Intelligent microscope III

PDEng THESIS REPORT

Intelligent Microscope III

Mark Laane
October / 2020
Department of Mathematics & Computer Science

PDEng SOFTWARE TECHNOLOGY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Intelligent Microscope III

Mark Laane

Eindhoven University of Technology
Stan Ackermans Institute - Software Technology

Partners	Thermo Fisher Scientific Eindhoven University of Technology
Steering Group	Mark Laane Remco Schoenmakers Mykola Pechenizkiy Yulong Pei
Date	October 5, 2020
Confidentiality Status	**Confidential** until October 5, 2021
PDEng report	PDEng 2020/048

Composition of the Thesis Evaluation Committee:

Chair:	ir. Harold. Weffers, PDEng
Members:	dr. Remco Schoenmakers
	prof. dr. Mykola Pechenizkiy
	dr. Yulong Pei
	Joost Dierkse
	dr. Nikolay Yakovets

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Contact Address
Eindhoven University of Technology
Department of Mathematics and Computer Science
MF 5.080A, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
+31 402472759

Published by
Eindhoven University of Technology
Stan Ackermans Institute

PDEng report
PDEng 2020/048

Abstract
Thermo Fisher Scientific (TFS) aims to become the leading company in electron microscopy by driving the innovation in the company and the electron microscopy industry. The purpose of this project was to identify and de-risk novel technologies that have the potential to significantly improve the ease of use of electron microscopes in five years. In this project, three technologies were selected for investigation, evaluation, and prototyping: (1) multilingual VUI for operating the microscope, (2) VR for interacting with the microscope, and (3) P2P architecture for sharing neural networks between microscopes. The study investigated the feasibility of applying those technologies through prototyping. For each technology, a prototype was built to test the feasibility of applying it in the electron microscopy domain and to identify the technical challenges that TFS would encounter. The prototypes in this project show how each technology could be implemented and the report highlights the limitations of each approach and technology. From those three prototypes the following major conclusions can be drawn: (1) Translating an English-speaking chatbot, using a general-purpose transcription and translation services from Google, Amazon or IBM is not a viable approach due to insufficient transcription accuracy; (2) Using VR as an alternative UI to interact with the microscope is viable, but finding a use case where the user would clearly benefit from the immersiveness from the technology is challenging; (3) Distributing neural networks using P2P architecture is possible, but building a reliable P2P network from scratch is challenging. To build one, relying on existing P2P protocols such as IPFS could reduce the development effort significantly and therefore warrant a future evaluation.

Keywords
Software Design, Electron Microscopy, Voice User Interfaces, Virtual Reality, Peer-to-Peer Architecture, Artificial Neural Networks

Preferred reference
Mark Laane, Intelligent Microscope III. Eindhoven University of Technology, PDEng Report, PDEng 2020/048, October 2020.

Partnership
This project was supported by Eindhoven University of Technology and Thermo Fisher Scientific.

implied, nor does it assume any legal liability, whether direct or indirect, or responsibility for the accuracy, completeness, or usefulness of any information.

Trademarks

Product and company names mentioned herein may be trademarks and/or service marks of their respective owners. We use these names without any particular endorsement or with the intent to infringe the copyright of the respective owners.

ABSTRACT

Thermo Fisher Scientific (TFS) aims to become the leading company in electron microscopy by driving the innovation in the company and the electron microscopy industry. The purpose of this project was to identify and de-risk novel technologies that have the potential to significantly improve the ease of use of electron microscopes in five years. In this project, three technologies were selected for investigation, evaluation, and prototyping: (1) multilingual VUI for operating the microscope, (2) VR for interacting with the microscope, and (3) P2P architecture for sharing neural networks between microscopes. The study investigated the feasibility of applying those technologies through prototyping. For each technology, a prototype was built to test the feasibility of applying it in the electron microscopy domain and to identify the technical challenges that TFS would encounter. The prototypes in this project show how each technology could be implemented and the report highlights the limitations of each approach and technology. From those three prototypes the following major conclusions can be drawn: (1) Translating an English-speaking chatbot, using a general-purpose transcription and translation services from Google, Amazon or IBM is not a viable approach due to insufficient transcription accuracy; (2) Using VR as an alternative UI to interact with the microscope is viable, but finding a use case where the user would clearly benefit from the immersiveness from the technology is challenging; (3) Distributing neural networks using P2P architecture is possible, but building a reliable P2P network from scratch is challenging. To build one, relying on existing P2P protocols such as IPFS could reduce the development effort significantly and therefore warrant a future evaluation.

FOREWORD

What a difficult time it is to graduate a PDEng. What started out as a ''normal'' year ended up in all of us trying to adjust to a ''new normal''. It meant less personal contact (in fact, I have not been in the office at all since early March) and less opportunity for ''bouncing ideas back and forth''. In this setting, Mark had to create his research, develop his code, and write his thesis. That was tough, and the fact that he persisted in doing his work is laudable.

Mark had the additional challenge that he had to further extend the work of two previous PDEng's who both did a very good job in scooping the best and most obvious use cases in a project that has as an onset the exploration of new and upcoming digital technologies in the context of electron microscopy. The depletion of the ''low hanging fruit'' basket would have been challenging under normal circumstances, but it was even harder in the current time, where simply walking around some offices to get inspiration is not possible, and Mark has had little time to build up a network of contacts in the company before the lockdown started.

Nevertheless, we had identified a couple of nice opportunities: what can you do with Virtual Reality? Can we build further on the successful demonstrator of voice control of the previous PDEng and make it language agonistic? What can we do if we connect microscopes together and have them exchange AI-related information, and how can we do that?

The risk of science and research is that there is a serious possibility that a path of exploration leads to no result or at least the identification that the topic of research is not leading to a situation that requires follow-up development or productization. That does not degrade the result of the research, on the contrary, it means you have provided an answer to the scientific question being asked. That the answer is not what you would have hoped for, does not diminish the relevance of it. This is the case with the three topics that Mark examined. We found that VR, although shown technically feasible by Mark, does not appear to have a really compelling use case for end-customer use in electron microscopy (we already knew it does have a position in Service, so that was not examined). This is extremely valuable information, as every now and then, questions arise around the usefulness of VR. We now have a report in hand that has data, rather than gut-feelings, to accurately address these questions.

The second question around language-agnostic voice control was a high-risk high-reward topic. If it would work, it would have been a ''killer app''. Mark tried many things, and the conclusion was that the available translation services at this moment in time are not mature enough to get a sufficiently high rate of understanding. So, if we keep monitoring the progress of these services, we may revisit this once there are signs that they may be significantly better, and we now have the architecture and design in place to quickly benchmark, and perhaps even productize these future developments.

The final part was on instrument connectivity. We already have and use a ''central database'' system, where instruments upload their information to a central store that can be queried by the instruments or outside parties. But why do we need such a ''central point of failure''? Could we set up a much more dynamic system of shared resources and responsibilities and what challenges, benefits and drawbacks do we encounter? Unfortunately, time was running short, so this has not been explored in full detail, but I am sure we are going to revisit these ideas at a later stage.

All in all, Mark has given us significant contributions and information around the (in)feasibility of these hyped technologies in the context of electron microscopy, in an environment that would have led many people to simply give up. It is a sign of character that he managed to finish the code, the thesis, and his graduation in such a commendable way.

Dr. Remco Schoenmakers
Director of Digital Science Technologies
September 15, 2020

ACKNOWLEDGMENTS

Without the support of many people around me, this project would have never been accomplished. I would like to take this opportunity to thank everyone who helped and supported me during this 10-month-long journey

First, I would like to thank my company supervisor, Remco Schoenmakers. Thank you for opening the door to electron microscopy. It is amazing how much concentrated information I could get in an hour of meeting, but all delivered in a simple and easy-to-understand language. You invested a lot of time showing me how my work relates to other things happening in the company. I am also very grateful to my TU/e supervisors Mykola Pechenizkiy and Yulong Pei for their feedback, report reviews, tips, and suggestions.

I would also like to thank the Software Technology program director Yanja Daisuren, and management assistant Désirée van Oorschot for propelling this program forward. During the last two years, I had the chance to work and study in TU/e with remarkable colleagues and I would like to thank you all for the time spent together.

I would also like to thank my friend Arne, who sparked the idea of pursuing this program and has always motivated me to explore the world and expand my comfort zone.

Most importantly, I am thankful to my parents, grandparents, and my family, who have always been there for me. Thank you, my grandmother, for wise counsel and encouragement during hard times. Last but not least, I am thankful to my girlfriend Laura for her unwavering support and kindness.

Thank you!

Mark Laane
September 2020

PREFACE

This document is the technical report of the project Intelligent Microscope III. The main objective of this project was to explore novel technologies that could change the way users interact with electron microscopes in the future.

This project was carried out by Mark Laane within Thermo Fisher Scientific between January 2020 and October 2020. It is the author's ten-month final project required to obtain a Professional Doctorate in Engineering (PDEng) degree in Software Technology at the Eindhoven University of Technology.

This document is primarily intended for readers with a software engineering background who are interested in the application of novel technologies in the electron microscopy domain.

Mark Laane
September 2020

EXECUTIVE SUMMARY

This document is a technical report of the Intelligent Microscope III project. The purpose of this project was to identify and de-risk novel technologies that have the potential to significantly improve the ease of use of electron microscopes in five years.

From a pool of emerging and novel technologies listed in Chapter 3, three technologies were selected for investigation, evaluation, and prototyping. The selected technologies were: multilingual voice user interface (VUI) for operating the microscope, Virtual Reality (VR) for interacting with the microscope, and peer-to-peer (P2P) architecture for sharing neural networks between microscopes.

For each technology, a prototype was built to test the feasibility of applying it in the electron microscopy domain and to find out the technical challenges that Thermo Fisher Scientific (TFS) would encounter.

The main results from experimentation and prototyping presented in Section 5.1 are:

- Translating an English-speaking chatbot, using a general-purpose transcription and translation services from Google, Amazon or IBM is not a viable approach. The main limiting factor is the accuracy of general-purpose speech-to-text services. The multilingual voice user interface prototype was uncomfortable to use because it was not accurate enough in recognizing the user's intent.
- VR can be used as an alternative UI to control and interact with microscopes, given the needed functionality is available through an API, not just GUI. Nevertheless, finding a use case where the user would clearly benefit from the immersiveness from the technology is not easy.
- Distributing neural networks using P2P architecture is possible by transferring neural networks in SavedModel format between microscopes. Building a P2P platform from scratch as done in this project is not suggested.

Recommendations discussed in Section 5.2 of this report include:

- For multilingual voice user interfaces, additional experimentation is needed to find a more accurate multilingual speech recognition service. Alternatively, we could wait for general-purpose speech-to-text services to improve over time and then repeat the experiment.
- For Virtual Reality, further exploration and experimentation are needed to find a suitable use case that would benefit from the immersiveness of the VR medium.
- For the P2P model sharing platform, I suggest investigating existing P2P protocols such as IPFS to reduce the development effort of a P2P platform.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Innovation – the successful exploitation of ideas – is the key competitive advantage for a company in a highly competitive and dynamic market. Innovation enhances the competitiveness of a company in two distinct ways. First, it is the source of new products and services that can be sold. Secondly and more importantly, the process of innovation transforms the company itself, making it more adaptive, better able to learn, and better in exploiting new ideas, and better in adapting to changing markets. [1]

This project aims to spark innovation and inspire people in Thermo Fisher Scientific by demonstrating the usefulness of novel technologies in electron microscopy.

In this chapter, I introduce the Intelligent Microscope III project and its background. Section 1.1 presents the context of the project. Section 1.2 defines the aim and goals of this project and the general approach taken in this project. The structure of the rest of the report is described in Section 1.3.

1.1 CONTEXT

Intelligent Microscope III (IM3) is a research project that was executed in Thermo Fisher Scientific (TFS). This project is also a ten-month graduation project of the PDEng Software Technology program. As the name suggests, it is the third successor of a series of PDEng graduation projects: Intelligent Microscope I (IM1) by Dmytro Kondrashov in 2018 and Intelligent Microscope II (IM2) by Konstantinos Smanis in 2019. Therefore, this project builds upon an existing software system by extending and improving it.

TFS is an international company with a mission to serve science by enabling its customers to make the world healthier, cleaner, and safer. The company is engaged in a wide range of activities in advancing science – from provisioning reagents and consumables and building scientific instruments to providing software and services.

TFS aims to become the leading company in electron microscopy through full automation of the electron microscopes and a software ecosystem that partners with researchers to gain knowledge. This vision serves as a common goal for many projects in TFS.

This project was executed in the department of Advanced Technologies (AT) in Eindhoven. The department is investigating how novel technologies could innovate electron microscopy and focuses on researching and testing innovative technologies to answer the question: how will users interact with the microscope in the future?

The vision of the project series

The Intelligent Microscope (IM) project series was started in TFS to boost innovation in the company. To inspire people inside the company to adopt new technologies and to demonstrate the usefulness or un-usefulness of new technologies to developers and managers. Testing novel technologies first in a separate project can reduce the risks when adopting them in main projects.

Uncovering innovative, useful, and exciting use cases for novel technologies is relevant, as it would give TFS a competitive advantage in the market. In electron microscopy, TFS is competing with companies such as JEOL, Nikon, and Zeiss.

The overarching goal of the Intelligent Microscope project series can be captured in the following vision statement:

> *The intelligent Microscope project series aims to spark innovation and inspire people in Thermo Fisher Scientific by demonstrating the usefulness of novel technologies.*

Project series strategy
While the vision statement expresses the motivation of the project, it does not define how to achieve the described change; this is captured by a strategy. Several strategies can be devised, but vision guides in choosing a suitable strategy. Therefore, a clear vision is a prerequisite for choosing a suitable strategy towards it.

1.2 PROJECT GOAL

This project aims to identify novel technologies that have the potential to significantly improve the ease of use of electron microscopes in five years. The main output of this project is knowledge and insights about the chosen technologies: the opportunities and technical challenges that they bring. This knowledge is used to spark innovation and potentially start new software development projects in the company. The insights generated from the Intelligent Microscope III project aim to reduce the uncertainties that these new software development projects face when trying to use novel technologies.

Because this project aims to envision microscope usage in the future, a "Technology Push" approach was taken. The project started with a selection of novel technologies and continued with a series of experiments trying to investigate the feasibility of applying those technologies in electron microscopy. These experiments aimed to uncover the technical difficulties and opportunities brought forward by the technologies. The opposite approach would be "Market Pull" – starting from an end user's problem and trying to solve it with different technologies. Starting from the technologies instead of market research meant that the project did not directly try to solve the problems users have today. Instead, the project was expected to generate insights into the technologies and uncover opportunities for improving the experience of using a microscope in the future.

1.3 OUTLINE

The report is outlined as follows. Chapter 2 describes the problem domain and previous iterations in the IM project series. In Chapter 3, the list of candidate technologies is presented along with research questions. Chapter 4 covers the three experiments conducted during this project. Chapter 5 provides high-level conclusions from the experiments. Project management has been described in Chapter 6. Finally, Chapter 7 presents the author's reflections on the project in a retrospect.

2. DOMAIN ANALYSIS

This chapter gives an overview of the domain analysis performed during the Intelligent Microscope project. Section 2.1 provides a brief overview of electron microscopy.

Domain analysis is the process of collecting information about the domain and problems in the domain. In software engineering, the term domain refers to the general field of business where the upcoming software system is going to be used. This information can be acquired from different sources: existing software systems, literature, and domain experts. This background knowledge is useful for making good decisions during the whole software engineering process. [2]

2.1 ELECTRON MICROSCOPY

To take an image of a flower, we can just use a camera and take a photograph. The camera captures the visible light bounced and scattered from the flower and directs it onto a light-sensitive surface, such as a photographic film or a digital sensor, creating an image. To take images of small objects and samples such as human hair and skin cells, we can combine the camera with an optical microscope that uses a system of lenses to magnify the small detailed samples.

Nevertheless, taking images of very small objects such as MEMS-devices, nanomaterials, crystals, viruses, and proteins is not that easy. As the objects and their features get smaller, we hit the resolution limit of optical microscopes that use visible light. With a modern microscope, we can distinguish features if they are at least 200nm apart [3]. To see objects smaller than that, we need a more powerful imaging technique.

Electron microscopy is a technique for obtaining high-resolution images of very small objects, providing a spectacular level of detail. It can distinguish features as small as 0.1nm and the very best electron microscopes can show us individual atoms. Instead of visible light, electron microscopes use a beam of high-energy electrons. [3] The higher resolving power of electron microscopes comes from the lower wavelength associated with electrons.

2.1.1 Applications

Electron microscopy has a wide range of applications ranging from semiconductor failure analysis, drug research, and forensics to structural biology. Electron microscopes and are used by research laboratories, universities, and companies worldwide to investigate the world at the nanoscale.

In biology and life sciences, electron microscopes are used to study living organisms: animals, plants, bacteria, and viruses. Figure 1 shows an image of an ant taken with a scanning electron microscope. The high resolving power of electron microscopes is even more important in cellular and structural biology research. Cellular research focuses on individual cells and how they organize into tissues and organs. Structural biologists explore the sub-cellular components such as organelles. It enables the research and exploration of the molecular mechanisms of infectious diseases and pathogens, and for the development of new drugs. Figure 2 shows an image of COVID-19 viral particles taken with a transmission electron microscope.

In material sciences, electron microscopes help scientists to understand the link between the structure, composition, and properties of materials. This enables the researchers to design new materials and improve the existing ones. [4]

Figure 1: An image of an ant in a scanning electron microscope. [5]

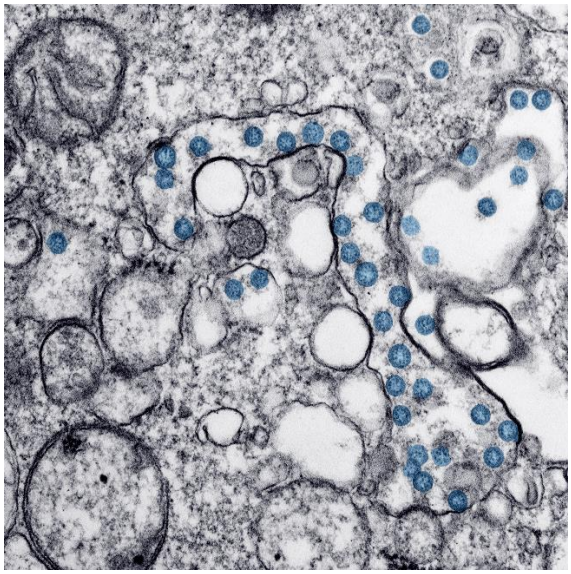

Figure 2:Transmission electron microscopic image of COVID-19. The spherical viral particles are colorized blue. [6]

2.1.2 Users of Electron Microscopes

Electron microscopes are highly specialized tools with a high cost and are therefore considered to be a long-term investment. In addition to the high cost of the machine itself, the costs include installation, servicing, repairing, and operating costs.

Currently, the operators of the electron microscopes are scientists with MSc and Ph.D. degrees in Life Science, Material Science, and Physics, working in a wide range of industries. Operating an electron microscope is not a simple task. To acquire good-quality EM images, the operator must have sufficient knowledge of the principles of electron microscopy and must have completed training for operating procedures.

To expand the market reach, TFS is trying to widen the range of the potential user base by lowering the barriers to operating a microscope. The first goal is to reduce the expertise and training needed to operate a microscope. Another goal is to allow non-English speaking users to control and use the microscope.

2.2 Overview of Previous Projects in the Series

Intelligent Microscope III (IM3) is the continuation of two past projects – Intelligent Microscope I (IM1) and Intelligent Microscope II (IM2). The first project was a green-field project. The result of the IM1 project was a software system with the same name. The second project built on the work of IM1 by extending the functionality of the software system.

The authors of the first two iterations of the project (IM1 and IM2) chose to build and extend an experimentation and demonstration platform called Intelligent Microscope. It is a software system where many different novel technologies are explored and combined. For the management of TFS, it is a source of knowledge of what is and what is not achievable with the chosen technologies. For developers of future products, it is a working example and a technical guide. For developers of the Intelligent Microscope, it is a platform that provides complete freedom to experiment and generate new insights.

Intelligent Microscope I

The first Intelligent Microscope project resulted in a platform that allowed the user to control the Scanning Electron Microscope (SEM) with voice commands such as “zoom in” and “please, acquire an image.” The system was also able to detect cells and mitochondria in the images. Figure 3 illustrates the scope of the system that was built during the Intelligent Microscope I project.

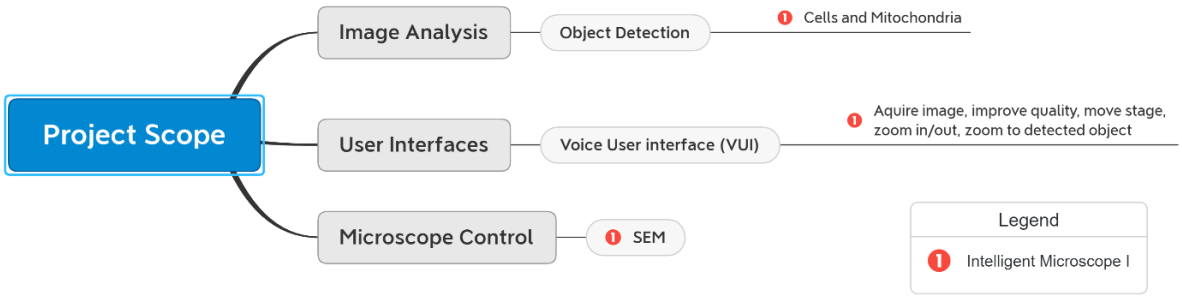

Figure 3: Scope of the system built during the Intelligent Microscope I project

Intelligent Microscope II

The second Intelligent Microscope built upon IM1 by extending the capabilities of the Intelligent Microscope. As can be seen in Figure 4, Intelligent Microscope can detect asbestos fibers from the SEM images and de-noise and colorize images. The system was extended to support additional devices such as Transmission Electron Microscopes (TEM) in the STEM mode and Phenom microscopes. Based on this, the Voice User interface was proposed as a solution for unifying the UIs of different microscopes. A command pattern was used to allow the user to undo and redo commands.

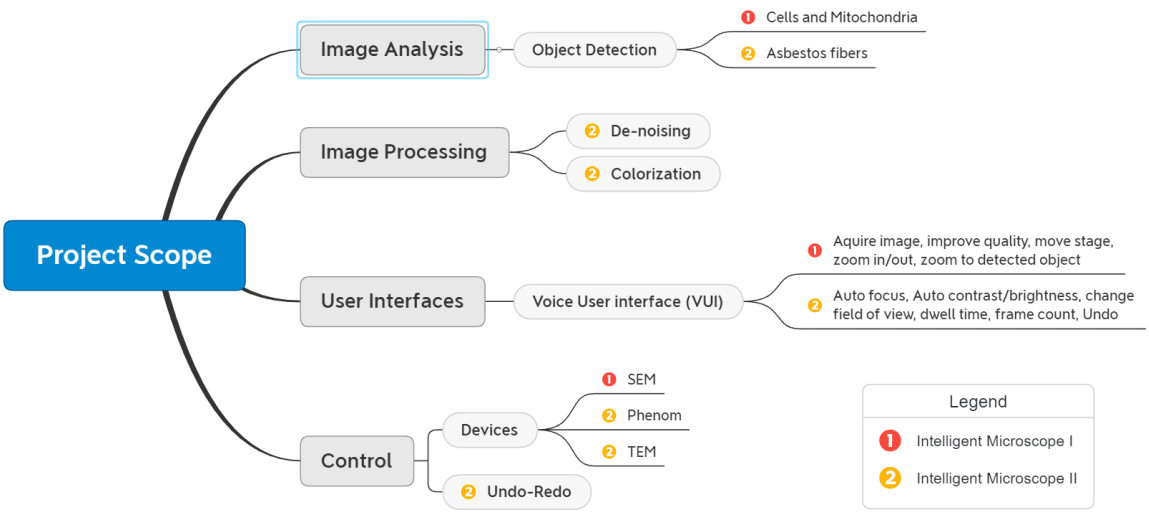

Figure 4: Scope of the system built during the Intelligent Microscope II project

3. CANDIDATE TECHNOLOGIES IN IM3

In this section, the technologies that TFS was interested in exploring are presented. The section also provides an overview of how the technologies were selected for the IM3 project. Finally, the three chosen technologies are presented.

The first two iterations in the IM project series, IM1 and IM2, left a lot of interesting technology challenges untouched. IM3 project was initiated to explore and test a selection of those technologies. Additionally, other innovative technologies were identified as possible exploration areas for IM3.

At the start of the project, I conducted multiple interviews with Remco, the main stakeholder in TFS, to identify the candidate technologies for exploration. I identified the following broad technology areas that TFS was interested in exploring:

- Virtual and Augmented Reality (VR and AR)
- Voice User Interfaces (VUI)
- Semantic Web and Knowledge Graphs
- Artificial Intelligence and Machine Learning (AI and ML)

Within those technology areas, the following functionalities were identified to be of interest:

- VR and AR:
    - Ability to operate the microscope in VR.
    - Enhanced user experience in managing a fleet of instruments in an immersive VR environment.
- VUI:
    - Ability to control the microscope in a wide range of spoken languages
- Semantic Web and Knowledge Graphs:
    - Giving meaning and context to objects observed with the microscope
    - Helping the user to choose a suitable experiment with TFS tools.
- AI and ML:
    - Microscopes sharing knowledge and learnings
    - Incorporating any algorithm that is easy to integrate and relevant

Investigating all of them would not fit into the scope of a ten-month project. Therefore, I prioritized the list based on the degree of interest of the customer. At the end of each experiment, we reprioritized the list and chose the topic of the next experiment.

In the following sections, I present three experiments – each one investigating a different technology or novel functionality. For each experiment, I present the main research question or challenge and short reasoning why it was interesting for TFS.

3.1 MULTILINGUAL VOICE USER INTERFACE

TFS is a global company and therefore is interested in offering service also to non-English speaking users. IM1 and IM2 have a voice user interface that allows the use of voice commands to control the microscope. Given that the user knows what she wants to do, she can just say it to the application and it would fulfill the command. It works surprisingly well, but the limitation is that it supports only the English language.

Voice User Interface was seen by TFS as a technology that would be easy to automatically translate and localize to other languages. The idea was to try using general-purpose machine translation services such as Google Translate to turn an existing English-language VUI into a multilingual VUI. The feasibility and accuracy of this solution had to be tested. If this approach would work well enough, it would save TFS time and money in translating the user interfaces.

Research question: **Is it possible to automatically translate the Voice User Interface of Intelligent Microscope by leveraging existing automated translation services? If so, how?**

We expected that this kind of system would work worse than the original English-only VUI, but well enough for most of our use cases. I believed the system to have systematic errors in translations, where the translation service would systematically translate specific words or phrases in the wrong way. If that was the case, it would be possible to simply post-process the translations to fix the systematic errors and improve the system's accuracy in understanding the foreign language.

Hypothesis: Yes, it is possible and the resulting multilingual VUI works well enough for most of the use cases but would have systematic errors caused by mistranslations.

3.2 Virtual Reality

TFS was interested in applying VR in the electron microscopy by allowing the user to operate the microscope in VR. VR seemed interesting to TFS because of its ability to improve the focus, awareness, and performance of the user while interacting with the microscope in the artificial environment.

This experiment aims **to assess the realizability and usefulness of applying VR in operating an electron microscope**.

Hypothesis 1: It is possible to show microscopy images and interact with IM in VR.

Hypothesis 2: VR can be used to intuitively and comfortably navigate (pan and zoom) the sample in the electron microscope.

3.3 Peer-to-Peer Platform for Sharing Neural Networks

Today, TFS electron microscopes sometimes need to use ML-models to perform advanced tasks such as anomaly detection and object classification. The distribution and deployment of those models on the microscopes is still an open problem. TFS has currently implemented and created a central machine-learning service that the microscopes can access to perform inference. The goal of this experiment was to explore the opposite direction: how we could share knowledge in the form of neural networks directly between the microscopes without relying on any external servers.

The research questions: **Is it possible to set up a P2P system for sharing neural networks? If so, what kind of opportunities and technical challenges would it bring?**

Hypothesis 1: Sharing neural networks is possible by exchanging the trained neural-network-model files between microscopes.

Hypothesis 2: Compared to the client-server architecture, sharing neural-network-model files directly between microscopes would result in a highly fault-tolerant system that would be resistant to hardware and software failures.

4. TECHNOLOGIES INVESTIGATED

During the Intelligent Microscope III project, I investigated three different technologies: multilingual voice user interfaces (VUI), virtual reality (VR), and peer to peer sharing of neural networks. The following three sections cover each one of them in detail.

4.1 MULTILINGUAL VOICE USER INTERFACE

This chapter describes how to improve the VUI of Intelligent Microscope, by making it multilingual and reusable. It covers available technologies, different possible approaches, the chosen architecture, implementation, deployment, and validation of the result.

4.1.1 Introduction

Why is supporting multiple languages important for TFS?

Allowing non-English speaking users to interact with microscopes is important for TFS because it is a global company with a mission to be the world leader in serving science. Adding support for multiple languages in user interfaces can drastically improve TFS's foothold in non-English speaking markets. The most important direction is China because it is a large and fast-growing market, but other regions where Spanish, Japanese, and French are spoken are also of interest. Each additional language that TFS microscope UI offers expands the potential user base TFS can serve.

In non-English speaking markets, a multilingual user interface reduces the cost of ownership by reducing the operating and training costs. The owners can hire operators locally and forego extra training of operators needed to give them the ability to understand the English user interface. This makes the TFS microscope financially more compelling over the competitors' products.

What are the challenges of translating existing user interfaces to multiple languages?

Today, TFS microscopes can be operated through GUIs that are available only in English. Localizing a GUI can be a challenge for numerous reasons. First, adding the support of switching languages to an application can be a technical challenge. All the strings must be extracted from the application code and dynamically loaded. Even when TFS achieves the technical capability to support localization of all their applications, there is more work to do. All the texts and strings in the application have to be translated and localized by language experts one by one. The translation can affect the layout of the GUI too. The size of the translated text can fluctuate from language to language. Some languages such as German tend to have longer words causing the need to adapt the layout of the GUI. The reverse can also be true: Chinese for example tends to be very compact, leaving a lot of unused space in the UI.

What are the ideas that led up to trying out a multilingual VUI in microscopy?

Voice User Interface was seen by TFS as a technology that could be automatically translated and localized to other languages. The idea was to try using general-purpose machine translation services such as Google Translate to turn an existing English-language VUI into a multilingual VUI. The feasibility and accuracy of this solution had to be tested. If this approach would work well enough, it would save TFS time and money in translating the user interfaces.

Additionally, Voice User Interface was seen as a technology to avoid the challenges that building a GUI brings. When building a VUI, there is no need to worry about the length of the text impacting the graphical layout of the UI as is the case with GUI. If there would exist an easy way to build multilingual VUIs that perform as good as current GUIS, then TFS could avoid spending time and money on translating GUIs for each target market.

4.1.2 Domain of VUIs

This section gives an introduction to voice user interfaces. First, it covers some domain terms and then describes technologies that are used to build a VUI.

Voice User Interface

Voice User Interface (VUI) is a user interface that can be used to control a system with voice. Figure 5 illustrates how VUI listens to the user's speech, predicts its intent, and responds to the user also with speech. The predicted intent can be used to trigger actions in the system.

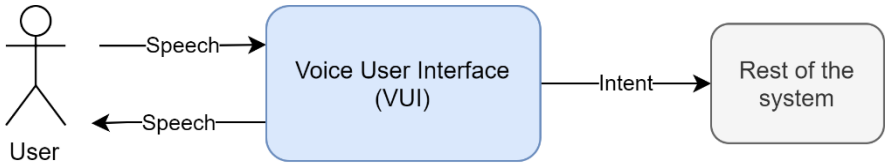

Figure 5: System context diagram for Voice User Interface

Figure 6 illustrates how three main technologies can be combined to build a Voice User Interface:

1. Automatic Speech Recognition (ASR) – for converting speech from audio into text
2. Natural Language Understanding (NLU) – for understanding the intent of the text and generating a textual response
3. Speech Synthesis – for converting the textual response to speech

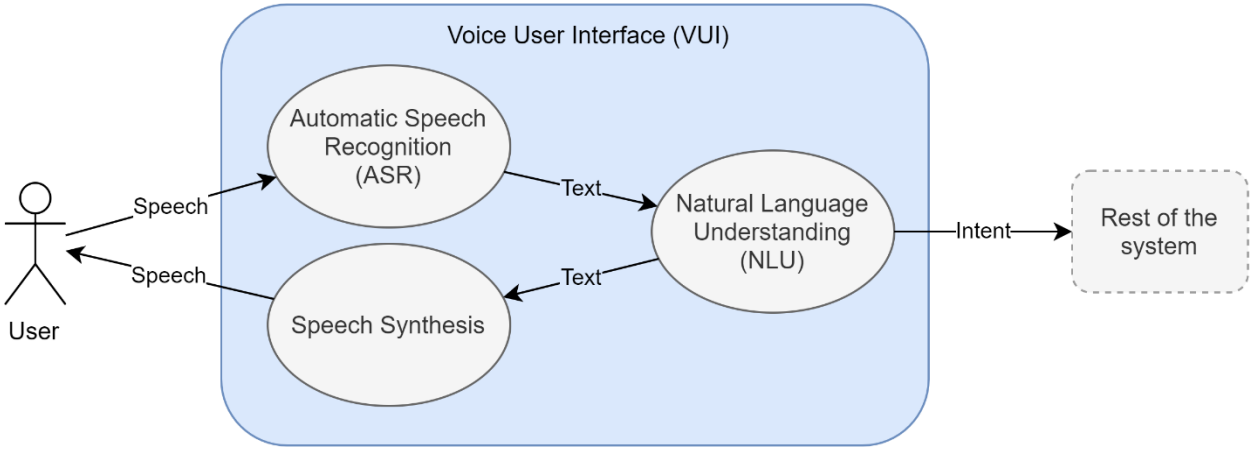

Figure 6: Technologies used in VUI

Virtual Assistant

A Virtual Assistant (also known as a chatbot) is a software agent that can perform tasks or provide services for an individual based on commands and questions. A virtual agent usually emulates a conversation with a real human. The conversation can be in the form of text or voice. If the voice is used to hold a conversation, the assistant is considered to have a Voice User Interface.

SWOT analysis for VUI

Table 1: SWOT analysis for VUI

	Beneficial	**Harmful**
Internal	Strengths • Enables hands-free and eyes-free interaction • No searching for functionality	Weaknesses • Low discoverability of functionality • The information must be consumed at the speed of Voice UI. • Need for VUI specialists for good UX design.
External	Opportunities • Improvement in speech recognition, natural language understanding, and speech synthesis technologies widen the range of actions that can be done. • Users are already familiar with the technology: Other Voice Assistants such as Apple's Siri and Google Assistant have already popularized the interaction method.	Threats • Users reject VUI due to non-intuitiveness or unreliability and fall back to using GUIs

4.1.3 Technical Challenge

Voice User Interface for IM1 and IM2 was built with Amazon Lex[1] – a fully managed service for building chatbots. This service combines Automatic Speech Recognition (ASR), Natural Language Understanding (NLU), and Speech Synthesis. [7] Unfortunately, Lex supports only one language – English. This severely limits the audience the system can serve. The main objective of this experiment is to make the VUI of the IM platform multilingual.

From interviews with the stakeholders, I extracted three main high-level business requirements for the multilingual voice user interface:

- Voice assistant must support at least four languages, including English, Dutch, and Chinese
- Voice assistant must be easy to integrate into new and existing TFS applications
- Voice assistant's voice should sound the same across devices, platforms, and applications

4.1.4 Choice of Approach

There are two main ways I could make the existing chatbot multilingual. The simplest way would be to try another VUI platform that already supports multiple languages. The more generic way would be to add a translation layer between the user and Lex.

Potential Approach 1: Using a VUI platform with support for multiple languages.

Some VUI platforms such as Dialogflow and Wit.ai, support multiple languages. They achieve this support by allowing the user to define multiple instances of the bot, one for each language. The bots usually share the underlying intents, but the utterances that they react to and responses they send must be translated manually. This means that the maintainer of the bot must create and maintain the translation of each bot separately.

While TFS acknowledged this approach as a viable solution to making the bot multilingual, it was not preferable, as the burden of translation would be considerably high. TFS would like to avoid the need to translate each bot separately to every language. Having to translate bots would also slow down the development process, as every change in the bot would have to be translated again.

[1] Amazon Lex: https://aws.amazon.com/lex/

As of writing this, there are no VUI platforms available where a developer could define a bot's behavior in one language and then expect it to automatically work in other languages too. The absence of this kind of platform leads us to a second potential approach, where the translation of the chatbot would be automated.

Potential Approach 2: Automated Translation

Another way to make IM VUI multilingual is to place a translation service between the user and Lex. Automated translation services such as Google Translate could be used to automatically translate all users' utterances to English so they could be understood by the NLU platform. Similarly, all the responses coming from the Lex can be translated back to the native language of the user.

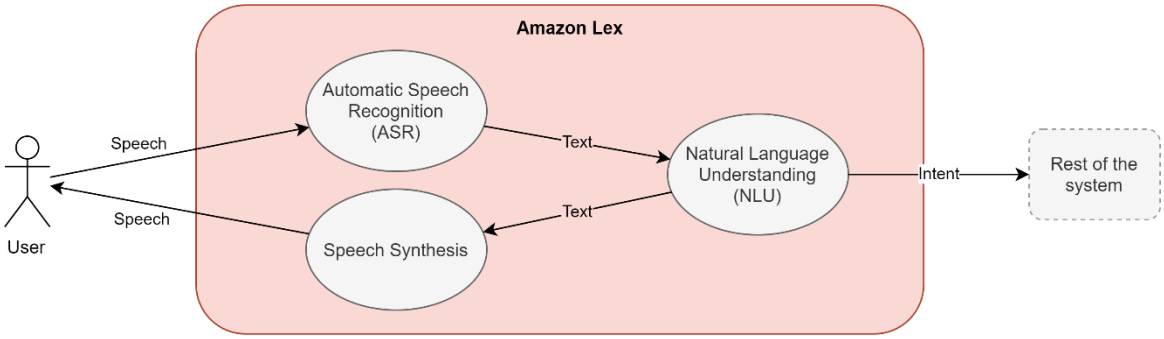

Figure 7: Amazon Lex used with speech audio. Amazon Lex combines ASR, NLU, and speech synthesis to provide the service.

This approach radically changes the way IM interacts with Lex. In IM1 and IM2, Lex was used as a full VUI platform. It provided ASR, NLU, and Speech synthesis functionality. Speech audio was sent to Lex and Lex responded with synthesized speech audio, as illustrated in Figure 7. Now, because translation APIs such as Google Translate and Amazon Translate operate only on text, the interaction with Lex must also happen over text. This means that we cannot leverage the ASR and speech synthesis functionality of the Lex. Instead, separate services for ASR and Speech synthesis must be used for this purpose. This approach has been illustrated in Figure 8.

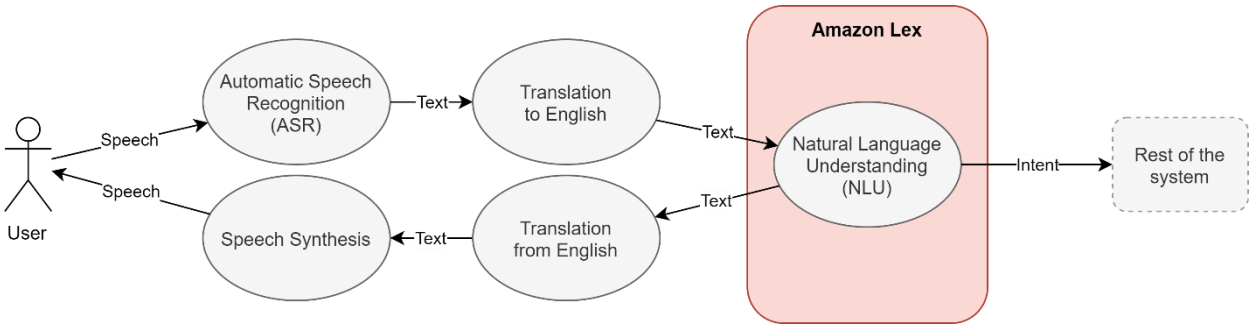

Figure 8: Automated translation by using external translation, ASR, and speech synthesis services in front of Lex.

In this approach, the Natural Language Understanding is always carried out in written English, regardless of the language spoken by the user. This means that we can reuse a single English-speaking virtual assistant for serving an audience with a wide variety of spoken languages. It reduces the maintenance burden from the maintainer of the UI, as there is no need to maintain a virtual assistant instance for every language.

Choice of the approach

Out of the two approaches, I chose the second one, the approach with an automated translation service. It was interesting to TFS because of its potential to eliminate the burden of translating the bot to multiple languages. At the same time, we had to accept that the automated translation will

introduce errors in the process, reducing the bot's ability to understand the user and the quality of its responses. The choice was made to investigate how far we can push this approach and what measures we can take to keep the quality of the bot on an acceptable level.

This choice also reaps the benefits of continuously improved translation services. The expectation is that as the automated translation services get better and better over time, the quality of the bot comes closer to the manually translated one.

4.1.5 External API Selection

Selecting APIs to use was an important step in the design of multilingual VUI. This section gives an overview of services and libraries available for speech recognition, translation, natural language understanding, and speech synthesis.

Choice of Speech-Recognition API

The available speech-recognition APIs can be broadly divided into three categories: APIs offered over the internet as a managed web service, APIs offered by the application platforms, and APIs offered by offline libraries. The differences between these categories of APIs have been summarized in Table 2 and discussed further in the next few sections.

Table 2: Comparison of speech-recognition API types

API offered by	**Managed web service**	**Platform/OS**	**Offline library**
Development effort	Low	Very Low	High
Portability	High	Low	Medium
Internet needed	Yes	Sometimes	No
Customizability	Medium	Low	High

APIs offered as a managed web service

APIs offered as a web service are accessible over the internet through a web API such as REST or gRPC. The main benefit of using a managed web service is the low development effort needed to get started. The complexity of speech recognition has been hidden behind a simple API call. Another benefit is that they can be used to build very portable applications. The same service can be used by applications built on any platform or programming language. On the other hand, the downside is that an internet connection is always needed to use the service. Some managed web services allow customizing language models to improve the accuracy in specific domains. Examples of speech recognition APIs offered as web services are:

- Cognitive Speech Services[2] by Microsoft
- Speech-to-Text[3] by Google
- Amazon Transcribe[4]
- Watson Speech to Text[5] by IBM

APIs offered by the application-platforms

Some application platforms and operating systems also offer speech recognition capabilities:

- Web Speech API in browsers
- Speech recognition in Windows
- Speech recognition in Android
- Speech Framework in iOS and macOS[6]

The main benefit of using an API provided by the platform is the ease of integration. On the other hand, it has low portability; If the application is used on different platforms, integration with API

[2] Azure Cognitive Speech Services: https://azure.microsoft.com/en-us/services/cognitive-services/speech-services/
[3] Google Cloud Speech-to-Text: https://cloud.google.com/speech-to-text
[4] Amazon Transcribe: https://aws.amazon.com/transcribe/
[5] Watson Speech to Text: https://cloud.ibm.com/catalog/services/speech-to-text
[6] Speech Framework in iOS https://developer.apple.com/documentation/speech

must be done for each platform separately. Furthermore, the behavior of the application is now dependent on the characteristics of each API, making it act differently across platforms.

Some of those APIs use local, offline algorithms to transcribe on the device of the end user, but more commonly the actual speech is sent to remote servers for transcription. For example, the Web Speech API in Chrome uses Google's speech-to-text web service to transcribe audio. On Apple devices, recognition of some languages is done on the device, but for others, a network connection is needed to send the audio to Apple servers for processing.

APIs by offline libraries

Lastly, the speech recognition can be done completely offline and independently, by using a suitable library such as CMUSphinx[7]. Nevertheless, it would mean a higher development effort both in getting started and in the later development of the application. Most notably it would mean that the language model management must be handled by the application developers. The language models must be installed on the client side or loaded separately. Nevertheless, this allows using completely custom language models, potentially improving the accuracy in specific use cases.

Selection Criteria

The criteria for selecting a speech recognition service are:

- Development effort
  - How easy is it to add to an existing or new project?
  - Does it abstract away most of the complexity of speech recognition?
- Portability – the possibility of using the same service across different TFS applications
  - Can it be used by a Python web service? A JavaScript web application? A C# Desktop application?
- Language support
  - How many different languages does it support?
  - Does it support English, Dutch, and Chinese?
- Accuracy of the recognition results
  - Can it recognize less common words and phrases such as "mitochondria" or "field of view"?
- Cost of usage
  - How much does it cost per request or minute of speech?
- Speed of recognition
  - Does it support real-time recognition?
  - How long does it take to transcribe a complete sentence?

Web Speech API

Web Speech API is an experimental API offered by browsers. Currently, this API is only available in WebKit based browsers, such as Google Chrome[8], but Firefox is also considering adding support[9]. Chrome uses Google's servers to perform speech recognition [8]. Audio recording is sent to Google servers, and the result of speech detection is returned to the web application as a text.

In this project, I chose to use Web Speech API, mainly for its low development effort and wide language support. It supported all the languages that TFS was most interested in – English, Dutch, and Chinese, and many more. The simple API allowed me to quickly prototype the solution. The positive and negative sides of using this API are summarized in Table 3.

[7] CMUSphinx: https://cmusphinx.github.io/
[8] Availability of Web Speech API: https://caniuse.com/#search=SpeechRecognition
[9] Web Speech API in Firefox: https://platform-status.mozilla.org/#webspeech-recognition

Table 3: Considerations of using Web Speech API

Positive	Negative
• Low development effort – It has a simple API and is easy to get started for programmers • Free – it uses a speech recognition system available on the end user's device. • Automatic speech start and end (silence) detection. • It can be made to listen continuously by restarting it when it stops, taking away the need to click a button every time user wants to say a voice command. • Wide language support	• Currently supported only by Chrome • Experimental API – no assurance that the API would not change or be removed in the future. • It is unknown how platforms might limit this free service when used a lot by one device or application.

Choice of Speech synthesis API

The main criteria for choosing a speech synthesis API is the quality of the voice and the number of languages supported.

In our application, it is preferred to keep the sound of the virtual assistant's voice the same regardless of the device or browser the user is using. We cannot rely on platform-specific APIs, such as Web Speech API in browsers or Speech Synthesis framework in iOS, as each one of them has a different sound and feel. Therefore, platform-specific APIs were not considered. The number of languages provided by each service can be seen in Table 4.

Table 4: Comparison of speech synthesis APIs

	Amazon Polly	Google Text-to-Speech API	Microsoft
Number of supported languages	29 standard TTS 4 neural TTS [9]	40 standard TTS 31 neural TTS [10]	45 standard TTS 31 neural TTS [11]

Each service offers standard and neural voices. Neural voices are powered by deep neural networks. Neural-network-based speech synthesis produces the most natural and human-like voice but has a higher price than a standard service. Using neural voices makes the interaction with chatbot and voice assistants more engaging and natural.

In IM3 I continued using Polly because it was already used in previous iterations of IM and it was good enough for this project. It supported all the languages I wanted to synthesize. Nevertheless, to increase the number of supported languages, it is easy to switch to Google's or Microsoft API, by replacing the speech-recognition API adapter in the solution.

Choice of Translation API

The main criteria for choosing translation API was the number of different languages offered by the service.

I decided to only consider managed services offered by Amazon, Google, and Microsoft and not use local translation to benefit from the progress and improvements made by the service providers over time. As can be seen in Table 5, Google has the largest selection of languages and was therefore chosen.

Table 5: Comparison of managed translation APIs

	Amazon Translate	Google Translate	Microsoft
Number of supported languages	55 [12]	109 [13]	77 [14]

It is interesting to note that even though Google Translate offers 109 languages, we cannot benefit from this large number as we cannot recognize nor synthesize speech in that many languages.

4.1.6 Architecture

This section presents the architecture of the multilingual VUI.

Conceptual view

The conceptual view describes how the system works and what it does on a high-level without describing the realization details.

The conceptual flow of the solution is shown in Figure 9. The flow starts when the user utters a command. The speech is captured as a speech audio. The audio is transcribed to a text. This text can then be translated into English. Now, we can use Lex or some other NLU service to understand the meaning of the translated text. After extracting the intent, NLU service Lex usually responds to the user with some predefined phrase. This response is translated back to the native language of the user. Next, the response speech is synthesized and played back to the user.

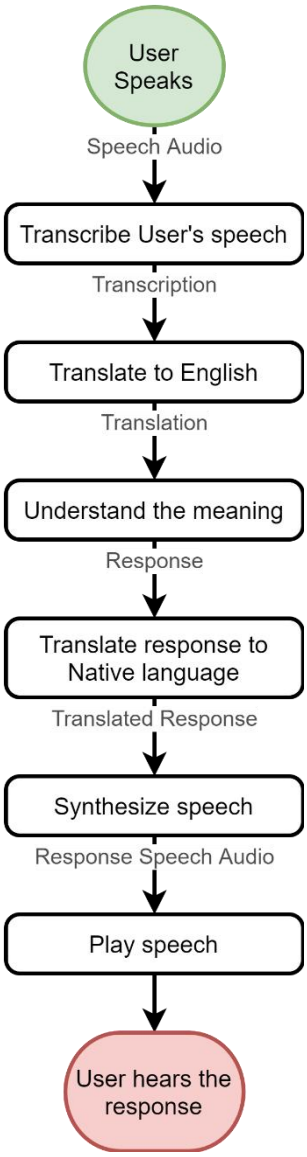

Figure 9: Activities performed during one request-response cycle

Design Alternatives

There are multiple ways to map functionality, such as speech recognition, translation, and synthesis, to architectural components. In this section I discuss three architectural approaches and their trade-offs:

- All in the client
- All in the back end
- Audio processing in the client, text in the back end

The main driving forces of architecture are reusability and performance. The system must be easy to reuse in another project, to add a VUI for another TFS application. From the performance side, the application has to be quick in responding to the user's requests.

There are two extremes: Either control everything directly in the browser or hide everything behind a backend service.

All in client

On way to tie all the services together is to do it in the client application. This is the simplest approach, as it completely avoids building a backend service.

Example technology choices for each functionality:

- Speech Recognition – Web Speech API[10]
- Translation – Google Translate API[11]
- NLU – Amazon Lex[12]
- Speech Synthesis – Amazon Polly[13]

[10] Web Speech API: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API
[11] Google Translate API: https://cloud.google.com/translate/docs
[12] Amazon Lex: https://aws.amazon.com/lex/
[13] Amazon Polly: https://aws.amazon.com/polly/

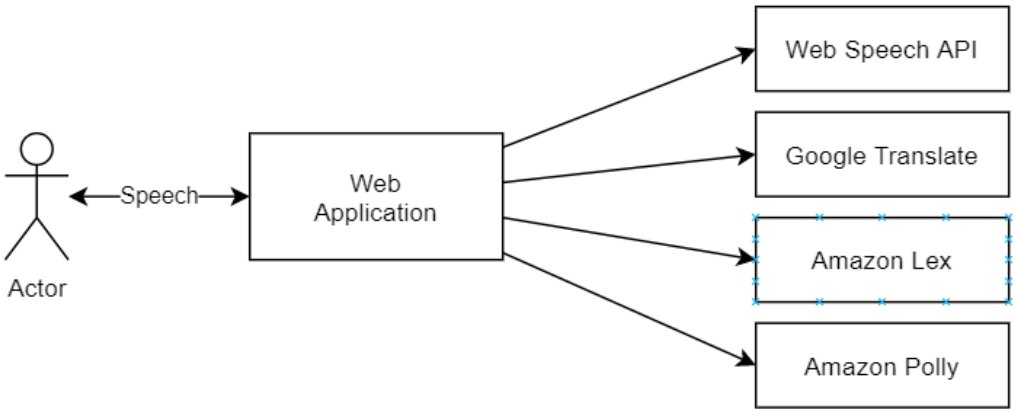

Figure 10: Architectural option: All functionality in the Web Application

Nevertheless, it comes with some consequences: There can be no reuse between UI platforms. When adding a new user interface on another platform (VR, Android, iOS), all the functionality would have to be recreated. Therefore, this design alternative does not satisfy the reusability requirement.

All in backend

To satisfy the reusability requirement, we can move as much logic as we can into a separate web-accessible backend service as illustrated in Figure 11. In this architecture, the core of the application is in the Backend Service. The client applications only capture the user's speech and playback the response. This allows reusing all the core logic with many different client applications, except voice capture and playback. This solution also allows connecting other types of clients such as a VR client or mobile client easily, as we only must integrate each client with Backend Service, not four different external services.

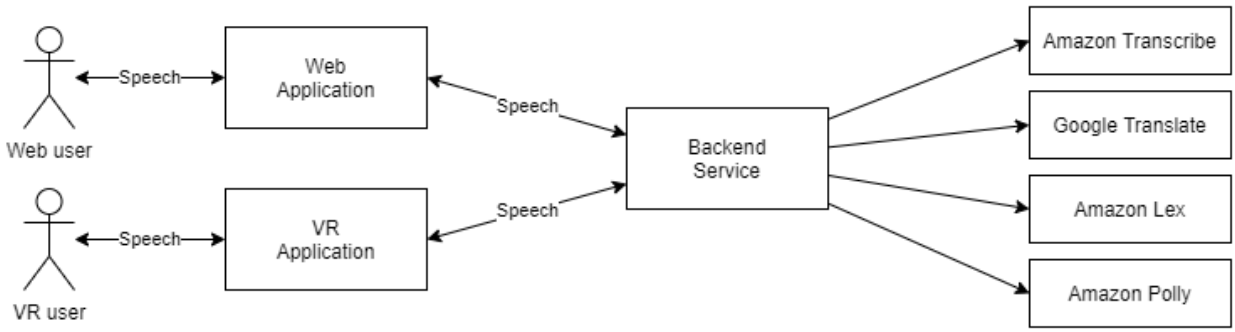

Figure 11: Architectural option: Core functionality in the backend

Nevertheless, it does not make sense to continuously stream audio from all applications to a backend service, as it would be a huge waste of network bandwidth and strain for backend service to process.

Audio in the client, text in the backend

To cut down the strain on the bandwidth, we can alter the architecture and make the speech bypass backend service entirely as illustrated in Figure 12.

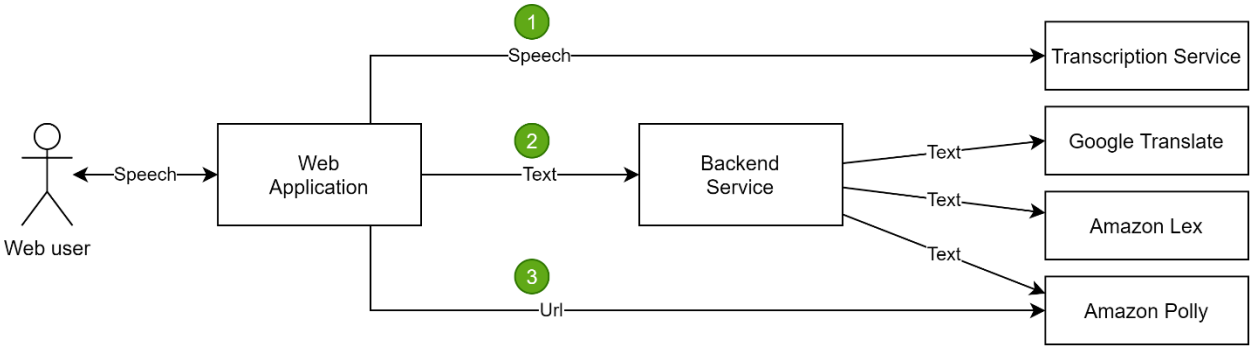

Figure 12: Architectural option: text in the backend.

When the user utters a sentence, (1) it is transcribed by the web application either by relying on a Web Speech API or some web service such as Amazon Transcribe. (2) It sends the transcription to a backend service. The backend service translates it, predicts its intent, formulates a response, translates the response, and sends the translated response to the synthesis service. The synthesis service provides the backend service with an URL where an audio file can be found. The backend returns the URL to the web application. (3) The client application downloads the audio and plays it to the user.

This solution allows medium code reuse across platforms (everything that is in the backend service) while avoiding a high network load on the backend service.

4.1.7 Design

This section describes the design of the translated bot service. It is the service, ties together three external services to provide a multilingual chatbot service. It is used by the IM webpage to provide multilingual VUI. The context diagram of the service is shown in Figure 13.

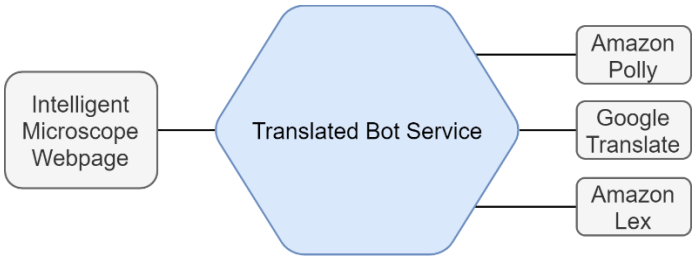

Figure 13: Context diagram of Translated Bot Service

Onion architecture and Ports and adapters

The design of the translated bot service has been heavily influenced by two related architectural patterns: “Ports and Adapters” [15] (previously also known as “Hexagonal Architecture”) by Alistair Cockburn. and “Onion Architecture” [16] by Jeffery Palermo.

Onion architecture is an architectural pattern that uses the concept of layers where each layer can only use layers below, but the use is different from the classic layered architecture. In the classic layered architecture, the data access lowest layer is the data access layer. The lower layers depend on infrastructure - databases and external APIs. In onion architecture, the central layer is the domain layer that holds the domain objects and entities. These entities do not have any external dependencies. Furthermore, the infrastructure dependencies have been moved to outermost layers, making the core of the application independent of specific infrastructure.

Ports and Adapters architectural pattern specifies that the application’s communication to external infrastructure should happen through **ports** and **adapters**. Each port represents a connection from the application to the outside world. A port represents the purpose of interacting with the external

world without a knowledge of a concrete implementation. Usually, in the code, a port is the interface declaration while the adapter is the technology-specific implementation of that interface.

Translated Bot Service has 4 ports that are shown in Figure 14 in yellow. API port is the driving port that defines the interface of the Translated Bot service. It is used by the HTTP API adapter, to provide this functionality over HTTP. The other three ports are driven by the application and used to query external services. Each port has a respective adapter that defines how exactly the external service is queried.

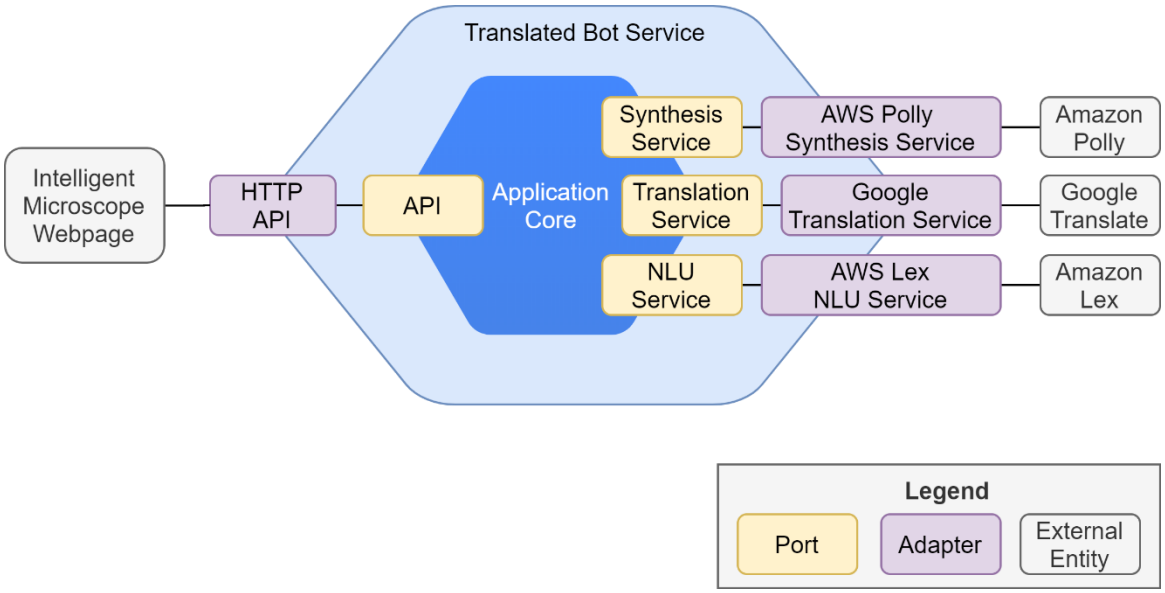

Figure 14: Ports and Adapters in Translated Bot Service

Both architectural patterns make a clear distinction between the application core and the infrastructure. Infrastructure is the code that connects the application core to databases, third-party APIs, and user interfaces. The application core contains the business logic and has no knowledge of the external entities and infrastructure. Both architectural patterns focus on externalizing the infrastructure to ensure that the application core is not dependent on the infrastructure.

I followed these patterns to create a technology-independent application core and easily exchangeable infrastructure. It means that the external infrastructure such as databases, messaging queues, user interfaces, and externals APIs can be switched without the need to modify the application core. The resulting architecture allows the developers to easily switch technologies while retaining the business logic, making it easy to quickly test out different technologies. Also, it improves the testability of the service, by simplifies testing of the application core in isolation, without making queries to external infrastructure.

4.1.8 Deployment

The translated bot is a serverless application deployed to Amazon Web Services (AWS) using Serverless Framework[14] – an open-source web framework for building serverless applications. For software developers, Serverless Framework increases developer velocity by simplifying the process of deploying serverless applications. The framework is provider agnostic, meaning that switching the cloud service provider to Microsoft Azure or Google Cloud Platform provider is easy. Compared to using CloudFormation directly, this avoids vendor lock-in. The following design decision summarizes the justification for and against the choice.

[14] Serverless Framework: https://www.serverless.com/

Design decision:
In the context of creating a multilingual voice user interface,
facing the need to deploy the back end to AWS
we decided to use the Serverless Framework
and not Elastic Compute Cloud(EC2), CloudFormation, or Serverless Application Model (SAM),
to achieve: **increased developer velocity, ease of deployment**, no server administration, automatic scalability, vendor independence
accepting that long-running processes cannot be run on serverless functions

Using serverless architecture eliminates the need to manage the infrastructure such as servers and virtual machines. Instead, the code is uploaded as a serverless function. The provisioning of the runtime environment for that function is fully managed by the cloud service provider. This means that as a developer I can focus on designing applications and not have to worry about provisioning infrastructure. The cloud service provider scales the service automatically with demand, charging only for the number of executions and their duration, making it also a very cost-effective solution. For example, if there are no requests in a certain time-frame, the cost of the service is 0. This automatic scalability is especially useful in cases of unpredictable workloads.

Serverless functions have some limitations concerning execution time and resource usage. Most notably, a single execution of the function cannot run for very long and has a limited amount of RAM. Each service provider has its own quotas, but in AWS, the Lambda cannot run longer than 15 minutes or use more than 3GB of RAM. [17] For our use case, all these limitations are completely acceptable.

Serverless Framework deployment on AWS
Serverless Framework uses AWS CloudFormation[15] to create and configure resources on the AWS cloud. When building a serverless application, it creates an API Gateway for accepting HTTP requests, and Lambda functions for executing code. Additionally, it creates an S3 bucket for storing deployment template and IAM roles for lambdas. Figure 15 illustrates how Serverless Framework uses CloudFormation to deploy an application stack on AWS.

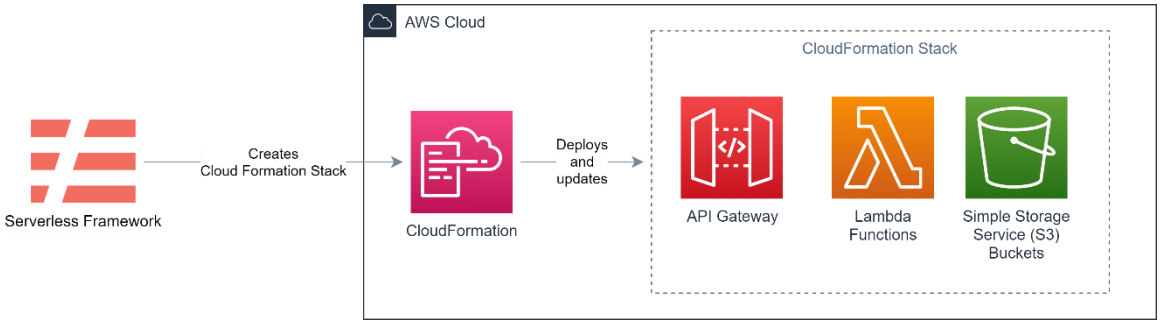

Figure 15: Serverless framework uses CloudFormation to deploy a serverless application

Figure 16 illustrates the resources deployed to the AWS cloud and their interactions with other components of the system. The API Gateway provides an HTTP interface with two endpoints: POST /askBot and POST /botSpeak – each responsible for invoking the respective lambda functions. The lambda functions contain the code and business logic and interact with external API's to fulfill their goals.

[15] AWS CloudFormation: https://aws.amazon.com/cloudformation/

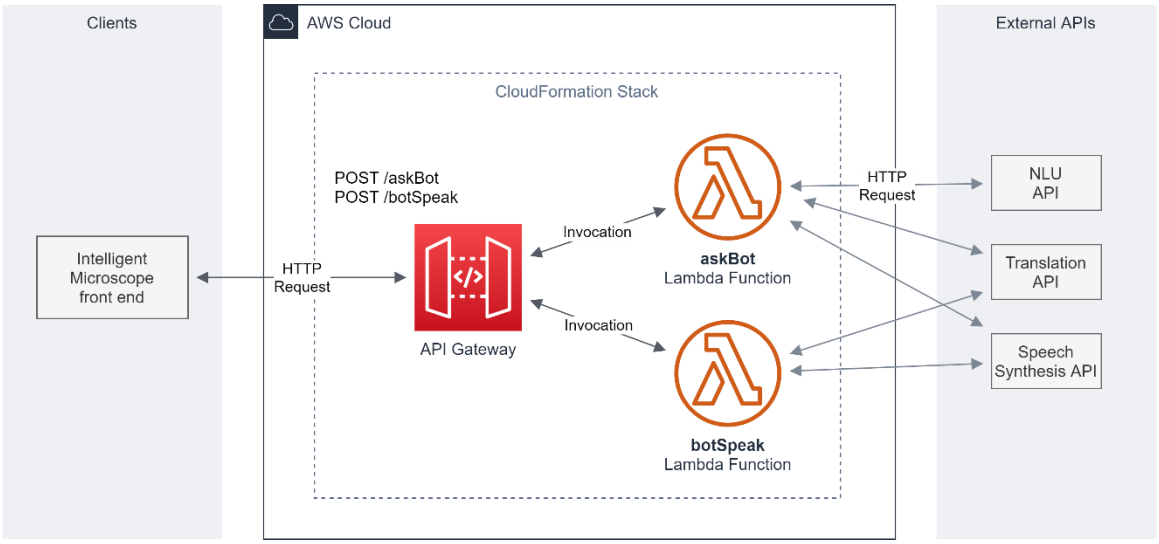

Figure 16: VUI resources deployed in AWS

API Gateway acts as a "front door" for the application and is usually responsible for access control, throttling and monitoring. For example, the endpoints in this experiment are throttled to three requests per second to avoid excessive costs in case the client code misbehaves and starts generating a lot of requests. No access control is currently performed on the endpoints.

4.1.9 Implementation

The multilingual bot was written in TypeScript[16] – a language that builds upon JavaScript by adding syntax for static typing. This allows the code to be type-checked before execution for errors, improving the reliability of the application. Furthermore, static typing improves developer experience by allowing the IDE to provide better code completion and refactoring functionality.

I picked TypeScript because I built the first iterations of this experiment to be run in the browser. In this environment, the only natively available language is JavaScript, but languages that can be transpiled[17] to JavaScript can also be used. To get the benefits of static typing, I chose to use Typescript and transpile it to JavaScript. Later, the same code was deployed as a serverless function and developed further, demonstrating the high portability of this language.

Code Quality Aspects

To improve code reliability and maintainability, I used ESLint[18] to lint the code and Prettier[19] to automatically format it. ESLint is a linting utility that can detect defects, problematic patterns, and possible bugs in the code. It warns the developer of problems during development and suggests improvements. Prettier is an opinionated code formatter that can be used to automatically format code. This ensures that the code formatting is consistent across the whole codebase. Consistency of the code has a positive impact on the readability and maintainability of the code. Together those

[16] TypeScript: https://www.typescriptlang.org/
[17] Transpiling is the process of transforming one programming language to another, similar language.
[18] ESLint: https://eslint.org/
[19] Prettier: https://prettier.io/

tools make sure that the code adheres to a consistent set of formatting and coding standards, making the code easier to read and modify.

Source code organization

Source code is organized according to the layers in onion architecture and by ports in "ports and adapters" architecture. The content of the *src* folder is shown in Figure 17.

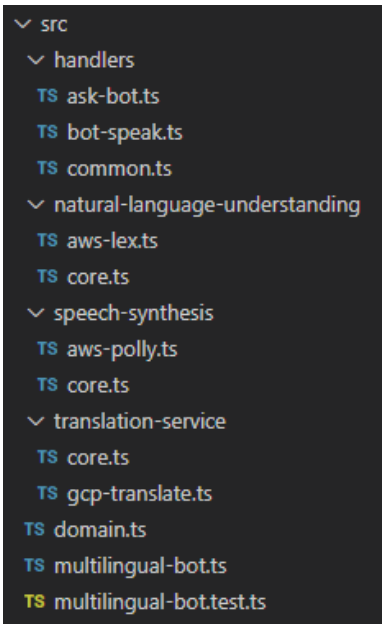

Figure 17: Source code organization in the src folder

Handlers define the HTTP API of this service. They act as adapters between the HTTP API and the technology-independent API offered by the application core.

Multilingual-bot.ts represents the core of the application. It defines the conceptual API and contains the main business logic of the application.

Natural-language-understanding, speech-synthesis, and translation-service folders contain the ports and the adapters to respective services. I declared the technology-independent interface in *core.ts* files and each service-specific adapter in a separate file.

Domain.ts contains all domain objects that are used across the application such as enumerations of written and spoken languages.

HTTP API

The backend service exposes two endpoints: askBot and botSpeak. The first one is for interacting with the multilingual bot and the second one is for synthesizing speech.

POST /askBot

The main interaction with the multilingual bot happens through the POST /askBot endpoint. This is the endpoint for speaking with the bot. The endpoint expects the JSON payload to contain the transcription of the user's utterance and a language of that utterance:

```
{
    "language": "et",
    "utterance": "Liigu vasakule"
}
```

The supported languages and language tags are defined in an enum and are as follows:

```
enum WrittenLanguage {
  Dutch = "nl",
  English = "en",
  Estonian = "et",
  French = "fr",
  German = "de",
  Mandarin_Chinese = "cmn",
  Turkish = "tr",
  Russian = "ru",
}
```

The language tags conform to IETF BCP 47– best current practice published by Internet Engineering Task Force. [18]

The language parameter is optional. When the payload contains only the utterance as in {"utterance": "ga naar links"}, the service tries to predict the source language of the utterance by relying on the external translation service.

An example JSON response structure from /askBot endpoint looks as follows:

```
{
  "message": "OK! Ik zal het podium naar links verplaatsen!",
  "intentName": "MicroscopeMoveDirection",
  "dialogState": "Fulfilled",
  "slots": { "moveDirection": "left" },
  "synthesisUrl": "https://polly.eu-west-1.amazonaws.com/v1/speech[...]"
}
```

It contains the response in text format on the “message” field that can be shown to the user and the “synthesisUrl” points to the mp3 file of the synthesized speech that can be played back to the user. “intentName”, “dialogState”, and “slots” are Amazon Lex specific fields. Notable ones are the name of the intent and slots that were detected from the user’s utterance. These are used to trigger respective actions in the microscope backend.

POST /botSpeak

botSpeak endpoint can be used to synthesize translated speech in any supported language. The endpoint expects a message to be synthesized in English and the target language. For example:

```
{
    "message": "I found 10 mitochondria",
    "language": "nl-NL"
}
```

The list of supported synthesis languages are as follows:

```
SpokenLanguage {
  Dutch = "nl-NL",
  English_British = "en-GB",
  English_US = "en-US",
  Estonian = "et-EE",
  French = "fr-FR",
  French_Canadian = "fr-CA",
  German = "de-DE",
  Mandarin_Chinese = "cmn-CN",
  Turkish = "tr-TR",
  Russian = "ru-RU",
}
```

As with written-language tags, the spoken-language tags also conform to IEFT BCP 47. Spoken-language tags have an additional subtag to differentiate between language variants spoken in different regions. For example, the multilingual bot supports British English and US English. It also supports French and Canadian French.

After the translation and synthesis are done, the endpoint responds with the translated message and an URL of the synthesized speech that can be played back to the user. The JSON response from /botSpeak looks as follows:

```
{
```

```
    "message": "Ik heb 10 mitochondriën gevonden",
    "url": "https://polly.eu-west-1.amazonaws.com/v1/speech?OutputF..."
}.
```

Incompatible tags between services

Each external service used by the multilingual bot has its own system of tagging languages. This makes integrating them cumbersome, as the tags have to be converted between services. when using the multilingual bot in Chinese (language tag: `"cmn-CN"`), the google translate service expects a language tag `"zh-CN"`, and Amazon Polly expects the name of the voice `"Zhiyu"`. To have a clear separation between the used tags, the application core uses IEFT BCP-47 tags for each supported language. Each service adapter converts this tag to their service-specific tag.

What makes it even more cumbersome, is that there is no one-to-one mapping between languages between the services. Services for speech synthesis and speech recognition usually provide multiple versions of English based on dialect: British English, American English, Australian English, and others. On the other hand, text translation services have only one version of English. To reflect this, there are two sets of languages used in the application: written languages and spoken languages. The application contains an explicit conversion table from spoken languages to respective written languages.

4.1.10 Validation

Testing speech-recognition services in English

In IM1 and IM2, the transcription of the speech happened in the AWS Lex service. The transcription-accuracy in this service is commendable, as it almost always transcribes the commands exactly as uttered. Most probably, Lex service trains a new speech transcription model whenever the list of expected utterances is changed. This allows the service to be very accurate in transcribing this limited and defined set of utterances.

In the pursuit of making the IM3 multilingual, I tried using general-purpose transcription services instead. This would allow us to transcribe other languages than English. The first pick for the system was Google Transcribe because of its wide range of supported languages, but we also tested Amazon Transcribe and IBM Watson.

To assess the accuracy of each service, we subjected them to the same utterances in English that the original Lex bot would need to transcribe. We repeated each utterance at least three times and up to ten times and recorded the percentage of correctly transcribed utterances. We chose 75% transcription rate as "reasonable"; if a transcription is correct three out of four times, it is good enough.

Testing results

As can be seen in Table 6, out of the 45 chosen utterances, Lex bot transcribed 42 with a transcription rate higher than 75%. Most of the utterances were correctly understood every single time. Only three posed some problems for Lex. For example, "Switch to microscope six" was frequently transcribed as "select microscope zero." It is unknown why number six causes problems, but four, five, and other numbers are transcribed correctly almost 100% of the time.

Table 6: Comparison of accuracy of speech-recognition services

Given 45 utterances	Lex (Original)	Google Transcribe	Amazon Transcribe	IBM Watson
Number of utterances with correct transcription rate >75%	42	20	9	4

General-purpose speech-recognition services did not fare so well. Their accuracy in transcribing the same utterances was far worse. The best one, Google Transcribe, was able to transcribe only 20 out of 45 utterances with a reasonable >75% transcription rate. Amazon Transcribe and IBM Watson were even worse.

These results highlight the inaccuracy of general-purpose transcription models in transcribing the utterances needed for the Intelligent Microscope. Nevertheless, based on those results, we continued using Google Transcribe for building the multilingual VUI to see how well it would work in a complete setup, as Lex is sometimes able to understand the intent even when the speech is slightly mistranscribed.

Testing VUI in four languages

To assess the usability and accuracy of the multilingual voice user interface as a whole, we tested it in four languages. There were four participants, each proficient in a different language: English, Turkish, Chinese, and Dutch.

The participants were asked to interact with the application and record the resulting transcriptions and intentions that the system recognized. Each participant was given a list of intents that can be triggered (taking an image, zooming in) and example phrases in English ("capture an image," "zoom in," "set field of view to 20 micrometers"). The participants were asked to utter those phrases in their language, observe the results, and record them to the spreadsheet.

Process

Each participant received an e-mail with a link to the application to test and a spreadsheet to fill out. Participants were asked to try each utterance at least three times. On utterances that sometimes failed, we asked the participant to repeat it six to ten times to get a better feeling for the accuracy of that particular utterance.

Testing Results

The initial plan was to test a wide range of VUI functionality, but manual data collection using a spreadsheet proved to be too cumbersome for the participants to support it. Therefore, most participants stopped after testing the first three main functionalities: making an acquisition, changing the zoom level, and moving the stage.

The results of the test are summarized in Table 7. The accuracy of the system in detecting a particular event was rated in three levels: "High" when an utterance would trigger the correct intent more than 90% of the time, "Medium" if more than 75% of the time, and "Low" if less than 75%.

Table 7: Accuracy of the system in detecting an Intent

Intent \ language	English	Dutch	Turkish	Chinese
Taking an image	High	High	High	Medium
Zooming	High	Medium	Medium	Low
Moving the stage	Medium	Medium	Medium	High

The worst results came for the Chinese language. Based on the responses, there is a suspicion that noise cancellation was interfering with the speech recognition (See section "Sensitivity to noise cancellation software"), as there are many errors in transcribing the first word of the utterance.

Taking an image works very well in all languages except Chinese where the first part of the utterance is sometimes mistranscribed, failing the whole intent detection flow.

Zooming works well in English, but in other languages not that much. In Dutch and Turkish, "zoom in" works well, but "zoom out" is problematic. While the problematic intent is the same, the reasons for

failure are different. For Dutch, the problem lies in transcription: the *zoom uit* can be transcribed as *zo meid.* For Turkish, the problem lies in translation: "*uzaklaştır*" is transcribed correctly, but translated to English as “too.” In Chinese, the participant zoomed in using the utterance "放大". The translation service translates this into "Amplification,” which the Lex does not understand. This could be solved either by using a custom translation model or adding "Amplification" to the list of utterances Lex should recognize.

Setting the field of view directly in Dutch with the utterance *"zet gezichtsveld naar 20 micrometer"* works perfectly with many different numbers. It seems that longer utterances help in transcribing the neighboring words with higher accuracy. The speech-recognition service struggles the most with short utterances such as "move up" or "zoom out."

Surprisingly, moving the stage works best in Chinese, even better than in English, but that comes with a caveat. In Chinese "move left" can be formulated into a two-word or four-word expression and the system works best with the longer form. In other languages, moving the stage in some directions is easier than others. For example, moving stage up in Turkish works perfectly, but trying to move it right succeeds only 20% of the time. A similar issue comes up in Dutch, where moving the stage works well in directions left and up, but poorly in others (right and down). This shows how in each language, the general-purpose speech-recognition service struggles with some words but excels in detecting others.

Selecting a microscope was tested only in English and Dutch. In this use, the speech recognition worked fairly well in both languages, but some numbers did provide challenges. As an example, in English, the number four is frequently transcribed as “for,” leaving the user unable to select the microscope number four, as the command is transcribed as "Select microscope for.” In Turkish, the translation service provides a translation with an unexpected word order with numbers two and four: “microscope choose two,” “choose four microscope,” also causing the intent detection to fail. On numbers one and ten, the translation is even worse, changing the whole meaning of the request by translating it to "choose on from the microscope" and "choose it from microscope" respectively.

These results indicate a strong need for improving the accuracy of both the transcription of speech to text and accuracy of translating text to English.

Sensitivity to noise cancellation software

Many laptops come today with noise-canceling software preinstalled, to help their users have a pleasant video and voice-calling experience. These algorithms suppress background noises and bring out the voice of the user. Usually, when no speech is detected in front of the computer, the microphone signal is muted. Unfortunately, this extra processing hinders speech recognition, reducing its accuracy. The largest symptom of this issue is skipping of the first uttered word or mistranscription of the first word; "**Move** stage up" is transcribed as "stage up" and "**向左移**动" is transcribed as “**左移**动” or “**做移**动.”

In practice, this means that each user must be instructed to manually turn off any noise cancellation or extra processing of the microphone signal.

Retrospective on the testing process

Filling out a spreadsheet is error-prone and tedious for the participant. Each participant filled the sheets in a bit differently, sometimes summarizing the results in the notes section: "worked 3 out of 10.” For this quick test, this method was acceptable, as we only needed to get an approximate feel for accuracy.

For the future, nevertheless, it makes sense to automate the data collection process by recording the results directly inside the application. Furthermore, this approach would allow the application to guide the participant through the testing process step by step, making it easier for the participant.

Conclusions from testing

Overall, the testing results point to the fact that the accuracy of the system is not satisfactory for comfortable use. Although some commands work well, most of the time the user has to be careful to pronounce all the words properly to invoke the correct action. Even then, mistranscriptions and mistranslations are frequent.

4.1.11 Conclusions

The goal of this experiment was to assess the feasibility of using general-purpose machine translation services such as Google Translate to turn an existing IM English-language VUI into a multilingual VUI.

Is the chosen approach viable as a solution for building a multilingual UI?

At this moment, using general-purpose speech-recognition and translation services is not a viable approach for creating a multilingual VUI for microscope operation. The main obstacle in this approach is the insufficient accuracy of general-purpose speech-recognition services in transcribing the commands.

Is VUI easier to translate than a GUI?

Indeed, the length of the text does not impact VUIs as much as it impacts the layout of a GUI. Nevertheless, VUI is not easier to translate than a GUI, because having a Voice User Interface brings its own technical and usability challenges. The largest technical challenges revolve around capturing and understanding the user's intent. It needs an excellent speech-recognition system to capture the user's speech and natural-language-understanding service to interpret it.

4.1.12 Future Work

How to go forward with multilingual VUI?

One approach would be to accept the current accuracy and wait for general-purpose transcription and translation services to improve over the years. The hope is that as time goes by, those services improve, and their accuracy becomes high enough for our use case.

Another approach would be to train custom transcription and translation models for our specific use case - models specifically trained to recognize only our commands and translate them correctly. The effort to do this would be high, as domain-specific transcription and translation models have to be trained for each language. This approach would probably mean collecting hours of audio for training the transcription models.

As a next step, I suggest evaluating the accuracy of (1) Speech to Text service from Microsoft[20] or (2) alternative ML-models offered by Google Cloud Speech-to-Text service[21]. This is because as of June 2020, Microsoft's Speech to Text service and the alternative models by Google are reported to have lower mean word error rates than the default ML model from Google. [19] The current solution relies on the standard model from Google, used by Web Speech API, but other models such as the command-and-search recognition model or enhanced video model could result in higher accuracy.

What else did I learn from this experiment?

I had a very pleasant experience with Serverless Framework and serverless architecture in general. Serverless Framework is a very simple-to-use tool for deploying serverless applications. It abstracted away a lot of deployment details and automates the deployment process.

This framework allowed me to get the benefits of serverless architecture (extreme scalability, cost efficiency) without the drawbacks of vendor lock-in and complicated setup of cloud resources. It

[20] Microsoft Azure Speech to Text: https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/
[21] Google Cloud Speech to text models: https://cloud.google.com/speech-to-text/docs/basics#select-model

allowed me to quickly iterate on the code and immediately have it deployed in the cloud. Based on this experience, I suggest trying out the Serverless Framework in new cloud projects in TFS.

4.2 VIRTUAL REALITY

This challenge started from a general question: how would the future microscope operators use the microscope through VR? In this chapter, I investigate the usefulness and feasibility of applying Virtual Reality in electron microscopy in TFS. The following sections will cover the experiments I carried out with VR to assess the feasibility and usefulness of applying the technology.

4.2.1 Introduction

Virtual Reality (VR) is a technology used to create an interactive, simulated experience where the user has the feeling of being present in a virtual world. [20] Out of all the identified novel technologies, VR seemed interesting because of its ability to immerse a person in a completely artificial environment. This environment could be designed to improve the focus, awareness, and performance of the user while interacting with the microscope. The expected benefits and possibilities of using VR in microscopy that led up to trying out VR in this project are the following:

- Immersion and presence
- Large working space
- "Wow"-factor
- Collaborative environments

Immersion, presence, full engagement, and absorption

Virtual reality can improve the performance of the user by improving her attention and focus on a given task. This improvement can be achieved through mental immersion – a state of being deeply engaged in the activity at hand. Another word to describe this is "presence." A well-designed VR experience can create a feeling of presence – a perception of being physically present in a non-physical world – immersing the user in the activity.

Large working space

VR offers the possibility of creating a large, 360-degree working area around the user. All this space can be used to show a lot of images captured with a microscope or controls to interact with it. Even the floor and sky can be filled with controls and resulting images. Compared to traditional GUIs, where the size of the screen limits the available space, in VR the limiting factor is the 3D space around the user. This 3D space can offer a much larger area for UI designers to work on.

Furthermore, using a 3D space, we can tap into the spatial memory of the user. Instead of forcing the user to switch windows or tabs in a flat GUI application, the user can look around in the 3D world to find the desired functionality.

Excitement and "wow"-factor

VR is currently mostly used in the entertainment industry. A lot of gaming arcades have popped up around the world and VR experiences are used in museums as VR has a high "wow"-factor, especially among people who have not tried it yet. Nevertheless, outside of the specific installations and specialized rooms, VR is currently not widely used, as it needs significant monetary investment.

Collaborative Environments

Virtual space can be shared with other users, creating a shared, collaborative environment. Collaborative environments enable users to interact with one another over large distances.

SWOT analysis for VR

Table 8 presents the strengths, weaknesses, opportunities, and threats being faced when applying VR in the microscopy domain.

Table 8: SWOT analysis for VR

	Beneficial	Harmful
Internal	Strengths • Might improve users' efficiency by making them feel more present and immersed (when the experience is well designed) • Ability to trigger empathetic reactions through the first-hand experience (this is a strength of VR that is not useful in microscopy)	Weaknesses • Users need specialized hardware; High cost at this moment. • Users need extra training for unconventional UI • Technological annoyances: VR can cause motion sickness; headsets are uncomfortable to use with glasses. • To create a well-designed experience, developers, and designers with very specialized knowledge are needed.
External	Opportunities • High VR adoption improves interest, demand, and the use of TFS solutions.	Threats • Excitement about VR dies • Users consider the VR experience as just a showy toy, not a serious tool.

4.2.2 Hardware

For the experiment, I used HTC VIVE Pro VR Full Kit[22] - a full VR kit that allows the user to see, move in, and interact with the 3D VR environment. It includes a VR headset, two handheld controllers, two base stations, and all the accessories to set up the system. The main components of the kit are shown in Figure 18.

Figure 18: The main components of the VIVE Pro Full Kit. From top to bottom: two base stations, headset, and two handheld controllers. [21]

For the user, the headset is the "window" to the VR world. It covers the user's vision and contains two AMOLED screens, one for each eye, that show the virtual environment to the user. The two handheld controllers can be used to interact with the virtual environment. They have multiple input methods, such as a trackpad, grip buttons, trigger, and buttons. The location and the orientation of the headset and the controllers are tracked using base stations. Base stations emit infrared pulses that are picked up by the sensors on the headset and controllers, allowing the system to determine their position and orientation. The base stations are mounted in opposite corners of the physical environment. [21]

[22] HTC VIVE Pro full Kit: https://www.vive.com/eu/product/vive-pro/

4.2.3 Research Objective

The objective of this experiment was to assess the realizability and usefulness of allowing the user to operate TFS electron microscopes using VR. In Chapter 3.2, we defined two hypothesizes:

Hypothesis 1: It is possible to show microscopy images and interact with IM in VR.

Hypothesis 2: VR can be used to intuitively and comfortably navigate (pan and zoom) the sample in the electron microscope.

The first hypothesis focuses on the realizability of making the connection between VR and the IM platform. The second focuses on the usefulness and comfortability of using the VR medium in zooming and panning. To test the second hypothesis, I looked for ways the user could intuitively explore images in VR and later integrated it with the existing IM platform, to test the first hypothesis.

4.2.4 Zooming in VR

This section focuses on exploring the ways the user could intuitively explore microscope acquisitions in VR. Exploring an image involves zooming and panning. While zooming and panning are simple on flat-screen devices, they do not translate well to immersive VR environments. Trying to map head motion to panning runs into complications such as simulation sickness. [22]

I replicated the "Circle mode" from [22] where the center of the viewing field is covered by a "magnifying glass." The solution offers a huge improvement in usability over the naïve approach of just reducing the field of view, but as it obscures part of the viewing field, it makes it hard to locate objects. The paper proposes two improved modes, "Alpha Circle Mode" and "Zoom Circle Mode," to reduce this problem. These two modes make the magnifying glass disappear when the user makes rapid head movements. Therefore, the zooming is triggered whenever the user stops the head movement. Although these two modes solve the problem of obscuring the view, they suffer from a high chance of triggering zooming inadvertently; as soon as the user stops, the zoom is triggered.

Ways to trigger zoom

There are many ways to trigger an action, such as pressing buttons, giving voice commands, or stepping on foot switches. In Chang's paper [22], the trigger to start zoom was a stop of head movement. In our project, we have handheld controllers available with position sensors, a trackpad, trigger, buttons, and haptic feedback. They provide an additional input modality, allowing us to avoid the drawback of an inadvertent triggering of functionality.

Table 9 illustrates the differences between three different methods of triggering zoom action: a head movement stop, a controller button click, and a voice command.

Table 9: Comparison of zoom triggers in VR

	Method for triggering zoom:		
Aspect	**Head movement stop**	**Controller button click**	**A voice command**
Latency	Medium	Very low	Medium
Risk of inadvertent triggering	High	Low	Medium
Other	No need for extra hardware	Immediate tactile feedback	Ability to trigger many different actions

The latency of using head movement as a trigger depends on the dwell-time. If dwell time is set to low, then triggering is quick, but it has an increased risk of inadvertent triggering. When the dwell-time is set higher, the risk of inadvertent triggering decreases, but it also forces the user to stay still for longer to trigger the zoom, making the whole process slower.

The button click triggers the action virtually immediately and gives immediate and clear tactile feedback to the user.

Voice command has a clear advantage in giving a user the ability to trigger a wide selection of different actions based on an utterance. Compared to the button, voice command is slow, as it takes time to speak, analyze, and process the utterance. Length of utterance is important to reduce the time from user intent to action. An utterance "Zoom" is preferred to "Microscope, please zoom in," as it takes less time to say and less time to process. Nevertheless, the shorter utterances increase the likelihood of inadvertent triggering by noise or other similar utterances.

It makes sense to use trigger methods based on the ratio between the latency of the trigger and the length of the action. Higher latency is acceptable for triggering actions that take a long time or are infrequently used. Therefore, for zooming, an action that is used frequently, I suggest using a controller button click as a trigger.

Ways to aim

Table 10 illustrates the differences between three main ways of pointing and aiming in VR: pointing with a laser pointer, head orientation, and eye gaze

Table 10: Comparison of aiming methods in VR

	Method for aiming:		
Aspect	**Controller – laser pointer**	**Head orientation – the center of the view**	**Eye gaze**
Speed	High	High	Extremely high
Intuitiveness	High	Medium	Low
Hardware availability	Yes	Yes	No

Using controllers to point VR is very intuitive and provides a quick way to indicate the location with a laser pointer. Using the Head Mounted Display (HMD) to point is a good option when controllers are not available. It is fairly intuitive, but not as prevalently used as pointing by hand. Eye gaze is an extremely fast pointing method, as eyesight can quickly and easily jump from one object to another. The eye gaze arrives at the target even before the user is conscious of the fact [citation needed]. Nevertheless, it is an extremely non-intuitive way of interacting with the system as eyes are not used to interact with the world in daily life, but only to ingest information.

Choosing methods

The main criteria for selecting a trigger and aiming method for the Intelligent Microscope are:

- **Availability** of hardware (to end users)
- **Intuitiveness**
- **Speed** or **latency** of the interaction method

Based on these criteria, I suggest the following:

- **When possible, use the controller both for aiming and triggering the zoom.**
- When controllers are not available, Use the head for aiming. To trigger an action, use dwell-time or a short utterance.

Therefore, for our project, using controllers for aiming and triggering zoom is the best choice.

4.2.5 Architecture

Integration with the Intelligent Microscope
Virtual Reality UI was integrated with the existing IM platform to allow the operator to benefit from the functionality already present in the IM. The IM platform handles the microscope control, AI functionality, and voice interactions. VR UI presents an interactive 3D world to the operator where she can interact with microscopy images. To interact with the system, the operator must have both applications running at the same time: the existing IM web application and the new VR UI.

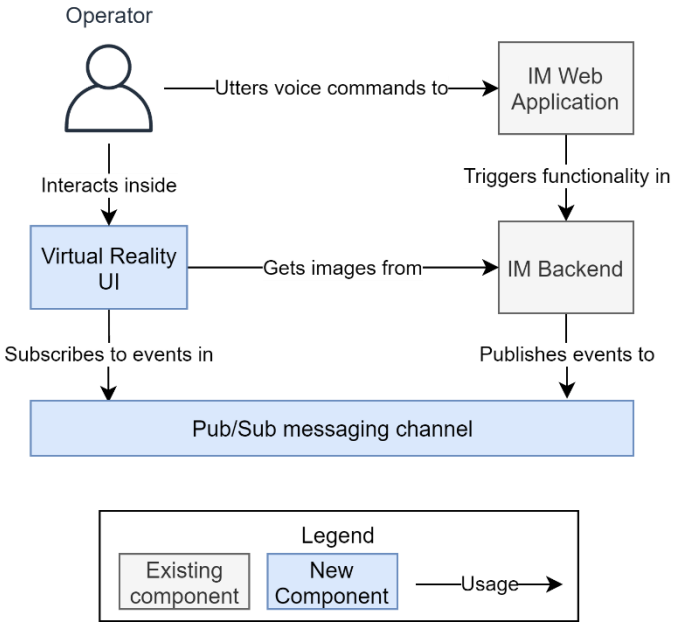

Figure 19: VR UI interaction with existing Intelligent Microscope components

Figure 19 illustrates how the new VR components interact with the existing IM platform. VR UI shows images from IM backend and subscribes to the events that the IM backend publishes. The voice commands are still handled by the existing IM Web application. Now whenever a command is fulfilled, a message is published to notify VR of a new result.

The same communication process is also illustrated in Figure 20. The figure shows the process followed during one voice command. The flow starts when the operator utters a voice command. In response to that, the IM web application triggers the respective functionality in the IM back end by making an HTTP API call. IM backend fulfills the request and as soon as the new image is ready, notifies the VR UI through the Pusher messaging channel. When the VR UI receives a notification about a new image, it downloads it from the backend and presents it to the user.

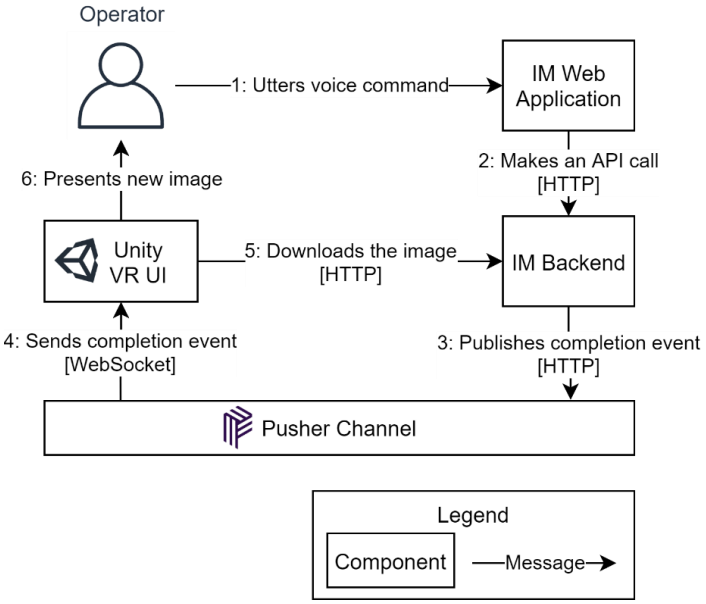

Figure 20: Communication process between the components when the user utters a voice command

4.2.6 Implementation

The VR user interface was built using Unity – a development platform for creating 2D and 3D games and interactive experiences. [23] It supports deployment to a wide selection of platforms. In addition to desktop and mobile devices, Unity supports deployment to VR and AR devices including Microsoft HoloLens, Oculus, and HTC Vive.

The wide support of VR devices has been achieved using SteamVR. SteamVR provides a high-level Input API for VR controllers and Interaction System. It is a collection of scripts, prefabs, and assets such as hands, VR controllers, and a VR camera that you can use in Unity. The most important asset is the Player prefab. By adding this to a Unity scene, a VR HMD can be used to look around in the scene and VR controllers can be used to interact with the scene. The player object contains the VR camera and all the logic handling the controllers.

The initial scene is simple; There is a floor to stand on, a simple light source, and a SteamVR Player object. Additionally, there is one canvas floating in front of the player with a high-resolution image for testing the zooming ability and another canvas for displaying the images coming from the Intelligent Microscope.

To create the magnifying-glass effect, I added a zoom camera and a zoom-projection plane to the 3D scene. The camera is used to capture a zoomed-in version of the view that the user is currently seeing through the VR. The camera is configured with a very narrow field of view (FOV), emulating the telephoto lens. Because of this, the zoom camera captures just a small area in the center of the player's view. The difference in FOV is illustrated in Figure 21. The resulting zoomed-in image is then projected onto the zoom projection plane in front of the player. I scripted both to follow the movement of the user's head: the camera follows exactly the position and orientation of the VR camera and the projection plane floats 1.8m in front of the player's current view.

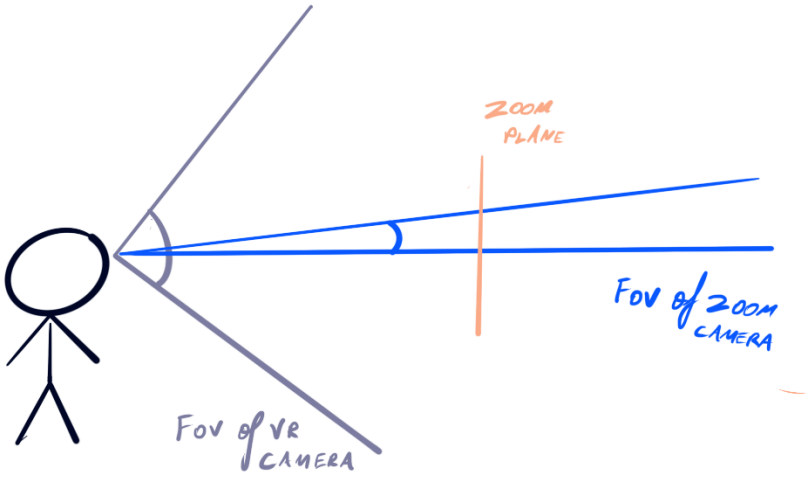

Figure 21: FOV of the VR camera and the zoom camera. The zoom-projection plane is in the center of the player's view.

The zoom projection plane is configured to be invisible to the zoom camera because the projection plane is right in front of the zoom camera. Otherwise, the zoom camera would zoom into the zoomed image creating an infinite loop.

4.2.7 Conclusions

This experiment aimed to assess the realizability and usefulness of applying VR in operating an electron microscope. With the prototype, I confirmed the hypothesis that it is possible to transfer images from the Intelligent Microscope platform to the VR environment over the existing REST API. Additionally, I found that it is possible to have real-time updates from the microscope to VR UI by using Pusher messaging channel.

Within this experiment, I successfully replicated the zooming functionality described in the work of Chang et al. I can conclude that the solution provided for navigating high-resolution images is pleasant to use when the hardware is limited only to an HMD. Nevertheless, when controllers are available, I suggest using them with laser pointers for more intuitive zoom control.

During the experimentation with the VR systems, I uncovered some technical risks and difficulties that are important to keep in mind when pursuing a VR project with Unity.

- Difficult to use NuGet packages – Unity does not have native support for NuGet- a package manager for .NET. This makes it difficult to reuse code written by the .NET community. The lack of native support forces the developer to resort to manual copying of DLLs that is an error-prone or to use non-standard solutions provided by the Unity community.
- Asynchronous code in Unity is handled with coroutines. This is incompatible with the Task-based asynchronous programming model in C# where async/await keywords are used.

4.2.8 Future Work

For TFS, I recommend further experimentation with VR. Using it as an alternative to classical GUIs is technically feasible but finding a suitable use case is not trivial. I suggest looking for specific use cases or workflows where the user clearly benefits from the immersiveness of the VR medium. For the users to embrace VR over existing GUIs, the benefits have to be higher than the cumbersomeness of using an HMD.

To support the development of alternative user interfaces such as VR and VUI, it would help to have microscope functionalities available through technology-independent APIs using standard protocols such as HTTP or gRPC. This would enable developers to quickly experiment with new UIs and services.

4.3 Peer-to-Peer Platform for Sharing Neural Networks

This section covers the third and final experiment in the Intelligent Microscope III project.

The challenge started from a set of broad questions from TFS: how could electron microscopes share knowledge between each other? If one microscope has learned something new, how could it share that knowledge with other microscopes? Would peer-to-peer architecture be useful in such a system?

The following sections cover the background of the challenge, the research questions to answer, different aspects of the challenge, the overview of implementation, and finally, conclusions and recommendations for future work.

4.3.1 Introduction

Every electron microscope made by TFS works as a standalone, independent system. One microscope system consists of a microscope enclosure and a Windows workstation that is used to control everything happening inside the enclosure. Because each system is independent and there is no central management, it means that when the user needs to set up, configure, or use multiple microscopes, she has to do it one by one through the workstations attached to the microscopes, as depicted in Figure 22.

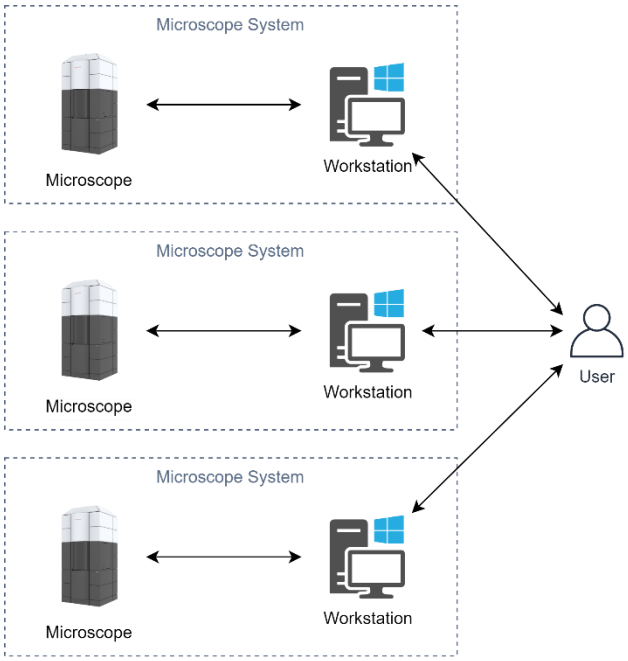

Figure 22: User working on multiple microscopes

Today, one of the customers of TFS has 50 identical microscopes on their premises. Currently, if the customer needs to use machine learning on the microscopes, then the model must be deployed manually on each microscope – one by one. Furthermore, if a new model is trained in one of the microscopes, then the manual deployment must be repeated on all the microscopes, which is a tedious process. It would be more convenient for the customer if the microscopes could share the models between each other automatically; if one microscope learns something, then others learn it too.

TFS has already created a solution for this problem, by creating a central machine learning service that each microscope can use to execute inference tasks. The goal of this experiment was to explore the opposite direction: how we could share the neural networks directly between the microscopes without relying on any external servers.

4.3.2 The Domain of Machine Learning

This section serves as an introductory overview of basic supervised machine learning processes and transfer learning. This overview is given to familiarize the reader with commonly used terms such as training, inference, machine-learning model, transfer learning, and fine-tuning.

Supervised machine learning processes

Supervised machine learning can be split roughly into two major processes: **training** and **inference**, as depicted in Figure 23. Training is the process of preparing the machine learning model to be useful by showing it labeled training data that it can learn from. Inference is the process of using an existing trained machine learning model to make useful predictions about the new unlabeled data.

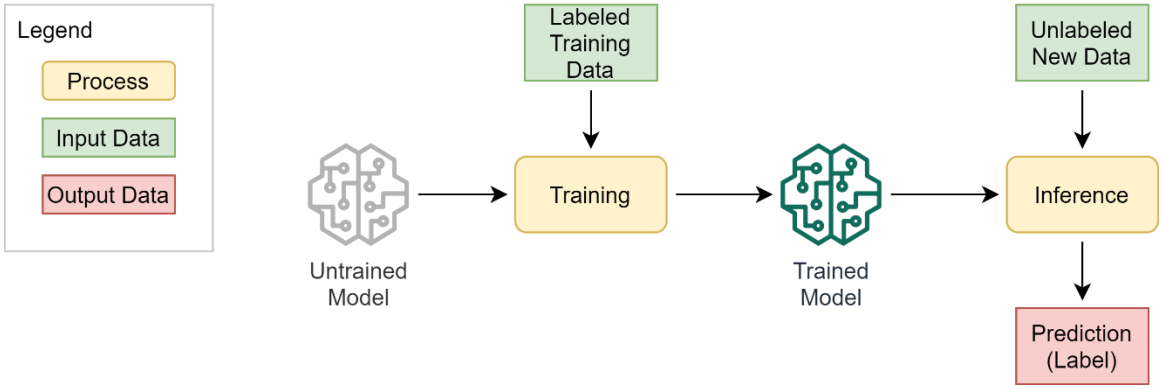

Figure 23: Traditional Machine Learning process. The model is trained using a training dataset. Later, the trained model can be used in the inference process to make predictions on new data.

When training a deep neural network from scratch, a lot of labeled training data is needed for the training process. Furthermore, in traditional machine learning, this training process has to be repeated for every different prediction task. This means that each task also needs a separate training dataset. Sometimes, creating a large dataset of labeled samples can be difficult or just economically infeasible. To overcome this difficulty, it is common to use pre-trained deep learning models as a starting point and then train it further a particular task. Transfer learning is the key machine learning approach for achieving this.

Transfer learning

Transfer Learning is a method in deep learning where an existing well-trained model is retrained for another similar task. With this method, the training time and the number of needed samples can be greatly reduced. Instead of starting the training from a completely randomly initialized and untrained model, training is started from a model that has been trained for a similar task. Using an existing pre-trained model as a starting point for training is illustrated in Figure 24. [24]

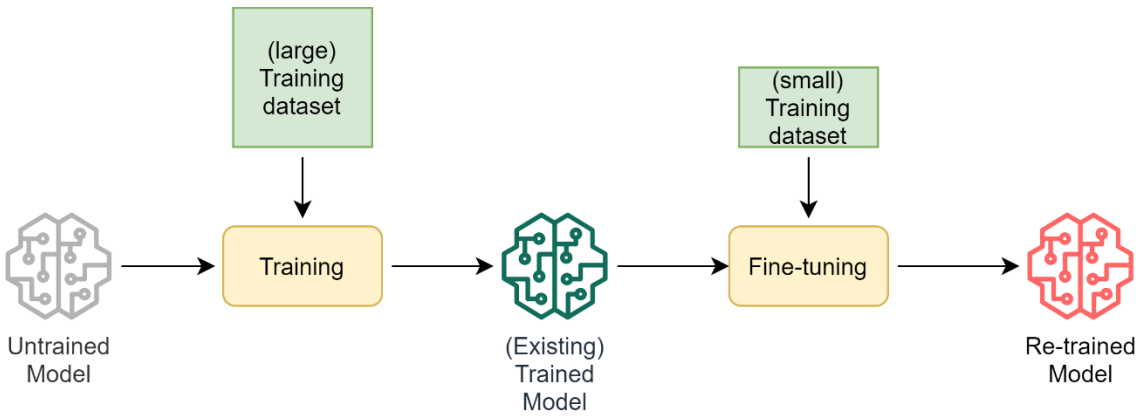

Figure 24: Transfer learning. A pre-trained model has been trained on a large training dataset. A small training dataset is used to fine-tune the pre-trained model to suit a particular task.

For TFS customers, transfer learning can be used to train many similar, but task-specific neural networks quickly and without the need to build large datasets for training each model.

Domain example

For example, neural networks can be used to detect defects in the manufactured MEMS devices by analyzing microscope acquisitions. I assume if a microchip manufacturing company has a trained neural network to detect defects in one type of MEMS device, it can be retrained to detect defects in another MEMS device. Compared to traditional machine learning, transfer learning allows the company to train the second model faster and with far fewer training samples.

4.3.3 Research Objectives

As stated in Section 3.3, the research questions are: Is it possible to set up a P2P system for sharing neural networks? If so, what kind of opportunities and technical challenges would it bring?

To answer this question, the objective of this experiment was to build a prototype system where multiple nodes exchange trained neural networks between each other in a peer-to-peer manner.

Because the broader goal of the IM project is to de-risk novel technologies and to uncover novel opportunities, then with this investigation, we tried to answer the following questions:

- What challenges and risks appear when trying to build this system using peer-to-peer architecture?
- Is it more fault tolerant to use a peer-to-peer system over a centralized system and why?
- What other opportunities does this approach open?

The main benefits that TFS expects from a peer-to-peer model-distribution system are cost reduction and fault-tolerance to hardware and software failures. Cost reduction is expected from not having a separate centralized system for model storage and inference. Fault tolerance is the system's ability to maintain an acceptable level of service in the event of a failure in some of its components, such as hardware or software failure. This fault tolerance of a P2P system is expected to come from not having a single point of failure. Each node in the system is independent and can continue working if others fail.

4.3.4 Problem Analysis

What does "sharing knowledge" mean?

In our experiment, "sharing knowledge" means sharing trained models between microscopes. Nevertheless, depending on whether we focus on sharing information during the training or the inference process, the word pair "sharing knowledge" can have two distinct interpretations. We focus on the inference side and assume that the models have already been trained. The challenge here is to distribute trained models across a large number of microscopes efficiently and reliably.

Where to perform the inference?

There are two main options in choosing the deployment location for inference: local and remote. The local deployment means placing the inference service in the workstation that is directly connected to the microscope. The remote deployment means placing the inference service into another network-accessible computer or cluster.

This remote deployment could be on the premises or in the public cloud. Private cloud deployment means placing the service in a network-accessible computer or cluster on the customer premises. Public cloud deployment leverages the cloud resources (storage and servers) owned and provided by a third-party such as Amazon Web Services or Microsoft Azure.

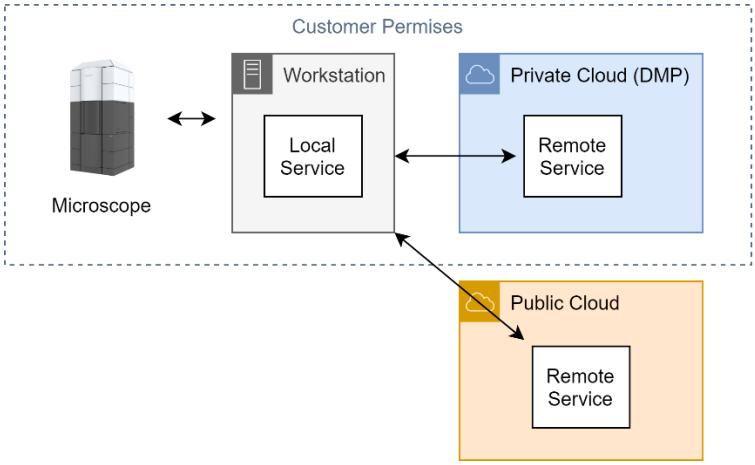

Figure 25: Deployment options for inference service

All three options are depicted in Figure 25: Local inference on the workstation, remote inference in a private cloud, and remote inference on a public cloud. A comparison of inference deployment can be seen in Table 11.

Table 11: Comparison of inference deployment options

Type	Local	Remote	
Location	Workstation	Private cloud	Public cloud
Horizontal Scalability	Limited – single workstation	High	Nearly unlimited
	Only vertical scaling is possible. (Adding more computational power and storage to the machine attached to the microscope) There is exactly one inference node for each microscope.	Both vertical and horizontal scaling is possible. Computational load can be distributed across multiple remote nodes. The number of inference nodes can be changed independently of the number of microscopes as computational demand changes	
Network load	Low Only inference results and models are transferred over the network. (Network load is low only when inference results and models are small)	High Full acquisitions are transferred over the local network to inference machines.	High Full acquisitions are transferred over the **internet**

The main benefit of local inference is that the data produced by the microscope can be processed locally and does not have to be sent over the network. By performing inference on the machine attached to the microscope, we can avoid building high-bandwidth networks. Furthermore, we can reuse the computing power of the machine that is already supplied with the microscope.

Design decision:
In the context of machine learning in electron microscopy,
facing the need to perform inference on electron microscope acquisitions
we decided to perform inference on the local machine
and not in a remote computer or cluster,
to achieve low network load
accepting that only vertical scaling of computational power is possible.

How to distribute the models?

To allow microscopes to perform machine-learning inference locally, the trained models must be present on each microscope. Therefore, one of the challenges of this experiment is distributing the models.

There are two main architectural approaches in building a system for distributing neural networks between the microscopes: centralized distribution or peer-to-peer distribution. The approaches are visualized in Figure 26. Centralized architecture would mean creating central storage of models that can be accessed by all microscopes deployed in the company. Each microscope can download existing networks and use them for local inference. The second would be to build a peer-to-peer network of connected microscopes that all work together to distribute known models among themselves.

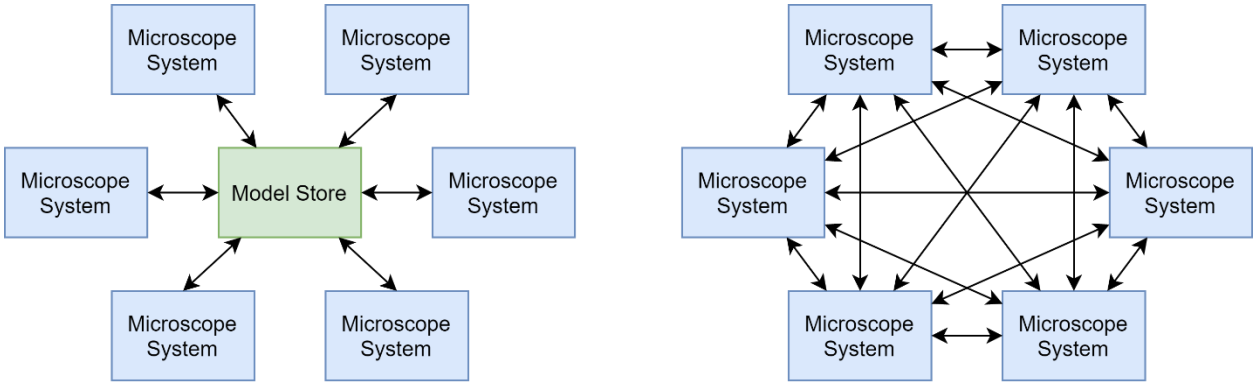

Figure 26: Centralized and peer-to-peer model distribution

For this experiment, we have chosen to investigate only peer-to-peer architecture for model distribution. We believe that centralized model distribution is straightforward to build and therefore does not need de-risking. Building a peer-to-peer model distribution system poses more uncertainties and unknowns and therefore is chosen for de-risking.

The main benefits that TFS expected from the peer-to-peer model distribution system were cost reduction and fault tolerance. Cost reduction was expected from not needing separate hardware for the central model store. Instead, existing workstations already attached to microscopes could be used for this purpose. Because of the lack of a single point of failure, the peer-to-peer architecture was also expected to be more fault tolerant in the case of hardware and software failures. Table 12 summarizes the differences between the two approaches.

Table 12: Comparison between centralized and decentralized model distribution

	Centralized (client-server)	**Decentralized (Peer to Peer)**
Summary	The Server provides service (model storage service) to clients. The clients use the service to upload and download models.	Each node in a network can request for models and can also provide models.
Cost	Extra hardware for central model store	
Complexity	A standard solution. Straightforward to develop and understand.	Complex to develop unless an existing solution is found and leveraged.
Fault tolerance	The central store is the single point of failure. If this stops, the sharing stops.	No single point of failure.
Storage	Server stores all models in a centralized location	Each node stores all models
Scalability	Each client has one connection. (const) For the server, each new device adds one connection. (linear)	Each new device adds one potential connection to each peer in the network. (linear)

Risks
Building a peer-to-peer model distribution system brings out some risks:

- If each node has to store all of the models, then the total storage space needed by the system grows fast. (The storage load grows in steps of M*N, where M is the size of a model and N is the number of microscope nodes in the network)
- For each peer, the number of possible connections rises linearly, as more nodes are added.
- Communication cost – all models are transferred to all the microscopes.

Design decision:
In the context of using machine learning models in electron microscopes
facing the need to distribute models to microscopes
we decided to use peer-to-peer architecture
and not centralized client-server architecture
to achieve **fault tolerance of the system** by eliminating a single point of failure and removing the need for extra hardware.
accepting the complexity of development, complexity of debugging, and reduced visibility of the system's state.

4.3.5 Requirements

Functional requirements
Functional requirements describe **what** the system does without specifying the solution, or **how** the system does it. The main functional requirements that this system has are:

- Local Inference: Each node in the network can perform local inference.
  Each microscope must be able to perform inference on local hardware, without sending the data to a remote node.
- Exchanging models: When a new model is added to one of the microscopes, then eventually it will be available in all other microscopes.

Non-functional requirements
- Fault tolerance: When there is a hardware or software failure in one node, the other nodes will stay operational and continue sharing models.

Design constraints
- Must be a peer-to-peer system – This is the technology TFS is interested validating
- Must work "offline", without the connection to the Internet – the system cannot rely on external services or servers that are outside of the local network because many of TFS customers keep microscopes isolated from the internet for security reasons.

Conceptual view (Product "How")
As a solution, we decided to build a simple application that would be able to exchange models between other instances of itself.

- Each node publishes a list of available models.
- At the user's request, the system performs a local inference using one of the locally available models.
- Each peer can upload any model to any peer
- When a new model is added to one node, it will notify other peers.
- When a node gets a notification from a peer about a new model and it is not present on the node, the node will try to download it.
- Alternatively to the previous two points, the nodes could just "Push" the new model to other peers directly by upload.

4.3.6 Architecture

This section presents the system architecture that was created based on the requirements defined in Section 4.3.5.

Context diagram of the expected use of P2P model service:

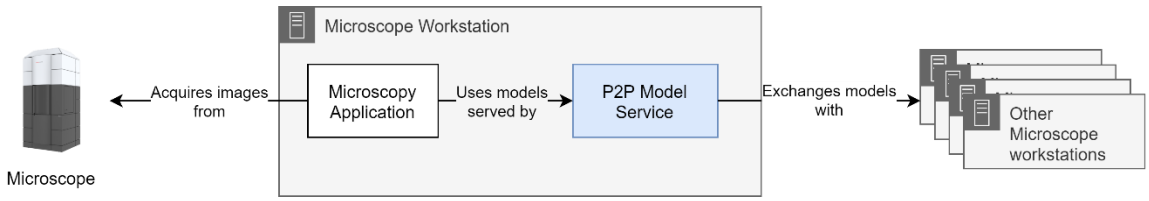

Figure 27: Context diagram of the expected use of the "P2P Model service"

The P2P Model service is envisioned to run on the microscope workstation, next to the microscopy application that needs to use ML models as illustrated in Figure 27. The Microscopy Application acquires images from the microscope and uses the P2P Model service to perform inference tasks. At the same time, the P2P Model service exchanges models with other microscope workstations that have the same service installed.

The model Service consists of two applications:

- P2P Model Exchange - for exchanging models with other nodes
- Inference service - for serving ML models

These two applications with their interactions with other components are shown in Figure 28. The Model Exchange service accepts models from other nodes and saves them to the filesystem. TensorFlow serving can then load the models from the filesystem and provide inference service.

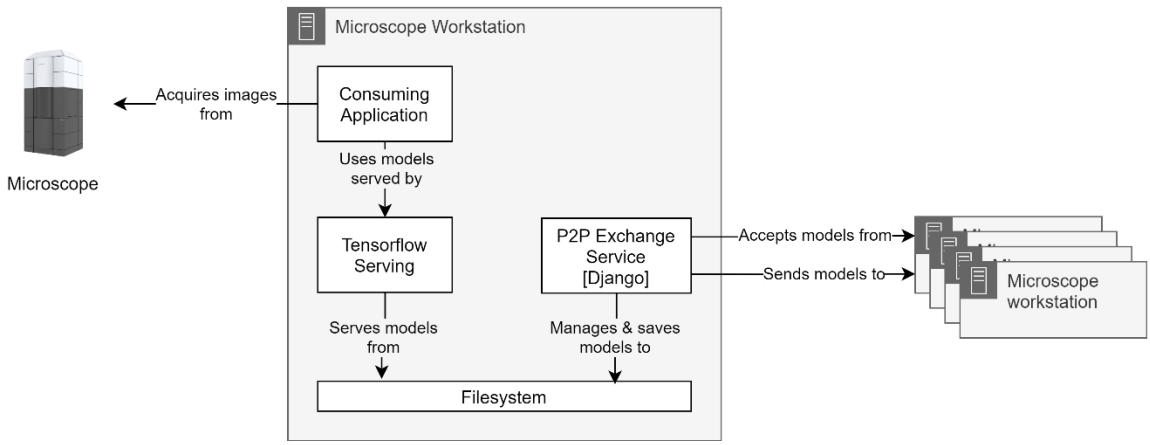

Figure 28: Applications inside a microscope workstation

Data delivery: Pull versus Push

In a distributed system, there are two classic approaches used to disseminate files and data: push-based data delivery (upload) and pull-based data delivery (download). The difference lies in which side of the transfer is active and which is passive.

Traditional client-server systems transfer data using pull-based data delivery. In a pull-based data delivery, the clients actively request data from the passive server node. The advantage of a pull-based approach is that the client nodes can play an active role in requesting the data they need rather than relying on the pushes from the server. However, there is also an important scalability disadvantage in a pull-based system: as the number of client nodes and the number of requests rises, the higher is the risk of overloading the source node with pull requests. [25]

In a push-based data delivery, the source node is actively uploading data to passive destination nodes. This allows the source node to be in control of the dissemination of the data, circumventing the risk of overloading the source.

Design decision:
In the context of machine learning model distribution
facing the need to transfer large files to a large number of nodes
we decided to push-based data delivery
and not pull-based data delivery
to avoid overloading the source node
accepting that receiving nodes are not in control of what is being sent to them

Peer discovery
Decentralized applications require some way of discovering other nodes in a network before any data transfer can happen. The process of locating the nodes in a peer-to-peer network is called peer discovery. Many techniques can be used for this purpose. Peers can be discovered via multicast DNS (mDNS), Distributed Hash Tables, and other techniques. The selection of the optimal technique depends on the use case, as each one has its own strengths and weaknesses. [26]

In our implementation, we skip the peer-discovery process and just assume that each peer is accessible through a predictable hostname such as “microscope-1” and “microscope-2”. When this is not possible anymore, we can take the next step and start using mDNS.

4.3.7 Implementation

P2P Model Exchange Service – Django application
Model exchange service is a custom Django application responsible for accepting new models and exchanging models with other nodes in the P2P network. Figure 29 illustrates how P2P Model Exchange Service relates to the user and other components in the system. It is mainly responsible for accepting models from data scientists and peers on the network. It stores the models on a filesystem and instructs the TensorFlow Serving to load the models whenever there is a change in the list of models. The service is also responsible for distributing the model to other peers by uploading it to the HTTP endpoint.

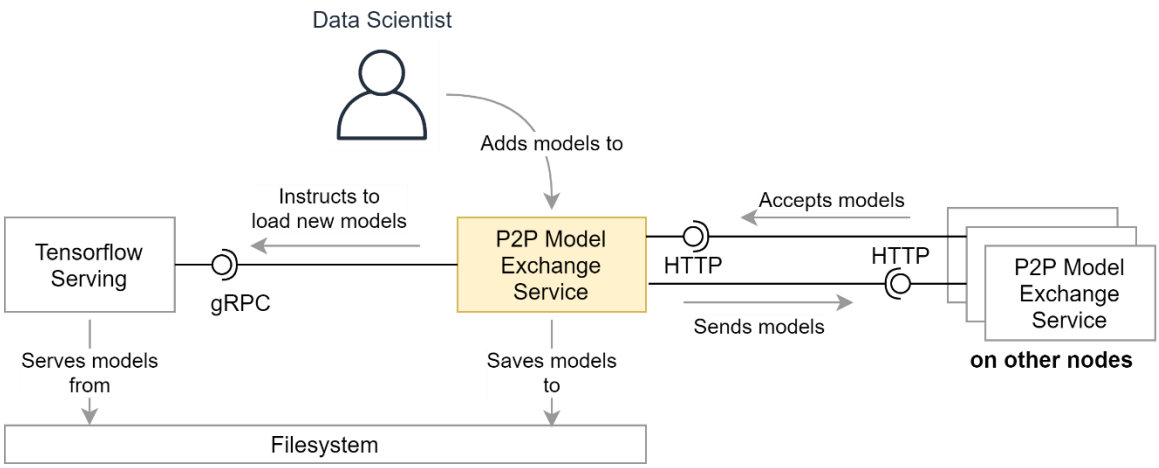

Figure 29: P2P Model Exchange service

Whenever a new model is received it is checked for validity and stored on the file system. Next, the TensorFlow serving is notified of the new model so it can start serving it. Finally, the model is distributed to other peers. This flow is illustrated in Figure 30.

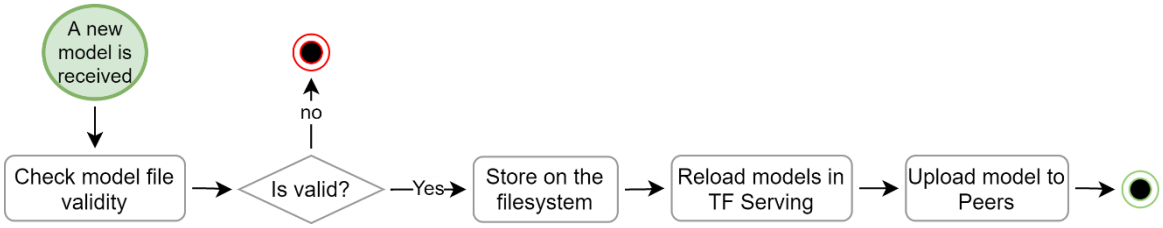

Figure 30: Process of accepting a new model into the system.

Inference server – TensorFlow Serving

For serving the models and performing inference, I used TensorFlow Serving [27] – a high-performance serving system for ML models. It provides a standard gRPC and REST APIs for inference tasks such as classification and prediction as illustrated in Figure 31. By default, it supports serving TensorFlow models stored in SavedModel format that are loaded from a filesystem.

Figure 31: TensorFlow Serving serves an ML model to a client application

TensorFlow Serve is deployed inside a Docker container as it is the easiest deployment options. There is one limitation though - Docker on Windows does not provide access to GPUs on the host machine. This might make inference in containers not a viable option on current microscope workstations if GPU acceleration is needed.

For our experiment, going forward without GPU support was acceptable, but if GPU is needed, then there are three options to overcome this limitation:

- Deploy ML services natively to the Windows workstation, without a container
- Move ML service to separate Linux node
- Start using Linux on workstations

I would suggest the first option, as is the easiest to apply immediately but comes with the downside of having to forego the benefits of containerized services. The second one needs an additional PC or cluster. The last solution is not feasible, as it would mean rewriting all the Windows-dependent code currently running on the workstation.

Deployment view

During this experiment, all the components were deployed on a single development-machine. To simulate multiple P2P nodes, I used docker-compose to spin-up multiple identical sets of containers.

Development view

The development view describes the system from a programmers' perspective.

Pipenv

To ensure future developers can easily reproduce the experiment, I decided to use Pipenv. Pipenv is a tool for managing Python virtual environments and source code dependencies. With a single command "pipenv install" it sets up a python virtual environment with using the correct python version and recursively installs all dependencies needed to run the source code. The expected python

version and the dependencies of the source code are declared in *Pipfile* and the exact pinned versions of the packages are stored in *Pipfile.lock*, ensuring deterministic setup every time.

4.3.8 Conclusions

The goal of this experiment was to explore how neural networks could be directly exchanged between the microscopes using a peer-to-peer architecture.

Generally, it seems that P2P architecture is a viable solution for distributing neural networks, although technically more complex than a system with a central control. Transferring SavedModel files over the network to a node and then deploying them in TensorFlow Serve works well. This proves the first hypothesis defined in Section 3.3. The complex part of the system is building a reliable and efficient P2P network, where this transfer could happen.

The solution built during this experiment is not a final solution for building a decentralized model sharing network, because of multiple problems affecting its scalability. First, there is no peer discovery and the node addresses are fixed. This makes adding and removing nodes to the network cumbersome. Second, the peers have a naïve approach in handling the model exchange – they push models to each other. The receiving node has little control over the upload process. In Django, the request handler is run only *after* the upload has been completed. The more nodes there are in the system, the more there is a chance that a single node is going to receive multiple uploads of the same model. Nevertheless, this could be avoided by having a better communication model where nodes would agree to the transfer prior to the actual upload.

4.3.9 Future Work

This project focused on the deployment and inference part of the machine learning and covered only the distribution of trained models between the microscopes on the premises of one customer. It only scratched the surface of what is possible in this domain. This section presents some pointers for future experimentation.

Federated Learning

It would be interesting to also investigate the training part of the machine learning process. Federated Learning could be used to train models without exchanging data samples between microscopes or even different TFS customers. For this, TensorFlow Federated[23] or OpenMined's PySyft[24] could be used. PySyft would be extra interesting in use cases where the data must stay private. This is the case when multiple customers of TF would like to collaborate on training a model, but not share the original data.

Multicast Push

In a unicast transmission, IP packets are sent to a single recipient on a network. Given there are many microscopes on the network, distributing large models to many nodes on a network in unicast transmissions is not the most efficient use of the network bandwidth. A multicast transmission could be used to send the data to a group of hosts on the network instead. This would allow pushing a model to multiple nodes with only a single transmission, greatly reducing the network load. For this, UFTP, UDP-based FTP with multicast, could be used. This is a file transfer program designed to distribute large files to a large number of receivers simultaneously [28]

Data Replication Using IPFS and IPFS Cluster

Building a model distribution platform on existing P2P protocols could vastly reduce the development effort needed to build a reliable and scalable platform. One such protocol is the InterPlanetary File System[25] (IPFS).

[23] https://www.tensorflow.org/federated
[24] https://github.com/OpenMined/PySyft
[25] IPFS: https://ipfs.io/

IPFS is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. The system consists of many IPFS peers that store and exchange files.

IPFS Cluster[26] is a distributed application (runs as a sidecar to IPFS peers) that maintains a cluster pinset. Pinning is the action of keeping the file locally available on the nodes. By default, when adding a file to an IPFS cluster, the file would be pinned (downloaded and kept locally) by all of the nodes.

Using a private IPFS cluster could be one approach to distribute models across a cluster of microscopes using peer-to-peer connections.

[26] IPFS Cluster: https://cluster.ipfs.io/

5. CONCLUSIONS

This chapter presents the conclusions of this report. Section 5.1 presents the obtained results and the lessons learned. Recommendations for future work are presented in Section 5.2.

5.1 RESULTS

This section covers the most important results and insights gained from the three experiments. It contains the high-level answers to research questions defined in Chapter 3. More detailed results and insights can be found in the conclusions section of each respective experiment in Chapter 3.

5.1.1 Multilingual Voice User Interfaces

The goal of this experiment was to assess the feasibility of using general-purpose machine translation services such as Google Translate to turn an existing IM English-language VUI into a multilingual VUI. To answer the research question defined in Section 3.1, I built the multilingual VUI and tested it in four languages. The hypothesis was that the system would work well for most of the use cases but would have systematic errors caused by mistranslations that could be easily corrected.

After building and testing a multilingual VUI described in Section 4.1, I can conclude that at this moment, using general-purpose transcription and translation services from Google, Amazon or IBM is not a viable approach for creating a multilingual VUI for microscope operation. The main obstacle in this approach is the insufficient accuracy of general-purpose transcription services in transcribing the commands. The inaccuracy of transcriptions results in poor recognition of commands and makes the VUI uncomfortable to use.

5.1.2 Virtual Reality

This experiment aimed to assess the realizability and usefulness of applying VR in operating an electron microscope. TFS was interested in whether it is possible to operate the microscope from a VR environment and what kind of technical challenges would it bring. Additionally, TFS was interested in learning about the opportunities and benefits VR can bring.

To assess the realizability, I built a VR environment in Unity and integrated it with the existing Intelligent Microscope software system. To assess the usefulness of VR in sample navigation, I recreated the zooming technique described in [22] and compared that to other zooming methods.

After experimenting with VR as described in Section 4.2, I can confirm that VR can be used to control and interact with TFS microscopes, as long as the needed functionality is available through existing APIs. Interfacing with IM over HTTP and Pusher messaging channels is enough to trigger actions and show microscope images in VR as they are taken.

Based on my experimentation, I could not make definitive conclusions on whether VR is better for panning and zooming microscopy images compared to existing GUIs. VR is an immersive medium and can potentially be used to enhance the user's immersion in the task. Whether this benefit can be leveraged, depends highly on a particular use case.

5.1.3 Peer-to-Peer Model Sharing platform

The goal of this experiment was to explore how we could share knowledge in the form of neural networks directly between the microscopes without relying on any external servers. TFS was interested in whether it is possible to create such a system using a P2P architecture and If so, what kind of technical challenges would arise.

To guide this exploration, I built a prototype model sharing platform using Django and TensorFlow Serve. This prototype could transfer models in SavedModel format from one peer to another and load them into the local TensorFlow Serve instance.

Based on my experimentation, I would not advise TFS to follow the approach I took – building a P2P platform from scratch. Instead, I suggest investigating into existing platforms such as IPFS and leveraging those for reliable P2P file transfer and peer discovery.

Generally, it seems that P2P architecture is a viable solution for distributing neural networks, although technically more complex than a system with a central server. Having the inference available in a local machine, makes the microscope independent from the rest of the system, improving its fault tolerance. Transferring SavedModel files over the network to a node and then deploying them in TensorFlow Serve works well. The complex part of the system is building a reliable and efficient P2P network, where this transfer could happen.

5.2 RECOMMENDATIONS FOR FUTURE WORK

IM3 project started with a very broad goal: to identify novel technologies that would improve the ease of use of electron microscopes in five years. This goal can be used to research a myriad of different technologies. IM3 project touched only 3: VUIs, VR, and P2P systems. To pursue the same goal further, one can start from either re-evaluating the candidate technologies listed in Chapter 3 and pick a new potential disrupting technology to investigate. Another option is to go further with the technologies selected in this project.

This section covers high-level recommendations for future work for each technology that was investigated in IM. The more detailed version with specific technical recommendations can be found in Section 4, in the last section of each experiment. For multilingual VUI: Section 4.1.12, for VR: Section 4.2.8, for P2P platform: Section 4.3.9.

Multilingual VUI

The multilingual VUI designed in this project was not comfortable to use, because it was not accurate enough in recognizing the user's commands. To improve this, I suggest trying out other transcription services such as Microsoft Speech to Text. Another approach is to wait for general-purpose transcription and translation services to improve over time and perform this experiment again. The hope is that as time goes by, those services improve, and their accuracy becomes high enough for our use case.

Virtual Reality

In this project, confirmed the feasibility of bringing images from the IM into the VR world, but I could not find a use case that would clearly benefit from that. For this reason, I suggest further experimentation with VR, but this time starting the search from specific use cases where the user would benefit from the immersiveness of the medium.

Peer-to-Peer Platform for Sharing Neural Networks

Although P2P architecture seems viable for sharing neural networks, building one from scratch is not an easy task. For this reason, I suggest investigating existing P2P protocols and platforms such as IPFS. Leveraging existing protocols for reliable P2P file transfer and peer discovery could vastly reduce the development effort needed to build a reliable model-sharing platform.

In this project, I only investigated distributing already trained models. If the ultimate goal is to learn from a distributed set of microscopes then I suggest investigating Federated learning. For this, TensorFlow Federated or OpenMined's PySyft libraries could be used.

6. Project Management

Project management is an important part of a PDEng graduation project, as the trainee is expected not only to design software, but also define, monitor, and manage the processes that will lead to a successful conclusion of the project. This chapter presents the project management techniques and processes that were used during this project.

6.1 Introduction

Project management is the application of knowledge, skills tools, and techniques to meet the project requirements. It includes planning and executing the project, setting up and maintaining effective communications with stakeholders, and balancing project constraints such as scope, schedule, and resources. [29]

The decades-old iron triangle models the constraints of project management as three interlocked components. It is a simplistic model to convey the idea that it is not possible to change one constraint without also changing the others. Therefore, the job of a project manager is to balance the following three constraints to ensure the successful conclusion of the project. [30]

- Scope – The number of features and their complexity and quality.
- Time – The amount of time allotted for the project
- Resources – available budget and the number of workers

For a project, it is important to clearly separate which of these components are fixed constraints and which are negotiable. This separation allows the project manager to decide where to focus the management activities. In the PDEng graduation project, the timeframe of the project has been fixed to ten months and the human resources are fixed to one student. This leaves the PDEng trainee only to vary the scope of the project as illustrated in Figure 32. By clearly separating the fixed and negotiable components, I could choose the component that I should place my focus on managing. Defining and controlling the scope was the main tool for keeping the project on track.

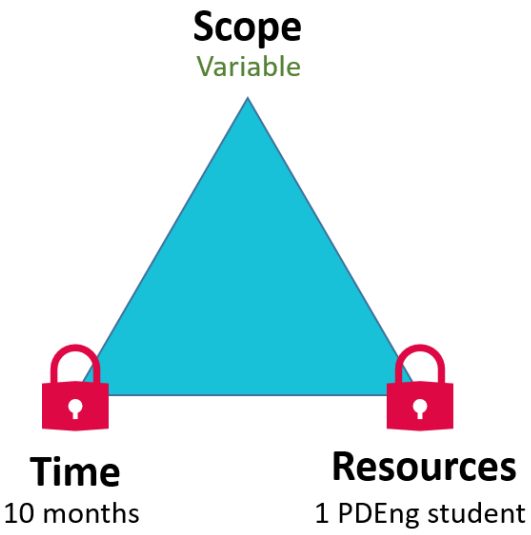

Figure 32: Iron triangle with variable scope

6.2 STAKEHOLDERS

A project stakeholder is an individual, group, or organization that may affect, be affected, or perceive itself to be affected by a project. [29] The main identified stakeholders of the IM3 project are shown in Table 13, along with their role in the project and their main interests.

Table 13: Stakeholders of the project, along with their roles and interests

Name	Role	Interests
Mark Laane	PDEng Trainee	• Successfully graduating from TU/e with the PDEng degree • Building an interesting, exciting system • Gaining experience in project management
Remco Schoenmakers	Supervisor (TFS)	• Sparking innovation in the company • De-risking novel technologies • Identifying valuable use cases for novel technologies
Mykola Pechenizkiy	Supervisor (TU/e)	• Ensuring the academic quality of the final report • Contributing ideas towards the content and technology
Pavel Potocek	Domain Expert	• Having a platform to showcase his innovations
Yulong Pei	Co-Supervisor (TU/e)	• Sharing insights about Natural Language Processing, Machine Learning, Artificial Intelligence, and Graph Mining. • Gaining experience in supervising
Yanja Dajsuren and Peter Heuberger	PDEng Program Director	• Improving the reputation of the PDEng program • Improving the quality of the project outputs and results

Stakeholder management is critical for the success of every project as no project exists in isolation. The input from main stakeholders helps to define the goals of the project and the output of the project has to be communicated effectively. This is needed to ensure the project produces useful output and the stakeholders would understand the benefits of it. Effective communication is critical to gain and keep stakeholders' support throughout the project.

During the IM 3 project the following four steps in stakeholder management were taken:

- Identification of the stakeholders
- Prioritization/categorization of the stakeholders
- Establishment of the communication plan
- Engagement with the stakeholders

To develop an appropriate communication strategy with each identified stakeholder, the stakeholders were prioritized and categorized using the power-interest grid shown in Figure 33. The power-interest grid classifies the stakeholders into four categories based on their power over the

project and their interest in it. The categories with their respective general communication advice are as follows:

- High interest, high power – manage closely
- High interest, low power – keep informed
- Low interest, high power – keep satisfied
- Low interest, low power – monitor

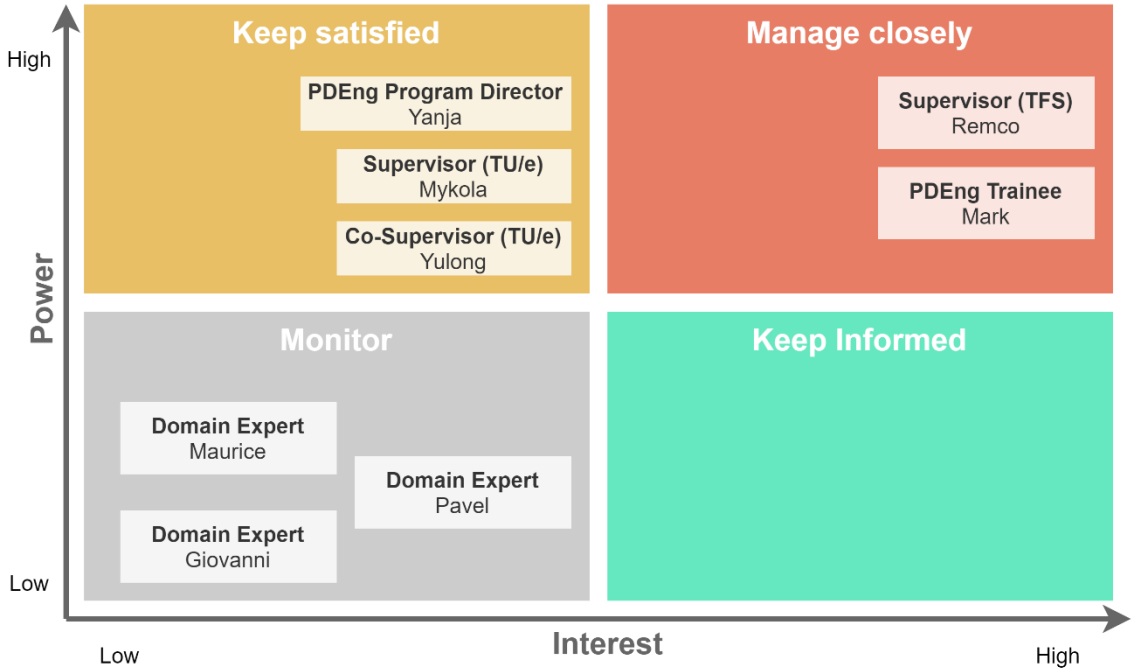

Figure 33: Stakeholder power-interest grid

The categories were used to develop an appropriate communication strategy for each stakeholder and the strategy was recorded in the communication plan. This communication plan was used throughout the project when engaging with the stakeholders.

6.3 Way of working

The communication plan specified two types of recurrent meetings: weekly progress meetings and monthly PSG meetings. Weekly meetings were held with the company supervisor on Mondays, to share the latest developments and discuss the next steps. These meetings were used to reflect on the last week and plan for the next one. Project steering group meetings were held monthly with all supervisors, to share the project's general progress and steer its direction if needed. Meetings with other stakeholders were held on an ad hoc basis. For running matters, e-mail was used with all stakeholders.

The scope of the project was expected to change throughout the project. The scope was adapted continuously to fit the time and resource constraints as I learned more about the domain, the problems, potential solutions, and relevant technologies. Until the end of April, I was working in short iterations and close customer collaboration on a meeting-to-meeting basis and adjusting goals weekly. The risk with this kind of approach is that it is very easy to get sidetracked from long-term goals and just build features that seem interesting at that moment. At the end of April, I created a roadmap to visualize the long-term direction of the project.

For communicating the high-level strategic vision, direction, and progress, I used a roadmap. It is a visual tool that is useful for aligning stakeholders setting priorities. An example of the roadmap used

in the project can be seen in Figure 34. The roadmap contains initiatives – descriptions of expected outcomes and a broad timeline (past, now, next, later). The roadmap does not specify exact dates for initiatives because its main goal is to convey priorities and strategy, not predict the future. Predicting the future is difficult, especially with the uncertainty coming from the project's explorative approach. Creating a roadmap forced me to make hard decisions about priorities – what to tackle first and what to leave for later. It also helped me to align the stakeholders on priorities of different initiatives, as I used it during the PSG meetings.

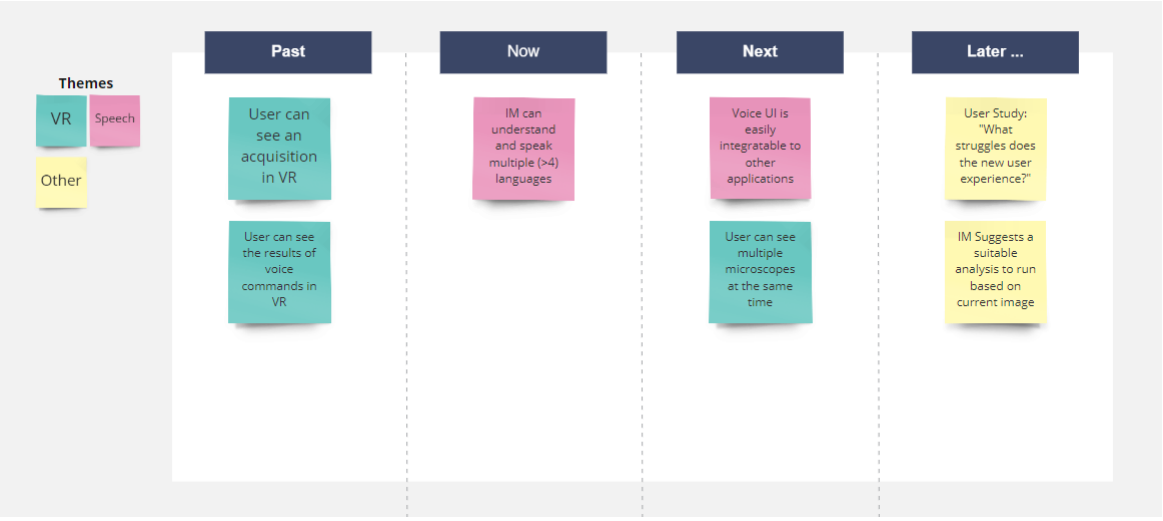

Figure 34: The product roadmap used during the project

The near-term features and tasks were managed using a Kanban-style board that contained the next tasks in a prioritized backlog, currently active task, and done or discarded tasks. Figure 35 shows the physical board on the office wall.

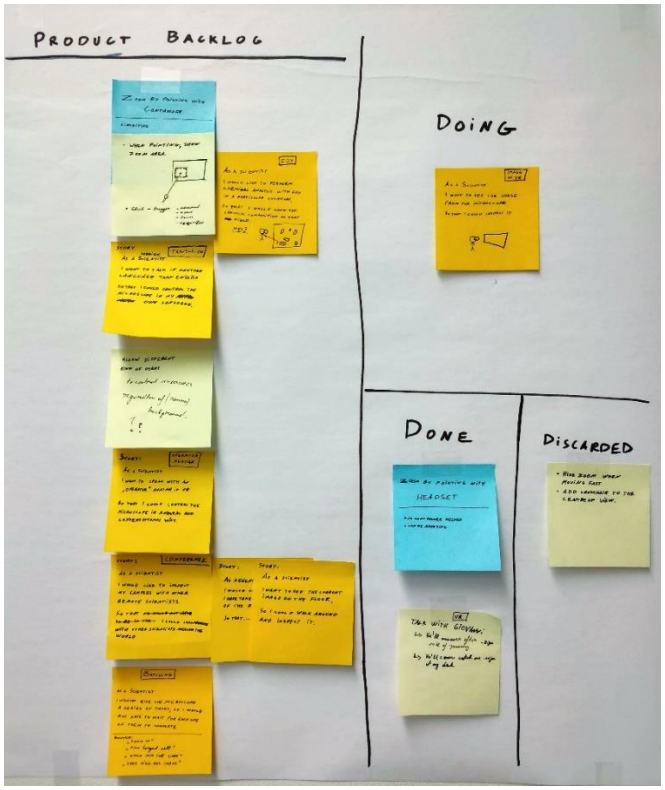

Figure 35: The physical Kanban board for managing near-term features and tasks.

6.4 SCHEDULE

The IM3 project was a 10-month-long project with a fixed duration spanning from January 2020 to October 2020. For development and exploration activities were executed in rapid iterations and only a couple of weeks were planned ahead. From the project's timeline, three major phases can be identified in retrospect: initiation, experimentation, and closing. Figure 36 illustrates the major phases of the project and shows the activities performed. The following sections describe the activities carried out in each phase of the project.

6.4.1 Initiation

The initiation phase took place in January. During the initiation phase, I familiarized myself with the goals of the company and with the existing work done in previous iterations of the project. Through interviews, I extracted a list of potential goals and objectives for the project, mapped them, and prioritized them according to the interest shown by TFS. This formed the basis for choosing the technologies to investigate in the experimentation phase.

During the initiation phase, I also created the first versions of various project management documents, such as stakeholder map and analysis, communication plan, and risks table and analysis. These documents were later updated during the project as needed. This time was also used to set up recurrent planning meetings and scheduling the first few PSG meetings.

6.4.2 Experimentation

The experimentation phase was the longest phase lasting 7 months from February to August. During this period, the development of three experiments happened. First, the experimentation with VR, then with multilingual VUI, and finally with P2P model sharing. The experimentation included setting the goals for experiments, designing and implementing the software systems, and validating the results. Additionally, during this time, I wrote parts of this project report iteratively, recording the goals, results, and findings of the experimentation.

6.4.3 Closing

The closing phase took place in September and October. This time was used for writing and finalizing this project report and preparing for the final defense presentation. The project report went through multiple reviews by the supervisors to ensure and improve its quality. The end of September was used to prepare for the final defense. Finally, the defense presentation was given at the beginning of October.

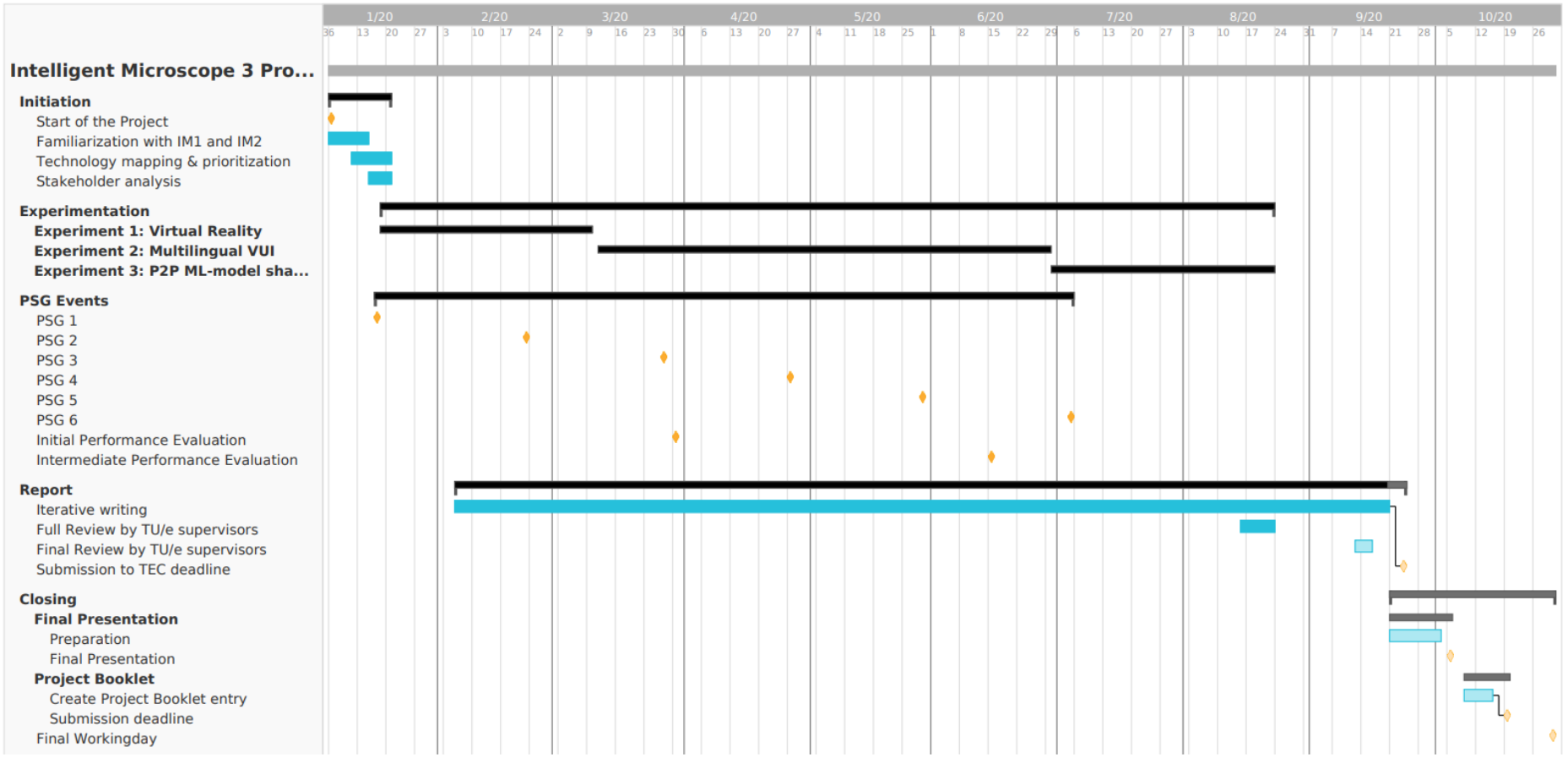

Figure 36: Gantt chart of the activities and deadlines in the project

6.4.4 Tracking Project Deadlines

The most important deadlines related to the project and the report were tracked in Notion[27] using a simple to-do list in a table format. Each entry has a status, deadline date, and a next step. Figure 37 shows the state of the deadline table in August.

Name	Status	Deadline	Deadline Co...	Next step
Submit Initial Performance Evaluation forr	Done	Mar 30, 2020		
Presentation on Come-back day III	Done	May 25, 2020		
Submit Intermediate Performance Evaluati	Done	Jun 15, 2020		
Submit TSP - Training and Supervision Pla	Done	Jul 01, 2020	must be approved (	
Submit Personal Dossier Overview	Done	Jul 01, 2020	With TSP V2	
Submit Report for Intermediate review by	Done	Jul 31, 2020	at least 7 days befo	
Submit Final Presentation Plan	Done	Sep 14, 2020		
Apply for PDEng number	Ready to Do	Sep 21, 2020		Investigate where to apply for t
Submit Deposit Agreement	Ready to Do	Oct 08, 2020	Submitted with the	Fill out the form with "1 year er
Submit Final Report Concept to TEC meml	on hold	Sep 18, 2020	at least 7 *working*	Prepare the PDF version of
Send Data for Diploma and Ceremony	on hold	Sep 21, 2020		Wait for Desiree to send the ex
Final Defense	on hold	Oct 05, 2020		Create first draft of the present
Submit Final Report	on hold	Oct 08, 2020	within 3 days after (	
Submit Article for Project Booklet	on hold	Oct 19, 2020		

Figure 37: Deadlines related to the project and their progress as of August

These deadlines together with the project start and end date were used during project planning as time constraints for the activities in the project plan.

6.5 Risk Analysis

During the project, I identified, documented, and kept track of risks related to the project. Table 14 shows the identified risks along with the probability, impact, and the planned response in case the risk should realize. Risks were discussed during the meetings with supervisors.

Table 14: Project risks

Title	Description	Probability	Impact	Planned Response
Expected low interaction with TU/e Supervisors	Interaction between PDEng Trainee and university supervisors is low due to working in physically separate locations.	Medium	Medium	Propose working from TU/e once a week, providing an opportunity to interact more frequently.
Low interaction due to COVID lockdowns	Interaction with the TFS supervisor is hindered after COVID related lockdowns.	High	Medium	Mitigate: Schedule weekly or bi-weekly recurrent video meetings.

[27] Notion: https://notion.so

Title	Description	Probability	Impact	Planned Response
Uncertainty in the final result	It is an exploratory project, meaning it is not exactly known what will be learned or how will the final result look like.	High	Medium	Accept, but manage the risk: Keep a prioritized list of initiatives and work with the most valuable items.
Negative experiment results	The result of an experiment is negative. For example: Using the selected technology in a chosen way is not feasible.	High	Low	Accept: This is the expected result of an explorative
A TEC member is not able to attend the defense	The supervisor not able to attend the defense	Low	High	Mitigate: Communicate dates early and ensure everyone is on the same page
Final Report not ready for submission	The final report is not ready by the submission deadline.	Medium	High	Mitigate: Create report iteratively: First, focus on covering mandatory sections and then iteratively revise, refine and improve the report
The trainee has no experience in developing VR, VUIs, or P2P systems	All of the technologies are completely new to the trainee.	High	Medium	Accept: Learn as you go – this is an expected part of the project.

6.6 Reflections on project management

At the beginning of this project, I decided to use a highly adaptive project lifecycle with rapid iterations with fixed time and variable scope. This project lifecycle is preferred when the requirements and scope of the product are difficult to define in advance. With fixed-time iterations, I reduced the impact of unknown risks to the project. At the end of each iteration, the current state of the product was demonstrated to the stakeholders.

Because the product that I was building was not well understood in the beginning, I could not have used a predictive lifecycle. In a predictive lifecycle, the deliverables and the schedule are fixed right from the start of the project. This is useful for projects where the risks are known and the product has to be delivered in full to provide value for stakeholders. Luckily, in my project, I could avoid this lifecycle by making sure that I deliver value incrementally.

In retrospect, I find that during this project the actual lifecycle reminded something in between. The lifecycle was iterative, but I did not apply time-boxing consistently. I usually continued pursuing a goal through multiple iterations. For this reason, the actual lifecycle was iterative, but not fixed-time, but rather fixed-scope.

7. Project Retrospective

Without a doubt, the last ten months have been challenging. I have learned and developed a lot both personally and professionally. In this chapter, I present my personal reflections on the process and the work done during the last ten months.

This project was an individual assignment and came with all the benefits and challenges of working alone. Compared to working in a software development team, working alone on a project comes with a wide range of challenges. First, I had to play all the roles in a software development project. Secondly, without a team, there were fewer opportunities for bouncing ideas, discussions, and shared problem solving. Finally, there is the pressure of full responsibility for the project's outcome and the challenge of keeping the motivation up. The latter was especially difficult when dealing with setbacks and negative results. I learned that communication is very important – especially at low points. When working on an individual assignment, it helps to actively involve stakeholders to gain more perspective and support. Frequent communication is also useful in managing their expectations. Communicating slow progress or negative results keeps the expectations realistic and serves as an opportunity to find a solution together.

The challenges of working alone were further accented by the COVID-19 lockdowns that closed the offices and forced me, my stakeholders, and other TFS employees to work from home. This made the threshold for communications higher, as there were fewer chances for simple informal communications. Every connection had to be premeditated either by requesting an online meeting or crafting an email. Having regular, repeating meetings alleviated this issue, providing space for sharing progress and ideas without the overhead of needing to schedule individual meetings. Additionally, the lockdown of the office also meant that there was no more access to VR hardware, prompting me to continue with other topics.

During this project, I had the opportunity to assume a wide range of roles in the software development process. I worked as a project manager, architect, tester, and developer to name a few. Juggling many hats was challenging because I had to prioritize tasks across multiple roles rather than concentrate only on the responsibilities of one role. Nevertheless, this opportunity gave me a chance to explore non-technical roles that I did not have much experience with before. Coming from a software engineering background, taking the project manager role was new and challenging for me. In that role, I developed a project plan, defined and managed deliverables, communicated with stakeholders, and managed their expectations. For me it was the first project where I felt I was fully responsible for planning, communicating, and managing the whole project and not just a technical side of it.

One of the challenging aspects of this project was that it was an exploratory project without pre-defined goals, only an open-ended general vision. On the one hand, this was very liberating as I had the freedom to choose my own path rather than a predefined one. On the other hand, however, with freedom comes great responsibility – It was my responsibility to define the specific goals for the project. Defining the goals was not an easy task and as the project evolved, so did the overall goals.

Furthermore, I found it difficult not to be able to start from specific end user problems. Rather than solving the specific problems users have today, this project was focused on uncovering opportunities that novel technologies might bring in the future. These opportunities might not be immediately applicable to the end users today. As a designer, I am an end-user-centric problem solver. I like to imagine and visualize how the solution is going to be accepted, adopted, and operated by the end user. Visualizing the users' goals and usage scenarios is important to me as it helps me to define the goals and constraints that guide me towards a technical solution. In this project, I could not interview the future user, but I had to envision the problems of a future user and make intuitive assumptions. This left some uncertainty in the design. This uncertainty is acceptable in the project as it is just an exploration, but uncomfortable and challenging for me to work with. All in all, I learned that working at the forefront of technology needs a balance of both ways of thinking.

ABBREVIATIONS

API	Application Programming Interface
ASR	Automatic Speech Recognition
AWS	Amazon Web Services
HMD	Head-Mounted Display
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IM	Intelligent Microscope
IM1, IM2, IM3	Intelligent Microscope I, II, and III (specific projects in the series)
IPFS	InterPlanetary File System
ML	Machine learning
P2P	Peer to Peer
TFS	Thermo Fisher Scientific
TU/e	Eindhoven University of Technology
UI	User Interface
URL	Uniform Resource Locator
VR	Virtual Reality
VUI	Voice User Interface

BIBLIOGRAPHY

[1] A. Neely and J. Hii, "Innovation and business performance: a literature review," 1998.

[2] R. Lafaniere, "Domain analysis," [Online]. Available: http://www.site.uottawa.ca/~laganier/seg2500/domain. [Accessed 29 07 2020].

[3] Thermo Fisher Scientific, "An Introduction to Electron Microscopy - Types of Microscopes," [Online]. Available: https://www.fei.com/introduction-to-electron-microscopy/type. [Accessed 29 07 2020].

[4] T. F. Scientific, "Practical Applications of Electron- and Ion-Beam Microscopy," [Online]. Available: https://www.fei.com/introduction-to-electron-microscopy/applications/. [Accessed 01 09 2020].

[5] US Government, "Wikipedia - SEM image of an ant," [Online]. Available: https://commons.wikimedia.org/wiki/File:Ant_SEM.jpg. [Accessed 10 09 2020].

[6] H. A. Bullock and A. Tamin, "Public Health Image Library (PHIL)," 2020. [Online]. Available: https://phil.cdc.gov/Details.aspx?pid=23354. [Accessed 10 09 2020].

[7] Amazon, "Amazon Lex - Conversational AI for Chatbots," [Online]. Available: https://aws.amazon.com/lex/.

[8] Google, "Google Chrome Privacy Whitepaper," [Online]. Available: https://www.google.com/chrome/privacy/whitepaper.html#speech. [Accessed 28 August 2020].

[9] "Voices in Amazon Polly," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/polly/latest/dg/voicelist.html. [Accessed 02 09 2020].

[10] "Google Cloud Text-to-Speech - Supported voices and languages," [Online]. Available: https://cloud.google.com/text-to-speech/docs/voices. [Accessed 02 09 2020].

[11] "Microsoft Azure - Language and voice support for the Speech service," Microsoft, [Online]. Available: https://docs.microsoft.com/en-us/azure/cognitive-services/speech-service/language-support#text-to-speech. [Accessed 02 09 2020].

[12] "What Is Amazon Translate?," Amazon Web Services, [Online]. Available: https://docs.aws.amazon.com/translate/latest/dg/what-is.html. [Accessed 02 09 2020].

[13] "Language support," Google Cloud, [Online]. Available: https://cloud.google.com/translate/docs/languages. [Accessed 02 09 2020].

[14] "Microsoft Translator - Languages," Microsoft, [Online]. Available: https://www.microsoft.com/en-us/translator/business/languages/#list. [Accessed 02 09 2020].

[15] A. Cockburn, "Hexagonal architecture - Ports and Adapters," 4 January 2005. [Online]. Available: https://alistair.cockburn.us/hexagonal-architecture/. [Accessed 25 08 2020].

[16] J. Palermo, "The Onion Architecture," 29 July 2008. [Online]. Available: https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/. [Accessed 25 08 2020].

[17] AWS, "AWS Lambda quotas," [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html. [Accessed 18 09 2020].

[18] "BCP 47 - Tags for Identifying Languages," Internet Engineering Task Force (IETF), September 2009. [Online]. Available: https://tools.ietf.org/html/bcp47. [Accessed 04 September 2020].

[19] J. Jarmulak, "Speech-to-Text Accuracy Benchmark - June 2020 Results," 10 September 2020. [Online]. Available: https://www.voicegain.ai/post/speech-to-text-accuracy-benchmark-june-2020-results. [Accessed 21 September 21].

[20] J. Bardi, "What is Virtual Reality?," [Online]. Available: https://www.marxentlabs.com/what-is-virtual-reality/.

[21] "VIVE Pro Full Kit | The professional-grade VR headset," HTC, [Online]. Available: https://www.vive.com/eu/product/vive-pro-full-kit/. [Accessed 20 September 2020].

[22] H. Chang and M. F. Cohen, "Panning and Zooming High-Resolution Panoramas in Virtual Reality Devices," in *UIST '17: Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, Québec City QC Canada, 2017.

[23] "Unity Platform," Unity, [Online]. Available: https://unity.com/products/unity-platform.

[24] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural," *Advances in Neural Information Processing Systems,* vol. arXiv:1411.1792 [cs.LG], no. 27, pp. 3320-3328, 2014.

[25] S. Acharya, M. Franklin and S. Zdonik, "Balancing Push and Pull for Data Broadcast," *ACM SIGMOID Record,* vol. 26, 1998.

[26] J. Santell, "P2P Peer Discovery," 14 01 2020. [Online]. Available: https://jsantell.com/p2p-peer-discovery/. [Accessed 09 08 2020].

[27] "TensorFlow - Serving Models," Google, [Online]. Available: https://www.tensorflow.org/tfx/guide/serving.

[28] D. Bush, "UFTP - Encrypted UDP based FTP with multicast," [Online]. Available: http://uftp-multicast.sourceforge.net/. [Accessed 31 07 2020].

[29] PMI, Ed., A Guide to the Project Management Body of Knowledge (PMBOK Guide), 5 ed., Project Management Institute, 2013.

[30] Atlassian, "The iron triangle of planning," [Online]. Available: https://www.atlassian.com/agile/agile-at-scale/agile-iron-triangle. [Accessed 16 09 2020].

[31] JEOL, "JSM-F100 Schottky Field Emission Scanning Electron Microscope," [Online]. Available: https://www.jeol.co.jp/en/products/detail/JSM-F100.html. [Accessed 2020 08 21].

[32] ZEISS, "ZEISS ZEN Intellesis for Image Segmentation in Microscopy," [Online]. Available: https://www.zeiss.com/microscopy/int/products/microscope-software/zen-intellesis-image-segmentation-by-deep-learning.html. [Accessed 2020 08 21].

[33] serverless, "Serverless Framework," [Online]. Available: https://www.serverless.com/.

About the Author

Mark Laane received his bachelor's degree in Computer Engineering from the University of Tartu, Estonia, in 2015. In 2017 he graduated with a Master of Science in Software Engineering from the University of Tartu. His master's thesis, "EyeTal – A Fully Eye-Controlled Map Editor," involved creating an eye-controlled application for editing spatial data. During his master's studies and afterward, he worked in an international software development company Datel as a software engineer.

Aiming to develop his skills further, in 2018, he joined the PDEng Software Technology program at the Eindhoven University of Technology. During the program, he was involved as a developer, a Scrum Master, and an architect in several projects with various industrial partners such as Philips, ASML, and Thermo Fisher Scientific. He has a keen interest in DevOps, lean and agile development practices, cloud and serverless architectures, and full-stack web development.

PO Box 513
5600 MB Eindhoven
The Netherlands
tue.nl

PDEng SOFTWARE TECHNOLOGY

EINDHOVEN
UNIVERSITY OF
TECHNOLOGY