Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Emil Elton Nilsen

Practices and Challenges in Assuring Software of Quality in the Norwegian Public Sector

Master's thesis in Computer Science
Supervisor: Babak A. Farshchian
June 2023

NTNU
Norwegian University of
Science and Technology

Emil Elton Nilsen

Practices and Challenges in Assuring Software of Quality in the Norwegian Public Sector

Master's thesis in Computer Science
Supervisor: Babak A. Farshchian
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Practices and Challenges in Assuring Software of Quality in the Norwegian Public Sector

Emil Elton Nilsen

June 2023

Preface

The following document is a multiple case study on the practices and challenges of software quality assurance in the Norwegian public sector. Written as the master's thesis at the master's degree program in computer science at the Norwegian University of Science and Technology (NTNU). The multiple case study was conducted by Emil Elton Nilsen, a master's student at NTNU and Software Developer at the Norwegian Labour and Welfare Organization (NAV). The master's thesis was supervised by Babak A. Farshchian, Associate Professor at the Department of Computer Science at the Norwegian University of Science and Technology (NTNU).

I would like to thank my supervisor Babak for his honest and useful feedback during the writing of this thesis. His deep knowledge of both academic writing and digitisation in the Norwegian public sector has been highly valuable in helping me write the thesis, and guiding me throughout the process. I would also like to thank the employees at NAV who helped me with the study, especially Parastoo Mohagheghi, scientist and coordinator of science at NAV. Her help in acting as a second supervisor has been of great help in giving a second perspective on the thesis.

Abstract

This study aims to uncover the practices and challenges of software quality assurance in the Norwegian public sector. A multiple case study with interviews of employees from certain agencies in the Norwegian public sector was used to obtain the results of this study. Where the employees interviewed have a strong influence or ownership over technology in the agencies chosen. The results show several practices being used to assure the technical quality of software and different software development methods used to assure quality. Challenges in software quality assurance relating to organisational aspects in the Norwegian public sector were discovered, such as budgeting and lack of resources. Forcing the agencies to use the criticised project wizard delivered by the Norwegian digitisation agency. Relevant scientific literature suggests that the practices followed by the Norwegian public sector are recommended for assuring quality software. However, the challenges presented in the study must be addressed before the Norwegian public sector can elevate the quality of its software.

Contents

Tables

Glossary

Kubernetes System for the management of containerised applications [1]. 34

Microservices Architectural pattern where a piece of software is split up in a collection of highly specified services [2]. 28, 34

Mob programming Software development technique where the entire development team work together in the same space at one computer [3]. 35, 48

NAIS Application platform developed by NAV and used for developing and maintaing typical "NAV-applications" [4]. 34

OpenAPI Standard for interfacing with HTTP API's [5]. 31

Pair programming Software development technique where to developers work together at one computer [6]. 7, 28, 35, 48

Publish-subscribe Pattern for messaging between computer systems. 31

ROS-analysis Analysis of risks and vulnerabilities in a piece of software. 27

Service-oriented architecture Architecture that focuses on discrete services rather than monolithic design [7]. 34, 44

Technical debt The cost of fixing an easy but limited solution instead of the more time consuming, but better solution [8]. 7, 29, 30, 37, 42, 48, 51

Chapter 1

Introduction

1.1 Motivation

Software quality is how well a piece of software is measured against a chosen set of non-functional requirements, chosen based on the software's functional requirements [9]. These non-functional requirements are defined as when it can be determined that a piece of software delivers value [9]. Software quality assurance is the planning, controlling, and executing of processes which measure these non-functional requirements or other defined quality standards defined by a product team or at the organisational level [10][11].

As of 2020 in the Swedish public sector, 88% of agencies claim to use agile methodologies in their software development to assure quality [12]. Out of this 88%, only 45% agree/strongly agree that they include the use of test automation to assure quality in its software, even though papers suggest that test automation and continuous testing are necessary is important to meet user requirements in an agile development workflow [13]. Other than this, little research has been done on software quality assurance in the context of the Norwegian public sector, nor the public sector in general.

The Norwegian government is pushing that the Norwegian public sector should move towards offering connected social services, provided by a shared digital platform throughout the Norwegian public sector [14]. This includes the municipalities, counties and the Norwegian state to share data and increase cooperation by using digital tools [14]. As the digital service in the Norwegian public sector moves closer together in responsibility and services, it can be speculated that they will have the same non-functional requirements which need to be measured to assure software quality. This has been described as being the case for similar systems, such as Sharing Economy systems. Having non-functional requirements elicited from requirements such as information security, privacy, and personal data

protection due to their importance in creating trust [15]. These requirements are also described as important in public sector software for increasing trust among its users [16], and trust among its users being of importance for a well-functioning Norwegian public sector [17].

The researcher's role as a software developer in the Norwegian Labour and Welfare Administration (NAV) has given him perspectives on software quality assurance in the Norwegian public sector. Perspectives such as agility are important so that a piece of software can change with ease due to unforeseen events or user feedback. As well as organisational culture is important in software quality assurance, with a culture for psychological security where employees are able to learn from their mistakes and not be punished. However, the agency of NAV is large, handling about 1/3 of the Norwegian national budget [18][19] and employing about 22 000 [20]. Which could mean that the researcher's views are not representative of all employed at NAV, or the Norwegian public sector at large.

The objective of this study is to understand what the IT professionals with high influence or ownership over technology in certain Norwegian public sector agencies consider important in assuring quality in the software being used to serve their users. What challenges the agencies might be encountering in the quality assurance of their software. And how these findings compare with the Norwegian government's digitisation strategy. Leading to the following research questions:

- **RQ1:** What practices do agencies of the Norwegian public sector use to assure quality in its software?
- **RQ2:** What challenges are agencies in the Norwegian public sector encountering in the quality assurance of its software?

1.2 Report outline

The rest of the report is structured as follows.

Chapter 2 - Theoretical Background Introduces scientific literature relevant to this study.

Chapter 3 - Case Background Introduces background knowledge needed for the case study of this report.

Chapter 4 - Method Contains a description of the methodology used to obtain the results of the case study.

Chapter 5 - Results Presents the results collected in the study.

Chapter 6 - Discussion Contains a discussion of the findings of the study and the scientific literature in chapter 2 in relation to the objective of this study.

Chapter 7 - Conclusion Concludes the study and its findings.

Chapter 2

Theoretical Background

This chapter introduces scientific literature relevant to the study and its discussion about practices and challenges in software quality assurance in the Norwegian public sector. Previous research has sought to understand how the quality of the software is assured, the relationship between different software development methodologies in organisations and the quality of their software and what benefits or disadvantages this brings.

Not a lot of this research is done with the Norwegian public sector as a context. The context of the public sector is important, as it is said to differ from the private sector in several aspects, such as organisational structure, corporate governance structure, the capacity of IT, organizational IT competence, lack of market competition and stability, and government regulations and policies [21].

Budgetary constraints in the public sector are more challenging. Resulting in projects that are perceived as risky being less likely to be considered, even if these projects are good investments [21]. The public sector is said to be obliged to address government agendas, resulting in the public sector potentially being more susceptible to political changes [21]. Political cycles can cause disruption to top-level management, and priorities in the departments can change significantly with each new administration [21]. Public sector policy processes can make results and decision-making difficult for IT managers in the public sector and could reduce implementation success [21]. A lack of in-house shortages has led some public sectors to hire contract staff and/or outsource IT functions [21]. Public sector agencies are able to participate in cross-agency IT governance, however, being limited in their effectiveness due to the structure of ministerial portfolios and administrative control of agencies [21].

DevOps is a software development methodology used to integrate development and operations to shorten the development life cycle [22]. The most recent factor

for assuring quality in software is the implementation of DevOps in an organisation. This is due to the ability to combine continuous integration with automatic testing [23][24][25]. DevOps also help with assuring quality as it increases cooperation and knowledge sharing between employees, leading to an overall improvement of a "quality culture" in the organisation [24][26]. However, it was stated that the effects of testing in DevOps environments had not been studied systematically by scientific literature [27][28].

No literature has so far verified any specific effects on software quality by introducing DevOps to the public sector, however, it was mentioned the expected benefits of introducing DevOps to the public sector [26]. DevOps is expected to bring the different agencies in the public sector closer, increasing knowledge culture and collaborative work [26], which is shown to increase software quality [24].

It is also shown that introducing DevOps to an organisation can hinder the assurance of software quality if not addressed properly [27]. It is emphasised that continuous testing is not the same as test automation and manual testing is still needed to achieve quality [27]. Product teams have to rethink their roles and responsibilities towards quality assurance, by shifting from specific roles to raising the quality assurance competence of all members of the product team [27]. The product teams have to take end-to-end responsibility for the content being produced, meaning that the entire product team must be involved in the quality assurance of the content in its entire life [27].

Collaboration between the development and operations components of an IT function is crucial to ensuring that problems are fixed and the quality and resilience of the software are enhanced [29]. And the members of cross-functional IT DevOps teams have to broaden their skill set and adopt a general competency set. [29].

Several other factors which are related to DevOps, but not necessarily directly connected are considered to be important in improving software quality. Automation is considered an important factor in increasing software quality [24]. Culture is also considered important, due to it boosting software quality through integration, evaluation and sharing of knowledge [24].

It is shown that assuring quality in software reduces costs and increases security [30]. Continuous focus on quality and testing in the development of software reduces costs, as the cost of fixing bugs after the software is delivered to the customers is higher than before [30]. This also implies security, as weak security of software can have larger consequences and costs to fix if discovered after the software is delivered to the customer [30].

Certain software development practices are said to be beneficial for assuring software quality. In the development of complex tasks, techniques such as Pair programming are said to be beneficial [31]. Registering Technical debt is seen as important for improving software quality, as it enables organisations to have more optimal use of their resources [32], and enables development teams to identify and repay Technical debt in a timely fashion [32]. The metrics collection of software in runtime helps us to make meaningful estimates for software products and guides us in taking managerial and technical decisions [33].

Chapter 3

Case Background

This chapter introduces relevant background theory for understanding the cases of the study, its findings and its discussion. The agencies studied and relevant organisations in the Norwegian public sector are presented. And processes and practices used for software development and software quality assurance in the Norwegian public sector are presented.

3.1 Norwegian Public Sector

The Norwegian public sector is a collective term for the state and municipal administration in Norway. It is an entity which includes public goods and government services, such as military, infrastructure, public education, and health care. The public sector differs from the private sector, as it does not concern privately owned organisations, nonprofit organisations or households [34][35].

3.1.1 Norwegian Labour and Welfare Organization (NAV)

The Norwegian Labour and Welfare Organisation, or NAV, is a central agency in the Norwegian public sector which provides social and economic security to the citizens of Norway while encouraging a transition to activity and employment [36]. NAV is partnered with the municipalities in Norway and has about 22 000 employees, where about 15 500 are employed in the Norwegian state, and 6 500 are employed in the municipalities [20].

3.1.2 Norwegian Tax Administration (Skatteetaten)

The Norwegian Tax Administration, or Skatteetaten, are responsible for ensuring that taxes and other claims in Norway are correctly assessed and paid. It also ensures that the Norwegian National Registry is up to date [37]. As of 2019, Skattetaten has about 6 300 employees [37].

3.1.3 Norwegian Food Safety Authority (Mattilsynet)

The Norwegian Food Safety Authority, or Mattilsynet, are responsible for promoting animal and human health in Norway by supervising the different food chains in Norway. All from the farms or fisheries, to the food Norwegians eat from said farms and fisheries [38]. As of 2023, Mattilsynet has about 1 300 employees, stationed in offices in all regions of Norway [39].

3.1.4 Norwegian Digitisation Agency (Digdir)

The Norwegian Digitisation Agency, or Digdir, are responsible for the development and maintenance of development and services used across the Norwegian public sector. Digdir also has the responsibility of strategic coordination of digitisation across the Norwegian public sector [40].

3.1.5 Norwegian Public Service Pension Fund

The Norwegian Public Service Pension Fund delivers pensions, insurance, and loans to employees in the Norwegian public sector and other organisations connected to the Norwegian public sector. In total the Norwegian Public Service Pension fund manages the pension of 1 Million Norwegian who are, or have been employed in the Norwegian public sector [41].

3.1.6 Norsk Tipping

Norsk Tipping is a gambling company owned by the Norwegian government and managed by the Norwegian Ministry of Culture. Norsk Tipping has a monopoly on a range of gambling games and their rules in Norway [42].

3.1.7 Entur

Entur is an organisation responsible for providing a national travel planner for Norwegian public transit, including, buses, trams, trains, subways etc. Entur provides a single interface to buy multiple tickets from multiple public transit providers [43].

3.1.8 Norwegian Directorate of Agriculture

The Norwegian Directorate of Agriculture is responsible for implementing the agricultural policy of the Norwegian government. And in general, facilitates the agriculture and food industries in Norway [44].

3.1.9 Nortura

Nortura is one of Norway's largest providers of food, mostly focusing on different red meats and chicken. Nortura has over 30 production units and yearly processes 350 000 tonnes of meat, which is delivered to grocery stores and hotels [45].

3.1.10 Norwegian Broadcasting Corporation (NRK)

The Norwegian Broadcasting Corporation, or NRK, is a media company owned by the Norwegian government, which provides media on TV, radio, and the Internet. NRK is funded by Norwegian taxpayers through the Norwegian television license [46].

3.1.11 The Brønnøysund Register Centre

The Brønnøysund Register Centre is an agency which has the responsibility of providing a range of registries storing information about Norwegian society, such as organisations operating in Norway [47].

3.1.12 Norwegian Public Sector Project Wizard

In order for the agencies of the Norwegian public sector to obtain funding for larger projects, a project wizard is delivered by the Norwegian Ministry of Finance. The project wizard is a set of requirements that any digitisation projects with a total cost equal to or above 300 Million NOK have to follow. The requirements include the project to be externally evaluated before being presented to the Norwegian government and the Norwegian Storting. This is to avoid faulty investments and keep control of spending in the project, assuring efficient use of the funding from the Norwegian taxpayer [48].

The project wizard has the following phases which are to be followed by any project: (I) Idea, (II) Concept, (III) Preliminary project, and (IV) Implementation.

Idea The idea phase is used to identify any problems that could be suited as a project in the Norwegian public sector and consider if the problem should be investigated further. For the larger projects, the result of the idea phase will become a mandate for the concept phase [48].

Concept In the concept phase, the problem is described in detail and how the project should solve the problem. The future needs of Norwegian society and the goals of the performed projects are also described. Multiple concepts of the project should also be proposed in this phase, which are then subject to a socioeconomic analysis. This is to decide which concept is the best and what circumstances are needed for further planning to be successful [48]. This choice of concept is performed by independent, external experts before a concept can be approved by the Norwegian government.

Preliminary project The preliminary project phase then has the responsibility of setting the basis for management and estimates of costs for the chosen project. In this phase, documents describe how the project should be performed by further planning of what should be built or developed. And developing more detailed cost estimates and how uncertain these cost estimates are. How the project should be managed to keep eventual costs controlled and the project reaching its set goals. The preliminary project phase also considers which contracts are needed with possible suppliers to deliver what the project needs to meet its set goals. The quality of the basis of management and cost estimates are then assured before the project investment and cost framework can be approved by the Norwegian Storting [48].

Implementation After the project has been approved by the Norwegian Storting, the implementation of the project is commenced.

The Norwegian Digitisation Agency has also described two phases which occur after the implementation phase for digitisation projects: (V) Closing, and (VI) Realisation.

Closing After what is built or developed is complete, the closing phase is started. The closing phase ensures a structured and formal closing of the project, as well as a good handover of the project to the organisation responsible for manageability [49].

Realisation When the project has been transferred, the project is evaluated for further realisations of gains and if the project has reached the goals which were set in the concept phase [49].

3.1.13 Digitalisation in the Norwegian public sector

The government of Norway has in the last decades an increased focus on the digitalisation of the Norwegian public sector, with the most recent digitalisation strategy from 2019-2025 [14]. The main goal of the digitalisation strategy is for different actors in the Norwegian public sector to cooperate through the development of shared digital ecosystems. The goal of this development project is to aid the development of user-centred system development for more effective and coordinated exploitation of the Norwegian public sector IT systems [14].

3.2 Software Quality

As stated in section 1.1, software quality is defined as how well a piece of software is measured against a chosen set of non-functional requirements, chosen based on the software's functional requirements [9].

3.2.1 Non-functional requirements

Non-functional requirements are defined as when it can be determined that a piece of software delivers value, with the non-functional requirements chosen based on what value the piece of software is supposed to generate [9]. These are some examples of the most common non-functional requirements:

- **Performance** - The amount of useful work accomplished by a piece of software [50].
- **Scalability** - How a piece of software can handle a growing and shrinking amount of work by adding and reducing resources [51].
- **Portability** - The effort required for a piece of software to run on different devices [52].
- **Reliability** - How long a piece of software can function without failure [53].
- **Maintainability** - How easy a piece of software is to repair or update [54].
- **Security** - How protected is a piece of software from attacks by malicious actors [55].
- **Usability** - How a piece of software provides conditions for its user to perform tasks safely, effectively, and efficiently [56].

3.2.2 Software Quality Assurance

As stated in section 1.1, software quality assurance is the planning, controlling, and executing of processes which measure non-functional requirements or other defined quality standards, which are defined by a product team or at the organisational level [10][11].

3.3 Software Testing

Software testing is a core activity in software development, where the behaviour of a piece of software is examined through validation and verification. Software testing can give information about the quality of a piece of software and any eventual faults in the software [57].

3.3.1 Acceptance Testing

Acceptance testing is a variant of software testing where a piece of software is tested to determine if the software meets its requirement specification [58]. Acceptance testing might involve black-box testing, which involves testing a piece of software's functionality without inspecting its technical structure [59].

3.3.2 Automated Testing

Automated testing is the use of software separate from the software being tested to control its quality through automated mechanisms. In automated testing, the predicted outcome of functionality is compared with the actual outcome. And is suited for repetitive testing tasks in a formalised testing process which would be difficult to do manually [60].

3.4 Software Development Methods

3.4.1 Waterfall

The waterfall method is a software development method in which the phases of a software project are split up into sequential phases. The following phases are common in using the waterfall method: (I) Requirement specification, (II) Design, (III) Implementation, (IV) Verification, (V) Delivery, and (VI) Maintenance. The waterfall method is described as less iterative and flexible than other software development methods as progress flows in a singular direction [61].

3.4.2 Agile

The agile software development method is designed to bring the user/customer closer to the development team, in order to give continuous feedback on the software being developed. The agile method also enables adaptive planning and more flexible requirement specifications, which can be changed during the development project, based on user/customer feedback [62].

3.4.3 DevOps

DevOps, short for development operations is a software development methodology where a set of tools is used to integrate development and operations in order to shorten the development life cycle. DevOps is a methodology which is complementary to the agile software development method [22]. The DevOps methodology has 3 main components: (I) Continuous Integration, (II) Continuous Delivery, and (III) Continuous Deployment.

Continuous Integration

In continuous integration, all developers in a development team are continuously merging their work into the main code repository. Each merge usually results in a workflow being triggered, where tests are run, and the current code repository being built [63].

Continuous Delivery

In continuous delivery, the development teams produce a piece of software in short cycles, to ensure that a piece of software can be delivered at any time through an automated delivery pipeline. Allowing more incremental updates to be delivered, reducing the risk of delivering faulty software [64].

Continuous Deployment

Continuous deployment is the technical aspect of continuous delivery and is the automated system, set up to deliver updates through its delivery pipeline. This delivery pipeline is usually set up to run automated tests, which block further delivery if any test fails. If all tests pass, the software update is built and delivered [65].

3.5 Product Teams

Product teams are cross-functional software development teams which are focused on measuring the outcomes of their work, rather than output. Product teams are empowered to find the best way to solve their problems for their customer/user, yet follow the strategy of the organisation for which the teams are working [66]. Product teams differ from more traditional feature teams, which focus on delivering features from a prioritised list of requirements [66].

Chapter 4

Method

As stated, the objective of this study is to understand what the IT professionals with high influence or ownership over technology in certain Norwegian public sector agencies consider important in assuring quality in the software being used to serve their users. Including what challenges the agencies might be encountering in the quality assurance of their software, and how these findings compare with the Norwegian government's digitisation strategy. Due to this being a broader theme or phenomena [67] without a clear answer as shown in chapter 2, it is decided that a multiple case study with the context of certain agencies in the Norwegian public sector is used as an appropriate method. The study uses the experience of the researcher as a software developer in the Norwegian public sector and interviews as data generation methods.

4.1 Selection of method

There are several reasons why a multiple case study is chosen as the method for this study [67], where the main reason is that software quality assurance in Norwegian has little research. As well as the term "software quality assurance" can be interpreted in multiple ways from different points of view. A multiple case study, therefore, allows the researcher to gain a deeper understanding of the topic from multiple sources, both qualitative and quantitative, which gives the researcher a larger range of data to be interpreted [67].

4.2 Multiple Case Study

4.2.1 Type of case study

Exploratory study As there is little research on the research topic in this study, and what research does exist is mostly broad and qualitative, it is decided that an exploratory study is used to gain a deeper understanding of the research topic [67].

Short-term, contemporary study The multiple case study is a short-term, contemporary study as the researcher is interested in what is the understanding of software quality assurance in the Norwegian public sector at the time of this study [67]. Is also decided due to the limited time resources in the production of the study, hence a longitudinal study would not be possible to produce.

4.2.2 Selection of case

The chosen cases for this multiple case study are certain agencies in the Norwegian public sector with a substantial IT department, used for providing digital services to their users. It is the following reasons why this case is chosen to answer the research questions described in section 1.1.

Typical instance Software quality assurance is important for other areas than just agencies in the Norwegian public sector. The results of this multiple case study could be applied to other nation's public sectors, or other organisations in different private sectors.

Convenience As the researcher is employed as a software developer in the Norwegian public sector, it is a convenient case to study. The researcher already has contact with potential participants for the study and in-direct contacts with other potential participants in the agencies of the Norwegian public sector.

Unique opportunity The researcher's unique position as both a researcher and software developer in a Norwegian public sector agency gives a unique opportunity for the chosen case. The researcher is able to bring first-hand accounts of their own experience while also conducting research which could support or contradict those experiences.

4.2.3 Interviews

The main data generation method in the multiple case study is interviews as this allows the researcher to gain a deeper understanding of a topic or area of research. Participants in the interviews were experts on digitisation in the Norwegian public sector and employees of certain agencies in the Norwegian public sector.

Experts Experts in the field of software development in the Norwegian public sector and/or large Norwegian organisations are interviewed to gain their views on the study's objective. These experts are researchers in the mentioned field and have several publications related to the study's objective.

Agencies Employees at IT departments in different agencies of the Norwegian public sector are interviewed, where the employees have a high influence on technology and/or methodology in the IT department. These employees have positions similar to that of leaders of IT areas inside an IT department, or as having the responsibility of deciding the technological direction the IT department should follow. These are chosen due to their higher perspective on software quality and software quality assurance in the particular agency. And being able to connect this higher perspective to the mission of the agency and the Norwegian public sector at large. The exact position of the employees has been altered due to their position revealing their identity, preventing anonymity. The agencies where the participants are employed are:

- NAV (Norwegian Labour and Welfare Administration)
- Skatteetaten (Norwegian Tax Administration)
- Mattilsynet (Norwegian Food Safety Authority)

Participants

Table 4.1 and Table 4.2 show the different interview objects participating in the multiple case study.

Experts

Participant ID	**Organisation**
#1	Norwegian University of Technology and Science
#2	Sintef - Research Organisation

Table 4.1: Experts on the Norwegian public sector participating as interview objects

Agencies

Participant ID	Agency	Role
#3	NAV	Technical Advisor
#4	NAV	Area Leader
#5	NAV	Section Leader
#6	NAV	Technical Advisor
#7	Skatteetaten	Manager
#8	Skatteetaten	Section Leader
#9	Mattilsynet	Area Leader
#10	Mattilsynet	Architect
#11	Mattilsynet	Section Leader

Table 4.2: Employees from Norwegian public sector agencies participating as interview objects

Structure

The interviews were structured as semi-structured interviews, which allowed the researcher to diverge from the interview plan to ask unplanned questions which the researcher finds of interest to answer the objective of the study. The questions in the interviews were created based upon the research questions in section 1.1 and the research in chapter 2. The interview held for the experts and the agencies is worded somewhat differently, as there is a difference in perspective. The experts are asked about the whole of the Norwegian public sector, while the interviewees from the cases are asked about the situation in their agency.

Expert interview The expert interviews were structured as follows:

1. What kind of research are you conducting?
2. What comes to mind when you hear "software quality"?
3. Can you say something about software quality in the Norwegian public sector?
4. What comes to mind when you hear "software quality assurance"?
5. How does the Norwegian public sector assure quality in the software they develop and maintain?
6. What challenges does the Norwegian public sector face in the assurance of quality in their software?

Agencies interview The agency interviews were structured as follows:

1. What is your role in your agency?
2. What comes to mind when you hear "software quality"?
3. Can you say something about software quality in your agency?
4. What comes to mind when you hear "software quality assurance"?
5. How does your agency assure quality in the software you develop and maintain?
6. How do you think the assurance of software quality is in the rest of the Norwegian public sector?
7. What challenges does your agency have in the assurance of quality in its software?

Recording

In order to gain complete and contextual data from the interviews, field notes and audio recording is used as recording methods. The field notes allow the researcher to note the context of the interview and anything noteworthy to be explored during or after the interview. While the audio recording is used to listen to the interview multiple times or to be transcribed and used in a textual data analysis [67].

4.2.4 Data analysis

After each interview in the multiple case study, the records were subject to data preparation through transcription. The transcriptions were then subject to a thematic analysis.

Data preparation

Before any analysis of the data collected in the multiple case study could be performed, it needed to be prepared. For the interviews this was done in three ways: (I) live transcript in Microsoft Teams, (II) transcription using Whisper, and (III) transcription using Autotekst. As the interviews were held in Norwegian, the language settings of the transcription services were set to Norwegian.

Transcription using Microsoft Teams Transcription in Microsoft Teams is available in any ongoing meetings and can be set to either be just visible during the meeting or be persistent and stored in a Microsoft Teams group. After starting any meeting, the host can start to record the audio from the meeting, in the recording menu there is an option to also record transcription. When starting the transcribing, a specific language can be chosen, all of the digital interviews were held in Norwegian, so the transcription language was set to Norwegian. When the host stops recording, the transcription files will be available as **.docx** or **.vtt** files in the Teams group [68].

Transcription using Whisper Whisper is an open-source speech recognition software, built by OpenAI to increase the performance and accuracy in recognising speech [69]. Whisper has three modes of accuracy in speech recognition: (I) small, (II) medium, and (III) large. With small being the most efficient, but least accurate, and large being the least efficient, but most accurate. This study used the large model, as this yielded the most accurate transcribing. Whisper takes an **.wav** file as input, and can output in multiple file formats, such as **.txt** and **.vtt**.

Transcription using Autotekst Autotekst is a transcription service delivered by the University of Oslo and is based on Whisper. Autotekst provides more accurate transcriptions than by using Whisper as a standalone service and is faster than running Whisper on normal computers [70].

All of the mentioned transcription services do not provide 100% accurate transcription, yet being accurate enough to understand what message and meaning is being communicated in the interview. This is considered accurate enough for this study, as it still enables the researcher to perform a thematic analysis on the transcribed interviews.

Thematic analysis

In order to generate results from the interviews in the multiple case study, a thematic analysis was performed. The first step of the thematic analysis is to create general categorisations from the interviews in relation to the objective of the study [67]. The general categories are drawn from the field notes of each interview, which then later makes it easier to categorise the different sections of the interviews. Table 4.3 shows an example of this:

Example interview - John Doe/01.01.2022

Section(s)	**Theme**
We mostly use unit testing and regression tests to ensure that the quality is good...	Technical testing
There is an increased focus on knowledge sharing through the organisation	Culture
...	...

Table 4.3: Example table for categorising sections of an interview

To categorise which themes are shared between interviews, a table of interconnected themes between interviews is created [67]. Table 4.4 is the following result of this. This table however is only an rough overview used to aid the researcher in presenting the results, some of the themes might be falsely identified, but it is still of great help in presenting the results from the interviews.

Theme / Participant ID	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11
Non-functional requirements	X	X		X	X		X	X	X		X
Agile	X		X					X	X		
Simulation	X										
Security	X						X	X	X		
Inspection	X									X	
DevOps	X		X	X	X		X			X	
Architecture	X						X				
Team	X	X			X	X		X	X	X	X
Testing	X		X	X		X	X	X		X	
Technical debt	X		X	X	X			X	X	X	X
Knowledge	X			X		X	X				
Method	X	X			X		X	X	X	X	
Project	X	X	X	X	X	X	X	X	X	X	X
Resources	X					X	X				
Feedback	X	X	X								
Definition		X				X			X	X	X
User		X									
Contextual		X				X					
Measurements					X		X				
Revision					X						X
Legal		X					X			X	X
Modelling										X	
Languages										X	
Domain										X	

Table 4.4: Table of interconnected themes between interview participants

Chapter 5

Results

The results of the interviews conducted for the multiple case study are divided into four primary categories:

1. *Perspectives* - What perspectives the participants had on "software quality" and "software quality assurance" in the Norwegian public sector
2. *Technical* - Aspects relating to the assurance of the technical quality of the software in the Norwegian public sector
3. *Methodological* - Aspects relating to software development methods used to assure software quality in the Norwegian public sector
4. *Organisation* - Organisational aspects influencing the quality assurance of software in the Norwegian public sector

5.1 Perspectives

Participants #1, #2, #3, #4, #6, #7, #8, and #9 conveyed their perspectives on software quality and software quality assurance.

5.1.1 Definition

The word "*quality*" is described by participant #2 as a word that is difficult to describe: "The word "quality" is a very difficult word. It's a word that everyone feels they understand and know what it is, but no one can really define. Although everyone has some sort of idea that they know what quality is". Further describing that "*software quality*" are words which are not widely used in the Norwegian public sector: "I don't hear many people talking about software quality in the Norwegian public sector".

Participant #6 gives multiple definitions to the words "*software quality*". Firstly describing it as the ability to modify a piece of software. Secondly describing "*software quality*" as a piece of software solving a particular problem for a long period: "NAV, for example, has "Infotrygd" which is about 40 years old, and which is nation-supporting, and if that is not quality, then I don't know. It has little ability to change, and as far as I know, it has very few tests. But it has solved a problem over 40 years".

5.1.2 Context

It is described by Participant #3 that what context the software is situated in, is important for how its quality should be assured. In the context of deciding if software quality should be assured manually by humans, or automatically with computers, participant #3 describes both have pros and cons for their respective context: "If you're trying to do most of your quality assurance with computers, then you have the effect that you can do this much faster. However it will mean that something will slip through". The description of context deciding what software quality assurance methods should be used is shared with participant #6. Describing that NAV's systems for disbursing social securities and sick pay do not use the same quality assurance methods due to their context being different.

Other situations are also important for the context of deciding what software quality assurance methods should be used. Participant #6 explains that sometimes there is no need for quality assurance: "If you are going to develop something that will be used once, then modifiability is not the most important thing. Then it is more important that it just works". Further explaining that in the described situation, a heavy focus on software quality by the development team can lead to unwanted complexity in the software, which does not help the software in solving its intended problem.

5.1.3 Non-functional Requirements

As described by participants #1, #2, #4, #7, #8, and #9, the definition of software quality is associated with non-functional requirements. Software quality relates to usability, maintainability, performance, security, or the mutability of the software's architecture. As well as manageability, described by participant #4 as important for software quality.

At NAV, non-functional requirements have been described as quality requirements or quality properties by participant #1. Also explaining that in another part of the Norwegian public sector, a team of software developers once was delivered a list of 250 such requirements or properties which the system had to maintain. Further explaining that following such a list is challenging: "If you come with a very long list of things, then it is difficult to check that everything has been done. Not to mention actually doing it".

5.2 Technical

Participants #1, #3, #4, #5, #7, #8, #9, #10 described aspects relating to the assurance of the technical quality of the software in the Norwegian public sector.

5.2.1 Security

How quality in security is assured, is explained by participant #1 to differ between agencies and projects. In the development process of NAV's new systems for distributing parental benefits, how security has been assured in the system has changed. Initially, only specific persons had the responsibility for security, however, when the project was completed in 2019, each team had a specific person with knowledge of security. Other projects such as the "Perform Project" at the Norwegian Public Service Pension Fund is described by participant #1 as having more central control of security in its requirements, which the development teams had to take into account.

The requirements for security at Skatteetaten are described by participant #7 as always being strict: "We are constantly exposed to attacks, so the requirements and quality of security solutions have always been very high", explaining that the focus on security and a general lack of development resources has resulted in Skatteetaten prioritising security above other requirements and functionalities, therefore Skatteetaten's users might have to accept software which lacks usability and functionality. Participant #7 described this focus on strict security are the same for NAV, as it is for Skatteetaten.

For Skatteetaten to ensure their strict requirements for security, participant #8 explains that all events in Skatteetaten's systems are logged. For example, whenever a network call is made or someone does a direct look-up in a database, it is audit logged. The reason is explained as when either someone with authorised permissions is abusing their permissions or someone with unauthorised permissions will be logged and exposed.

To ensure that quality requirements such as security are met at Mattilsynet, participant #9 explains that analysis such as ROS-analysis is used. Also explaining that the ROS-analysis is used to ensure that other quality requirements than just security are met.

5.2.2 Inspection

It is the belief by participant #1, that many development teams in the Norwegian public sector use code inspection as a tool to find faults in their software. As new code has to be inspected as well before it is allowed to be merged into a code repository. Participant #1 explained that this was done at the "Perform project" at the Norwegian Public Service Pension fund.

Some developers are described by participant #1 to be hesitant about certain code inspection tools to increase software quality, such as developer code review due to it being time-consuming: "The whole thing can take a number of days, and the one who has written the code must understand what you did again when you received the results of the inspections". Participant #1 recommending to perform Pair programming, as to code would have been inspected during the coding.

Mattilsynet's developers recently started to use static code analysis tools. Participant #10 explained how this is said to find 145 critical faults, enabling Mattilsynet to know that these faults exist, and can start processes to handle them. This is in contrast with an alleged study from Microsoft, described by participant #1. This is said to conclude that such code inspection tools only find a limited amount of critical faults. They mostly find faults in code style and not much else.

5.2.3 Testing

The use of regression testing is described by participant #1 as used by most agencies in the Norwegian public sector to ensure quality in their software, as well as trying to automate as much of the testing as possible. Yet some of the testing activities described by participant #1 are still done manually, such as exploitative testing and involving the customer in the testing activities.

At the "Perform project" at the Norwegian Public Service Pension fund, it was described by participant #1 that supporting systems for testing their solution with simulated data were created. While at the development of the new system for distributing parental benefits at NAV, more traditional methods were used to test the system with simulated data.

Writing tests is seen as important by participant #3 when trying to include proper continuous delivery in a software development project: "It's about making sure that you've spun a good enough safety net around the code you write so that the probability of you letting something go wrong is as small as possible. It will never be zero, but you have to make it as good as possible". Further explaining that writing tests are also important for the ability to more securely deliver smaller and more frequent changes to the software being created.

The importance of writing tests is shared with participant #8. Describing that to be able to modify a piece of software, the software needs to have high test coverage. Participant # explains that a low-level method is first tested using unit tests, and then the whole software unit or application of which the method is a part is tested: "In a Microservices architecture where you have interaction and values going through, you have to tests at all levels. To make sure things work and nothing breaks if you go in and make a change".

The degree of automated tests is described by participant #7 to be high at Skatteetaten, and it is years since Skatteetaten stopped performing acceptance tests. Participant #7 further explains that Skatteetaten has had to experiment with writing the correct amount of automated tests: "Automated testing is that you shouldn't have too much, but you shouldn't have too little either. If you have too much, it becomes a big management task. ... so it has been challenging to find the correct balance".

5.2.4 Technical Debt

The ability to register Technical debt is described by participant #1 as an important measure to increase software quality. This is important as it enables knowledge of what should be improved in a piece of software, due to it uncovering potential faults and vulnerabilities in a piece of software.

NAV is described by participant #3 as having a large variety of software in their organisation to aid them in their operations: "Some systems are 40 years old, some are being written for the first time now, and everything else in-between". Further explaining that the age of the system decides how quality should be assured: "If you are going to make a change in a system that is 20 years old, then there is bound to be some manual testing in place. You have to have your own test environment ... It prevents us from achieving quality assurance". Participant #3 explained that this has led to NAV modernising their old software, which usually means creating new software to replace the old software and switching off the old software.

This process of replacing old software is explained by participant #8 to be similar at Skatteetaten. As Skatteetaen have old software, the re-writing and renewing of this old software is a challenge due to the low quality of the old software. Whereas the new software that Skatteetaten is creating is of high quality. The old software of Skatteetaten is described by participant #8 as being modernised slowly and is switched off as they are being modernised by newer software.

NAV is also described by participant #3 as trying to modernise their old software, without creating new software and switching the old software off. Instead modernising the old software while still in service. NAV's system for delivering pension benefits was initially created by consultants from 2007 to 2011, and in 2017 was described by Participant #3 as being technical bankrupt: "We just have to throw that away, because we can't change it anymore. It takes half a year to change a comma". Yet by working on and improving the pension benefits system for five years, participant #3 explained that it is still not at the point where changes are deployed daily. Emphasising the importance of creating modifiable software, as it does not matter how much testing has been done upfront, the requirements of the system will change from when it was ordered to when it is delivered, and after the users start using it.

Some older systems at NAV have been described as a challenge by participant #5, as they do not have automated tests, requiring more manual quality assurance methods. Explaining that these systems also have a lower rate of deployment, such as the system called ARENA, NAV's system for delivering daily allowance benefits. Its deploy frequency has changed from 4 deploys to 12 deploys a year. The deployment frequency has increased somewhat, but still being in-frequent due to the need for manual quality assurance methods.

Mattilsynet is described by participant #9 to have some older systems, however, with the new product teams organisation recently introduced, it has given better focus to these older systems. Participant #9 explained that the new systems created at Mattilsynet are created with better quality, yet that are still issues with the bad quality of their older systems when it comes to usability and Technical debt.

An alleged Gartner report on Mattilsynets system "Mats" in 2012 was mentioned by participant #10 and described that "Mats" will cost Mattilsynet 10 Billion NOK over the next 15 years, and its consequences for Mattilsynet are described by Participant #10 as being negative: "That money doesn't exist, so it can't happen. But the consequences are that the system loses functionality over time because the environment changes". Participant #10 explained that functionality loss has already occurred, when a data format from the Norwegian Directorate of Agriculture changed, due to Mattilsynet being unable to modify "Mats" to accept the new data format.

The ability of software developing teams to understand the necessity of maintenance and handling of Technical debt is described as an issue by participant #4. Development teams should use 20-25% of their time to improve Technical debt, as Technical debt is the opposite of software quality. The challenge is to make the development teams dedicate that amount of time to Technical debt.

5.2.5 Measurements

The systems NAV uses in their operations are described by participant #5 as having a strong focus on measurements, being that they are measured in how they behave, such as wherein the system is there downtime, are there any response problems, etc. Participant #5 also explains that it is up to each team at NAV how they measure their systems when in operations. However, it is encouraged that they do, as it enables a good understanding of what state their system is in at all times.

It is the belief of Participant #7 that the software at Skatteetaten is of few faults. Giving an example of the process for calculating how much taxes are owed in Norway. First Skatteetaten calculate how much each citizen in Norway owes them, or how much Skatteetaen owe the citizen. Then this amount is sent out to each citizen of Norway, which only takes place once a year. Each time this takes place,

Skatteetaten measures faults in the systems responsible for this process. Participant #7 explains that the faults measured then get fixed, and the next year none of these faults appear again: "Then we measure mistakes, and what we see is that the mistakes we make one year, we fix them, and we don't make the same mistakes the following year".

5.2.6 Software Properties

A property described by participant #9 as being an indicator for a piece of software to be of good quality, is that the data it serves is easily accessible. Explaining that if software quality at Mattilsynet should be assessed today, it would be based on how its data interfaces with other applications. Such quality assurance methods are described as important by participant #9: "It is extremely important to ensure that you have quality in your solution". Participant #9 explains that today it is expected for an application to have accessible APIs so that it is predictable and easy to retrieve its data. Something that is not possible to achieve without following all aspects of good software quality, such as correct architecture and mindset in building a solution.

This description of software quality has also led to new initiatives being started at Mattilsynet. Participant #10 explains that when a new piece of software is created, its API is created first. This is done by creating API contracts using OpenAPI and designing the API from these contracts. Further explaining that this form of API-contract-creating strategy is something that the product teams at Mattilsynet are going to start doing. Enabling the product teams to discover an API that has already been created by another team, preventing duplicate work.

The use of other data-sharing patterns, such as the Publish-subscribe pattern, is described by participant #10 to increase the quality of Mattilsynets services. Described as enabling Mattilsynet to share sets of data between applications through data streams, both internally and externally of Mattilsynet. Participant #10 provided an example with their partner Nortura: "Instead of iterating over a Mattilsynet API and asking for all these animals that they intend to slaughter have some restrictions. They can rather get the data stream from us, so then we don't have to ask". Creating a loose coupling between Mattilsynets and Norturas software, meaning that Norturas software will not fault if Mattilsynets software is unavailable.

5.3 Methodological

Participants #1, #2, #3, #4, #5, #7, #8, #9, and #10 described aspects relating to software development methods used to assure software quality in the Norwegian public sector.

It is suspected by Participant #2 that the assurance of quality used to be a more formal part of software development, when the software development processes were more defined, like the waterfall model. Explaining that now development teams in the Norwegian public sector themselves choose what should be prioritised and what is important. This is suspected by Participant #2 to have led to software quality assurance being prioritised lower and seen as less important. As the development teams instead think that if the users are satisfied, then the software is of high quality.

5.3.1 Agile

The Norwegian public sector is described by participant #2 as not being concerned with assuring software quality by strict regimes in projects which were quite common before. Instead describing it to be based on getting feedback from its users. If the user does not experience any faults with the software, then it is of high quality. If the user finds that the software is of value, then it is of high quality.

Retrieving continuous feedback on a piece of software is a method described by participant #1 as used to assess the quality of the piece of software. Explaining that at the "perform project" at the Norwegian Public Service Pension fund, receiving feedback through demonstrations of the software being developed was performed, and at each demonstration retrieving feedback on the quality of the software.

The principle of retrieving continuous feedback is described as a challenge by participant #2. As the citizens of Norway usually are not interested in giving feedback, only being interested in the software functioning correctly. Explaining that these challenges increase as handling feedback properly in a public sector from its citizens is important due to the public sector having power over its users. Participant #2 describes this challenge as larger in the public sector than in the private sector. Due to the relationship of power being different, and therefore the public sector users have to trust that the particular public sector will not use the feedback data against their users.

Other uses of agile practices to ensure software of high quality, are described by participant #1 to be used at NAV in their project to develop a new system for distributing parental benefits. The same agile practices are described by participant #2 to be used at Entur to ensure software of high quality. Participant #8 explains how the agile practices of delivering MVPs, development velocity, and how each release is to be published are used to ensure that Skatteetatens software is of high quality.

It is described by participant #9 that Mattilsynet could have more established routines for assuring quality in its software. Explaining that Mattilsynet's current focus is on development speed: "Mattilsynet are a very young organisation when it comes to development ... there has been a focus on productivity and getting the solution out and testing it, rather than on routines and quality". Describing once development speed is at a satisfactory level, the focus will be shifted from development speed to quality, testing and management of Mattilsynets software.

5.3.2 DevOps

There are allegedly studies on the difference of the review of software through demonstration or deployed software described by participant #1. Where the alleged studies conclude with the customer and users find more faults when the software is deployed, rather than in a demonstration. This is because when the software is presented in a demonstration, the context is artificial. Participant #1 explains as this is because user reviewing the software does not get to experience how it works in their own organisation or in their day-to-day life. Hence other aspects of quality are reviewed when the software is deployed and available to the user in their normal context.

The use of DevOps to receive feedback about quality in a piece of software is described by participant #1 to be used at NAV in the development project for disbursement of parental benefits. Explaining that the project was initially planned to be delivered through three large deployments, with the deployment plan changing for the last delivery: "They combined both development resources and business resources and where they then worked towards having a continuous deployment". Participant #1 explained that this change gave the teams more control, due to the feedback from the users being more authentic on a wider range of quality aspects.

For ensuring the development of robust software, participant #4 describes the need for continuous development. As well as high development speed- and flow are needed for the development teams to ensure that other quality aspects of the software are high when being developed. Further explaining that if software is not maintained the quality will decrease. Leading to any eventual fault in their software needs to be fixed quickly to ensure that the software constantly is of quality. It is for these reasons that the development teams at NAV follow the DevOps Principles, explained participant #4.

For helping the development teams at NAV follow the DevOps principles, participant #4 explained that NAV has created its own application platform, NAIS. With NAIS being designed in a way which gives strong recommendations on how software should be developed and deployed to NAV's infrastructure, including standardised services for typical NAV applications. Participant #4 describes this as enabling software of high quality: "The sum of NAIS makes it easier to create good software, which leads to increased quality. The teams can make their own choices, but it is recommended to make good choices with NAIS".

During the development of software, participant #3 explained that for the negative consequences to be as low as possible, it needs to be handled with continuous delivery. Explaining that in the past, NAV had 4 large releases every year for their new software, where the largest release included about 116 000 development hours. The only way to ensure that 116 000 development hours do not cause negative consequences, is to perform a large number of manual tests and a large number of checklists. Participant #3 now believes that more frequent and smaller releases decrease the consequences of eventual faults at NAV. This is why NAV has gone from 4 releases a year to 1500 releases a week.

The teams themselves at Skatteetaten are described by participant #7 as having the responsibility for delivering production-ready software. Explaining that Skatteetaten has worked with continuously deploying production-ready software since 2013 and that Skatteetaten has always had a focus on deploying smaller changes early: "This is all about getting things out early. If there is an error, it is sometimes limited in scope and we can quickly roll back or fix it". This as large and infrequent releases need large acceptance tests, resulting in large faults.

New software being developed at Mattilsynet is described by participant #10 as being created using the Microservices or Service-oriented architecture. It is then deployed to Mattilsynets own Kubernetes cluster, which gives a declarative way of deploying software. Participant #10 explained that Mattilsynet has a set of requirements for their software to be deployed to their servers. Being that the code has to stored in a GitHub repository which is a part of Mattilsynets GitHub organisation. Enabling Mattilsynet's platform team to know what the code is, and what it does, allowing them to scan the code for different reasons.

5.3.3 Software Development Techniques

During the "Perform project" at the Norwegian Public Service Pension fund, participant #1 describes Pair programming to be used as one of their software development methods. Explaining that not all development teams in the Norwegian public sector perform Pair programming, but the ones who do, use it for the development of more complex features. The development teams try to make sure that all team members have pair-programmed in a development iteration. The benefits of this are described by participant #1 as: "It gives a common ownership of the code base, which is quite preventive for introducing new bugs, and that developers understand what is happening in the rest of the code".

Similar software development methods are described by participant #1 to be used by some teams at NAV, methods such as software teaming or Mob programming, where the whole development team sit together and works on the same feature. Explaining that this enables more discussion about the feature than by performing Pair programming, which also has the effect of increasing knowledge in the whole development team.

NAV have a technical direction, which describes what software development methods are desired to be followed. This technical direction is described by participant #5 to recommend Pair programming as a preferred software development method, and is recommended to be performed by the development teams as often as possible. The technical direction is also described by participant #5 to recommend a focus on test-driven development and test coverage in NAV's development teams.

Skatteetaten is described by participant #7 to use something similar to NAV, but instead being called a method-framework. Which explains the best practice in software development and what development patterns should be followed at Skatteetaten. Further explaining that part of this method framework is knowledge sharing of these methods: "Then we enter it as part of our methodological work to communicate it out to share experiences". The method framework is also explained by participant #8 to include development practices such as KISS (Keep it Simple Stupid), Clean Code, and Separation of Concern.

One of the development patterns in Skatteetaten's method-framework described by participant #8 as creating good quality is doing things in the right order and having a good workflow, so that the competency of the developer is used in a good way. Participant #8 explained that as a software developer, there might be tasks that he/she gets stuck on, such as not understanding logic, organisation rules, or architectural decisions. In these scenarios, good quality is being preventative in making the correct decisions.

It is described by participant #7 that Skatteetaten has changed the time when the software should be quality assured, away from the traditional ISO mindset where the quality is controlled after the software is created. Explaining that for the last 15 years, Skatteetaten has instead been controlling the quality from the start of the development project, as a part of their development method. This control mechanism includes new code requiring to be through a test phase for the new code to be deployed to Skatteetatens production servers.

5.4 Organisational

Organisational aspects influencing software quality and its assurance in the Norwegian public sector were described by all participants in the interviews of the multiple case study. Organisational aspects were also the most talked about by the participants, whereas about 50% of the data was about organisational aspects.

5.4.1 Team

This change from quality assurance being a formal part of software development is described by participant #5 as being the case at NAV. With NAV changing from centralised requirements and checklists for the quality assurance of their software, to having no centralised management of quality assurance. Explaining that it is now up to each development team at NAV to understand how the software they develop functions. As NAV's management trusts that the teams have the best knowledge of their software, and know how to create software of quality.

Entur is described by participant #2 as basing their development teams on being autonomous and deciding what should be done at what point in the development project. With this autonomy in decisions including which methods should be used for quality assurance in the software being developed.

Autonomy is described by participant #10 as also being incorporated into Mattilsynets 10-11 product teams. However describing the lack of centralised authority is starting to become a challenge for Mattilsynet, and might lead to Mattilsynet implementing a central authority for architecture. Participant #10 explains that such lack of central authority has been deliberate, due to Mattilsynet's current strategy is to increase development velocity in the product teams before adjusting their direction. Instead iterating the product teams as new problems arise.

The autonomy of the product teams at Mattilsynet is also explained by participant #10 to extend to the choice of technology in their software. However, when recruiting new developers, skills in a certain set of programming languages are preferred, such as Kotlin/TypeScript. Explaining that Mattilsynet recommends that a certain set of programming languages be used: "You won't get any pushback if you want to use it, but if you use another language, for example, then you probably have to be prepared to argue for it". Participant #10 describes this as deliberate,

as it should be easy to follow the "main path" at Mattilsynet, without setting hard limits. As hard limits hinder experimentation by the product teams and slow their development velocity. However explaining that this practice enables the product teams to understand that going beyond the "main path" is going to increase costs, motivating the product teams not taking uncommon decisions without having a good justification.

Mattilsynets product teams are described by participant #10 as being a mix of consultant and permanent employees, no longer having teams purely consisting of consultants, explaining that this ensures that both consultants and permanent employees understand the costs of creating long-lasting software. As in the past, Mattilsynet has had experiences with consultants not taking into account the costs of maintaining software when creating it. Participant #10 also explained the product teams to be long-lasting, with no plan for these teams to be dissolved, a view shared by the leadership of the IT department of Mattilsynet. Explaining that this ensures that the teams most familiar with the software have long-lasting ownership of responsibility for that software. Ensuring that any Technical debt are decreased.

It is also explained by participant #10 that Mattilsynet has recently done a large reorganisation, by changing from project-based development to product teams. The product teams are also organised to have tech leads, where each tech lead has the responsibility for technical quality. This is described as enabling Mattilsynet to solve its strategic problems by making sure that to the highest degree possible, the correct problems are being solved. This helps to solve the issue of having software that is high in quality but does not solve the correct problems. Hence participant #10 explained that reorganising to product teams and making sure these product teams are looked after, has been an important decision for Mattilsynet. However, participant #10 also describes that the reorganisation of product teams has led to a loss of routines at Mattilsynet.

Governance principles are described by participant #11 which the product teams at Mattilsynet have to take into account when developing software, including for instance security, archive law, and GDPR. The product teams at Mattilsynet are described as quite new, with a high degree of newly employed members. Leading to Mattilsynet currently focusing on governance principles required by Norwegian law, being GDPR and security. To help with this, participant #11 explained that Mattilsynet has a central team for security, which works closely with the platform team to create a good security foundation for the product teams.

The teams at Skatteetaten are described by participant #8 as being cross-functional, with a range of roles being described in a team. Each team has a member responsible for the architecture and it follows the guidelines and patterns which are approved in Skatteetaten. There is also a member responsible for security which makes sure that the team deliver proper security in their product. Another mem-

ber has the responsibility of DevOps and ensures that the CI/CD pipeline is properly configured. The teams do also have a product owner, which is responsible for choosing what should be prioritised and what should be done at what times. There are also members who are representing the organization and making sure the organisational requirements are met. Another member has an extra focus on testing and testing-related activities. Participant #8 explained that roles such as security and testing responsible does not have full responsibility for this, they are only supposed to make sure that it happens, the whole development team are responsible for these activities.

5.4.2 Knowledge

Participant #1 explains that the Norwegian public sector participates in conferences to discuss competence development for their agency. Describing it as something which is not only done by the larger agencies: "It's not just Skatteetaten and NAV as the big communities that are present in places where this sort of thing is discussed, so I think it's quite widespread".

Sharing of knowledge is described by participant #11 as important for assuring quality in software, as well as high competence throughout an organisation is important for software quality. Participant #11 explains that Mattilsynet is trying to achieve this by having a tech lead forum for all tech leads at Mattilsynet to increase knowledge sharing between all the tech leads.

Lack of competency is described by participant #6 as a challenge for NAV, as it is quite clear that the people with the least experience in software development have little knowledge in testing or continuous delivery, so these have to be trained in such things. Lack of competency in the people maintaining Skatteetaten's older IT systems is described as a challenge by participant #7, as these people lack the competency to develop and maintain Skatteetatens newer software. These people are described by participant #7 as working with Oracle databases and do not know how to create a user story or a Java component. Explaining the need for different competencies at Skatteetaten, due to the large changes in their modernisation plan.

For the health platform developed for the counties in mid-Norway, it is described by participant #1 that it was known beforehand that it would not be user-friendly. The activities to improve usability were not performed, technical faults in the platform, and a testing rig was placed too far away from the real environment in which the platform would operate. Participant #1 thinks that this could be due to a lack of IT competence in the planning of the project.

5.4.3 Constraints of Development Projects

The majority of the data from the interviews about organisational aspects described organisational aspects relating to development projects as affecting software quality and its assurance. These aspects are (I) the Norwegian digitisation agency's project wizard, (II) Financing, (III) Contracting and Tenders, (IV) Budgeting, (V) Off-the-shelf Software, (VI) Ownership, and (VII) Strategy.

Project Wizard

Described by participant #1 as of great importance to the Norwegian public sector, is the Norwegian digitisation agencies project wizard. Describing the project wizard as not suited for agile projects, rather projects following methods such as the waterfall method. Participant #1 explains that this means that there are more projects following the waterfall method in the Norwegian public sector than in the Norwegian private sector.

The project wizard is described as a challenge for NAV and the rest of the Norwegian public sector by participant #3. As the project wizard is necessary for NAV to obtain the correct funding for their larger software proposals, however describing its quality assurance regime as more suited for building roads than software: "It is based on having a fairly perfect knowledge of what you are going to make before you start making it. We've never had that, and I don't think we ever will either". Participant #3 explains further when proposing software which will replace 40-year-old software, which has been altered to fit new laws and user needs through 40 years. You uncover new things which could never be planned ahead of time when starting to create the replacement.

The challenges with new knowledge and the project wizard were also explained by participant #11, describing the project wizard as hostile towards project scope creep. When in reality scope creep might mean new insights have been discovered about the problem being solved by the software. Thus having the possibility to give more value to the end user. Participant #11 also explains that scope creep can also mean that the original scope of the software has changed, so not changing the scope can mean solving the wrong problem.

The project wizard is also described as a resting pillow for not reorganising into product teams by participant #10. Explained as being the case with the old IT leadership at Mattilsynet, with Mattilsynet being focused on becoming better at projects suited for the project wizard. Participant #10 described the real need to stop with such projects, as actively recruiting staff to improve Mattilsynet on projects created slowness in the organisation. Further explaining that it then became a challenge for the staff specialising in projects when Mattilsynet changed from project wizard projects to product teams.

Both opportunities and problems are described by participant #7 to be given by the project wizard. Explaining that NAV had a financial model which was heavily dependent on funds from the project wizard, and buying software from external vendors. Participant #7 describes this as not being the case for Skatteetaten, as Skatteetaten has always developed and operated its own software. Explaining that this means that NAV has had an opportunity to save money as a consultant might cost 2.5-3 Million NOK, while a permanent employee might cost 1 Million NOK. Meaning that NAV has had a good opportunity to build up a sizeable development environment. This opportunity is explained by participant #7 as which Skatteetaten has never had, relying on the funds from the project wizard for their large initiatives. As well as being able to deliver as required by the Norwegian government. The IT department of Skatteetaten has a total of 1000 personnel, in addition to 300 consultants. The 300 consultants would never be possible without the funding from the project wizard.

Another challenge with the project wizard explained by participant #3, is that its funding stops when the software is put to use. Describing that NAV wishes to create software that is long-lasting, which requires steady funding for development and maintenance long after the software has been put to use. As there will always be changes in the laws surrounding NAV's software and other needs. Participant #3 explained that the stop in funding is hurting the Norwegian public sector: "It is one of the major headaches for the entire public sector, but perhaps now we at NAV are the most aware of it since we have come the furthest in this digitisation".

The project wizard is described by participant #11 as no longer being a challenge for Mattilsynet. As Mattilsynet stopped its use after Mattilsynets recently re-organising from projects to product teams.

Financing

The challenges relating to financing from the project wizard in section 5.4.3 are described as an issue of prioritisation, rather than financing by participant #5. It usually being a discussion about financing, when in reality it is a prioritisation issue, independent of financing. Explaining that financing might not always mean low funding, but the expectation of what should be delivered compared to the given budget is too high. Leading to over-promising which can create issues which are expensive to fix and clean up.

Firmer boundaries and more trust in the financing of NAV's software are described as desired by participant #6. As this would enable NAV to flexibly prioritise their funds. Yet also explaining that firmer boundaries and more trust in financing might not be correct for the whole Norwegian public sector, as it might lead to all agencies asking for too much funding.

A more predictable financing model is described by participant #8 as being better for Skatteetaten. Describing that a potential risk in a project with an unpredictable financing model is that the financing model does not take into account unforeseen events. This is described by participant #8 as being able to have negative consequences for the quality of what is being developed.

Contracting and Tenders

The health platform developed for the counties in mid-Norway is described by participant #1 as mainly created using the waterfall method, switching to an agile method later in the project. Explaining that most software professionals would agree not to follow waterfall methods, as most of the time it leads to the software quality suffering. Participant #1 explains that in such projects, it is common to define a set of requirements, send them to multiple contracts, and contractors fight to deliver the cheapest system. Describing that in such processes, the software quality usually suffers: "This will often mean that you have to take certain shortcuts, and if everything going into the solution is by contract the shortcuts will usually be in the quality of the software". Further explaining the issue with sending software systems on tender, as it also distances these requirements from the user of the software.

Before 2016, participant #3 explained how NAV had little knowledge or resources to develop software themselves, as all systems required for NAV's operations had to be put on tenders. Explaining that NAV has stopped this practice, as it was not sustainable and had negative consequences. Participant #3 explained how this was the case when NAV received delivery of their original system for delivering pensions, where NAV did not know how to maintain the system themselves. NAV then has to put the maintenance and further development of their core business on tender.

This way of putting software of tender is described by participant #7 as never being done at Skatteetaten. Explaining that since 2003 the operational and IT at Skatteetaten have worked together in product teams for IT-related projects. This is because when there is a drastic change in operations at Skatteetaten, it cannot simply be given to a single team. Described by participant #7 as requiring a change in the organisation on how things are solved and how people cooperate both inside and outside the agency so that the software is created in a new way.

One of Mattilsynets's older systems, called "Mats", is a system explained by participant #10 to be tailor-made for Mattilsynet by a contractor. Explaining that this system had a problem of Mattilsynet wanting functionality, without the contractor being motivated to refuse. Leading to the system's complexity growing organically. Participant #10 explained that Mattilsynet has not understood the consequences of their orders: "Functionality has been ordered and technical debt has been ignored. Just thinking that somebody else will fix the problems, which they do not".

A similar previous problem for Mattilsynets described by participant #10, is Mattilsynet having quite weak requirement specifications for their contractors. As there were not enough internal resources at Mattilsynet to create proper requirement specifications for their contractors. Resulting in the contractors themselves almost creating the whole requirements specification for Mattilsynet. Participant #10 explained that this was the case with Mattilsynets photo app, where Mattilsynets inspectors are able to take pictures and store them in Mattilsynets databases, without having to use iCloud. Since the requirement specification was weak, the team making it did not think that someone had to maintain it long after it was developed. Making the decisions for technology and the solution quickly, and Technical debt being created from the start.

The assurance of software quality is described by participant #10 as quite variable in the Norwegian public sector. Describing areas such as healthcare, municipalities and national defence to be heavily invested in putting software on tender and assigning contractors. While others such as NAV, Skatteetaten, NRK and Norsk Tipping are described by participant #10 as creating software themselves. The difference is that the consultants are hired as competency to an already existing development team.

Budgeting

Yearly budgeting in the Norwegian public sector is described as a challenge by participant #2. Explaining that in the private sector, if you can prove that a development project will be beneficial, it is easier to gain and spend money for that development project. While in the Norwegian public sector budgets are planned yearly. so it is not easy to change how the budget should be spent on different development projects. Participant #2 explains that if a certain software in the public sector experiences any sudden problems which need larger funding, it will be difficult to get the necessary resources. Describing this as an issue for software quality: "If you find out something, like a quality issue, it's not like you can just magically get more money".

Participant #2 explains that the challenge of yearly budgeting in the Norwegian public sector also affects the creation of new software. As the budget is being funded by the Norwegian tax-payers, leading to development projects in the Norwegian public sector usually not having enough funding. This situation is described by participant #2 as usually resulting in the software quality being prioritised lower than required functionality, due to there not being enough funding for both.

The IT budget of NAV is described by Participant #4 as not set correctly, as the budget should be placed under the specific service provided by the agency, instead of everything IT-related being placed under the IT budget. Explaining that for NAV, this means increasing the budget for a specific benefit, so that that ben-

efit can create the software that it needs. Participant #4 explained that this way, the management of NAV can be more flexible with their budget on what should be created for the benefit that needs it the most. However, now the budget is unreachable for the management of NAV and therefore not prioritised in a way that is beneficial for the users of NAV.

Off-the-shelf Software

Adjusting off-the-shelf software as a way of saving money is described as a challenge by participant #2 in the Norwegian public sector. As trying to adjust off-the-shelf software might lead to lower quality, or having to spend a lot of resources to adjust it to the point of its users being satisfied. Participant #2 explained that it might be a divide between the agencies in the Norwegian public sector which have their own in-house development environment, and the agencies that do not. This is due to the agencies with in-house developers who can create software themselves, having full control of the software quality. While the agencies without will need to adjust off-the-shelf software.

It is also explained by participant #2 that agencies with an in-house development environment are likely to know their agency quite well. Making the job of adjusting off-the-shelf software easier. However, when using off-the-shelf software, the rules of the software are already set which have to be taken into account. Participant #2 explains that when creating software from scratch, these rules do not apply anymore, letting the software be highly specified for the user's needs.

Ownership

NAV's success in digital transformation is described by participant #6 because of their focus on ownership. Including stopping NAV's software single-handedly created by consultants: "It is a problem with consultants who do not have the long perspective. That they think that consultants and projects are connected at NAV, and when the project has been created it is done. To me, that is the opposite of quality". Participant #6 explains that as time goes on and a piece of software at NAV is not maintained, it loses quality, so a constant fight against quality withering is necessary.

Participant #3 explained how organisations want to deliver software to their users as NAV has achieved. Explaining that it sounds quite attractive, and is spoken about at conferences, yet it requires high effort from the organisation. It mainly requires not having software delivered single-handed from consultants, instead software developed by in-sourcing at the organisation. Participant #3 explaining as soon as the software is created outside the organisation, the quality can decrease. Describing to do as NAV requires the organisation to take ownership of their software, however, not all agencies in the Norwegian public sector have this opportunity.

Strategy

Other agencies in the Norwegian public sector are described by participant #4 as following the methods NAV are using to improve their digital transformation methods. The methods include rigging the agency around DevOps, product teams, and small specialised applications. Participant #4 describes the Norwegian public sector as improving in these fields, while still having some way to go, to be as good as the private sector. Participant #4 emphasised that the agencies in the Norwegian public sector do not follow NAV because NAV is successful, but because the methods NAV are following being clever.

The researcher Torgeir Dinsgrøyr from NTNU was mentioned by participant #7 to talk about how NAV is coordinating teams and steering of their development direction. Participant #7 mentioned that the mission of Skatteetaten will always change: "Our politicians are constantly inventing new things, or updating existing things for us". Thus creating the need for coordination between product areas, multiple teams and multiple parts of the business operations at Skateetaten.

It is the perception of participant #7, that many might think of product teams as something which is quite stable. However explaining that this is something which can be unwanted from an organisational perspective, instead wanting product teams to be flexible. Explaining when the organisation tries to change the staffing in a product team, a high degree of resistance can be encountered. This is due to the staff enjoying the team or area to which they belong, with the people they already work with. Therefore organising around product teams can create stiffness in an organisation.

Well-managed projects are described by participant #10 as not without their issues. Such as the project at Mattilsynet for meat controlling, which was described as in general well managed and delivered with an Service-oriented architecture. It was so well managed that it became the template for projects at Mattilsynet. However, participant #10 explained that it had no real strategy on how it should be maintained and how further operations should be conducted after it was delivered. Even if the project was well done, it still left a large problem for Mattilsynet. Now explaining that the same system has been split up between two product teams, something which was described as not without problems by participant #10: "Now both of those product teams are sitting with components that they are not in a position to understand. With technology that they do not know ... and any functional changes in these components do not occur".

5.4.4 Resources

The main challenge for the Norwegian public sector is described by participant #1 as the lack of staff with knowledge in IT. Explains as more agencies in the Norwegian public sector are following NAV to achieve continuous deployment and maintenance, it requires the agencies to have their own IT resources.

The challenge of a limited amount of resources is described by Participant #7 as being true for Skatteetaten. Leading to Skatteetaten having to prioritise what should be included when creating new software. Participant #7 explains how this means that Skatteetatens projects usually end up with a backlog of tasks which have been given a lower priority. As there is not enough capacity to meet a large number of wishes and needs.

Some of the older systems at NAV are described by participant #6 as being of low quality, and that NAV does not have the necessary resources to handle the lower quality at this moment. Explaining that this is the case with NAV's payment system, which each year pays out 500 Billion NOK to the Norwegian citizens, being a third of Norway's national budget. Participant #6 explaining how it needs to be fixed: "It has to be rewritten, and we don't have the people for that right now, because we have other things that are even worse. The people who can change something like that are over 60 years old". Further explaining that general NAV is losing competency for their older systems.

Mattilsynet is an agency which is described by participant #10 as having a large domain-complexity, while also having relaxed non-functional requirements. Explaining the domain complexity to be large, due to Mattilsynet being descroned as a healthcare provider for a range of animal species. Leading to Mattilsynet not having the capacity to maintain its operational logic, being its largest challenge. Participant #10 giving an example that in a year, over 2000 new animal husbandry's are registered at Mattilsynet, with each being unique.

The importance of central resources in an organisation which the product teams can take advantage of is described by participant #11. Explaining that resources such as security experts and legal advisers which the product teams can use in their projects being hired at Mattilsynet. There are not enough resources for the product teams themselves to have this expertise, but this allows them to be somewhat available for the teams, which is important for software quality.

5.4.5 Legal Requirements

Software quality is described by participant #7 as a piece of software solving a problem, and that the software covers certain requirements. Not just user requirements, but operational requirements such as legal requirements. Participant #7 explains that Skatteetaten is driven by legal requirements: "Skatteetaten are heavily driven by legal requirements. It sets a lot of demands on how things should be developed, and we cannot create a solution which does not comply with the legal requirements". Explaining that Skatteetatens software should have good usability, but is mainly driven by legal requirements.

Legal requirements are also described by Participant #7 as a challenge for Skatteetaten's mission to create trust in the Norwegian populous. So that each Norwegian citizen is motivated to pay their taxes to fund the Norwegian welfare state. This is described by participant #7 as being achievable through sharing data about Norwegian society, as Skatteetaten has a wide knowledge of this. However difficult, as the legal requirements are described as putting limitations on data sharing.

Another challenge with legal requirements described by participant #7 is that these are not digitisation friendly, being difficult to simplify for Skatteetatens operations. In a project in cooperation with the Brønnøysund Register Centre, participant #7 describes how Skatteetaten and Brønnøysund wanted to create a single interface where businesses owners could see information about requirements from both Skatteetaten and Brønnøysund. Explaining that Skatteetaten did not receive any data from Brønnøysund, and Brønnøysund did not receive any data from Skatteetaten. However, it is described by participant #7 that the laws did not allow this, even though the business owners own the data. Yet Skattetaten is allowed to share data with other parts of the Norwegian public sector, such as the health sector. Leading to legal requirements setting hard limitations on delivering good user experiences.

5.4.6 External Revision

The process for software quality assurance at NAV is described by participant #5 as not being fully documented. Explaining that if a project at NAV is subject to a revision, it would be a challenge to get a complete overview of the quality assurance methods used. In such an event, the development teams would need to be contacted and asked to document these processes. However, participant #5 describes this lack of documentation to have some advantages: "The advantage is that we do not create documentation that is not actually used in practice".

It is explained by Participant #5, that recently there has been an external evaluation of the development of the system for sick pay at NAV. Explaining that this is something which occurs rarely, only done in this case due to the system receiving a lot of attention due to its development progress not being satisfactory. Resulting in external quality assurance of the system, including software quality.

The Norwegian Broadcasting Corporation is described by participant #11 as still being focused on projects with external revision to ensure that their software is of quality. Which is described by participant #11 as doing little to actually assure quality in software.

Chapter 6

Discussion

This section will discuss the results from chapter 5 in relation to the scientific literature presented in chapter 2, and the research questions and goals of this study presented in section 1.1 will be attempted to be answered.

6.1 RQ1: What practices do agencies of the Norwegian public sector use to assure quality in its software?

The practices relating to software quality assurance in the Norwegian public sector were described in chapter 5 as being in techniques and software development methodologies.

6.1.1 Techniques

A non-functional requirement described in subsection 3.2.1 often used as a measurement of quality in software, is security. The results from subsection 5.2.1 show that this non-functional requirement is also of importance to the agencies in this study. This has led to multiple practices described in subsection 5.2.1 used to ensure secure software, such as security knowledge, audit-logging and security analyses. It is shown that having a strong focus on security before a piece of software is delivered can be beneficial to the Norwegian public sector. Weak security in software can have large consequences and costs to fix if discovered after it is delivered to the user [30]. It could therefore be argued that in the context of weak security can lead to mistrust by Norwegian citizens. As their trust might be challenged if a Norwegian citizen discovers their sensitive data to be lost and shared online. Leading to negative consequences due to the importance of Trust in the Norwegian public sector [17].

Different code inspection methods are described in subsection 5.2.2 to be used by the agencies of this study to detect faults in their software. One manual method, code review, is described as being time-consuming and does not find any critical faults, while other automatic methods such as SonarQube is said to have found critical fault in Mattilsynet's system, "Mats". This difference in methods and their respective benefits and drawbacks implies that the Norwegian public sector has to choose such methods with care. As some methods might steal valuable time, without returning to enough value in software quality. It could also seem that if choosing a code inspection method, an automatic method is recommended, as it has shown to be important in increasing software quality [24].

Other techniques resembling the code inspection methods, as described in subsection 5.3.3 are used to assure software of quality in the Norwegian public sector. These techniques being Pair programming and Mob programming. While also acting as code inspection methods, these are described as increasing ownership and knowledge of the software, increasing its quality. Using pair- and mob programming in the Norwegian public sector is beneficial, as the context of a public sector is complex [21], leading to the software tasks being more complex, where techniques such as Pair programming are said to beneficial [31].

Specific testing methods used to assure quality in software have changed as shown in subsection 5.2.3, changing from manual acceptance tests to automatic tests. This change could have multiple reasons. As shown in subsection 5.4.3, the agencies are moving away from projects with a set deadline, usually funded by the project wizard described in subsection 3.1.12, requiring quality checks such as acceptance tests. Another reason could be due to the implementation of DevOps in the Norwegian public sector, shown in subsection 5.3.2, in order to be able to deliver frequent updates to the end user. As well as being able to deliver frequent updates with low risks, high test coverage with automatic tests is described in subsection 5.2.3. The agencies in the Norwegian public sector should still be careful about moving fully to automatic tests, as this can lead to lower quality, due to manual testing still being needed to achieve high quality [27].

The practice of documenting Technical debt, as described in subsection 5.2.4 is used by Mattilsynet to help in software quality assurance. This practice is seen as important, as it enables organisations to have more optimal use of their resources [32], and enables development teams to identify and repay Technical debt in a timely fashion [32]. The practice of documenting Technical debt can therefore be seen as quite crucial in assuring software of quality. As it was shown in subsection 5.2.4 and subsection 5.4.4 that the Norwegian public sector has a high degree of Technical debt.

Measurements are shown in subsection 5.2.5 as a method for agencies in the Norwegian public sector to measure the quality of software in runtime. Collecting software metrics is valuable, as it helps in creating meaningful estimates and guides development teams in taking decisions in their software [33]. The measurements could be used to compare if the quality of a piece of software has increased or decreased over a longer period. Helping to understand where in a piece of software faults or low quality are present, and guide on what decisions should be made to fix the faults or improve the quality.

Implementing certain properties to software is described in subsection 5.2.6 to be used for increasing the quality of software. It is explained how Mattilsynet is going to enforce such implementations soon. However little research suggested the implementation of such properties to increase quality, and the properties in general lacking research. Therefore enforcing such implementations without proper research to evaluate the enforcement, could lead to negative consequences. It is also shown in subsection 5.4.1 that the Norwegian public sector is implementing autonomous product teams. Enforcing such implementations could lead to the autonomy of the product teams being reduced, which is described as important in creating software of high quality.

6.1.2 Software Development Methodologies

Agencies in the Norwegian public sector are described in subsection 5.4.1 to organise their development teams as cross-functional in order to help in the development and maintenance of software of high quality. This is an important practice, as cooperation in IT teams, such as between operations and development is crucial to enhance the quality of software [29]. This could be the reason why the Helseplatform of mid-Norway is lacking in quality, as described in subsection 5.4.2, it was made without knowledge of usability in their development teams.

The practice of cross-functional teams is described in subsection 5.4.1 to be in addition to the development teams being organised as autonomous product teams, and this organisation is important in developing software of high quality. However, little research was found on the effects of product teams on software quality assurance, or in product teams as described in section 3.5. These teams being autonomous could be important for assuring software of quality, as team autonomy is described as being important in achieving agility, [71], and agility could be important in software quality [72].

DevOps as a practice to gain feedback more frequently and increase software quality is described in subsection 5.3.2 as being implemented in agencies of the Norwegian public sector. This is because the development teams are able to deliver small changes frequently to the users. This decreases the scope of which parts of the software the users have feedback about and the area affected by eventual faults are limited. The development teams can therefore use the same methods to quickly fix their software based on the feedback, and quickly get new feedback from the user. This use of DevOps has been found to assure quality due to its ability to continuously deliver new updates to the user [23][24][25].

Sharing of knowledge and increasing the level of knowledge in their agency was described in subsection 5.4.2 as important in assuring the quality of software in the Norwegian public sector. As well as the use of DevOps is an important method for assuring quality through increased knowledge sharing between employees [24][26]. However, none of the agencies described DevOps as a method for increasing knowledge in their agency. Rather as described in subsection 5.4.2, using more traditional methods, such as participating in conferences or leaders of the development teams meeting to share experiences. It could therefore be the case that DevOps have effects on knowledge sharing and its effect on software quality which the Norwegian public sector are unaware of.

With all the seemingly positive effects of DevOps on software quality described in subsection 5.3.2, it could be quite tempting for other agencies in the Norwegian public sector to implement it. However, if not implemented correctly, DevOps can hinder software quality assurance [27]. In order for DevOps to increase quality, the development team have to take shared responsibility for quality assurance. Rather than a single person having all the responsibility for quality assurance [27]. This shared responsibility also extends through the software's life, from development to its phase-out [27]. If more agencies in the Norwegian public sector want to introduce DevOps correctly, it needs long-lasting product teams which share responsibility, or else the software quality can suffer.

This temptation combined with it being stated that the effects of testing in DevOps environments had not yet been studied systematically by scientific literature [27][28]. This could mean that the agencies of the Norwegian public sector should be even more considerate if implementing DevOps is going to increase software quality. There could be hidden risks or benefits which are not uncovered by scientific literature.

6.2 RQ2: What challenges are agencies in the Norwegian public sector encountering in the quality assurance of its software?

The challenges relating to software quality assurance in the Norwegian public sector were described in chapter 5 as being in organisational aspects of software development.

The main challenge for software quality assurance in the Norwegian public sector, as described in subsection 5.4.4, is the lack of IT resources available. As it was described NAV lacks sufficient resources to deal with the Technical debt in its payment system. If such a system fails, it could have negative consequences reaching beyond the specific citizen not receiving their payment, but the Norwegian society and economy being negatively effect. In 2018, it is estimated that NAV constituted for 15.5% of Norway's gross domestic product [73], totalling to about 434 Billion NOK in 2018 [74]. It could therefore be argued that the Norwegian economy has a heavy reliance on NAV's payment system, and any amount of downtime can have negative consequences for the Norwegian economy and in turn, the Norwegian society.

The practice of yearly budgeting in the Norwegian public sector is described in section 5.4.3 as a challenge for performing software quality assurance. An agency might have enough resources at its disposal to improve the software quality of existing software, however, the resources are not being specified in the budget to be used for software quality assurance, preventing the available resources to be used. Yearly budgeting is also described in section 5.4.3 to be a challenge for the creation of new software, as the yearly budget is usually limited. This could lead to software quality being prioritised lower than functional requirements, resulting in the same challenges described with Norwegian public sector software today in subsection 5.2.4. Section 5.4.3 also describes that software quality is not a good enough reason to change the budget. Even if software quality assurance efforts are shown to be good investments in the public sector, budgetary constraints can hinder its consideration [21].

Using external vendors is described in section 5.4.3 as a challenge in the Norwegian public sector, with it being described as resulting in lower quality. This lower quality was described for three reasons: (I) distance between the user and the developers, (II) lack of operational knowledge of the external vendors, and (III) hard transition between development and operation at project handover. However, with these described challenges in external vendors, it can still seem that this practice is used in a large part of the Norwegian public sector as described in section 5.4.3. Something that is described as being the case for other public sectors due to shortages in in-house resources [21], and described in section 5.4.3 to be true for the development environments in the Norwegian public sector.

The lack of resources and budgeting has led to agencies in the Norwegian public sector requesting funding through the Norwegian digitations agency's project wizard, as described in section 5.4.3. However this necessity is described to have several challenges: (I) challenges in mutability, (II) hostility towards new knowledge, (III) hard transition between development and operation at project handover, (IV) lack of funding when a project is complete, and (V) faulty quality control regime. Such public processes are said to have the possibility to reduce implementation success [21], which could lead to faulty software that is expensive to resolve after the software is delivered [30]. With its challenges, it could seem like the project wizard is not assuring high quality and reducing costs as described in subsection 3.1.12, instead reducing quality and increasing costs.

Legal requirements either requested by the Norwegian government or required by Norwegian law are described in subsection 5.4.5 to create challenges in the assurance of quality in the Norwegian public sector. The challenges are described as (I) resources only covering legal requirements, (II) GDPR, and (III) laws unfit for digitisation. It can be argued that these challenges are changing in magnitude for software quality assurance, as the public sector is obliged to address these legal requirements [21]. As the legal requirements are susceptible to changes from political change and cycles, the priorities in legal requirements changing significantly with each new administration [21].

6.3 Cooperation Between Agencies

One of the current digitisation goals of the Norwegian public sector, set by the Norwegian government, is to increase cooperation between agencies through a shared digital platform [14]. However, it could seem like the agencies are lacking in resources to provide software of high quality to their own domain. To then use the already precious resources on co-operation, could lead to providing software of lower quality in their own domain.

Lack of resources was described as an issue, as agencies are only able to achieve what is legally required. It would therefore seem that if the Norwegian government want more cooperation between the agencies, more funding and resources is required. Or obtaining exemptions from certain legal requirements to work on digital cooperation. However, obtaining exemptions could be argued as less desirable, as it can decrease the value the agencies provide to Norwegian citizens.

None of the participants mentioned that cooperation through a shared digital platform would increase the value delivered to Norwegian citizens by the agencies. It could therefore be worth to investigate if these goals by the Norwegian government stem from a real problem. Or if the digitisation goals of the Norwegian government are the result of wishful thinking by leaders who are positioned far away from the agency's users.

It could also be worth investigating if digital cooperation is something which will yield any value to Norwegian citizens. It is the perception from the interviews in this study that the agencies have moved away from large applications which cover a wide range of needs. To smaller applications which cover more specialised needs. Something described as giving higher value to the Norwegian citizens, and by the participant's definition, increasing quality. It could therefore seem that developing large technical solutions that cover a wide range of needs mean stepping back in time.

The digital platform that the Norwegian government might wish to possess, is something resembling the NAIS platform, used by NAV. However it is important to remember that the NAIS platform is built specifically for NAV applications and NAV's domain. It could therefore be argued that if a digital platform across the Norwegian public sector wishes to be successful and deliver services of quality to Norwegian citizens, it needs to be as specialised as the NAIS platform.

The question is what organisation with enough domain knowledge of all the agencies in the Norwegian public sector are going to be responsible for the development and maintenance of the digital platform? The Norwegian digitisation agency might be a potential candidate, as they already develop shared solutions for the Norwegian public sector. Yet might not have enough resources and IT staff to develop and maintain a digital platform of the scale needed to cover the Norwegian public sector.

The criticism mentioned in this section could seem unjustified, as recent literature has explained that DevOps is expected to bring the different agencies in the public sector closer, increasing knowledge culture and collaborative work [26], which is shown to increase software quality [24]. The use of DevOps in the Norwegian public sector seems to be quite recent, and might not have time to be fully established in all the agencies using it. Therefore it might be the case that the Norwegian government preemptively released their strategy, without the Norwegian public sector being ready for its implementation and use.

At the time of this study, a new digitisation strategy is beginning to be planned. It will therefore be interesting to follow if the Norwegian government follow the same strategy, with the mentioned challenges. Or choose a different strategy with a lower level of ambition and a more detailed description of the problem trying to be solved. Leaving the agencies of the Norwegian public sector with a better understanding of what is to be expected from the Norwegian government, other than legal requirements.

6.4 Researchers Perspective

Even though the development environment of NAV has grown significantly with in-house developers, it can still be a challenge to acquire the necessary development resources. When a development team is not receiving enough resources, only the legal requirements and core needs of the users are met. Resulting in shortcuts in testing and code quality being made to meet the lowest criteria of the software.

NAV has also made efforts to make it easier to create NAV applications of high quality by creating tools specifically for the development of NAV applications. However, when not developing a typical NAV application, it can lead to the specialised assisting tools not being helpful. Resulting in the developers re-inventing the wheel to make the application fit their specific domain, using the already limited resources on tasks that do not directly benefit the Norwegian citizens.

These tools can increase the already steep learning curve for inexperienced developers at the agency. Platforms such as NAIS are simple to use when in possession of knowledge in DevOps and software development, as most time-consuming tasks have been automated and abstracted. However, without the knowledge of how such platforms function, it is difficult to understand what value is to be gained from using tools such as NAIS. Resulting in a large amount of resources being spent learning to use the platform, instead of the platform saving resources. The researcher described that he worked a year as a software developer at NAV, before having a good knowledge of how to use NAV's tools to create software of high quality. Due to the researchers not having the required knowledge in DevOps and software development.

6.5 Limitations

As this study is a case study with interviews as the main data generation method, the study could risk generalising the results and its discussion to the entire Norwegian public sector. As only 3 of the agencies in the Norwegian public sector were included, and only a few employees of these agencies were interviewed.

The participants in this study have also signed non-disclosure agreements, not to share data about the users of the services being provided by the particular agency. Leading to either the employees not sharing data relevant to the study, or the researcher not revealing data which could be considered sensitive to the particular user or employee.

The researcher's role as a software developer in the Norwegian public sector could be both positive and negative for the validity of the data collected in the study. It could be that the participants would be more open to sharing sensitive data. While at the same time being afraid to share the correct data, due to being scared of offending the researcher by criticism through the sharing of data.

Bias by the researcher, due to their position as a software developer in the Norwegian public sector can challenge the validity of the results presented. Due to the researcher having a conflict of interest in how the results are presented and discussed, in a way that could favour the agency the researcher is employed at, or the Norwegian public sector at large.

The lack of scientific literature on software quality assurance in the context of the Norwegian public sector could lead to validity challenges of the results presented in this study. As the results and arguments conducted have limited external data to be validated against. It could be that there is scientific literature relevant to this study, however, the limited time and resources for this study have halted any further search for relevant scientific literature.

Chapter 7

Conclusion

As the Norwegian government is moving the Norwegian public sector to offer connected social services through a shared digital platform. A range of non-functional functional requirements becomes shared between agencies in the Norwegian public sector in order to maintain the trust of Norwegian citizens. Leading to what can be suspected as a shared view on quality assurance between the agencies in the Norwegian public sector. The goal of this study has therefore been to understand through a case study, how software quality assurance is being practised, and what challenges are encountered in software quality assurance in the Norwegian public sector.

The case study shows that several agencies have moved away from the traditional ISO mindset, instead focusing on feedback from their users. Leading to the agencies implementing methodologies such as DevOps and Agile to assure quality in software through user feedback. The assurance of technical quality in software is also shown to be of importance for the agencies in this study, as practices such as inspection, measurements and testing are used to ensure high technical quality of the software. The challenges revealed in software quality assurance were shown to not be directly connected with the practices, but rather organisational factors relating to the context of the Norwegian public sector. These are the lack of resources and budgeting pushing the agencies to use the criticised project wizard delivered by the Norwegian digitisation agency. And legal requirements that leave little room for assurance of non-functional requirements such as usability.

When comparing the software quality assurance practices of the agencies in this study with the scientific literature, it can seem that the practices are what is recommended. However, being in the context of the public sector is the main challenge for producing software of high quality. These challenges need to be addressed for the Norwegian public sector to be able to create software of high quality in order to better serve the Norwegian public. As well as these challenges must be addressed before the Norwegian government can move forwards to deliver shared social services through a shared digital platform.

The findings of this study do have some limitations. As only 3 of the many agencies in the Norwegian public sector were included, and a limited number of employees at each agency were interviewed. As well as the research role as a software developer in the Norwegian public sector could lead to bias in favour of the agencies, especially NAV, and in disservice to the Norwegian government. The lack of scientific literature in the context of the Norwegian public sector could also lead to validity challenges for the findings of this study.

The results of this study should be used by agencies in either the Norwegian or any other public sector that is struggling with software quality assurance. As the practices discovered in this study can be useful. While the challenges revealed are recommended to be further studied to uncover methods to solve the challenges and reveal the potential benefits of these challenges being solved.

Bibliography

[1] Kubernetes. "Kubernetes." (2023), [Online]. Available: https://kubernetes.io/.

[2] Wikipedia. "Microservices." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Microservices.

[3] Wikipedia. "Mob programming." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Mob_programming.

[4] NAIS. "It is nais to be a developer at nav." (2023), [Online]. Available: https://docs.nais.io/.

[5] Swagger. "Openapi specificaiton." (2023), [Online]. Available: https://swagger.io/specification/.

[6] Wikipedia. "Pair programming." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Pair_programming.

[7] Wikipedia. "Service-oriented architecture." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Service-oriented_architecture.

[8] Wikipedia. "Technical debt." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Technical_debt.

[9] I. O. for Standardization. "Iso/iec 25010:2011." (2023), [Online]. Available: https://www.iso.org/standard/35733.html.

[10] I. of Electrical and E. Engineers. "Ieee standard for software quality assurance processes." (2023), [Online]. Available: https://standards.ieee.org/ieee/730/5284/.

[11] Wikipedia. "Software quality assurance." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Software_quality_assurance.

[12] M. Borg, J. Wernberg, T. O. U. Franke, and M. Andersson, "Illuminating a blind spot in digitalization - software development in sweden's private and public sector," *42nd Internation Conference on Software Engineering Workshops*, pp. 299–302, 2020.

[13] V. Kettunen, J. Kasurinen, O. Taipale, and K. Smolander, "A study on agility and testing processes in software organizations," *19th international symposium on Software testing and analysis*, pp. 231–240, 2010.

[14] Regjeringen. "En digital offentlig sektor." (2019), [Online]. Available: https://www.regjeringen.no/no/dokumenter/en-digital-offentlig-sektor/id2653874/.

[15] "Towards systematic specification of non-functional requirements for sharing economy systems," *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, vol. 15, pp. 423–429, 2019.

[16] L. Alzahrani, W. Al-Karaghouli, and V. Weerakkody, "Analysing the critical factors influencing trust in e-government adoption from citizens' perspective: A systematic review and a conceptual framework," *International Business Review*, vol. 26, pp. 164–175, 2017.

[17] OECD, *Drivers of Trust in Public Institutions in Norway*. OECD, 2022.

[18] Regjeringen. "Økte kostnader på 40 milliarder." (2022), [Online]. Available: https://www.regjeringen.no/no/aktuelt/budsjett-pengene-i-folketrygden/id2930068/.

[19] Regjeringen. "Statsbudsjettet 2022: Statens inntekter og utgifter." (2021), [Online]. Available: https://www.regjeringen.no/no/statsbudsjett/2022/statsbudsjettet-2022-statens-inntekter-og-utgifter/id2873448/.

[20] NAV. "Organisering av nav." (2023), [Online]. Available: https://www.nav.no/no/nav-og-samfunn/om-nav/fakta-om-nav/organisering-av-nav.

[21] J. Campbell, C. McDonald, and T. Sethibe, "Public and private sector it governance: Identifying contextual differences," *Australasian Journal of Information Systems*, vol. 16, pp. 5–18, 2010.

[22] Wikipedia. "Devops." (2023).

[23] A. Mishra and Z. Otawiw, "Devops and software quality: A systematic mapping," *Computer Science Review*, vol. 38, 2020.

[24] S. M. Mohammad, "Improve software quality through practicing devops automation," *Internation Journal of Creative Research Thoughts*, vol. 6, pp. 251–256, 2018.

[25] M. Lazaurdi, T. Raharjo, B. Hardian, and T. Simanungkalit, "Perceived benefits of devops in organization: A systematic literature review," *International Conference on Software and Information Engineering*, vol. 10, pp. 10–16, 2022.

[26] M. Mubarkoot, "Assessment of factors influencing adoption of devops practices in public sector and their impact on organizational culture," *ICST conference*, vol. 2, pp. 475–483, 2021.

[27] D. S. Cruzes, K. Melsnes, and S. Marczak, "Testing in a devops era: Perceptions of testers in norwegian organisations," *International Conference on Computational Science and Its Applications*, pp. 442–455, 2019.

[28] J. Angra, S. Prasad, and G. Sridevi, "The factors driving testing in devops setting- a systematic literature survey," *Indian Journal of Science and Technology*, vol. 9, pp. 1–8, 2016.

[29] A. Wiedemann, M. Wiesche, and H. Krcmar, "Integrating development and operations in cross-functional teams - toward a devops competency model," *Computers and People Research Conference*, vol. 19, pp. 14–19, 2019.

[30] C. S. Wright and T. A. Zia, "A quantitative analysis into the economics of correcting software bugs," *Lecture Notes in Computer Science*, vol. 6694, pp. 198–205, 2011.

[31] J. E. Hannay, T. Dybå, E. Arisholm, and D. I. Sjølberg, "The effectiveness of pair programming: A meta-analysis," *Information and Software Technology*, vol. 51, pp. 1110–1122, 2009.

[32] M. Vidoni, Z. Codabux, and F. H. Fard, "Infinite technical debt," *Journal of Systems and Software*, vol. 190, 2022.

[33] J. K. Chhabra and V. Gupta, "A survey of dynamic software metrics," *Journal of Computer Science and Technology*, vol. 25, pp. 1016–1029, 2010.

[34] SNL. "Offentlig sektor." (2022), [Online]. Available: `https://snl.no/offentlig_sektor`.

[35] Wikipedia. "Public sector." (2023), [Online]. Available: `https://en.wikipedia.org/wiki/Public_sector`.

[36] Regjeringen. "Norwegian labour and welfare organization (nav)." (2023), [Online]. Available: `https://www.regjeringen.no/en/dep/bfd/organisation/tilknyttede-virksomheter/Norwegian-Labour-and-Welfare-Organizatio/id426155/`.

[37] Regjeringen. "The norwegian tax administration (skatteetaten)." (2023), [Online]. Available: `https://www.regjeringen.no/en/dep/fin/about-the-ministry/subordinateagencies/the-tax-administration---the-directorate/id270405/`.

[38] Regjeringen. "The norwegian food safety authority." (2023), [Online]. Available: `https://www.regjeringen.no/en/dep/hod/organisation-and-management-of-the-ministry-of-health-and-care-services/etater-og-virksomheter-under-helse--og-omsorgsdepartementet/Subordinate-institutions/Norwegian-Food-Safety-Authority/id279765/`.

[39] Mattilsynet. "Om mattilsynet." (2023), [Online]. Available: `https://www.mattilsynet.no/om_mattilsynet/`.

[40] Regjeringen. "Digitaliseringsdirektoratet." (2023), [Online]. Available: `https://www.regjeringen.no/no/dep/kdd/org/etater-og-virksomheter-under-kommunal--og-distriktsdepartementet/underliggende-etater/digitaliseringsdirektoratet/id2684200/`.

[41] S. Pensjonskasse. "Om oss." (2023), [Online]. Available: `https://www.spk.no/om-oss/`.

[42] Wikipedia. “Norsk tipping.” (2023), [Online]. Available: https://no.wikipedia.org/wiki/Norsk_Tipping.

[43] Entur. “Om entur.” (2023), [Online]. Available: https://om.entur.no/om-entur/.

[44] Landbruksdirektoratet. “Om direktoratet.” (2023), [Online]. Available: https://www.landbruksdirektoratet.no/nb/om-direktoratet.

[45] Nortura. “Nortura er en av norges største matprodusenter.” (2023), [Online]. Available: https://www.nortura.no/om-nortura.

[46] Wikipedia. “Nrk.” (2023), [Online]. Available: https://no.wikipedia.org/wiki/NRK.

[47] Wikipedia. “Brønnøysundregistrene.” (2023), [Online]. Available: https://no.wikipedia.org/wiki/Br%C3%B8nn%C3%B8ysundregistrene.

[48] Regjeringen. “Hva er statens prosjektmodell?” (2019), [Online]. Available: https://www.regjeringen.no/no/tema/okonomi-og-budsjett/statlig-okonomistyring/ekstern-kvalitetssikring2/hva-er-ks-ordningen/id2523897/.

[49] Digitaliseringsdirektoratet. “Hva er prosjektveiviseren?” (2023), [Online]. Available: https://prosjektveiviseren.digdir.no/prosjektfasene/hva-er-prosjektveiviseren/5.

[50] Wikipedia. “Computer performance.” (2023), [Online]. Available: https://en.wikipedia.org/wiki/Computer_performance.

[51] Wikipedia. “Scalability.” (2023), [Online]. Available: https://en.wikipedia.org/wiki/Scalability.

[52] Wikipedia. “Software portability.” (2023), [Online]. Available: https://en.wikipedia.org/wiki/Software_portability.

[53] Wikipedia. “Reliability engineering.” (2023), [Online]. Available: https://en.wikipedia.org/wiki/Reliability_engineering.

[54] Wikipedia. “Maintainability.” (2023), [Online]. Available: https://en.wikipedia.org/wiki/Maintainability.

[55] Wikipedia. “Computer security.” (2023), [Online]. Available: https://en.wikipedia.org/wiki/Computer_security.

[56] Wikipedia. “Usability.” (2023), [Online]. Available: https://en.wikipedia.org/wiki/Usability.

[57] Wikipedia. “Software testing.” (2023), [Online]. Available: https://en.wikipedia.org/wiki/Software_testing.

[58] Wikipedia. “Acceptance testing.” (2023), [Online]. Available: https://en.wikipedia.org/wiki/Acceptance_testing.

[59] Wikipedia. “Black-box testing.” (2023), [Online]. Available: https://en.wikipedia.org/wiki/Black-box_testing.

[60] Wikipedia. "Test automation." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Test_automation.

[61] Wikipedia. "Waterfall model." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Waterfall_model.

[62] Wikipedia. "Waterfall model." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Agile_software_development.

[63] Wikipedia. "Continuous integration." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Continuous_integration.

[64] Wikipedia. "Continuous delivery." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Continuous_delivery.

[65] Wikipedia. "Continuous deployment." (2023), [Online]. Available: https://en.wikipedia.org/wiki/Continuous_deployment.

[66] M. Cagan. "Product vs. feature teams." (2019), [Online]. Available: https://www.svpg.com/product-vs-feature-teams/.

[67] B. J. Oates, M. Griffiths, and R. McLean, *Researching Information Systems and Computing*. Sage Publishing, 2022.

[68] M. Teams. "View live transcription in a teams meeting." (2022), [Online]. Available: https://support.microsoft.com/en-us/office/view-live-transcription-in-a-teams-meeting-dc1a8f23-2e20-4684-885e-2152e06a4a8b.

[69] OpenAI. "Introducing whisper." (2022), [Online]. Available: https://openai.com/blog/whisper/.

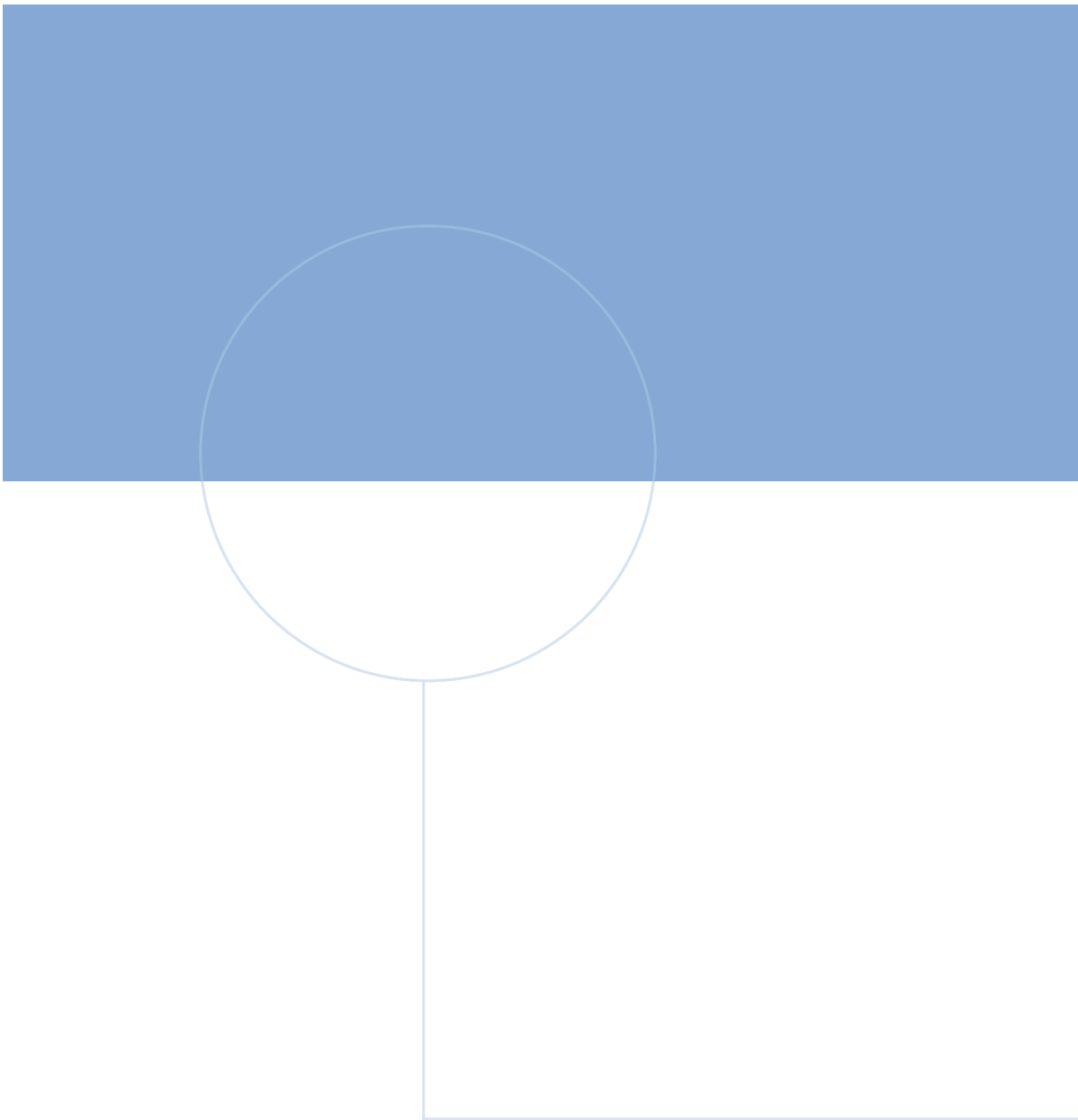
[70] U. i Oslo. "Transkriber med autotekst." (2023), [Online]. Available: https://autotekst.uio.no/.

[71] G. Lee and W. Xi, "Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility," *MIS Quarterly*, vol. 34, pp. 87–114, 2010.

[72] M. Huo, J. Verner, L. Zhu, and M. A. Babar, "Software quality and agile methods," *International Computer Software and Applications Conference*, vol. 28, pp. 520–525, 2004.

[73] NAV. "Nav-ytelsene frem mot 2060." (2019), [Online]. Available: https://www.nav.no/_/attachment/download/6cd7f195-3d79-4abc-a185-c55ba9e97a65:3da22b0bd21a5f920b5a7a8c52c211c846689039/NAV_Notat_022019_NAV-ytelsene%5C%20frem%5C%20mot%5C%202060_En%5C%20oppdatering.pdf.

[74] Finansdepartementet. "Hovedtall for norsk økonomi og i statsbudsjettet." (2020), [Online]. Available: https://www.regjeringen.no/contentassets/fada85d4686f486ebc31dd278a7a7778/faktaark_hovedtall.pdf.



NTNU

Norwegian University of
Science and Technology