This week it was educative to see how comprehensively the other manual tester in author's team writes his test cases.

Learning from his examples will be beneficial for the author's professional growth.

When the author first read about shift left testing, it was unclear how to execute tests even earlier than currently. There were many questions raised especially about even quicker cycle in release testing than currently. Pugh (2020, referred to Apr 9, 2021) explains that shift left testing is not about executing the test earlier, rather it means writing the test earlier. There should not be implementation before there is a test, which matches the requirement. Tests and requirements are interrelated, you cannot have one without another.

One method to achieve Built-in quality is test-first, meaning the implementation of TDD and BDD. On top of these Pugh (2020, referred to Apr 9, 2021) explains about HDD, which stands for Hypothesis Driven Development. HDD is used in feature tests. The process starts with idea of a new feature with hypothesis that customer will use that feature. Then it should be tested if the right thing is being build. The criteria are that if customer does not need this feature, then the wrong thing is being build. Also, a bug discovered in this phase is cheapest to fix.

In shift left, the idea is to test in every phase of the SDLC, instead of merely testing at test phase. This will reduce loopbacks and delays related to testing. This is presented in Figure 5.

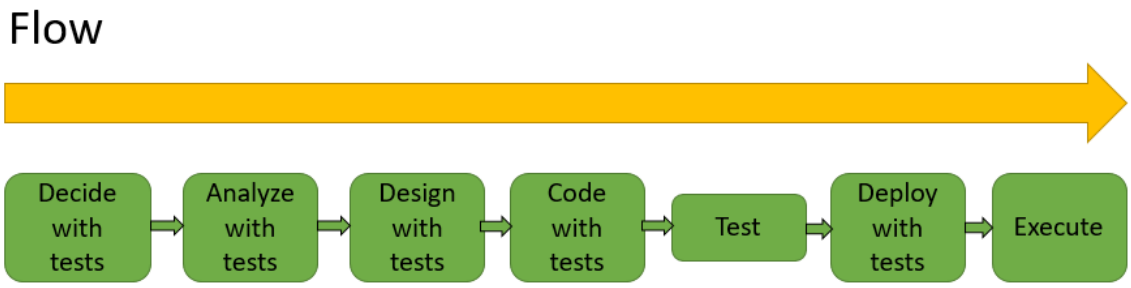

FIGURE 5. SDLC flow with shift left testing decreases the amount of testing in actual testing phase (Pugh 2020, referred to Apr 9, 2021)

One of the reasons for the importance of shift left thinking is that it will help with test automation. When the test is written before any coding is done, then automation is easier to do. If it is done after implementation, then automation must be done via user interface, which is often time-consuming and fragile. (Pugh 2020, referred to Apr 9, 2021.)

Pugh (2020, referred to Apr 9, 2021) discusses the triad when aiming to achieve built-in quality. Triad is the discussion between tester, developer, and customer, latter referring usually to PO. This was also conversed by Crispin et al (2017a. Referred to Mar 15, 2021) when they were talking about the “power of three” discussions. The idea is to create tests to a new functionality before implementation to be sure the right thing is being build. The triad is responsible for the quality of the product.

TestingWhiz (2017, referred to Apr 6, 2021) introduces ways to get started with shift left testing and receive the benefits of it. As organizations identify and plan the whole testing lifecycle, testers and developers understand the objectives, tasks and expected outcome of the project. One way of achieving this is to include testing requirements to planning and requirements specification phase. The second way is to integrate project management, development, and operation processes with testing. This will help in testing cycle time and effort estimation, and in understanding when and in which stages testing should happen.

Having defined quality standards and controls for all stages of SDLC will help to identify deviations from the expected outcome, but more importantly check whether the development lines up with QA. As these standards and controls are well set, it will be easier to take corrective actions in time. Planning departmental releases wisely and before-hand is recommendable while shifting to left in testing. This will help in resource planning and understanding the possible delays. (TestingWhiz 2017, referred to Apr 6, 2021.)

Test automation plays key role in successful adaptation of shift left testing. With test automation, testers and developers can automate entire build to test processes, which will increase continuous delivery, create better integration between processes and bring more confidence to each release. (TestingWhiz 2017, referred to Apr 6, 2021.)

Regardless of the indisputable benefits of the shift left testing, it is not yet widely implemented. Shahaf (2020, referred to Apr 9, 2021) has collected opinions from five experts on what hinders organizations from shifting left in testing. Lack of leadership support can result to organization not to be able to shift left. The reasons why leadership support is lacking comes from the plan to meet deadlines and delivering the functionality. Any additional, delaying work will meet resistance. Also changes in processes and tools, for example in CI/CD pipelines or developer environments, may be seen as productivity killer, if R&D is not involved in the tool selection.

Insufficient testing capacity is keeping testing teams from adopting shift left or making them sceptical. If a tester is already doing testing according to existing plan and then new project is requiring early testing, the workload is doubled. Some may as well still ponder, if all cases could be found and covered right in the beginning of testing. (Shahaf 2020, referred to Apr 9, 2021.)

Developers can be overloaded with work too, and they do not have resources to support testing. Usually, one reason for this is that testers are finding lots of bugs affecting the quality and developers work hard to get them fixed. In this kind of situation developers do not have the time to test as well. (Shahaf 2020, referred to Apr 9, 2021.)

Testers should be involved in the early phase of design and development. If this is neglected and test automation is thought to be the main testing solution, then shift left testing does not produce positive results. Everything cannot be automated, and it is important to get testers opinions in the opening phase. The scope of testing should also be well-defined. If the scope is too broad, there is no capacity to integrate it to the build cycle to shift left. (Shahaf 2020, referred to Apr 9, 2021.)

4.6 Week 6, April 12 – 16

This week author's child was sick, which naturally affected how much the author could concentrate on thesis work. The author analysed her current work as a test engineer and thesis objectives and wrote about those. It was decided together with the supervisor that the planned interviews will be left out from the thesis.

On this week, the objective was to understand how to affect organization's QA and what the relation between traditional QA and agile development is.

The purpose of software quality assurance is to minimize the risk of providing software that does not meet the wants, needs, and expectations of stakeholders within schedule and budget (Laporte & April 2018a. Referred to Apr 14, 2021). Usually, QA is expected to come with high costs, and therefore implementing it meets resistance. But delivering non-quality software costs a lot more. The most essential question is how much it will cost not having quality management system instead of how much quality management system costs. However, the cost of quality cannot be calculated in the same way in all organizations. (Laporte & April 2018b. Referred to Apr 14, 2021.)

Laporte et al (2018b. Referred to Apr 14, 2021) point out the organization culture factors that boost software quality. Good team spirit makes everyone commit to their work and follow agreed processes also on busy times. The input of skilled managers cannot be underestimated, and they should set a good example for the whole team. Also, the skills of the other members of the organization play a significant role, because if the people are not competent, the efficiency will decrease accordingly.

Effective communication plays a key role in organization's software quality. When customers, managers and the colleagues can communicate on issues, then the quality also satisfies better all parties. The organization should also recognize and value initiatives to improve quality. (Laporte & April 2018b. Referred to Apr 14, 2021.)

Leadership team should communicate organization's values and principles, and ensure they are respected by the personnel. Clearly defined roles and responsibilities guide the daily work of the personnel. (Laporte & April 2018b. Referred to Apr 14, 2021.) Enerim is going through organizational changes and one aim is to define the responsibilities and establish new positions to divide the workload more evenly.

Lastly, it is important to involve the customer in the project because this will have the most impact on software quality. Then organization can be sure it is building the right thing, which the customer wants. If customer representative is involved throughout the project, then any ambiguities can be discussed along the way and need for costly late fixes will decrease. (Laporte & April 2018b. Referred to Apr 14, 2021.)

Itkonen, Rautiainen & Lassenius (2005, referred to Mar 5, 2021) have discovered problems in agile development versus traditional QA. It should be noted that SAFe was invented in 2011, many years after this article was written.

According to Itkonen et al (2005, referred to Mar 5, 2021) agile methods have been well adapted on the developer level, but testing practices remain challenging. Hence, there was a need for this study to compare traditional QA with agile QA. In agile development, rapid release cycles put pressures to testing with fixed deadlines and does not allow prolonging the testing period if more defects are found than estimated. At Enerim, this has been faced several times in service pack testing and in release testing. Too optimistic predefined schedules can affect the quality when there is not enough time to test all possible scenarios. The other option is to work overtime, but in a long haul it is not a sustainable solution. However, testers have been inquired if they need extra time for completing testing and that time has been arranged.

Changing requirements can happen late in agile development, which causes challenges for QA. This is because in traditional QA, the specifications work as a basis for test design. Agile development also relies on direct communication between business people and developers. Documentation is kept in minimum. However, traditional testing has relied on documentation, and it is difficult to

know the expected results, if these are only in the heads of developers and business people. In agile, testing cannot be left as the final phase in SDLC, since working software is wanted to be released to customers often. (Itkonen et al 2005, referred to Mar 5, 2021.)

Itkonen et al (2005, referred to Mar 5, 2021) consider problematic that in agile developers would test their own code or that organization would test their own code. They base this on the fact that one of the fundamental principles of testing is independency. Having the customer involved in development and acceptance testing can also pose problems but will work if the customer has the required skills and expertise.

Testing should be performed with valid and expected as well as invalid and unexpected results, and this can be difficult to achieve by a developer. If you are expecting to see only the correct behavior, then failures are more difficult to detect. All testing should not rely on unit tests or test automation, because also manual testing is required. Even though software passes all unit tests, it can still be broken due to inadequate manual testing. (Itkonen et al 2005, referred to Mar 5, 2021).

5 REFLECTION

The objective of this thesis was to make observations on the current level of SAFe model's built-in quality at Enerim Oy. Observations were planned to be gathered as improvement suggestions and ideas on how to carry on this work in the future to increase the overall built-in quality at Enerim Oy.

Themes, which were most mentioned in literature as a solution to achieve built-in quality are that in organizations, testers should be brought along in planning early and the whole team should be involved in testing. Also, collaborating with customers to ensure you are building what the customer wants and needs, and increasing the use of test automation will improve quality. It was emphasized that built-in quality will bring cost savings as bugs are being detected early.

Currently at Enerim the testers are brought along when planning the increment and individual sprints. The whole team is involved in testing. There are plans on increasing the utilization of test automation starting from autumn 2021. The author has limited knowledge on how customer collaboration works currently, but the new quality project is meant to respond to customer demands.

Nevertheless, it was noticed that some issues may need some further review. Shift left testing was not implemented in TDD, BDD or HDD. Usually when testing stories, test cases are written while doing the actual testing. Testers' participation to HDD is limited. This is because testing resources are not currently adequate to shift fully left. Triads or power of three discussions are not in use either. However, considering the current number of meetings caused by SAFe model, they can be heavy to implement. As Enerim is lacking test lead now, it is unclear to the author is anyone planning testing life cycle currently. Quality standards and controls seem also to be lacking or at least they do not show in author's daily work.

Increasing the level of test automation may provide a solution to ease the work of manual testers, though in the beginning it will require familiarization and time. Test automation could be applied in end-to-end testing, which is currently utilized limitedly. Though it must be noticed that long test automation scenarios

break easily. It must be noted though that test automation needs vary between different teams, so bringing in one general way of working will not create the best benefits. Instead, each team should plan their needs carefully before starting to implement test automation further. Hiring a person to lead test automation process could be recommendable.

Moreover, the approach to testing should be more proactive instead of reactive, which would help if shift left testing is decided to be implemented. This will require that there is a person coordinating manual testing. As remote work still seems to continue for a while and lack of direct communication poses productivity challenges, this person will take care of communicating testing issues not only for testers, but also for the other parts of the organization.

The biggest success of this thesis was keeping the tight schedule effectively and sticking to the original plan. Only few minor adjustments were made along the way. This is a good proof of project management skills. Author's personal learning objectives were achieved well. She got familiarized with agile testing and built-in quality of SAFe model, and benchmarked good test case writing practices, which she will continue doing after this thesis process is over. It is difficult to self-estimate how much own level of quality increased during release testing, but more attention was paid on it.

Making the thesis was relatively easy, since this was the second thesis written by the author, so the process was already familiar. Carefully planning and orientating to the work helped a lot with schedule pressures. Having a clear daily and weekly routine supported how the work progressed. The only change in the plan was that originally planned interviews were left out. This was partially because of the tight schedule and author's child being sick, which affected the working pace.

Thesis process revealed to the author how much there is still to be learned about testing and especially agile testing. The first year at Enerim Oy has been learning how EnerimCIS works, getting familiar with working according to SAFe and learning to use the tools. The author is considering taking ISTQB certification tests to become more qualified in testing.

Criticism can be posed to the fact that in the thesis the author could not combine the theory of built-in quality to release testing as well as she would have wanted to. As the theory of release testing was gathered, it was noticed that it handled exploratory testing. Including that did not suit the original scope of the thesis, so it was therefore chosen to be left out.

Current research concerning built-in quality of SAFe model itself is modest. The further research possibilities could include querying how and in which level different organizations implementing built-in quality practices are and what are their findings on this journey. It is also clear that test automation plays key role in this but getting the most benefit out of test automation remains problematic in many companies. This could also provide good research problem for future research. Shift testing left provides solutions on testing cost and quality issues, however it is not widely and fully implemented. This could also be a good starting point for future research, but it should be noted that literature covering shift left testing is even more modest than the one on built-in quality.

REFERENCES

Black, Rex 2017. Agile testing foundations: an ISTQB Foundation level agile tester guide. Referred to Mar 22, 2021. https://learning.oreilly.com/library/view/agile-testing-foundations/9781780173368/?ar/?orpq&email=%5Eu

Crispin Lisa, Gregory Janet 2017a. Shared Understanding through Early Testing. Agile Testing Essentials LiveLessons. Video. Addison-Wesley Professional. Referred to Mar 15, 2021. https://learning.oreilly.com/videos/agile-testing-essentials/9780134683287/9780134683287-ATEL_01_07_00

Crispin Lisa, Gregory Janet 2017b. Ingredients for Success. Agile Testing Essentials LiveLessons. Video. Addison-Wesley Professional. Referred to Mar 22, 2021. https://learning.oreilly.com/videos/agile-testing-essentials/9780134683287/9780134683287-ATEL_01_07_00

Itkonen Juha, Rautiainen Kristian, Lassenius Casper 2005. Towards Understanding Quality Assurance in Agile Software Development. Referred to Mar 5, 2021, https://www.researchgate.net/publication/242662177_Towards_Understanding_Quality_Assurance_in_Agile_Software_Development

Laporte Claude Y., April Alain 2018a. Software Quality Fundamentals. Software Quality Assurance. Referred to Apr 14, 2021. https://learning.oreilly.com/library/view/software-quality-assurance/9781118501825/?ar%2F%3Forpq=&email=%5Eu

Laporte Claude Y., April Alain 2018b. Quality Culture. Software Quality Assurance. Referred to Apr 14, 2021. https://learning.oreilly.com/library/view/software-quality-assurance/9781118501825/?ar%2F%3Forpq=&email=%5Eu

Nader-Rezvani, Navid 2018. Agile Quality Test Strategy: Shift Left. An Executive’s Guide to Software Quality in an Agile Organization: A Continuous

Improvement Journey. Referred to Mar 9, 2021. https://learning.oreilly.com/library/view/an-executives-guide/9781484237519/?ar/?orpq&email=%5Eu

Palamarchuk, Sofía 2018. Why Shift-Left Testing? Pros and Cons. Referred to Mar 29, 2021. https://www.testim.io/blog/shift-left-testing/

Pugh, Ken 2020. Shift Testing Left to Build in Quality. Video. Referred to Apr 9, 2021, https://www.youtube.com/watch?v=e4oB_9ThMao

Scaled Agile, Inc. 2021a. Core Values. Referred to Mar 16, 2021, https://www.scaledagileframework.com/safe-core-values/

Scaled Agile, Inc. 2021b. Built-in Quality. Referred to Mar 17, 2021, https://www.scaledagileframework.com/built-in-quality/

Shahaf, Tzvika 2020. 5 Reasons Why Shift Left Testing Is Not Happening. Referred to Apr 9, 2021, https://www.perfecto.io/blog/shift-left-testing

Shoham, Shani 2018. 5 tips for shifting left in continuous testing. Referred to Mar 29, 2021. https://www.atlassian.com/blog/software-teams/5-tips-for-shifting-left-in-continuous-testing

SmartBear 2021. What the Shift Left in Testing Means. Referred to Mar 29, 2021. https://smartbear.com/learn/automated-testing/shifting-left-in-testing/

TestingWhiz 2017. 10 Steps to Get Started with Shift Left Testing. Referred to Apr 6, 2021. https://www.testing-whiz.com/blog/10-steps-to-get-started-with-shift-left-testing