Ing.in Amila Kugic, BSc BSc

# Sample Management System for Fresh Frozen Samples

# Applications in Agile Software Craftsmanship

**MASTER'S THESIS**

to achieve the university degree of

Diplom-Ingenieurin

Master's degree programme:

Bioinformatics and Molecular Bioengineering

submitted to

**Graz University of Technology**

**Supervisor**

Univ.-Prof. Dipl.-Ing. Dr.techn. Wolfgang Slany

Institute of Software Technology

Graz, December 2020

## AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

December 10th, 2020

Date, Signature

# Acknowledgements

Throughout the writing of this thesis, I have received a great deal of support and assistance from the faculty at the Technical University of Graz, my colleges at the Medical University of Graz and from my family and closest friends.

First, I would like to thank my supervisor, Prof. Dr. Wolfgang Slany, whose expertise and knowledge on Agile Methodologies was invaluable in formulating the research topic and questions for this thesis.

I would like to acknowledge my colleagues at the Medical University of Graz for their collaboration and support in making this master thesis possible. Without their input and the opportunities that I gained, while employed at the company, this thesis would have turned out much differently.

Finally, my biggest thank you goes out to my family for their continued love and support. I would not have completed this thesis without you. Your encouragement has been invaluable in these last few months of writing this thesis and completing my master's degree.

Graz, December 10th, 2020  Amila Kugic

Date, Signature

# Abstract

This master thesis focuses on the creation of a fresh frozen sample management system for the biobank at the Medical University of Graz. This goal is inherently connected with finding a hybrid development model that works best in a waterfall-only organization and adapting the model to fit the requirements set by the developer.

In customizing the development model, the individual building blocks of software development models, with a focus on agile methodologies, were examined in regard to their characteristics and their advantages and disadvantages. This knowledge helped in creating a customized model for the development of the sample management system.

The frontend of the main interface was developed as a web-based application in Java and Java Server Pages, which operates by interacting with a custom database created in Oracle. The recording and transfer of sample data sets was implemented by creating an Android app that was installed on multiple tablets.

A hybrid model for a single developer was created, which demonstrated the incompatibility of some agile practices for a single user use case. The whole sample management system has the ability to manage sample and storage data, control the incoming and outgoing samples, retrieve specific datasets and record new sample information.

In applying the customized hybrid model, the productivity and adaptability of the workflow increases using agile methodologies and practices. Consequently, the final software product attained better quality and a longer product lifetime. The final software gives the users the ability to record and manage samples according to internationally known quality standards and thereby increase the worth of each fresh frozen sample being handled by the biobank.



Keywords: biobank, fresh frozen, agile, hybrid model, sample management

# “Kurzfassung”

Diese Masterarbeit befasst sich mit der Erstellung eines Managementsystems für frisch gefrorene Gewebeproben für die Biobank der Medizinischen Universität Graz. Dieses Ziel hängt inhärent mit der Suche nach einem hybriden Entwicklungsmodell zusammen, welches am besten in einem Unternehmen eingesetzt wird, das nur mit dem Wasserfall Modell arbeitet. Die Anpassung des Modells an die vom Entwickler festgelegten Anforderungen sind Teil der Arbeit.

Bei der Anpassung des Entwicklungsmodells wurden die einzelnen Bausteine von Softwareentwicklungsmodellen mit Schwerpunkt auf agilen Methoden auf ihre Eigenschaften sowie ihre Vor- und Nachteile untersucht. Dieses Wissen half bei der Erstellung eines maßgeschneiderten Modells für die Entwicklung des Probenmanagementsystems.

Das Frontend der grafischen Oberfläche wurde als webbasierte Anwendung in Java und Java Server Pages entwickelt, die durch Interaktion mit einer in Oracle erstellten, benutzerdefinierten Datenbank funktioniert. Die Aufzeichnung und Übertragung von Beispieldatensätzen wurden durch Erstellen einer Android-App implementiert, die auf mehreren Tablets installiert wurde.

Es wurde ein Hybridmodell für einen einzelnen Entwickler erstellt, das die Inkompatibilität einiger agiler Praktiken für den Anwendungsfall für einen einzelnen Benutzer demonstrierte. Das gesamte Probenverwaltungssystem kann Proben- und Speicherdaten verwalten, eingehende und ausgehende Proben steuern, bestimmte Datensätze abrufen und neue Probeninformationen aufzeichnen.

Durch die Anwendung des angepassten Hybridmodells steigt die Produktivität und Anpassungsfähigkeit des Arbeitsablaufes mithilfe agiler Methoden und Praktiken. Folglich erreichte das endgültige Softwareprodukt eine bessere Qualität und eine längere Produktlebensdauer. Die endgültige Software gibt den Benutzern die Möglichkeit, Proben gemäß international bekannten Qualitätsstandards aufzuzeichnen und zu verwalten und dadurch den Wert jeder Kryoprobe zu erhöhen, die von der Biobank verarbeitet wird.

Keywords: Biobank, Kryoproben, agil, Hybridmodell, Probenverwaltung

# Table of Contents

# 1 Introduction

This master thesis is implemented for and written in cooperation with the Biobank Graz. This biobank is a central research facility of the Medical University of Graz and a publicly owned non-profit organization, which is part of the research infrastructure of the university. This institution processes, stores and distributes human biological samples and handles the associated data, which is where the heart of this project was established. It is one of the largest biobanks in Europe and deals with a high number of samples each week. The multiple collections of human sample material spanning in some cases over 40 years of sample collections, encompass the following sample types: biological liquid samples, formalin-fixed paraffin embedded tissue, fresh frozen or snap frozen tissue samples and more. The handling and storage of samples is supervised around the clock for quality control and assessment of sample traceability and quality. [1] This master thesis focuses on the creation of a fresh frozen sample management system for this organization.

Fresh frozen samples or also known as snap frozen tissue samples are living tissue samples acquired during a routine procedure or operation from a patient. If the tissue is of interest and the patient has given their informed consent, i.e. approval for the scientific use of the biological material, the sample is stored in a barcoded tube and frozen to temperatures around -150 to -160 degrees Celsius. This way of preserving the tissue of biological specimen would make it possible to further evaluate the tissue and the DNA of the sample at a later date, if a study needed to answer a specific research question.

The origin of collecting samples has its roots in pathology. Ancient medicine relied on the dissection of human cadavers to understand the human body and be able to crudely fix, what was broken. Not many know that pathology dates back to Ancient Greeks, where the first scientific human cadaver dissections took place. [2]

As medicine and technology advanced, so did pathology. It played a huge role in bringing medicine to the point, where it is possible to personalize treatments of diseases and adjust the focus of the treatment onto the patient and not the disease, which is called personalized medicine. An existing example for this is breast cancer, where physicians can evaluate, based on existing medical research, which treatment options would serve the patient best depending on their preexistent conditions and medical history.

"*Many medical advances, including studies of heart disease, AIDS and cancer, have resulted from preliminary developmental studies that have relied on access to and proper use of the appropriate biospecimens.* […] *For molecular epidemiology studies, the ultimate success of a study depends on reliable laboratory analyses of these specimens. In order for laboratory analyses to be reliable, the*

*collection, processing and storage of specimens must be performed under strictly controlled procedures* […] *to assure that biospecimens are of the highest quality*." [3]

This shows that biobanks are essential for furthering the research to make personalized medicine possible, because without the high-quality samples and standardized storage conditions, the use of samples and their value would greatly diminish.

Consequently, samples are one of the key ingredients in doing medical research. Biological samples are commonly referred to as biological specimens or biospecimens and come from various biological sources from the human body. Storing conditions differ depending on the material of the specimen. The material can be any tissue or bodily fluid from the human body, which is then stored for research purposes. Storing samples is in most cases strictly regulated by quality standards and biobanks adhere to these standards, to be able to deliver consistent quality throughout the sample collection.

The consistent quality throughout the sample collection does not start with only one organization adhering to the quality standards. The sample acquisition and handling is a process that starts, when the sample or the blood is collected and ends, when the sample reaches its final storage place. During this time, in this preanalytical phase, any changes or deviations in the handling of the sample can affect the lab values during the evaluation of the samples. Not only is a consistent quality necessary, but standardized documentation is part of that. Cornes et al. states that "*the preanalytical phase is the part most prone to errors throughout the total testing process, a close inspection and continuous evaluation of this phase is essential to produce high quality test results. Our survey shows that among the 1347 participating European responders analysing blood samples, nearly all (94%) do monitor or document preanalytical errors.*". [4]

Globally, the act of using animals for testing of substances and medical remedies is still in use. Based on that, worldwide there are biobanks entirely dedicated to animal testing and research of diseases using animals, where samples from the animals are taken and stored for further testing and to answer future research questions. The biobank at the medical university in Graz does not house any animal samples, because the entire sample collection consists only of human sample material.

The focus of this master thesis is deals with the sample collection of fresh frozen samples. As the name already indicates, it refers to the act of freezing fresh tissue to extremely low temperatures to stop the decay of the sample. Tissue samples are primarily gained through procedures like biopsies, surgeries, transplantations and autopsies.

The tissue should be frozen or fixed as rapidly as possible after tissue collection to avoid morphological distortions and damage due to tissue drying artifacts, autolysis and putrefaction. The ideal time limit for the complete freezing process should take about half an hour. The tissue needs to be snap frozen,

so that the sample stays in peak condition and does not change its morphological structure during the freezing process. If the tissue is frozen slowly, ice crystals form, which changes the histological structure of the sample and makes it useless to pathologists or other researchers. [5]

The fresh tissue sample at the biobank is sectioned into appropriately sized pieces and submerged into precooled methyl butane, which has a temperature at around -160°C. The fresh tissue piece freezes instantly and the frozen sample piece is then transferred into a barcoded tube that is stored in a box in a dewar filled with liquid nitrogen. At the end of the day, the barcoded tubes are transferred into the cryo tanks that are filled as well with liquid nitrogen.

Conversely, should the sample come from a surgery, e.g. an oncology surgery, the surgeon sends the cancerous piece of tissue to pathology, where one part of the sample is treated a bit differently, i.e. it undergoes a frozen sectioning procedure, and based on the findings and if the sample is of interest, biobank employees are called for sample collection.

"*The frozen section procedure is a pathological laboratory procedure to perform rapid microscopic analysis of a specimen*. […] *The intraoperative consultation is the name given to the whole intervention by the pathologist, which includes not only frozen section but also gross evaluation of the specimen, examination of cytology preparations taken on the specimen (e.g. touch imprints), and aliquoting of the specimen for special studies (e.g. molecular pathology techniques, flow cytometry). The report given by the pathologist is usually limited to a "benign" or "malignant" diagnosis, and communicated to the surgeon operating via intercom. When operating on a previously confirmed malignancy, the main purpose of the pathologist is to inform the surgeon if the resection margin is clear of residual cancer, or if residual cancer is present at the resection margin.* […] *The key instrument for cryosection is the cryostat, which is essentially a microtome inside a freezer.* […] *The usual histology slice is cut at 5 to 10 micrometres. The surgical specimen is placed on a metal tissue disc which is then secured in a chuck and frozen rapidly to about –20 to –30 °C. The specimen is embedded in a gel like medium called OCT and consisting of poly ethylene glycol and polyvinyl alcohol; this compound is known by many names and when frozen has the same density as frozen tissue. At this temperature, most tissues become rock-hard. Usually a lower temperature is required for fat or lipid rich tissue. Each tissue has a preferred temperature for processing.*" [6]

When the sample is processed and reaches the final storage placement, the collection procedure is completed. One aspect of the collection procedure has not been explained, which is the documentation of the sample data and the processing steps during the collection. The documentation is predominantly done with a sample management system that is in use at the biobank.

In the industry, fresh frozen sample management has been implemented many times in the past years and is being offered as a software package by some biobanking companies. The features would include corresponding data management and sample tracking in some capacity. Though it is possible to purchase and use an already existing and implemented software solution for such a task, the missing adaptability and performance inconsistencies with higher amounts of datasets are key in being able to use the software effectively. Having a full understanding of the data management aspect of the software package, i.e. the procedures and functions necessary for the processing of the data in the backend, is essential due to the nature of biobanks and the management of patient data. This means the biobank should be able to say with certainty that no third party applications can access sensitive data sets of any kind, which is why the implementation of software needs special precautions.

Biobanks with largely uniform data sets or similar sample types would find that using excel files for sample tracking and data management sufficient for their use cases. Rising amounts of sample collections and different kinds of sample data needing to be recorded, results in the biobank employees needing technological assistance in recording and managing sample data.

A custom solution for the tracking problem of fresh frozen samples is necessary. The functions need to fit and be adaptable to the ever-evolving nature of different biobanking workflows and the changing requirements needed to make new projects and sample acquisition possible.

Sample management systems might be the solution to that specific problem. These systems are not new to the industry and have been used all around the world for managing samples and keeping track of them for storage, data and event tracking purposes.

Although the final product or functions of sample management systems are interpreted differently by each of the companies, which produce and develop the systems, there needs to be a common understanding of what sample management systems are and what features such a system should have.

In this field of sample management, companies contribute and develop hardware and software solutions for these types of systems. The system itself needs to be in some way connected to or become a Laboratory Information Management System (LIMS). The latter is defined as "[…] *a software system developed to support laboratory operations. This software system can track specimens and workflows, aggregate data for research or business intelligence purposes, and ensure laboratory operations are compliant with various standards and regulations. It is important to distinguish between a conventional laboratory information management system (LIMS), which is often used in the research laboratory, and something similar called a laboratory information system (LIS).* […]

*By contrast, a LIMS is often the best software for dealing with the kinds of bulk data that a research laboratory operates on. For example, a LIMS can analyze masses of collected outcomes regarding the*

*efficacy of a drug or chemical product, or screen repetitive tasks for daily operations. As mentioned, these tools can also analyze and support workflow, for example, by providing essential testing tools for every stage of the workflow process or by promoting the consistent use of chemicals and physical products in the laboratory for a kind of electronic quality control process.*" [7]

Gibbon explained in a brief history of LIMS, how the discipline of these systems evolved from 1973 to 1995, and that automation of laboratory work truly started to evolve during that time. [8]

Modern day LIMS deal with samples, automated testing, assigning and planning of tests, managing the results of sample testing, how to register the results and print the reports, data analytics and much more.

The difference between a LIMS and a sample management software lies within their use, where the focus of LIMS is to support laboratory operations in the processing and analyzing of samples. The focus for the sample management system relies on the sample tracking and data management capabilities. The assumption that sample management software would be easier to handle and have less data to manage might be true, though depends heavily on the type and number of samples being managed.

To give a more specialized view of the capabilities of LIMS or similar sample management systems, the more prevalent examples and their companies are going to be discussed here in this section in no preferential order.

The first organization to be discussed is Micronic. The company has its headquarters in the Netherlands and focus on advancing scientific research to enhance quality of life, and the product line revolves around labware, precision engineering equipment and sample preservation. Micronic does list sample management systems according to their website and catalogue, which are defined as systems that accelerate the sample storage workflow and include two tube handlers and one manual tube selector. The electronic systems mentioned primarily sort and pick sample specimens from a large group of samples and do this based on picking files, which are loaded into the system by an external drive. [9], [10]

On the software side, the company has had a sample tracking software called Track IT available, a so-called Laboratory Sample Management System, which features included a database system, alerts, cross database searches, data exports and administration tasks available, though has been discontinued since then and replaced by another software called Tracxer, which offers barcode scanning and tube management capabilities, which exports the data as files to be used in conjunction with the Micronic Scanner. [11]

The next company offering sample management software and similar lab tools is FreeLIMS, who offer a LIMS software for biobanks and biorepositories, which is their software solution to manage

biobanking operations. According to their website, the software features include sample management, management of patient information, storage location in a customizable and hierarchical storage pattern, plus test management of various lab activities. [12]

A further example of LIMS application is Illumina, which is a company based in the USA and deal with genomic data, and their processes are geared towards labs, which are designed for the genomics laboratories and next-generation sequencing studies. [13]

Sample management systems fulfill their purpose by having features and functions like tracking of samples, depicting storage capabilities and storage management, searching for sample data and various key figures for querying data. Furthermore, the import and export of the data should function with automatic quality checks and quality measurements. The quality standards, which are fulfilled by such systems, depend on the documentation of sample management and how the systems support the employees in the day-to-day activities.

Being equipped with these features makes a system functional and usable for a biobank. A step up from there would be the inclusion and support of all the already existent storage hardware of the biobank (e.g. tubes, racks, etc.). Similarly, the access to the systems backend should be accessible to the IT team on site to be able to manage minor errors.

Therefore, the final workflow should adapt and be influenced by the workflow of the employees, and not the other way around, in which the system predefines the workflow and the available tasks and tools for the users, which lacks adaptability. This type of adaptability is an appreciated feature in sample management industry.

In analyzing the given LIMS examples, each company establishes software solutions based on their datasets, workflow and preferences, which is what this master thesis is similarly going to accomplish. The development of the sample management system is accomplished using a custom methodology. To understand the reason for this decision, a general overview of software development is given.

Software development has changed in the past few decades. Beginning in the year 1990, waterfall models were widely used to give software projects structure and a general guideline for the developers and project managers. In the mid-1990's agile methodologies began to appear and in 2001, the scene fundamentally changed, and a new way of project management and software development was created in establishing the Agile Manifesto. [14]

From that point onward, many software developers and companies had the idea to adapt waterfall models and add agile practices of project management and software development to fit their personal needs and to solve problems they have been facing. This resulted in not only many new ways of dealing with project management issues, but conversely had the effect of only researching parts of the

manifesto and how these affect the project and software development life cycle. Since then, each of the created methodologies have matured and become known for their effectiveness. [15] More than 80% of participants in a survey stated that hybrid development approaches emerged from experience, which evolved gradually over the past almost two decades. [16]

Even now, many companies will still use waterfall models as development practices, just because companies and teams don't have the resources to make the switch and fully operate on agile practices and uphold them to a degree that is necessary. The mentioned resources include the personnel, time, know-how and the ability to adapt the workflow to their needs.

Development of new software projects using the waterfall model has established itself in the agile software communities, as a somewhat rigid model with too much documentation attached to the development of code.

Conversely, this concept has the benefit that various stakeholders and users of this software need to know the specific tasks and requirements, before these tasks and new projects reach the research and development team. On the one hand, the documentation of the requirements and the different specifications can only be of an advantage for any development of a project in its beginning stages. These steps bring clarity and the ablity to make sure that nothing has been forgotten in the development life cycle. On the other hand, the documentation is going to take up a lot of time, specifically if there are unexpected delays, which lengthens the software development life cycle considerably, if the model is followed to the letter.

The drawbacks include that only seldomly does a project follow all the steps as planned, as either the requirements change abruptly and the developer needs to start from scratch all over again, or the documentation is not finalized as planned, as other projects and priorities are more important than following the specific steps of the waterfall model. [17]

Some research suggests that using a hybrid waterfall model would be a possible remedy for this problem, in which it reduces the documentation to the essentials and allows for quicker development cycles and better turnaround in deliverables (code deliverables). According to a survey in 2016, about 70% of development engineers are satisfied with the agile-waterfall hybrid model and the methodologies themselves. [18]

Additionally, these drawbacks could also be remedied by using a purely agile based methodology. However, in software development there is no one-size-fits-all methodology. Every project might need a slightly different handling, though which company can afford to adjust and adapt the workflow each time a new project is started. Changing the methodology fundamentally or even just slightly can be a considerable effort for a company.

Nafchi et al. explained that it is not just an adaptation of a workflow, and it is much closer to a migration of tools and the software company needs to create a new workflow. [19] This is not feasible for various companies from a management point of view. Furthermore, the authors illustrated this by attempting to quantify using different scopes and techniques the amount of work needed for the migration of the workflows. Due to the nature of the methods and agile practices, it is not a straightforward process. Each of the assessment tools try to appraise slightly different metrics in regard to flexibility, speed, leanness, objective principles and practices, agile practices, or even comparing the agility of organizations themselves. Nafchi et al. conclude that there still is a gap in being able to assess the agility of software companies, and that further research is needed to find or establish a different assessment model, which does not have the drawbacks of the existing ones.

Similarly, Anes et al. illustrates how agile methodologies have a wide range of applications in the industry, which has shown shortcomings in estimating and managing the risks in agile processes. This paper further introduces a new approach to remedy those shortcomings by using qualitative risk tools without compromising the flexibility of agile practices. [20] These so-called agile hazards are scope creep, unrealistic expectations, lack of cooperation and lack of communication. The scope creep describes the constant accommodation and requirements change, which results in continuous and uncontrolled growth of a project. Unrealistic expectations occur due to the different visions of what the customers or users want, without being able to gauge if those visions can be implemented. Lack of cooperation and lack of communication allude to there being factors, which inhibit the ability of the teams to perfectly work well together, be it due to health or social issues between team members. By employing various practices from fuzzy logic architecture to rules, the set of agile sprints can be adjusted to be able to do qualitative risk control. Such hazards, if not managed properly, would result in bigger concerns for the teams and their companies in managing the projects than working with a standard waterfall model.

All these papers highlight the difficulty in choosing one ideal workflow for a company or project group. Choosing the workflow that fits the team best is the main objective, because doing anything other than that would lead to missed deadlines or an unproductive working environment. Still, as mentioned by Nafchi et al. there is a significant barrier to entry connected to agile workflows in having to change the structure of the teams, team roles and how the whole company in the upper levels react to specific changes in the teams.

In 2011, a paper by Sletholt et al. further explained that there is a different approach to development in a scientific setting. The main goal for scientific software engineers is to perform science and not to write software, which makes this field of study much more adaptable to workflow changes as only the things that work for the engineers are used and other practices are disregarded. [21]

Based on those facts, the idea of following an agile methodology fully in every aspect is a choice and not a must have for every company. Taking this into consideration, this thesis will explore adapting a waterfall only organization and including more agile methodologies into the workflow.

To show how this is done, a new sample management system for fresh frozen samples will be implemented using the customized model for the management and development of this project. The code itself should follow the rules and guidelines of the book "*Clean Code*" by Robert C. Martin [22], which is as the subtitle suggests a handbook of Agile Software Craftsmanship, as well as using another Robert C. Martin book called "*Agile Software Development – Principles, Patterns, and Practices*" [23].

Agile Software Craftsmanship is a field encompassing the entirety of programming, no matter the language, and teaches practices and uses in real life, so that programming turns into a craft. The code that is written for this master thesis used these practices and applications throughout the given project.

The book "Clean Code" explains the craft of programming through another lens and using practical examples shows how to code in a clean and minimal way that benefits the company and the programmer coding the project. The practices in the book range from understanding what makes code good or bad, naming conventions, the attractiveness of small and compact functions, what role comments play in coding and much more. In Chapter 2 Methodologies, this topic will be explained in more detail and how it changed the project and its scope.

These books create the knowledge base from which the new methodology and sample management system are designed and adapted. To give broad overview of the management of samples before the start of this project, the structure and software applications of the preexisting system are described in the next paragraphs.

There are about 37,000 fresh frozen samples in specialized tubes that are stored in cryo tanks at the Biobank Graz. The data is stored in a software application called BioSamplePro, which was an existing cataloguing software adapted for the biobank by the Styrian company Joanneum Research.

The database design used by this software is very scattered and at scale inefficient. The database is implemented and programmed in Oracle.

The problem of the legacy software is that the use for this database is wide and varied and made to be as wide footed as possible. Through adapting the software and the database multiple times, the run time of the application suffered, in conjunction with the fact, that the system was not developed to handle massive amounts of data, which is needed in the case of sample management and storage infrastructure. Part of the reason for this is faulty sample management system is database design and

too much data for such a relationship entity model to handle, where a multitude of tables and their relationships are not able to be useful.

Another reason for building a new system to house the fresh frozen data sets is existing inconsistencies in the data sets themselves. The recording of new data sets occurred using mobile computers that included barcode scanners from the company Honeywell, previously Hand Held Products Inc. The mentioned company was an electronics manufacturer based in the United States, established in 1981, and focused on designing, developing and manufacturing image-based data collection systems. Additionally, it offered barcode scanner, mobile computers, vehicle mount computers and wireless infrastructure. At the end of 2007, the company Hand Held Products Inc. was bought up by Honeywell, and became part of their Honeywell Scanning and Mobility lineup. [24], [25], [26]

This hardware system consisted of multiple Honeywell Dolphin Handhelds, which are digital input devices that includes a touchscreen with keypad that operated on Windows ME CE Mobile.

The utilization of handheld products at the biobank created numerous benefits for the employees and the data acquisition using those products, but it also had its downsides. Through using that hardware the accuracy and correctness of data declined in the following years, based on data loss and input freedom by the users – in being able to input all kinds of data into the handheld device, without any input checks or restrictions. This user freedom was also given on the graphical user interface of BioSamplePro and changes could be made to all the data sets at any given time without recording the changes, who made those changes and the validity of the data being inserted into that database.

The legacy hardware system included a desktop software that would connect to the computer using Windows Mobile Device Center and adjust the settings of the Handheld of Programs and Services, send and upgrade Pictures, Music and Videos, exchange files with the device and set Mobile Device Settings. After the connection to the computer is established, another software is started to upgrade the custom software on the device and transfer the datasets from the hardware to an exchange folder on the server from the company. BioSamplePro would check the exchange folder every five minutes to see if new datasets could be found in the folder to import it into the database.

The preexisting system was convoluted and riddled with flaws from start to finish, and on top of that in some cases it did not even transfer all the datasets to the database. This coupled with the hardware problems mentioned above does not evoke much confidence in the continued workings of legacy system. Maintaining the existing system took a lot of work to keep it working smoothly, which resulted in the decision to create a completely new sample management system for the fresh frozen samples.

The goal of this project is to export the data from this legacy software packet and import it into a new custom database, which will manage all the information, as well as build a new graphic user interface to facilitate new possibilities in searching and working with the given data.

During this project, the legacy hardware to record sample datasets and additional information is replaced by tablets running Android Marshmallow (Android 6). The reason for the switch in technologies would mean more stability and better lifetime warranty for the biobank, which would allow users not to lose data sets, if the battery of the hardware gave out, be able to upgrade the software at a quicker pace and reduce unwanted costs. Comparatively, the tablets cost one tenth the price of a Handheld device, and added to that, the latter have long since been discontinued, which meant that an upgrade in technologies at that point was unavoidable.

This system should encompass a new graphic user interface that can be accessed using a computer. This would act as a management console for each employee at the biobank for fresh frozen samples. The tablets running a custom programmed app would record all sample and patient data and transfer that data to the database in the background, which has been designed from the ground up.

The functions for the graphic user interface comprises the ability to efficiently search for samples, to mark individual samples for a specific project, mark them as sent out and no longer in stock, check the storage of fresh frozen samples and find free spaces in the storage tanks. Additionally, the software allows users to query the database for frequently needed key figures, manage existing cohorts and list sample events for every single tube in the database.

Starting out the project, the applied methodology for the development of software exclusively consisted of the waterfall model. This model has its advantages and disadvantages, which will be covered in the Methodology section of this master thesis extensively. Software engineering may hold an answer in improving the process of developing new code, which is why the waterfall model got adapted for the development of a new sample management system.

Coming up in the next pages, in chapter 2 Methodology this thesis outlines the needed resources for this endeavor, showcases the main difficulties in switching from waterfall only to a more hybrid approach, allows for understanding the science behind the concepts of agile, and reiterates why such a custom model is a better fit for this company. Chapter 3 Results describes the resulting final method used to develop this project, the inner workings of all the parts of the project and how agile practices helped in the development of the final software for the sample management system. Chapter 4 Critical Reflection takes a deeper look at the structure of the project, the implementation of each of the features and points out, what could have been implemented better, as well as evaluates the project and the methodologies used in this project. Moreover, flaws or shortcomings of the methodologies

are pointed out and upgrades are discussed, which would make sense in the next development life cycle. Finally, in Chapter 5 Conclusion and Future Work, the final thoughts and next steps regarding this project are examined and noted.

# 2 Methodology

This chapter gives an overview of the basics in agile software craftsmanship and explains the idea behind the approach of finding a suitable hybrid methodology for the workflow at that company. Added to that, the parameters for choosing different agile processes are defined, as well as their requirements and what kind of impact that would have on the software development life cycle. On the more practical side, the methodologies for programming the software are analyzed, which includes the benefits of clean code and the changes it brings to the development itself.

## 2.1 The Beginning of Software Development

Before the 1950s, the term programming meant restructuring or rewiring hardware systems and did not have the same connotations that it has in the 2020s. Software development or programming at that point in time were not common or known about. The primary focus of that time period lied in the construction of hardware and how to make the production better and cheaper.

In the time period between the 1950s and 1960s, the first steps towards structured programming were taken, which had marginal improvements in time and development resources. This could be the time in which software engineering originated or was created. [27]

At the end of the 1960s, the so-called “software crisis” started. This term referred to a period, where the hardware costs had fallen, and the software development costs were at an all-time high. This phenomenon of software crisis showed itself in incomplete projects, time management issues, not being able to meet deadlines and to finalize projects. The maintenance and upkeep of software created problems, which could not be solved without falling behind in schedule. [28], [29]

## 2.2 Waterfall Model: The Reliable Choice

In the beginning of the 1970s, the first solutions to the problems seen in the 1960s were created. The waterfall model established itself as the gold standard of software craftsmanship, which still exists in the 2020s. The model is applied for smaller and standalone projects.

This software development model is a sequential model, which was introduced in 1970 by Winston Royce. It is the most basic of software development life cycles and separates the development of new software into different phases. Each phase of a project equals a specific part of a project, i.e. a specific task. This model consists of the following phases: Requirement Analysis, System Design, Implementation, System Testing, System Development and System Maintenance.

The Requirement Analysis lists all the necessary conditions that the software needs to meet, when delivering the final product. This should include all information available at this time. One way to accomplish this would be to write up a User Requirements Document, which lists all the conditions, wants and needs for this specific product and includes a rough estimate of the number of hours needed

to accomplish this task. This document will also include a list of use cases, which describe the way the software will be used in the future.

The System Design phase includes all the planning that needs to happen. Here all the data management and processing steps are written down in detail to make sure all the eventualities and problems have been addressed. The design element of this phase establishes all the interconnecting points and tasks of the software and draws or explains these in detail. At the end of this phase, the project or software needs to be fully planned and designed from the ground up.

The Implementation phase builds upon the already existing information of the previous phases and has to implement everything that has been planned in the System Design phase. Depending on the quality of work in the previous phase, the implementation phase is most likely a straightforward task of implementing and finalizing all the given tasks previously.

During the System Testing phase, the product should be tested for all the given use cases listed in the Requirements Analysis, as well as tested for any common user input mistakes, so that it can be determined whether the error handling has been implemented correctly.

The System Development phase includes the software being deployed and users using that software for the very first time and any small changes to the graphical user interface, if it can be accomplished in a short period of time.

The System Maintenance deals with any errors and working system failures of the software. This means dealing with false data sets, false inputs and workflow errors that do not deliver the data as promised.

These phases and this development model helped in completing the projects and meeting deadlines, though there cannot be a one solution fits all development model. The disadvantages of this model lie within having to have a rigid structure in place, before starting the development of the project. Should requirements change, this would mean going back to the drawing board and starting from anew, which does not sit well in most cases with managers, who require quick turnarounds, if possible.

## 2.3 Iterative, Incremental and Prototyping: The New Normal

From the 1970s until the end of the 1980s, newer and better solutions were created, which seem to work for project management and the development of new software. The practice of prototyping gained hold, which consisted of the developers showing users and other stakeholders unfinished and incomplete software to gain insight and essential feedback for the final product.

Other methodologies were adapted or changed slightly to fit the ideals of someone else. Examples for that being the spiral model that creates specific custom models based on the project itself, or a V model, which is a custom approach to the waterfall model.

## 2.4 A New Era: Agile Methodologies

From the 1990s onward, agile software practices were introduced in software development. These practices helped to bridge the gap between waterfall model and agile methodologies. The landscape of software development changed fundamentally in 2001, when the Agile Manifesto was created by Robert C. Martin and colleges. [14], [30]

Due to the internet and rapid development, the practices gained in traction. The ability to communicate and work closely together with people from all over the world marked an increase and a rapid growth in agile methodologies.

By not following the waterfall model and using agile practices for developing new software, specific parts of the development and project management are enhanced to fit the developers need. A few of the agile methodologies will be explained in the following subsections.

### 2.4.1 Scrum

This methodology is based on a lightweight agile project management framework. In using it, the projects are done incrementally and the product owner deals with the product features that need to be implemented, i.e. features, bug fixes, etc.

### 2.4.2 Lean and Kanban Software Development

This is methodology is split into two parts – the Lean Software Development and the Kanban Methodology. First and foremost, it is an iterative agile methodology, which was developed by Mary and Tom Poppendick. The main principles include eliminating waste, amplifying learning, deciding as late as possible, delivering as fast as possible, empowering the team, building integrity and seeing the whole. Kanban Methodology goes hand in hand with the Lean Software Development, and is a process, which is designed to help teams work together more effectively. The three basic principles of Kanban Methodology are to visualize what you do today (the workflow), to limit the amount of work in progress and to enhance the flow. [31]

### 2.4.3 Extreme Programming (XP)

According to R. C. Martin, Extreme Programming consists of a set of simple and concrete practices that combine into an agile development process. These practices encompass the customers becoming an integral part of the development, user stories, short development life cycles, acceptance tests, pair programming, test driven development, collective ownership and continuous integration of the code, planning, design and refactoring. All these parts work together to make agile development possible.

## 2.5 Advantages and Disadvantages of Agile Practices

Before being able to design a new software development methodology, one must have a solid foundation of knowledge of the basics, and this includes the understanding of what the advantages and disadvantages of using agile practices are. In this section, the agile practices from different methodologies are pooled together, primarily from XP practices, and their advantages and disadvantages are discussed.

### 2.5.1 Customers Becoming Team Members

The customers, i.e. the person or group that prioritizes features for the development of a project, are important assets to the developer. As required in the XP practices, they work closely with the developer and give feedback on the prioritization of specific tasks. The advantage lies within being able to distinguish, which features are needed for the day-to-day activities and having the ability to move up the timetable for specific tasks, if needed. The disadvantage is the knowledge gap in the understanding of the technical side of the software by the customers.

### 2.5.2 User Stories

Without the customers or users of the software, the developer cannot have the practical knowledge for the implementation of the software. In section 2.2 the user requirements document was mentioned. This is heavily influenced by the user stories and how the users would operate the software and is part of the initial documentation of the project. The use cases for this project are noted, and using these the developer can gauge, if the software project is completed or not. The customers rely on the capabilities of the developer to differentiate between features that can be implemented, and features that lie outside the realm of possibility.

### 2.5.3 Short Development Life Cycles

To understand why the development of software should have short cycles, one must understand the what software development life cycles (SDLC) represent. In general terms, the cycles represent the phases that a software goes through from its creation or inception, through development, testing and maintenance, and finally to its end of life. Between the inception and its end of life, there should be many iterations and smaller changes done to the software, due to the environment changing and needing to adapt to the rapidly changing requirements as well. The short life cycles represent the quick adaptation of software and making new features available to the users, i.e. a quick turnaround. The longer the life cycle becomes, the more stagnant and unusable a software gets, when it does not adapt to the changing requirements.

In XP, the life cycle length is set to an exemplary two weeks. This might not be the case for each company, and this project had a one to two-month life cycle length, depending on the features being implemented.

### 2.5.4 Acceptance Tests

These tests are automatically and repeatably runnable scripts, and their purpose is to test the requirements of the user. The tests are written in a scripting language of choice depending on the programming language used and give feedback on whether the functions are in working order.

The advantages to acceptance tests are to be able to consecutively conduct these tests and know when a build or test fails. This feedback is of importance, when more than one user is working on the development of the project. This makes error handling easier to handle and makes sure that the developer doesn't set the systems accessible to the users with the errors in place.

That practice makes it possible that a system is not offline for more than a few hours. It makes the maintenance and adding of new systems accessible and does not interfere with the development of new features.

The disadvantages of such tests are the primarily the added work for the programmers, i.e. the time spent on programming tests. Missing resources in some teams leaves that team no option but to leave out the systems tests or acceptance tests.

### 2.5.5 Pair Programming

As the name suggests, pair programming is the practice of programming being done by two developers together and simultaneously on one computer. One of the developers would be responsible to write the code and the other developer would sit beside the one programming, and looking for problems, errors and so on. It requires intensive collaboration on the part of the developers.

The advantages of two developers working side-by-side are the reduction of errors, as these are more easily spotted by more than one person. Furthermore, the collaboration increases the knowledge in the team regarding the running systems, as well as the craft of programming itself. Both programmers would benefit from those advantages through the roles being exchanged frequently.

The disadvantages of this method are having to have enough resources in the team to make this practice possible. The resources mentioned not only include programmers, but time and project allocation to help in programming of another software. Another disadvantage is that this practice is only effective, if both programmers are at a similar knowledge level, because drastic differences in knowledge would be a disadvantage for the other programmer.

### 2.5.6 Test Driven Development

This development style is influenced by and created for XP. Here the development of new software and the testing of said software go hand in hand. The procedure consists of writing a test, which is surely going to fail due to the missing functionality, and then implementing the function or feature corresponding to the test, so that the test succeeds. The general structure of test-driven development

is setup, execution, validation and cleanup. The setup refers to the writing of the test, the execution encompasses the implementation of the missing function, the validation is to make sure, if the test works and finally, the cleanup phase indicates the cleanup of the code.

The advantages of the test-driven development are the existent test cases at the end of development to test the required functions needed of the software and makes the checking of the functionality an automatic process.

The disadvantages of test-driven development are very similar to the disadvantages of acceptance tests. Added to that the mentality of programming in such a way is completely different, where the tests are at the forefront of the methodology and the functions are secondary to the functionality. Plus, this is only applicable for function-based programming. If the infrastructure or something similar is being programmed, this kind of development is of no use.

### 2.5.7 Collective Ownership

This practice describes the code being a shared resource of the development team, and each of the developers can check out the code and improve the code. Nobody has a monopoly or a sole ownership of the code.

The advantage of such a practice means that the developers themselves are not confined to one project or task. They can work on other tasks and implement and improve existing software for the users or help other developers out, which creates new collaborations between the team members.

The disadvantages, like many others, require employee and management resources. Furthermore, if tools like Github or a similar code tracking and version control tool is not used, many times it is quite the problem understanding, which tasks still need to be done, and what of the tasks are finalized. That would lead to inconsistencies and misunderstandings should the documentation of the project be insufficient, which are hurdles for developers, who want to adapt such practices in their own teams.

### 2.5.8 Continuous Integration

Continuous Integration refers to the action of checking out code from version control, plus merging it with software from other developers many times a day. The prerequisites for this would be having at least two employees working on the same project and having access to a version control tool.

The advantages of this practice are the constantly up -to-date software, the developers being at least passively familiar with almost all parts of the software and having one functioning version of the code available at all times, which is buildable and useable in a productive setting.

The disadvantages of continuous integration are especially taxing for resource management. The needed server resources, testing and development environments need to be very similar for all the team members, as well as team resources. Another disadvantage is the constant merging with the

code, which can create further problems or errors, in merging with the code of other developers. Dealing with those challenges can expand some development life cycles due to merging issues.

### 2.5.9 Planning

The general practice of planning a project at the beginning is at the core of every project. In XP, planning refers to the project managers and developers leading a discussion between business management and development sector. The decision is then made, whether the cost of developing the further features or of a new project fulfill the need, and when this step of starting that project should be taken. The business management looks at the need vs. the costs of the project and decides the next steps based on those factors.

The advantages of this variant of XP planning is to get the customers, developers and managers on the same page regarding the importance and costs of implementing that specific project. The development team further benefits from the planning, in which the deadlines, milestones and budget questions are preliminarily answered.

The down sides of this variant of XP planning means taking the management of the programming out of the developers' hands and deciding project relevant questions without the needed technical knowledge, only sometimes based on the cost of the development itself.

In some companies, the costs become a driving factor and simple solutions are prioritized, which leads to developers not being able to make the systems expandable or adaptable, on the grounds that the complicated designs of software do not get the approval of the management.

### 2.5.10 Refactoring

The action of refactoring describes the constantly making existing code better and frequently improving the code without affecting the function or its behavior in any way.

The advantage of refactoring means improving the code and the usability of the code regularly. This corresponds to a programmer never leaving code worse behind than at the beginning of a work session.

The disadvantages are without refactoring, the code would go stagnant and end up unusable. Added to that, should somebody refactor the existing code, there is always the risk of the developer introducing errors, which means having to work on error handling and software maintenance.

## 2.6 The Adaptation Process: Incorporating Agile Methods

During the Introduction it was established that a hybrid approach will be used for the implementation of the practical part of the thesis. This is accomplished by starting out with the waterfall model and changing the methodologies slowly to fit the needs of the project and the company. The adaptation of the workflow is a very gradual adaptation process.

Starting out the project, the first steps of documentation and workflow stayed the same. This included writing the user requirements document, which was essential in getting the project approved by the company. Listing all the necessary functionality and changes, which needed to happen to create a completely new sample management system, planning out the separate tasks and their priorities, and already establishing order of operations. By knowing approximately how much work is necessary to accomplish such tasks, the scope of the project can be measured and based on that the planning of the project is easier to handle.

Another key figure is the number of members of the project team working simultaneously on that project. In this instance, the author of this thesis is the project leader of the project and the only member of the software development team of the biobank working on the project from start to finish. Due to personnel and management reasons, all the programming and planning required is being performed by one person.

Recognizing the need for quicker iteration in the development of the functions, the workflow was adapted to reflect these changes. During this phase, the users concurrently gain an understanding of what is and is not possible in the adaptation of the code and program itself, while the swift turnaround makes it possible to test specific changes in the code and get feedback from the users on the efficiency and gain insight into whether the adaptations have a positive or negative impact on the workflow of the users. Based on the user feedback, the functionality of the final product is adapted to fit the new changes and requirements.

One further advantage lies within the code itself. The iterations of the code were done as Client software packages that connected to the active new database and updated datasets and showed up-to-date information, using the functions developed for this specific purpose. Even though the final graphical user interface uses Java Server Pages to interact with the users and their inputs, the client software allowed the developer to swiftly adapt the functions to the users' needs. The same functions developed in Java are then imported into the final product without having to adapt the functionality.

### 2.6.1 The Choosing Process

This section will cover the practices mentioned in above sections and explain why some practices were implemented and others were left out of the adaptation process.

The customers, in this case the lab personnel, became a useful source of information for the development of the new software and became part of the process.

The user stories and use cases flowed into the documentation and planning for the User Requirements Document.

Due to the situation of this project being a one-person project, the methodology needed to be adapted for the use of only one developer. This changes the strategy of incorporating certain agile practices, because some of them cannot be applied in that setting. The practices of pair programming, collective ownership and continuous integration were not implemented because of the mentioned reasons.

Short development life cycles depended on the software development life cycle. After setting parts of the software productive and giving users access to the software, the cycles in the beginning had a quicker and shorter cycle, and as the implementation of the software matured, the cycle expanded to two weeks or a month in length. This elongation can largely be traced back to too much tasks for a single developer.

Acceptance tests in a scripting language of choice has not become part of the adapted methodology due to resource constraints. Functionality testing is still part of the development though these are done in the form of alpha and beta tests.

Test-Driven Development is not implemented in the vast majority of the project, because of similar reasons for the acceptance tests. When the result can be tested, the unit tests and this development practices were applied; however, most of the tasks and programming could not be tested in such a manner.

The XP variant of planning got incorporated into the development life cycle, in which the project and its parts were presented to the management and they chose, whether the tasks fulfilled the need vs. cost function and the importance of the tasks. This meant that the tasks could be pushed back, if certain other tasks had higher priority.

The practice of refactoring code became an essential part of the life cycle and the idea of never leaving code worse off than how you found it, has shown to be a good mindset when programming, which makes sure the code itself is improving, no matter the circumstances.

### 2.6.2 Measuring Success and Failure

The measurement of success or failure of a project is predefined. Goals and requirements are set at the beginning of a project and in meeting the goals and requirements, the developer succeeds and otherwise fails to finish or complete the project.

In development, the methodologies and models set the rules for further tasks and how these should be accomplished. In applying the waterfall model, the progress is measured by moving through the phases of the model and delivering on the final product of the phase.

Agile methodologies are quite similar in the ability to manage the project, though measuring the success of each of the methodologies or practices is problematic, considering the existing overlap between the agile practices.

In the industry, various tools were created as mentioned in the Introduction, to be able to assess the metrics and the usability of each of the development models. For this implementation and hybrid methodology, the progress is not measured using metric based systems, but built upon the decided metrics by the stakeholders.

The success of the custom methodology depends on the progress being made by the developers, usability and adaptability of the software methodology. Corresponding to that, the success of the methodology is itself is very subjective to the one implementing it, and further depends on the resources available to the programmer and constraints placed by the stakeholders.

### 2.6.3 Measurement of Agile Variables

The ability to measure the agile variables is important for the development process, though as mentioned above sometimes not possible due to the requirements themselves.

The main idea in being able to track an agile variable would be to formulate a hypothesis, which specifies what effect a specific variable would bring, and see whether this hypothesis can be accepted or should be rejected at the end of the experiment. The experiment in this case refers to the agile practice being introduced to workflow for a specific amount of time and seeing or documenting the effects of these changes.

To accomplish this, specific questions were set at the beginning of the project, to track these variables and give feedback on the effectiveness. Three questions were formulated for this endeavor, which are listed here and are explained more thoroughly in the chapter Results:

- Can a hybrid agile model work in a setting, where a developer is solely responsible for the development of a bigger system?
- Documenting as little as possible and only what is necessary – is that the way to accomplish true agile productivity and performance?
- What risks or roadblocks are encountered during the process of adapting a workflow and developing a project at the same time?

## 2.7 Programming Infrastructure: Languages, Tools and Methods

The company has resources available through the hospital information system and the university resources. Both sources focus on patient data, data protection concerns, tools and needed software suits and serial numbers, to setup the development suite according to the requirements of the developer.

The company has multiple servers available, for the testing and productive use of software. These servers make testing for each of the developers possible and being able to test the software thoroughly before giving users access to the code.

The databases are all created and maintained using Oracle and the Oracle Environment Management system. This makes a standardized basis for database management of all databases possible and is aided by the university having a working relationship with Oracle.

The development platforms and suits can be installed and operated after gaining permission from the department of the hospital IT infrastructure. This external team is responsible for the general maintenance and installation requests made by the users employed at this specific hospital.

For the practical side of this projects, the main programming language is Java. This allowed the developer to use the same functions and objects throughout the individual parts of the projects.

Several different integrated development platforms (IDE) are used in implementing this project. For the graphic user interface, NetBeans is used for implementation. Reason for this being the availability of an integrated Tomcat and Glassfish server and being able to instantly test the changes of the Java Server Pages (JSP).

The main IDE for the all the batch jobs and progress tests is Eclipse. Both NetBeans and Eclipse are primarily used with the programming language Java and are evenly matched in their capabilities. These ideally support the developer in programming with Java.

For the development of the app, Android Studio is the IDE of choice. Android Studio is developed by Google and the development platform is based on the IntelliJ IDEA Community Edition. It is the fastest way to develop an Android app and test it, using the integrated tools of the IDE.

The act of programming has its own methods attached to it. The principles and rules of programming stem not only from the custom methodologies created for this project, but far surpass it in applying the material covered in Robert C. Martin books that reference and explain clean code, as well as the reasons and uses of Agile Software Craftsmanship. These practices are supremely connected and intertwined, because one cannot have one without the other.

# 3 Results

This section will cover the gained results of this master thesis. This chapter starts out in concisely describing the hybrid model used in development of the software, though the focus of this section lies within the implemented software using this hybrid model. The implemented software for this project is described from start to finish, which comprises of all the individual software parts, which includes but is not limited to the sample acquisition, processing, data cleaning, inventory of samples, and the approach to implementing this system and the process of designing it to fit the workflow of the biobank.

## 3.1 Hybrid Model: The Applied Model

The hybrid model, which was applied to create the sample management system, retained the characteristics of a waterfall model in the beginning stages, but deviated from there to incorporate agile methodologies.

In detail, this meant keeping the Requirements Analysis and Systems Design phases as is and deviating from the Implementation phase onward to attain a quicker turnaround time and higher quality development results, as well as better adaptation of the software itself.

During the implementation of the practical portion of the master thesis, the life cycle became shorter and prototyping of the software became a habit. The prototyping did not have the same end format as a web-based interface. Conversely, the prototypes were client applications, due to the easier handling of the software, when going live with the software itself. These software prototypes gave a good indication, which of the required features are fundamentally needed for the live use of the software. Additionally, it presented an opportunity for the developer to adapt the software for better usability.

Due to the nature of the project, individual parts of the implementation interlocked with each other in only one place, which is the database. The other parts of the system can operate independently from each other, if needed, or rearrange the use of the functions, should that be an issue.

The agility in the model during the Implementation phase is brought about by the customers, the user stories, XP planning, the mentioned prototyping and refactoring. The testing procedure included alpha and beta tests before deploying the software for the users.

## 3.2 Overview of the Sample Management System

Cryotracking is the overall term that describes the entire system that encompasses sample management of fresh frozen tissue samples, although in truth it only refers to the name of the database itself. It is developed to manage fresh frozen tissue samples at the Biobank Graz. The software needs to track the samples and their storage location, incorporate any new information from other sources seamlessly and establish a way to mark them using various flags for specific purposes.

The system consists of a database that was developed and designed to manage the data corresponding to the samples. The database is set up with Oracle and incorporates views of other databases to accomplish automatic validation and correction of the sample data. The connected databases on this level are internal databases holding patient information, which are updated regularly using interfaces to the hospital information system.

A batch job is implemented for the validation and updating of the data sets during the night, which activates once a week during off hours, i.e. during the night. This procedure pulls the datasets with the specific flag from the Cryotracking database and serves to correct the dataset and to complete it, i.e. some information is pulled from the pathology database, so that it is available for employees, when searching for those samples.

The data input occurs through the employees using tablets with custom software on it. The biobank is called if there are samples that need to be handed over and handled correctly. Whenever the pathologists working at the pathology institute at the Medical University of Graz have tissue samples for the biobank or for specific sample collections, the department then calls the lab personnel, where the tissue is then handed over to the employees with a set of information, i.e. organ, tissue specification, etc. that describes that tissue sample. This information needs to be typed into the custom software on the tablet by the lab personnel. This software running on the tablets is called “Cryo Recorder”. The software needs to record all the necessary data and assemble it into a predefined XML format, and the file should then be transferred to the server, where a listener is in place that imports the data into the database. This program is an app that was developed to run on Android 6.0. The tablets bought to replace the old hardware are the Asus Zenpad 8.

After the data has been validated, it is accessible for the employees at the biobank using the graphic user interface that is aptly named Cryo Sample Management, which is the central management software for all the software parts for the employees.

The described implementation of the sample management system is the end result and does not include all the iterations of the software prior to that software version. In the next section, the approach to the implementation of such a software management system is discussed.

## 3.3 Approach to Implementing A Sample Management System

This chapter will focus on the approach of implementing the sample management system. Due to the system consisting of multiple parts, the approach to implementing it needed to be checked with the customers and the management to be sure that the order of operations is accepted by them.

The mentioned order of operations indicates the order in which the multiple parts need to be developed, to make sure that all of them are compatible with each other. The points of contact include the graphic user interface interacting with the backend, which is the database. One further point of contact are the tablets, which generate sample datasets that are transmitted over the wireless local area network to the servers. These datasets need to have a standard format, so that the import of the data can function automatically.

The general approach is to start with the smallest common denominator, which is the dataset itself. This dataset still needs to be adaptable and expandable in the future, though the data types and data groups needed to be fixed before starting to implement the software attached to the data.

The next step turned out to be to work on the backend of the software, the database. This data structure needed to house the datasets and be functionally and logically assimilated together. The connections between the datasets should fulfill the basic requirements for the project and already have structures in place, if certain functions require it.

Third, the import of data is addressed, in which the fronted for recording new sample data is generated. It does not have the final form of the tablets, but an interim solution is created to be able to do data recording in any structured way. This became necessary, after the former data recording hardware had a hardware failure and failed to start anymore.

Fourth, the fronted is implemented to provide the users with the most common functions for the project for searching and managing sample data and see whether the functionality holds up to the expectations and the workflow of the users.

Fifth, the batch jobs are implemented to automatically insert, check and update data sets without the developer having to start it in the first place. The import of data and backups of datasets are both done using batch jobs, which register the uploaded new datasets and import the datasets into the database.

Finally, the app for the tablet is implemented, where the dataset is recorded in the graphical user interface and saved in a predetermined format.

## 3.4 Designing A Sample Management System

Figure 3-1 depicts the workflow, which has been constructed to showcase, which steps need to be taken to create a fresh frozen sample management system. It starts in the left upper corner by illustrating the old workflow existent before the introduction of the new systems that includes the handheld devices and the old software setup. This progresses to the upper right corner in roughly sketching the export of the data from the old system and import into the new system database, which is accompanied by data cleaning, validation and correction in the form of batch jobs and a inventory of the entire fresh frozen sample collection. From there, there is an arrow pointing down, which shows the import of the cleaned and ready to use datasets into the designed database for the software. On the left side of the Cryotracking database, the tablet software is portrayed, where the branches of the unique workflow on the tablet is shown and how the data is finally imported into the database. On the right side of the database, the graphical user interface is drawn to outline its existence in this complex workflow.

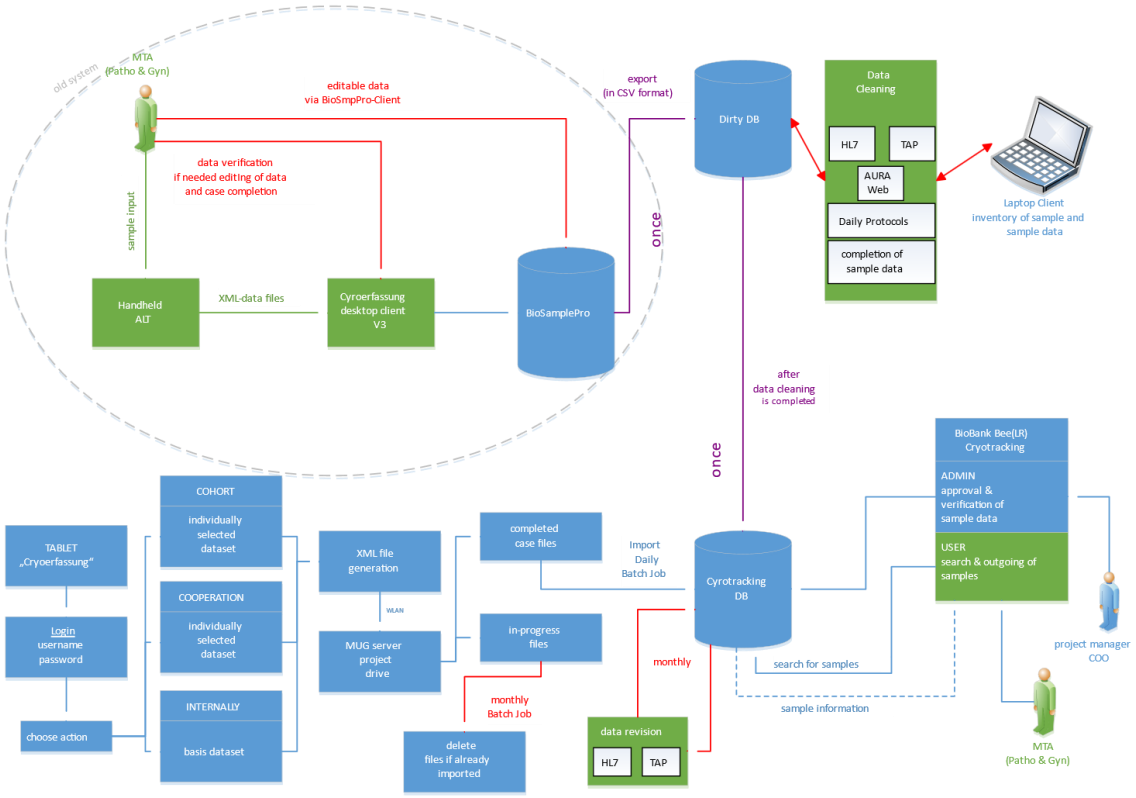



Figure 3-1: small portion of the workflow is shown that was used to accomplish the transformation from the old legacy system to the implemented software management system

The sample management system is designed to conform to the workflow of the employees and be adaptable. In designing the new system, the developer needs to take stock of the existent hardware and software, evaluate the costs and benefits of keeping the part of the system versus replacing parts of the management system during the implementation process.

In the next sections, the legacy hardware and software systems are assessed and their advantages and disadvantages to their operation in the biobank process are discussed.

### 3.4.1 Advantages and Disadvantages of Handheld Devices

As mentioned in the Introduction, the company used handheld devices from the company Honeywell to record the data from fresh frozen samples. The practice of recording data with such a device has advantages and disadvantages.

The advantages of using such handheld devices is the complete lack of connection to the internet and using the device for only one purpose, and that is for work related tasks. Other advantages included the scanning of barcodes using an integrated barcode scanner and the small pressure-based screen, that could be operated using a stylus, or the finger itself. The keypad allowed for limited typing, plus the hardware could be disinfected multiple times, which is necessary due to the environment, in which the product is handled. This refers to the different departments of the Medical University of Graz, which contribute to the sample collection at the biobank.

The disadvantages encompass losing all the data on the device, including the custom software itself as soon as the charge in the batteries runs out. The operating system is out of date, especially considering that the handhelds run on Windows ME Mobile Edition. That operating system is no longer fully compatible with modern operating systems like Windows 10. Similarly, the edition of handheld devices, which was used at the biobank, are discontinued and could not be bought anymore; conversely, they also had quite the steep price for the function they provided. The weight of the device and the uneasy handling of it is an additional factor, why these handhelds had to be replaced.

So, why do companies still use old systems and don't replace these with newer and better hardware? This question is answered in chapter 3.4.2 Advantages and Disadvantages of Legacy Software Systems.

### 3.4.2 Advantages and Disadvantages of Legacy Software Systems

Legacy hardware and software systems all have a lifetime. This type of software is explained as "[…] software that has been around a long time and still fulfills a business need. It is mission critical and tied to a particular version of an operating system or hardware model (vendor lock-in) that has gone end-of-life." Legacy systems conform to the same principles though include both hardware and software. [32]

This is widely known in the communities regarding hardware and software. Requirements change far too quickly in the world we live in, that the upkeep and maintenance of software needs to be calculated into the acquisition of any kind of product. The worst position to be in, is to have a software that cannot grow with the company and adapt to the changes that will come to pass, and the product, be it hardware or software, reaches its end of life far too fast.

In short, it means that the advantages of keeping a legacy software is that it fulfills a business need.

For the biobank, the maintenance of the devices kept growing and the advantages were slim to none. The software did the job of getting the information from the users and transferring them to the given database, but even that was not completely done right, though that is partly the fault of the receiving database, as some information that had to be transferred got lost each time in transmission. The reason for that being the missing data storage possibilities and expansion capabilities of the legacy database.

The disadvantages, as already mentioned, were numerous. From losing important data in the transmission, not having a clear understanding of how some errors come about or how users interact with the out-of-date software. The uneasy handling of the software itself, where you had to stepwise insert data, so that the information could be saved and allocated, is partly also the reason why it had to be replaced, as the data flow of the biobank itself adapted to the new workflows, and a more flexible way for data acquisition needed to be implemented.

### 3.4.3 Choosing the Right Hardware

The first step in replacing the legacy systems is to replace the hardware. The custom software solution would depend on the chosen hardware in relation to operating system, programming language, development suite and much more.

Criteria for the hardware are the following ones:

- The hardware itself should be able to be disinfected. The reason for that being the daily use of the hardware in pathology or the operating room.
- The hardware should not have a steep price tag, due to factors like overturn rate of hardware, changes in operating systems, etc.
- The operating system of the hardware should be mainstream and not a niche software and niche product. This would make the development of the software and any subsequent problem solving much easier.
- No ports or input jacks, like USB or anything similar, are necessary for this software, but the addition of the use of wireless transmission of data is expected. This would mean that the hardware should have wireless LAN and / or Bluetooth capabilities.

The chosen product that fit all the named criteria for this use case is an Android tablet. Such tablets are a mainstream product that allow for easy handling of the software in maintaining it and implementing new software. The hardware side fulfills the hygiene part of the criteria, as well as the price tag. The operating system is adaptable and the software development adaptable to new Android

software should the tablets reach their end-of-life. The wireless and Bluetooth capabilities are fulfilled as well.

After researching different brands and comparing the capabilities of the Android devices, the Asus Zenpad 8 tablets are chosen to replace the legacy hardware and to be the primary handheld device for recording fresh frozen samples (see Figure 3-2 for hardware depiction).

The specifications and characteristics of the Asus Zenpad 8 tablets are listed here: [33]

- 8-inch display, aspect ratio: 16:10 HD, resolution: 1280x800
- 64-bit media MT8163
- Quad-Core 1.3 GHz processor
- 2 GB of RAM, 16GB of internal memory
- android operating system: Marshmallow (Android 6.0)
- WiFi certified, Bluetooth 4$^{th}$ Gen
- model number: Z380M


Figure 3-2: hardware for installing the custom app, which is the tablet Asus Zenpad 8

## 3.5 Basics on Fresh Frozen Datasets

This section will focus on the data being gathered during each recording by using the tablets. Each of the variables chosen and recorded give the lab personnel, the project managers and the researchers at later stages the ability to use the samples for their intended purpose. The data can be grouped generally into two groups, which are the sample data that encompasses all organ and sample information, and storage data, which describes the location of the sample tube in the storage system of the company. In this section, we will focus primarily on sample data.

### 3.5.1 Admission Number

The admission number is a number that each patient receives at admission to a hospital. This admission number is given out for each separate visit to the hospital, and it does not differentiate between an ambulant or inpatient admission. There are certain exceptions to this rule, which occurs when a patient visits the hospital for follow up regarding the same issue. In those cases, the number will stay the same. The format of the admission number consists out of 10 numbers. The first four numbers are the year in which the admission takes place, and the following six numbers is a number that is incrementally raised, each time an admission at the hospital takes place. The clinical data connected to each admission number is saved in the hospital information system, which the biobank has limited access to in form of a HL7 interface.

### 3.5.2 Histologic Number

The histologic number has the same format as the admission number. This variable refers to the histologic reports being written in pathology for each tissue sample. Every histologic number has the connected histologic report in the pathology system. An extract of that data is forwarded to the biobank monthly in form of an XML file and saved in a separate database, which holds and manages all of that information.

### 3.5.3 Body Sites

The body site and body site code represent the place of the human body, which the sample belongs to. Examples for this can be liver, calf muscle, or even bones. These locations are predetermined by using the newest release of the International Classification of Diseases, called ICD-11.

### 3.5.4 Tumor Specification

This input field refers to the type of tissue regarding the sample. In simple terms, this variable describes, whether the sample consists of tumor tissue, or has been located near tumor tissue. Examples for tumor specifications are tumor, tumor near, tumor far, healthy tissue and inflamed tissue samples. This is vital information, because during medical procedures for instance, the whole lung could be taken out, and several samples need to be preserved and recorded for the biobank. Researchers sometimes need both the healthy tissue, as well as the cancer tissue.

### 3.5.5 Tube Content

The tube content refers to what part of the organ is stored inside the given sample. An example in this vein of thought is the organ or body site for the sample could be set as the heart, and the tube content is set as the right ventricle. If more than one sample is collected from each patient and from different parts of the heart, this information needs to be logged. A sample is basically worthless, if there is no data attached to that sample. The tube content makes the sample worth something, and have data, which can be used by the personnel at the biobank.

### 3.5.6 Warm and Cold Ischemia Time

The US National Cancer Institute of Health defines cold and warm ischemia time in the following way:

"*Warm Ischemia Time: In surgery, the time a tissue, organ, or body part remains at body temperature after its blood supply has been reduced or cut off but before it is cooled or reconnected to a blood supply."* [34]

"*Cold Ischemia Time: In surgery, the time between the chilling of a tissue, organ, or body part after its blood supply has been reduced or cut off and the time it is warmed by having its blood supply restored. This can occur while the organ is still in the body or after it is removed from the body if the organ is to be used for transplantation.*" [35]

In the beginning of preserving fresh frozen tissue samples, the ischemia time was documented as a time duration in whole minutes. Due to communication and documentation problems, this turned out to be inaccurate, which is why the biobank switched and records the actual time for instance, when the blood supply is cut off as a timestamp. This makes it possible to refer back to the preanalytical workflow at a later date and understand quality measures regarding sample acquisition.

### 3.5.7 Exclusive Rights

The exclusive right is information pertaining to the right to use the sample for research purposes. If the sample is marked as such, this would indicate that the sample can only be utilized by the sample owner. In the case of another researcher requesting one of those samples, the sample owner must give his or her consent for the use of their samples.

## 3.6 Data Acquisition and Recording

For the new system, the software Cryo Recorder, is used to record the fresh frozen sample datasets, and the files are sent to a storage drive of the medical university via wireless local area network.

The data being assessed and saved each time a new sample is brought to the biobank includes the already mentioned data in Chapter 3.5 Basics on Fresh Frozen Datasets. The datasets encompass data relevant to the sample itself and relevant to the biobank for storing said sample.

For storage information, this is dependent on the storage setup that has been implemented. At the Biobank Graz, cyro tanks filled with nitrogen are utilized for storing fresh frozen samples. The tanks are filled with racks, that contain boxes, in which samples are stored. This storage schema has been recreated in the database, which holds the data connected to the sample, which can be seen in Chapter 3.8 Importing Data into Cryotracking Database. Examples for storage data are the tank number, the rack number, the box level, at which the specific box is stored, the position of the tube inside the box, the box barcode.

A side note pertaining to the box barcodes: Until now, boxes have been labeled by hand using a cryo pen and names were assigned by the personnel handling the samples. Due to needing uniformity and uniqueness in the given names, this is changed, so that every box inside the tanks is a box with a unique barcode, which is labeled using laser technology. For that to be productively in use though, each sample needs to be inventoried and stored in a new pre-barcoded box. All this information is then saved in the database. Part of the reason for the inventory is the knowledge that a large amount of the fresh frozen samples at the Biobank Graz in storage are labeled by hand. This is not 100% truly connected to the data in BioSamplePro, due to problems in transferring the data and data acquisition changing through the years, i.e. from handwritten notes, to excel files, to different versions of the recording software on the legacy hardware systems. To streamline all that information, the inventory of all the samples is necessary, before the live operation of software projects can be possible.

For sample information, the data is partly an accumulation of regular data that is being provided by the pathologists, physicians or surgeons, and data that is part of the histologic report, which is sometimes accessible to the Medical Technical Assistants (MTA). Examples for sample data are histologic number, admission number of the given case, body site, body site code, tumor specification, etc.

From there, the data is imported daily into the database Cryotracking. The data that is imported must pass validation and verification checks before the import is complete.

## 3.7 Graphic User Interface for Recording of Sample Data

The datasets mentioned in Section 3.5 Basics on Fresh Frozen Datasets need to be recorded by a custom software. The graphic user interface is designed and programmed in Android Studio to run on Android devices. It consists of various activities, mostly so-called blank activities, which are used to implement the custom graphical interface that is needed. In the beginning steps of implementing the graphical user interface, these blank activities do not contain any graphical elements and are inserted to be used as a blank canvas, which can be adapted to the developers' needs.

### 3.7.1 Login Activity

The starting screen of this software is the login screen and users need to log in to be able to operate the software. This is mandatory, so that the handling of each sample and of each sample data set can be tracked using this software. For the login, users need to enter their login credentials. The username is to be selected from a drop-down menu, and the password is a specific input field for passwords, which masks the input of the password. A login button allows for the verification of the data using on-click activities. The user login is activated after the mentioned user underwent the training to use the software and the tablet.

### 3.7.2 Overview Activity

After correctly entering the login credentials, the overview activity is opened. This activity gives an overview of the capabilities of the app and shows the main tasks and actions, which can be accomplished using the software. These are the following ones: create new pathology case, open saved case files, box management, possible box placements, synchronization of cases and connecting to wireless local area network. Each of the listed functions or tasks represents a link, which leads to the corresponding activity. Every single one of these tasks has either a further activity to enter data, or only an on-click function, in the case of “synchronization of cases” and “connecting to a wireless local area network”.

The blank activity for “saved cases” consists of a drop-down menu of all the saved case files, which are not yet completed by the tablet users and transferred to the database to prevent data loss. This makes it possible for the user to select one of the saved case files and open the complementary activity for “new pathology case” for instance, and have the users pick up working where they left off.

### 3.7.3 Box Management Activity

The next activity named “box management” is a way to digitally create a box, which holds samples, and assign the box a storage space in the database, which reflects the state of the physical storage system. Furthermore, the user can choose an existing box from the storage tanks to gather more information.

The activity has input fields dedicated to the name of the box and the box barcode. The input for the placement of the box, in the case of tank number, rack number and box level are implemented using a drop-down menu. The option of placing boxes in freezers is possible and can be recorded as well. For that, a checkbox is ticked and the option of naming a storage space in a specific freezer is activated.

One of the buttons in the activity activates the camera to allow the user to scan barcodes, which are on the boxes themselves. The barcodes are automatically recognized and converted into human readable font and inserted into the correct input field.

### 3.7.4 Case Activity

The main use case for the software is to record the datasets for the fresh frozen samples. The case activity allows the user to insert the data into the given input fields and this activity generates the files in the background.

The case activity makes it possible for the users to scan barcodes and QR codes, to better support the users in the day-to-day activities and reduce human error from typing out the numbers, should the automatic scan of the barcodes not work.

The rest of the fields are dropdown of similar kinds of fields, which are partly filled automatically with predetermined values, should the sample be part of a study or cohort. These fields can be changed after the fact, but the autofill function makes the job of the users easier and less time intensive.

The placement of the tube in the box is input using a drop down of all the numbers between 1 and 81. The barcode of the box it is placed in, is scanned using the back facing camera, and the button beside it allows for a link to the activity of box management to either place the box correctly in the storage tanks by creating a new box with that name, or searching for a box with that name or barcode in the storage tanks and choosing from a drop down.

Most cases can have more than one tube per case, in which instance, a further tube section is added, where tube specific information is input. To explain this more thoroughly, this activity is split into two. The upper half contains all the information pertaining to the case and not the tubes themselves, and the lower part is only tube specific. This also allows for easier adding of more tubes, as only the lower part needs to be filled with information again, while the other information is saved.

This activity has an additional four buttons, which are the following ones: “new tube”, “save”, “complete case”, “show summary”.

The “new tube” button saves the currently input information of the tube and allows for the user to input the same variables for another tube.

The “save” button saves the information of the case that was input so-far, closes the activity and returns back to the main overview. Furthermore, in the background the case information is saved to the hard drive of the tablet, as well as sent to the main server of the medical university of Graz using wireless local area network, if that is available (or during the update process). Please note, that additionally to all of that information the information of which user edited the cases and which user completed the case are saved as part of the case in the database.

The “complete case” button does the same thing as the “save” button, only that it transfers the data to another folder on the server, where a batch job collects that case information and automatically imports it into the database.

The “show summary” button is for a quick and concise overview of all the information input for that case and should note any irregularities for the users to see quickly.

## 3.8 Importing Data into Cryotracking Database

Importing the data needs to be separated into two stages: initial import and productive imports.

The first stage refers to the initial importing of data from the previous database and into the newly designed database structure and is followed by cleaning and validating the datasets, before making it available to users. The data cleaning aspect of it is explained in depth in Chapter 3.10.

The second stage refers to the productive import of data, which is done automatically after a user marks case files as completed on the tablet. The files are generated by the app in the background and include the recorded datasets, and these custom datasets are saved in XML format, which is custom to the app. The files are transferred to a secure server using a wireless connection, and from there imported into the database by a batch job, which is activated once a day.

The database is designed for the optimal use of data processing and created for the saving the sample data effectively in a predefined structure. The entity relationship model of the database, i.e. a model that represents the relationships between each of the tables, can be seen in Figure 3-3.

In the illustration each of the tables are colored according to their overarching grouping. The blue tables are part of achieving a storage structure for the fresh frozen samples. The storage tables include data for tubes, boxes, racks, tanks and freezers. As describes in sections above, the tanks can hold up to 25 racks, each rack can hold up to ten boxes and each box can hold up to 81 samples. Freezers primarily are used for RNA later samples that are stored at temperatures around -80°C.

The green tables build up the data structure for the incoming and outgoing samples and make it possible for the users to reserve samples using the reservation tables. The incoming samples are inserted into the import table, which checks all entries for validity and cross references entries with other databases. After those checks are completed the data is inserted into the productive table of cryo data. By inserting the datasets into the cryo data table, the datasets are instantly available for search queries by the biobank employees. The separation of datasets allows the developer to only show verified datasets to the users of the software.

The orange tables represent the lookup tables. These tables contain all the active and inactive versions of the dropdown possibilities for the tablet app. An example for this would be the list of cohorts, where

only the entries that are active are listed in the app for the selection of one of the cohorts. The same principle is applied for the senders, the examination type, localization, ICD codes, pathologists, biobank employees and tumor specification. The ICD codes list is the only list that is used by both the organ and tube content dropdown listing.

Additional tables are listed in red, which don't have a core function and use for the graphic user interface of the productive system. The predominant functionality of these is to manage the introduction of new samples into the database. These tables include the initial import table, scanning problems table, where all the unscannable barcodes are saved, the events table that is the central place where the events of each of the sample is recorded. This latter table is not necessarily part of the core function of the database and sample management system, though still important to keep track of the datasets.

To sum up, the entity relationship model creates the possibility for the developer to show an excerpt of the data for the users depending on the view of the users, which is implemented by joining multiple tables together to query data across all the tables. That means that for the implemented general search, to be able to find a sample, at least five tables are involved in one single query.

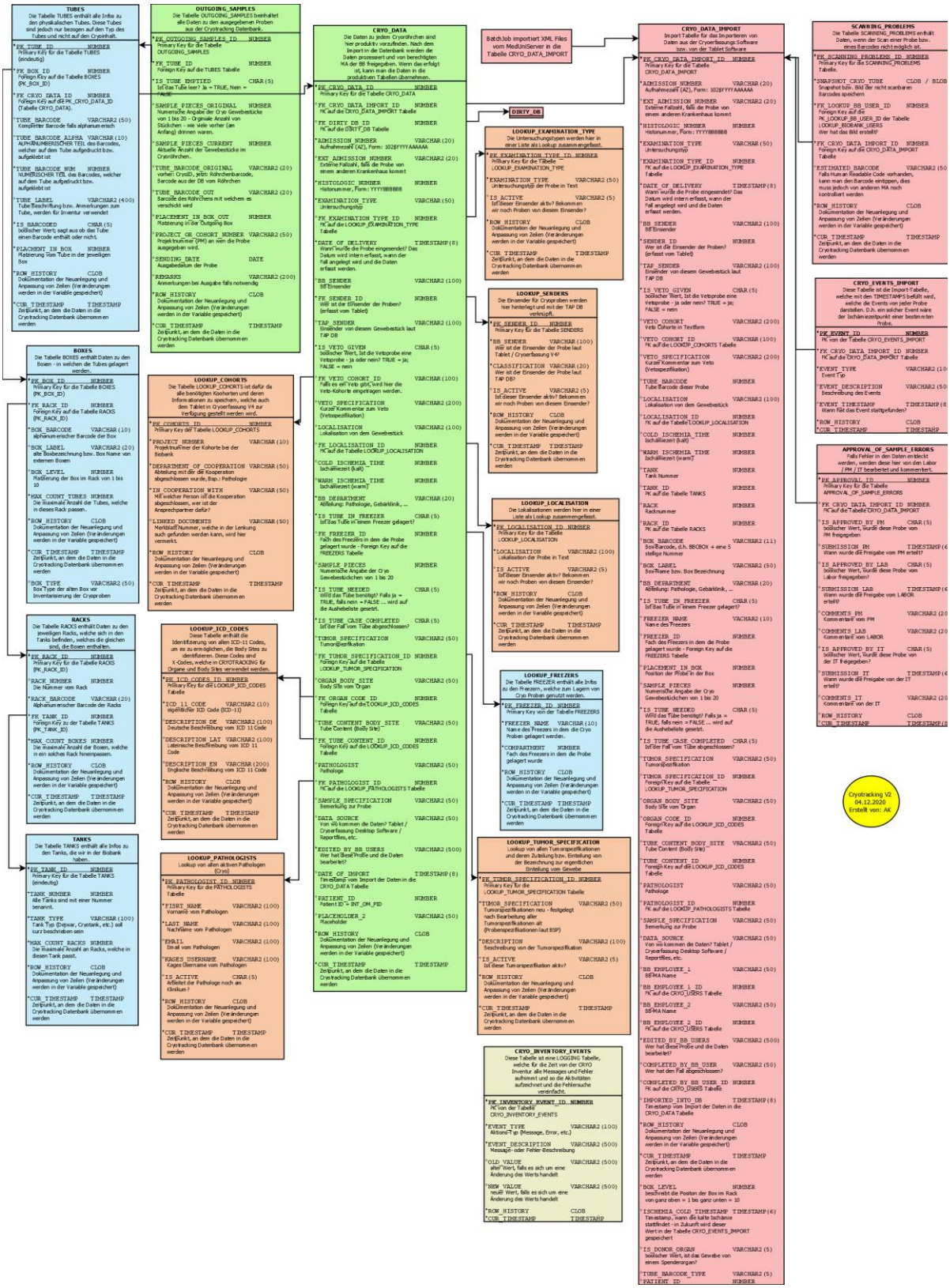



Figure 3-3: entity relationship model of the database called “Cryotracking”

## 3.9 Sample Acquisition and Processing

As described in the Introduction, the sample is acquired and processed by biobank employees.

That tissue is then prepared according to the guidelines and lastly stored in a tube. That tube has a predefined barcode and matching human readable code, which is important for further data acquisition purposes.

The app on the tablet allows users to scan the barcodes using the front facing or back facing camera and input the human readable code into the required field.

Before the tablets are included in the day-to-day data acquisition processes, a separate client software got implemented to record the datasets, because the legacy hardware, i.e. the handheld devices from the company Hand Held Products, stopped working and could not be repaired by the technicians at the company.

The client software was intended to be in operation for a short period of time, though changes occurred in the prioritization of tasks, which resulted in a longer life cycle for the software. The software is a runnable jar file, which consisted of a basic graphic user interface that made it possible for users to insert the data of the samples and transfer the data directly into the appropriate table in the database.

In Figure 3-4, the graphic user interface of the client software is shown. The input fields on the right need to be filled out first to create the case file for the sample in the database, and after that the tube data is input on the right hand side and committed using the button on the bottom right corner in the picture.

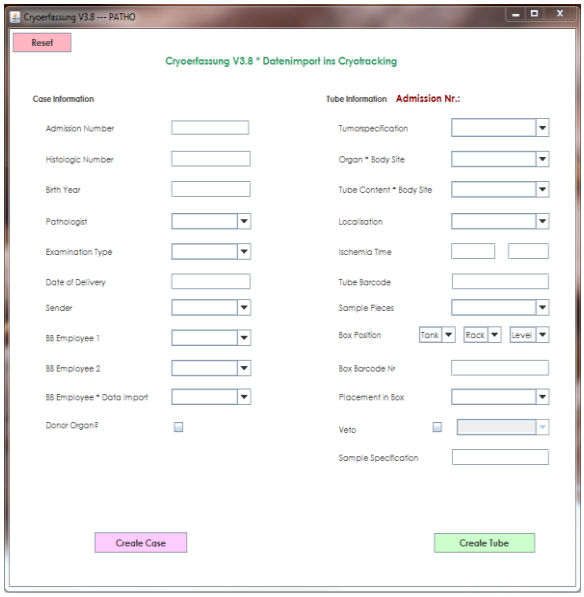



Figure 3-4: client software for the acquisition and recording of sample datasets before using the app

## 3.10 Data Cleaning

This section describes the changes that the datasets go through in this project before the initial import into the database is made.

Initial importing the data sets into the new database is done in various steps. The first step is to export the data from the old database. The columns, which we want to keep and are important to the sample, will be exported as a comma separated value (CSV) file. This file is parsed and imported into a database pro-forma.

After the insert, all the data is present in the database in one table, that is aptly named dirty db. Because various means of data acquisitions were used in the past, the data was in no way uniform or similar in any way. To change this, a lot of data transformations took place. For every variable in the table, a physician set the range, what was acceptable and what was not acceptable for that specific variable. Everything that did not fit the set standard would have to be changed to show the correct meaning. All the transformations are now listed below in this section.

The admission number is in most cases a number, which is given out by the hospital information system, for each new admission to the hospital. The admission number itself is normally only 10 digits long, where the first four digits indicate the year in which the admission took place, and the following six digits are a number from the hospital information system that counts upwards internally, and is assigned to patients in order in which they come to the hospital. The variable in most cases includes a three-digit code and a dollar sign at the beginning of the ten-digit number, to indicate which hospital that admission number is from. In the case of the hospital information system of Medical University of Graz, that number would be 102$. Due to the fact that that number is always input in this manner with the old system, this information is separated into another variable, as the manipulation of only numbers using databases is much faster regarding performance than searching for a text that matches or contains another text in the database. This is accomplished by selecting all the datasets that contain the dollar sign, and changing the datasets, so that only the admission number is input in that field.

The histologic number is the number that is linked to each histologic report in the pathology information system. There are different formats of the histologic number, mostly based on the preferences of the pathologists and how the system at the hospital developed over the past thirty years. To streamline that variable, each of the formats is input into the source code, each of the histologic number is checked to see what kind of format it is, and converted to the now official format, which aligns with the format of the admission number, i.e. a four digit year number, followed by a six digit number. All in all, it is also a ten-digit number, which is assigned by the system, depending on the order the tissues or samples are brought to pathology.

The variable “is used up” defines whether a sample was used up for a study or is still existent in storage. This was noted with a 1, 0 or null value. Because the given data format confused many users, the format was changed to a Boolean value, that only held the values true or false.

The variable “ischemia time” is generally the cold ischemia duration of the sample. The warm ischemia time is a new value that is being recorded with the new recording software Cryo Recorder. The cold ischemia time had various entries that all define the time in minutes. Some of those entries even had thousand minutes as an entry, which is entirely feasible, e.g. if the source of the sample is an organ transplantation. To weed out the false entries, two changes were undertaken. The first change to the data was that every entry over 10,000 is set to zero, which happened when autopsy samples got stored. The ischemia time is an input field, which must be filled to complete the case file, and to be able to do that, the users entered a value, which is at first glance not correct, to be able to spot the information and correct this later on. The second change to the data goes hand in hand with the first change and should the sample have an autopsy as an examination type, the ischemia time was updated to a null value, as an ischemia time is nonexistent postmortem. Furthermore, the value of the warm ischemia time is set to null for each data entry, because no other data sources held that information for the samples in the legacy system.

The variable organ had a multitude of different entries, where each entry was slightly different to other one, and included organs in multiple languages, various spelling mistakes and much more, which needed to be streamlined before importing the datasets into the database. To help with the classification of the organ entries, the ICD11 coding and two physicians helped to make this possible. This code contains body sites, which is preferable to organ codes like the ICD-O code for example. The samples themselves cannot always be classified as part of an organ. An example would be the muscles of any kind on the human body, or ligaments as well. This can be done with the ICD11 coding.

Considering the change in classifications, the existing datasets needed to be classified according to the new standards, which was accomplished through grouping the sample locations and then manually sorting these data sets and finding the correct body sites online and matching the samples to their correct code. This task was completed by two physicians manually, and using the resulting comma-separated value file, the matches are then imported into the database.

The same problems, which were explained for the variable organ, also apply for the variable tumor specification. Each of these though have full text entries, which cannot that easily cleaned up, so that users can use that data. In this case, each of the full text entries was matched to set values from study agreements and contractual obligations to insert and note, which study this data entry (=sample) is part of.

The tumor specification entries were once a free form text field, though that is changed to show fixed entries, where one of them can be selected from a given drop-down for each sample by the MTA. For data cleaning purposes the old entries are processed and adapted based on predefined rules. One of the reasons for this is that each tumor specification holds in most cases the diagnosis of each sample, or to be specific the type of tumor that can be found inside that sample. This is relevant for further studies, and for veto sample validation (also known as exclusive rights of a sample). After linking and processing all entries for tumor specification, each of the entries is checked against contractually set diagnoses and tumor specifications, which automatically flags a sample as a veto sample. This is the set procedure for retrospective data collection. For prospective data acquisition, the MTA must choose a cohort when recording the data, what study the sample is associated with.

The sender variables refer to the senders, i.e. the departments of the hospital or external partners, who send the data to the biobank. The variables hold only a few chosen sender values, which can be chosen from a drop down. Since no system is without faults, even here problems arose with streamlining the data, as employees would enter data into the database without using the recording devices, which would cause free text entries. Although this is problematic, the amount of variety here was minimal, which is why this part of the data cleaning was done manually, with a few manually written SQL statements, to update and change some entries and correct some spelling mistakes. It was an important step to separate the internal and external senders for future work with the datasets.

The variable pathologist was not adapted in any way. The data from the old database was imported as-is into the new database, because it only held the names of the pathologists, that documented and wrote the report for that specific sample. If there is no pathologist name, the sample should be from an external source. Should the data show that the sample is an internal sample, but not have a pathologist documented; this would require renewed documentation and validation of the names of the pathologists by one of the employees of the biobank.

Datasets that are used for storage information are updated using an inventory software. A first version of the inventory software was programmed in the beginning for this project, though that piece of software did not find any use, due to miscommunication errors. A second version was discussed in detail following the disuse of the first version and this part of the software was planned and programmed for finding samples and placing them correctly in the database.

Why was the data cleaning done in this manner? The data needs to be verified and validated by qualified people. This means that there are changes that need to be done, and each change needs to be logged in the database. When all the changes were finished and the data is completely and thoroughly finalized, then all the datasets are imported into the productive database. This distinction between the databases is only theoretical, as all this can be found in the same schema.

The validation of data includes merging further data sources and importing or inputting the information found from different documents or modules and adding that data into the database as well. One example for this, is the case, in which the pathologists are not logged for a specific duration in the past few years. The reason for this being the addition of a new dataset, and changes in the software resulted in the inability to save that information to the database, i.e. the database could not be adapted to hold the additional information that was input from the legacy recording system. This meant the recorded digital information was not saved to the database and lost.

## 3.11 Inventory of Entire Fresh Frozen Sample Collection

Before the sample management system could become part of the day-to-day management activities of the samples, the challenge of finding the correct samples created problems for the users. Due to the legacy software not being able to keep up with the changes in the datasets and concurrently, mark samples as given out for a specific study, the placement of the samples in the boxes did not match the data in the databases. To fix that problem, an inventory of the entire sample collection needed to take place.

While the initial ideas of doing the inventory of the samples using a file-based method, the users decided to conduct the inventory with IT support and requested for an implementation of an inventory software to inventory all fresh frozen samples.

The workflow for this undertaking was defined before starting the implementation of the software.

The first two iterations of the development of the software did not get employed by the users, due to workflow changes or undefined adaptations of the software by the users, which changed after assessing the software prototypes. The third and fourth iteration of the software fulfilled their purpose and the users could inventory the samples with the help of the software. The difference between the third and fourth iteration are minor functional changes and performance upgrades.

The fifth iteration of the software is a special edition to support employees with the inventory of existent samples that are stored in a separate cooling unit. The specialized cooling unit produced errors in temperature logging and storing of samples, which led to the decision to move the samples to a more secure storage placement in the cryo tanks, i.e. in the gas phase of the tanks, which are filled with liquid nitrogen. To be able to track the samples and import them correctly, the samples had to be inventoried by adapting the fourth generation of the software for that purpose. The reality of the samples all being barcoded helped in reducing the time spent on adapting the software for that purpose.

Figure 3-5 shows the fourth generation of the graphic user interface for the inventory software, which consists of two windows, or more accurately two layers. On the left side, the first window makes it possible for the employee to adapt and establish the storage structure of the box and the tubes. The workflow consists of the employee choosing the names of the employees performing the inventory, followed by three functions.

Firstly, the employee had to choose the name of the box from the dropdown, which is the box name that is written on the box of samples by hand, pick the box type and scan the box barcode of the new box to create the box in the database. In the case of an original box containing 100 samples, instead of the standard 81 samples, the software offered the users the possibility to scan two box barcodes,

which meant that the 100 samples would be separated between two boxes. Secondly, the box is placed in the digital storage by picking a tank, rack and box level. Thirdly and most importantly, the data can be retrieved from the database and inventoried, which opens up the second window in the illustration and allows employees to mark samples as existent or not existent, and note changes in the sample datasets, if necessary.

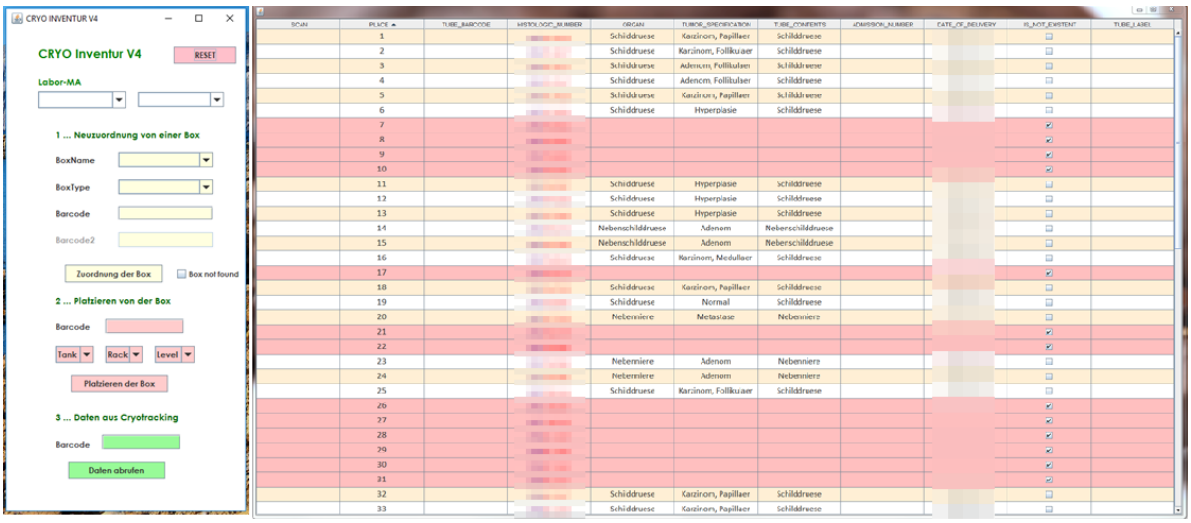
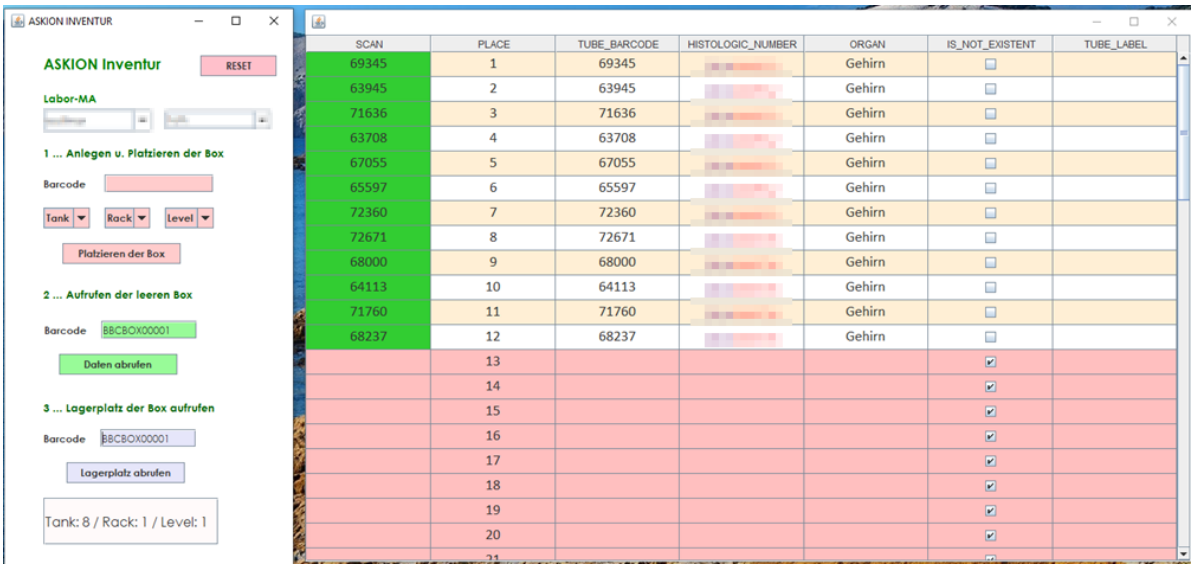



Figure 3-5: inventory software for the entire collection of fresh frozen samples

Similarly, Figure 3-6 depicts the graphic user interface for the fifth generation of the samples.




| SCAN | PLACE | TUBE_BARCODE | HISTOLOGIC_NUMBER | ORGAN | IS_NOT_EXISTENT | TUBE_LABEL |
|---|---|---|---|---|---|---|
| 69345 | 1 | 69345 | | Gehirn | ☐ | |
| 63945 | 2 | 63945 | | Gehirn | ☐ | |
| 71636 | 3 | 71636 | | Gehirn | ☐ | |
| 63708 | 4 | 63708 | | Gehirn | ☐ | |
| 67055 | 5 | 67055 | | Gehirn | ☐ | |
| 65597 | 6 | 65597 | | Gehirn | ☐ | |
| 72360 | 7 | 72360 | | Gehirn | ☐ | |
| 72671 | 8 | 72671 | | Gehirn | ☐ | |
| 68000 | 9 | 68000 | | Gehirn | ☐ | |
| 64113 | 10 | 64113 | | Gehirn | ☐ | |
| 71760 | 11 | 71760 | | Gehirn | ☐ | |
| 68237 | 12 | 68237 | | Gehirn | ☐ | |
| | 13 | | | | ☑ | |
| | 14 | | | | ☑ | |
| | 15 | | | | ☑ | |
| | 16 | | | | ☑ | |
| | 17 | | | | ☑ | |
| | 18 | | | | ☑ | |
| | 19 | | | | ☑ | |
| | 20 | | | | ☑ | |

Figure 3-6: fifth generation of the sample inventory software, special edition

For a complete overview of the workflow for the inventory of fresh frozen samples, see Figure 3-7. This illustration helped the employees during the inventory process decide, whether a sample is usable for research and where to place the sample, should that not be the case.

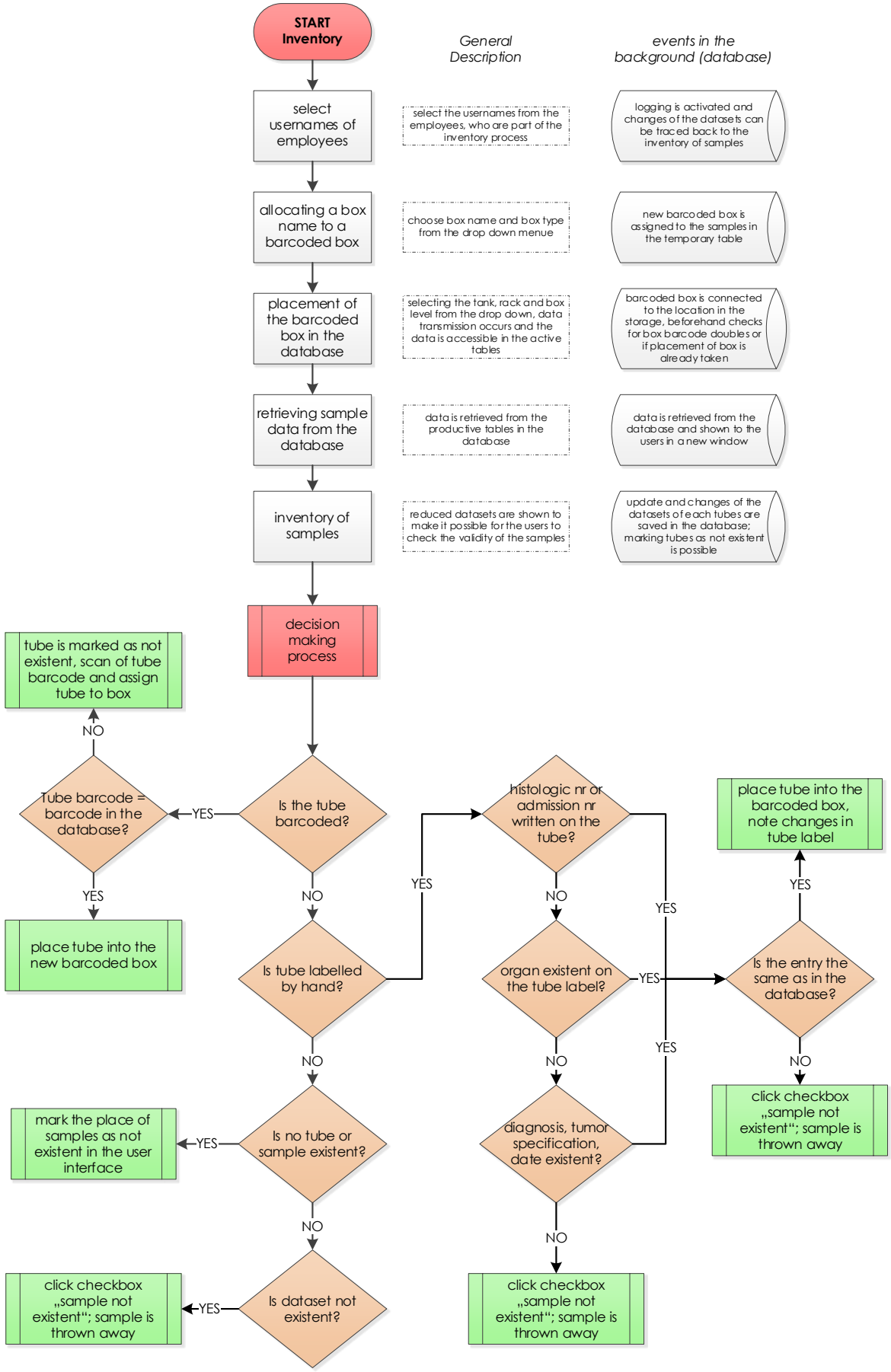



Figure 3-7: decision making process for the inventory of fresh frozen samples

## 3.12 Scripting Software: Automatically Generating Code

During the implementation phase, the software requirements of the multiple projects changed multiple times, which meant that the adaptation of the database and the code itself took a lot of time to complete and to adapt to the new structure. Especially, because the entire sample management system uses the base to model the Java object and SQL statements after – the core of the sample management system is the database itself.

To solve that problem, a scripting software was implemented to eliminate the adaptation of the Java objects and SQL statements in the software itself. The scripting software should fulfill the role of connecting to the database, retrieving all the information about the indicated running database instance, and create the Java code and objects, all the while adhering to clean code and agile methodology principles.

The code is split into three parts. One part is the graphic user interface, which allows the programmer to insert the database instance and connection details to the specific Oracle database, but makes it possible to save the credentials to a file on the computer, as well as open the file and load the credentials the next time the software is opened.

The second part is the scripting software itself. The connection to the database retrieves the table structures and meta data of the database, including if a variable of a table is required or not and what kind of variable type it is. All that information is necessary for the generation of the code. The Java objects are created and named the same as the tables and variables themselves. The SQL statements are grouped together in one file per table and include multiple functions for the adaptation of each dataset. Each file for the SQL statements includes multiple functions, which execute the statement of choice, and the file comprises of the following SQL statements or Java functions: one general insert statement, one general update statement, one general select statement, specific update statements for each variable of the table, specific select statements for each variable of the statements and distinct select statements for each variable.

The third and final part is the saving of the generated files into a file structure of choice. There is the possibility of saving these Java objects and files automatically as a .jar file, though that has not been implemented yet for the duration of this project. The files are saved to a hard-coded file path and from there can be accessed by the programmer and included in the new software projects or more replace existing Java objects and SQL statements in the software projects.

In the case of adaptations being made to the database, the developer would only need to adapt the database itself in Oracle and afterward start the scripting software. The software creates the code

quickly and efficiently, which makes adaptations to the datasets or changes in the database a lesser task than it would be, if such a scripting software is not available to the developer.

In Figure 3-8, the graphic user interface for the developer can be seen, which is described above, and in Figure 3-9, the Java objects and SQL statement files are shown.

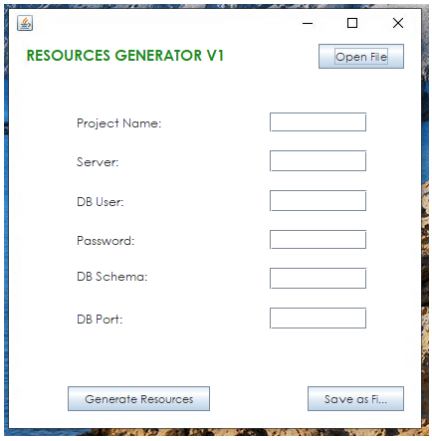
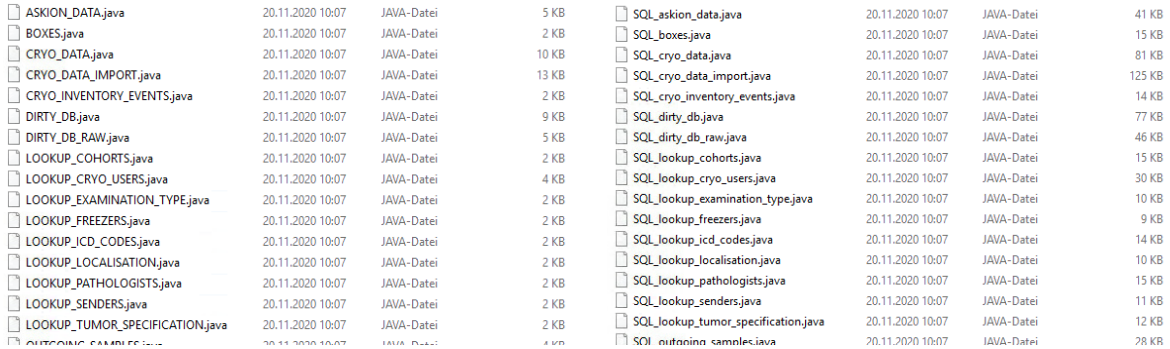



Figure 3-8: graphic user interface for the developer to generate the resources or files




(a) (b)

Figure 3-9: Java objects (a) and SQL statements (b) created by the scripting software

## 3.13 Cryo Sample Management

Cryo Sample Management is the management console for the Cryotracking system. The graphic user interface gives the users five different tabs to operate and manage sample data. Each of the users receives certain user rights during the training for the software, which determines which tabs can be accessed by them.

### 3.13.1 Overview of the Graphic User Interface

The first tab can be seen in Figure 3-10 and is called Sample Search. It offers the possibility to search for specific sample data, flag certain samples and call up timestamps regarding specific samples in the database. Samples can be flagged and reserved for a specific research question or study by choosing the sample datasets and clicking on the button “Reserve”. The “Reset” button resets the search form to its original state and the “Export” button allows users to export the shown datasets.

The timestamps pertaining to the samples themselves are shown, when clicking on the primary key of the sample data set. This opens a pop-up screen, which shows the timestamps regarding that sample.

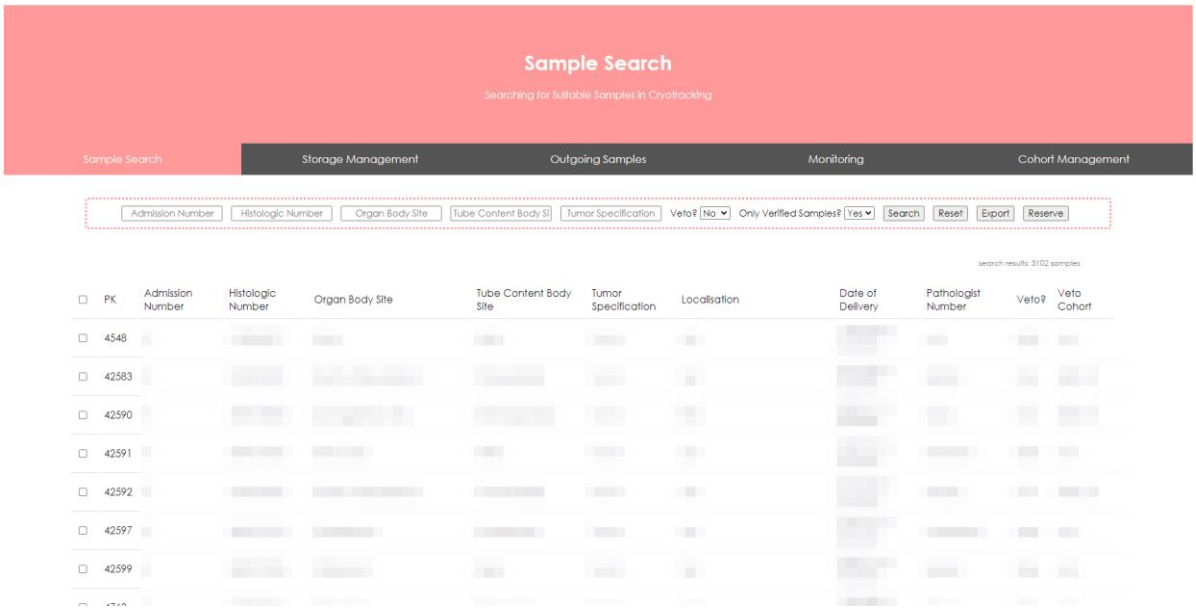



Figure 3-10: search interface for users to find suitable samples for research purposes; the datasets are blurred for data protection purposes

The second tab is called Storage Management. This can be seen in Figure 3-11, which is responsible for the sample storage management. This allows the users to find specific boxes for sample management, see the placement and storage structure in detail and find samples as needed. The sample datasets can be retrieved in the form of a table structure or shown in box structure similar to the real box, which is a grid of 9x9 tube slots, which hold the sample tubes. This gives the users the ability to quickly spot empty spaces and reorganize storage based on their needs.


Figure 3-11: storage management interface for the management of storage space for fresh frozen samples

The third tab is called Outgoing Samples and can be seen in Figure 3-12. Its primary function is to record the outgoing fresh frozen samples and make certain that the samples are marked correctly and no longer available for other research questions or purposes. The reservations from the tab sample search can be retrieved and listed by using the functions of this tab. Should single samples not fulfill the criteria needed, the reservations can be cancelled, which would mean that these samples are free to search for new purposes. By filling out the project number, the user can dispatch samples and start to fill out the rest of the auto filled delivery note, which needs to be done for outgoing delivery of samples.


Figure 3-12: outgoing samples interface for recording of outgoing fresh frozen samples

The fourth tab Monitoring can be seen in Figure 3-13. As the name already suggests, this tab gives the users the ability to monitor and supervise the incoming and outgoing samples. These are important key figures for quality management and control, and to make sure no major problems are occurring at the import or export of samples. The main key figures included in this are the number of incoming samples and number of outgoing samples with or without exclusive rights for the samples.

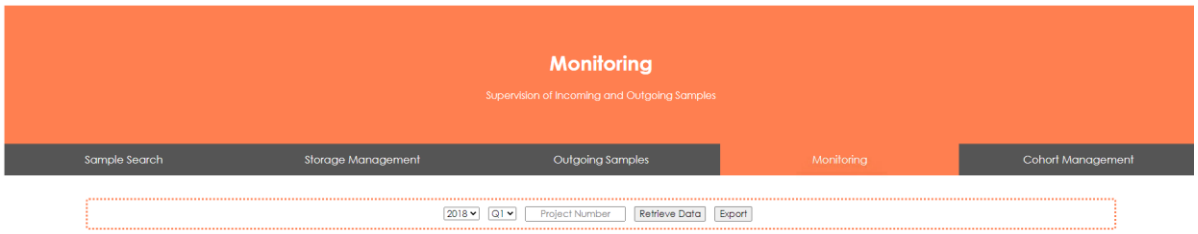



Figure 3-13: monitoring interface for the supervision of incoming and outgoing samples and retrieving important key figures for quality control purposes

The fifth tab Cohort Management can be seen in Figure 3-14. This tab offers functionality to manage and view fresh frozen cohorts. By clicking on the project number of one of the listed cohorts, the user

can retrieve all the data pertaining to that cohort and if needed mark them as reserved. Furthermore, a new cohort can be added in filling out the form on the top of the page.

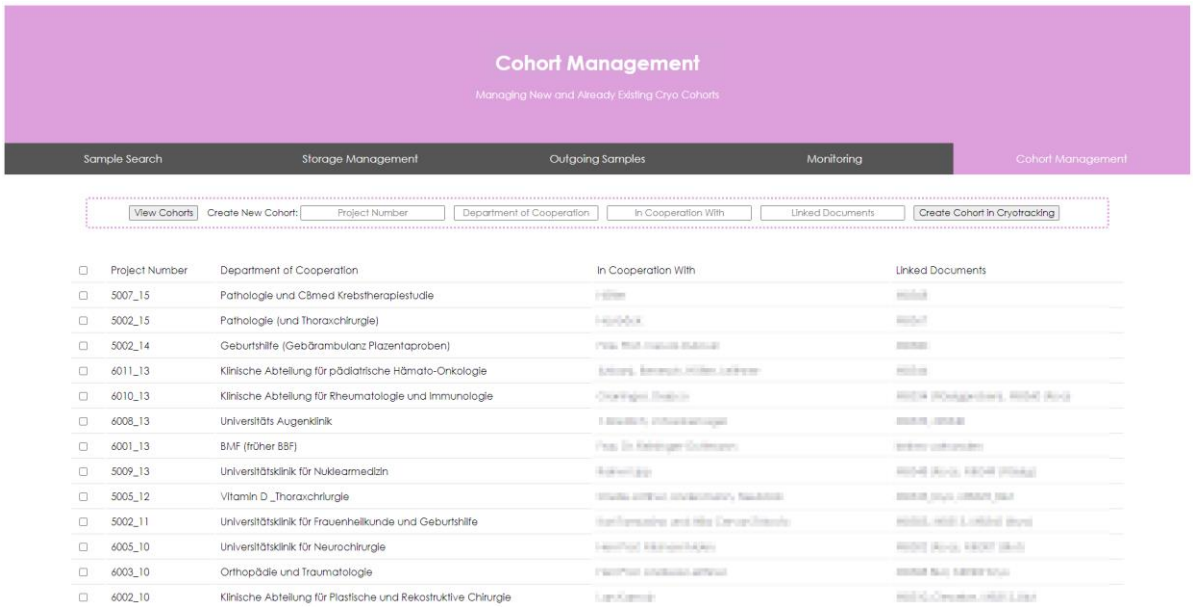


Figure 3-14: cohort management interface for the administration of fresh frozen cohorts

Before accessing the main page of the sample management interface, the user has to login to the system (see Figure 3-15). The login credentials are saved after the first sign up to the database, the password is hashed and saved as such in the database for security reasons. These login credentials are applied throughout the sample management system, e.g. for the tablets to login to the app.

Figure 3-15: login interface for cryo sample management

### 3.13.2 Establishing the Interface: Everything Is Functional

The technology behind Cryo Sample Management is a collection of Java Server Pages (JSP), which is a dynamic way for the creation of the interface as a web application. This made it possible to create one JSP homepage that includes and is linked up with all the CSS files to create the look and feel, which can be seen in Figure 3-10 to Figure 3-14.

The data output for each of the tabs are implemented in a separate JSP. The form in each tab is linked up with the correct JSP file, which outputs the data based on the entries in the form. Each form is roughly built the same, though when submitting the form, the application has to cross reference, which of the buttons in the form were clicked, to be able to function correctly. The functionality is made possible by using the Java objects and generated SQL statements, which are the building blocks for the creation of this software.

In the case of the search interface, the search functionality is a bit more complicated than it might seem at first glance. Each of the entries for the numbers in the search list (admission number, histologic number, patient ID) accept a list of numbers for easier search and collaboration between the existent systems at the biobank.

The entries for the organ, tube content and tumor specification in the search form are predefined by the user interface. The reason for this being that should users search for specific terms in these text fields, the results would not match their expectations. These entries are cleaned up in the database during data cleaning (see chapter Data Cleaning), but the predominant part of the entries for organ and tube content entries could not be fully cleaned up, due to the possibility of data loss, which meant classifying these datasets and grouping them according to their classification together.

The impact of this classification can be seen in the search form. An example: All tumor samples of the sample collections need to be retrieved from the database. There are various kinds of tumors in the database, from benign to malignant tumors and everything in between, and every one of them has a different name or tumor specification. Most of them have a specific name of the tumor written in, that a text search using the word tumor would find only a small portion of the samples.

In the search from, the user can only choose from a select few entries, which are tumor, tumor near, tumor far, normal, normal tumor near, normal tumor far and inflammation These are the classifiers for the tissue samples for tumor specifications. By selecting one of the entries, the foreign key is looked up in the database, which corresponds to that entry and all the entries with that foreign key in the tumor specification will be sought out by the software.

Similarly, this procedure is done for the tube content, organ body site and as explained for the tumor specification. This way of searching for the datasets delivers more results and offers better run times by the software.

Keeping in mind that the software is deployed on a Tomcat server, the accessible storage and server infrastructure is a valid concern, which is connected to the way the web application allocates storage during the search queries. Should a shortage of the resources on the server happen, then every application, which runs on the server is affected by that.

The testing of the JSP in its entirety got tested locally for development purposes. Subsequently, the testing of the software on the test server showed that the web application can be deployed for the use of the employees and be set into operation.

# 4 Critical Reflection

This chapter is going to address and reflect on the hybrid methodology adapted to fit the workflow at the biobank and provide more insight on whether the goal of the thesis has been fulfilled, what changed from the beginning of the thesis, and how did these changes impact the final product.

To structure this chapter a bit differently, the main points of the thesis are addressed and the role of clean code and what difference it made in applying the principles of clean code are expanded upon. Additionally, the hybrid methodology is further analyzed and compared to existing research, and which part of the hybrid methodology made the biggest change or impact for the development of the software.

## 4.1 Clean Code: Mind Over Matter

Any agile considerations touch upon being productive, never loosing time, improving and bettering processes until no time and work is wasted to meet the goals set for the given projects. To attain this level of productivity, one must have tools, which do not inhibit the work processes and one such tool is to be mindful and precise during programming sessions and aware of how coding is perceived by other developers in your team.

The book “Clean Code” by Robert C. Martin is called the handbook of agile craftsmanship and explains in detail every facet of coding and how to change the mindset to be able to generate clean code. This section will roughly explain the overarching theme of the book and summarize the main points.

The book starts by explaining what clean code is, as there are several definitions of what clean code is and what its purpose is. This includes understanding the difference between good and bad code, the mindset or attitude towards programming and how this needs to change and be much more intentional and how to act when encountering old code or bad code. The sections in the book describe naming conventions and properties of good functions, if comments are a necessary evil or should be banned from programming, what formatting has to do with programming and why a developer should care, object and data structures and error handling in general.

These practices impacted the process of developing and adapting code quite fundamentally and changed the approach to programming the software projects. The programming became an intentional and much more mind intensive task. This meant scrutinizing each line of code and each function being developed with the view of what characteristics clean code should have, and keep in mind all aspects of coding: the name of the variable, the simplicity, the functionality and the performance.

Especially at the beginning, there is a balance to be found between the productivity of the developer and adhering completely to the rules set forth by the books on clean code. In some cases, this resulted

in longer life cycles and better quality of code, which meant an overall improvement in understanding and effectiveness of the end result.

Additionally, the goal is not simply to create a great end product, but create usable code, which is by far more important, because this has the ability to be future proof and make it adjustable and readable by other developers, so that they have the ability in turn to improve and update the code.

## 4.2 The Biggest Impact: Recognizing Barriers and Finding A Custom Solution

Reflecting on the project, there is one approach that can be singled out from the rest, which had the most impact on the whole of the project.

During the act of programming, the developer is constantly problem solving and search for solutions to a given problem within the given requirements. This project made it easier to address the barriers and to overcome these using agile methodologies and practices from agile software craftsmanship.

Conversely, it is not the practices themselves that impacted the project at its core. While developing the software and the functions for each of the software parts, the project lacked flexibility. This means, should one of the variables change or another variable need to be added to the dataset, the developer would need to change not only the database, the batch jobs, the graphic user interface and the recording of the sample data, i.e. no part of the sample management system could be spared from having to be adapted in such a case. It became the prime example of a rigid and demanding project, which would require endless hours for maintenance and upkeep, after going live with the project.

Added to that, although the methodologies were agile and adaptable to the workflow, the workflow itself had to be changed to become more flexible and more forgiving in its nature.

How could this be done? The first step is to recognize the problem and where it lies in the case of emergency. The problem zone in this case turned out to be the central point, where all the systems interconnected: the datasets and the database itself. The solution would need to automatically update all the functions and objects for all the projects, which would change, should the database change. The updated objects and the functions could then be updated for each of the projects, which would create less work and friction, especially when creating a new sample management system, where the datasets change occasionally.

This scripting software, which was developed after dealing with changing requirements during the Implementation phase, was programmed to create the Java objects and SQL statements for a specific database schema. The developer would indicate the schema login details, and the scripting software would connect to the database and retrieve all the data and metadata about the schema from the database and generate the objects and SQL statements, which are from now referred to as resources.

These resources contain one object for each table in the schema and one Java class containing all the SQL statements per table. The SQL statements included basic data manipulation and retrieval functions, i.e. insert, select and update functions for each of the variables in the tables and objects themselves.

So, by programming this piece of software, the problem of nonexistent flexibility has been recognized and dealt with, which resulted in a more flexible and adaptable sample management system. Dealing with the change in datasets means that the database is adapted by the developer and the scripting software generates the new resources, which can be included as a new .jar file to the other projects. It saves time, effort and thought processes by the developer, as the generated code adheres to the guidelines of clean code and agile software craftsmanship.

## 4.3 Hybrid Methodologies: A Trending Topic?

It was mentioned in the Introduction that according to a survey in 2016, most of the development engineers were satisfied with the agile-waterfall hybrid model. But why would developers prefer to work with “old” development methodologies, when there are newer and better ways to develop projects?

Rightfully so, many companies and engineers try to find solutions, which work best for them and their workflow and don’t concern themselves with the idea of which new development model is on the rise again.

Even though there are new ways to develop and code, not everyone has the same requirements and the same tools at hand to be able to adhere or manage a project or an organization according to a specific methodology. The focus should be on whether the existing methodology works, and if not, to change and adapt to a new form of content management.

Assuming that the latest conferences on agile methodologies are the conferences that cover the newest technologies and research questions on agile methodologies, one can get a feeling of the covered topics in the agile research communities.

To follow that thought, here are some topics from the conference Empowering Agile Insights from October 2020, which gave an insight on the practical findings from Austrian companies [36] and the conference Agile World: Enabling Business Agility, which is a German based conference from July 2020 [37]:

- What kind of leadership does an agile organization need?
- Who is driving the agile transformation?
- How to embrace business agility?

- Work council, personnel and agility – a major catastrophe?
- Long-term thinking beyond sprints: sustainable agility starts with awareness
- Playing with matchsticks at work – how unauthorized experiments contribute to transformations

These are just some of the examples pulled from two agile conferences from this year, and as a rough overview, all these topics focus most their attention to the management and organizational leaders and not on the detailed smaller parts of developing software.

Even though this is a small sample size and not at all representative of the bigger picture, the idea of a hybrid methodology is not a new concept, and methodologies might change in the future though why change a system, which works for their own organization.

It may be an advantage to change the mindset of agile methodologies and see it as a toolbox full of tools, which help to make the management of developers and organizations easier, rather than a rulebook, which needs to be followed to gain the envisioned results.

## 4.4 Process Analysis: Avenues For Better Outcomes?

Developing a new sample management system according to the agile methodologies and practices turned out to be a positive experience and creating a high-quality result. However, are there processes, which could have gone better, or even improved the outcome of the development process?

First improvement is referring to the decision-making process. The scope of the project became bigger and better the more the users and personnel collaborated with the developer, especially as they kept on describing the requirements and necessary functions needed for the employees to do their jobs. From the point of view of the developer, the scope of the project expanded and changed to encompass a lot of different tasks that could not have been avoided, as many of the tasks were rightly part of the project. The management should have considered to prioritize the software of the fresh frozen samples above other tasks or projects, which would have made a difference in the scope and management of the project tasks themselves.

The second improvement concerned the changing requirements of the projects, after the requirements analysis phase. The management could have decided that these changes were unnecessary and preferred the prior configuration of the project. The changes would have been included in a further iteration of the project. Although the final product would have been in use much earlier, the users might have needed the aforementioned changes to effectively work with the software.

The third improvement would have been in the communication of the employees. This did change over the course of the project, as the input of the employees regularly flowed into the implementation and development of the project. One of the missteps, where communication is concerned, is the project management for the development of the inventory software. The communication before starting the development should have included more through documentation and meetings with the users, before programming the software, of which the first iteration sadly was not used for its purpose.

## 4.5 Hybrid Methodology: The Focus Points

As described above and in the Methodology section, the focus areas of this topic were to answer whether a hybrid agile model could work in a one-developer type setting, documentation questions and risks and roadblocks during development. These three overarching topics are going to be discussed in this section.

### 4.5.1 Single Developer Applying Agile Development Methods: A Smart Choice?

The majority of agile methods rely on a team working together to reach a common goal of completing a project. To make such a thing possible, many of the practices and methods are geared and created for productivity and frictionless teamwork, which is why Scrum and similar models became popular in recent years. Though single developers cannot attempt to apply Scrum methodologies because of their inherent use cases for teams and inter-team communication. Incorporating agile methodologies into the development life cycle is possible, but the question still lies in whether this should be done or not.

The decision of how to manage the development process should be a decision made by the developer themselves. The knowledge of how the company operates and how the individual fits into that picture establishes the requirements for the tasks. This further emphasizes that the workflow and the decision of how to manage the workflow should stay with the individual doing the work.

Agile methods bring great benefits to the development process, no matter the size of the team, be it one or ten employees. The wide range of practices and methods are existent, due to the programmers finding ways to create unique processes, which update and improve the adaptability of the workflow and their own productivity.

Customizing a model to their requirements is the definition of a hybrid model. The use of hybrid models has been around since the beginning, and not using such methods would just impact developers negatively in having to adhere to practices, which do not and cannot be fulfilled by the company or themselves acting as the sole owner of such a project.

The latter statement referring to working on a development project alone cannot be realized, in any way. The developer seldomly can only work on the required material given, without communicating with other team members or customers to verify the plans and gain more insight into the use of the software itself.

Conversely, the development of a project can be the sole responsibility of one developer, which refers to that developer being the only person assigned to a specific project. One must remember that the sole responsibility does not lie completely with the developer, but also with the management and other key players of the company.

In general, it is not a smart choice to give a single developer the main responsibility of a major project, because many factors can inhibit the progress and success of this project, should the management of the company not act accordingly and make sure that the developer has resources available to manage the project accordingly.

To summarize, giving the developer the tools to customize the workflow to their needs can only be an advantage for the company. Resting the responsibility and the development of new software on only one person is possible and accomplishable, but the management of the company needs to keep the resources and other projects in mind, when choosing to follow that path.

### 4.5.2 Documentation Reaches A Minimum: Has True Agility Been Accomplished?

"*The best documentation is self-documenting code and an intuitive user interface. -- a Bellevue Linux Users Group member, 2005*" [38]

Software documentation explains the concept of documenting the programmed function or software in further documents to write out the functionality and use cases in plain English. The view of software documentation is varied, and many would assume that should documentation reach a minimum, then the ideal scenario has been accomplished and we have reached the finish line with the aid of agile methodologies. This ideal goal does not exist, but there are ways to minimize the software documentation.

Applying the rules and methods learned by Robert C. Martin, one can learn to code in a clean manner, which makes the code readable and understandable, without having to use comments in the coding process to describe the action or function being developed. Understandable and clean code is the first step towards a minimalist approach to documenting your software.

The second step means designing the graphical user interface in such a manner that the users don't need a voluminous manual to guide them through the process of operating the software. As mentioned in the starting quote of this segment, an intuitive user interface contributes to the success of the software and the almost non-existent need for documentation.

The third step is non-negotiable documentation, which is written by the users or the developer. This kind of documentation is bound to the practice of the hybrid approach and encompasses the documentation of the requirements, the use cases and project specific forms for the company. These are written in a concise and simple manner to record the requirements and refer back to those, while being in communication with the management and the personnel.

In applying those three steps, any developer will become a documentation minimalist. Still, are these steps all it takes to reach the plateau of agility? No, it does not, though it is a step in the right direction. The changes mentioned in this section focus only on the developer and don't take the broader aspects

of the roles in the company and roles developers need to fulfill outside the projects into account, as well as their environment.

To recap, true agility is a myth and a goal that is not reachable by human standards. However, there are ways to modify one aspect of a project, in this case documenting the software differently, which will have an impact in how the flow and progress of the project is perceived.

### 4.5.3 Risks & Roadblocks: Developing Workflows and Software Projects Simultaneously

Developing and adapting the way the software, as it is being developed, is not recommended. The risks include time delays, changing priorities and the unfortunate risk of not finishing the project at all, considering all the changes occurring as a result of changing the workflows and requirements.

The minimalist approach to documentation has been helped in keeping up with the changes in requirements and being able to switch the direction of the development completely, due to the software being wide footed and adaptable, as intended.

The multiple changes impacted the project differently than expected. Changes in the workflow were predominantly perceived to be positive in nature and helped the workflow evolve to mold itself to the developer's needs.

Changes coming from the project itself came with a lot of problems and adaptability issues, where certain requirements are added that did not become known until then. This changed the structure fundamentally in a way, that in two cases two parts of the project were scrapped completely and the development started anew for those parts of the project.

## 4.6 Existing Limitations of the Sample Management System

The sample management system in its current form has limitations, like any other system. Reasons for this are often that the requirements in the beginning of the development did not cover a certain part of the system in such a detail that some functions were either missed or could not be implemented due to other factors.

One such limitation is the inability of the users to adapt the workflow for themselves on the tablets. This particularly stems from the various data acquisition probabilities and angles, where and how samples can reach the biobank and depending on the type of sample, the data acquisition differs in some instances. All of known workflows of sample acquisition are documented and are pictured in the recording software. In the case of a new type of sender or something similar, the developer is responsible for the adaptation of the software. In chapter Conclusion and Future Work, this is mentioned as one of the areas of the software that requires more attention.

Another limitation regarding the whole management system are the operating systems that are part of the system. There are a few different operating systems, which are part of the whole structure: The servers, which run the graphic user interface and batch jobs, operate on Windows machines, and the tablets run on Android Marshmallow 6. The upgrade cycle of Windows machines is of importance to keep track, what drivers or software is supported by windows itself and by the updates, which are installed regularly.

Additionally, the Android operating system deals with updates and new releases differently to Windows. Tablets and phones are part of a separate upgrade cycle, similar to other brands in this branch. Similarly, to Apple or other mobile software operating systems, the upgrade cycle of software and their iterations happens at minimum in most cases once a year. The mentioned operating system in use for the tablets is Android Marshmallow and launched in 2015 and corresponds to the API 23 (short for: Application Programming Interface). Since then more iterations of the operating system have come out, the last current one being from September 2020 called Android 11.

The operating system for tablets does not impact the performance of the custom software on the tablet, should the device not be updated by the users or the organization. In updating the device and their operating systems, the developer needs to check and update the code to make sure that the software supports and functions with the new API. The same idea corresponds to future plans to buy new hardware in the form of tablets, which means that the app software has to be refactored for the preinstalled API of the hardware. Such a process cannot be automated, as the customization has to be dealt with by the developer.

In a similar manner, the programming language can become a limitation of the development itself. A primary example are the licensing fees for the newer versions of Java. The licensing, support and release models changed in 2019, where certain models of the newer java version were no longer provided free of charge. This meant for this project in particular that an older version of Java is utilized to impart the indispensable functionality for Java projects. These facts hinder the possibilities in which the management system can evolve and continue to improve, should the system remain a purely Java based operation.

Finally, the last limitation is the lack of information recorded at the biobank on the sample quality before reaching the biobank. This kind of tracking information is found in other systems outside the biobank, which in future will be accessible if need be through its own interface connected to the biobank systems.

## 4.7 Sample Management Systems: A New Frontier?

Sample management systems have been implemented by many companies worldwide. A few examples have been listed in the Introduction of this thesis. Many companies might buy already implemented software. But should something change in the existent workflow or not work as needed, then the software company, where the software was bought from, can either refuse to adapt the software or adapt and change the software according to the wishes of the users. The adaptation of the software is inherently connected to costs and fees that the biobank in this case must pay.

Biobanks are companies, which require the acquired samples to have an excellent data and sample quality, so that the sample itself is of use for the researchers. Should that not be the case, the purpose of the biobank would disappear, and such a repository of samples would not be needed anymore.

The sample management system makes it possible to deliver and uphold those high standards. However, why not just see whether such a system is already existent? As mentioned in the Introduction, the similarity is given in the first few descriptions of the mentioned software implementations found online, but more or less each of the software companies include different processes and workflows in the software, which would not be of use or applicable for any processes of the biobank in Graz. Two examples will be taken and discussed here further, one is the Micronic Traxcer software and the second example is the FreeLIMS software.

Figure 4-1 shows a snapshot of the Micronic Traxcer software that deals with the scanning of the tubes of Micronic boxes or plates. [39]

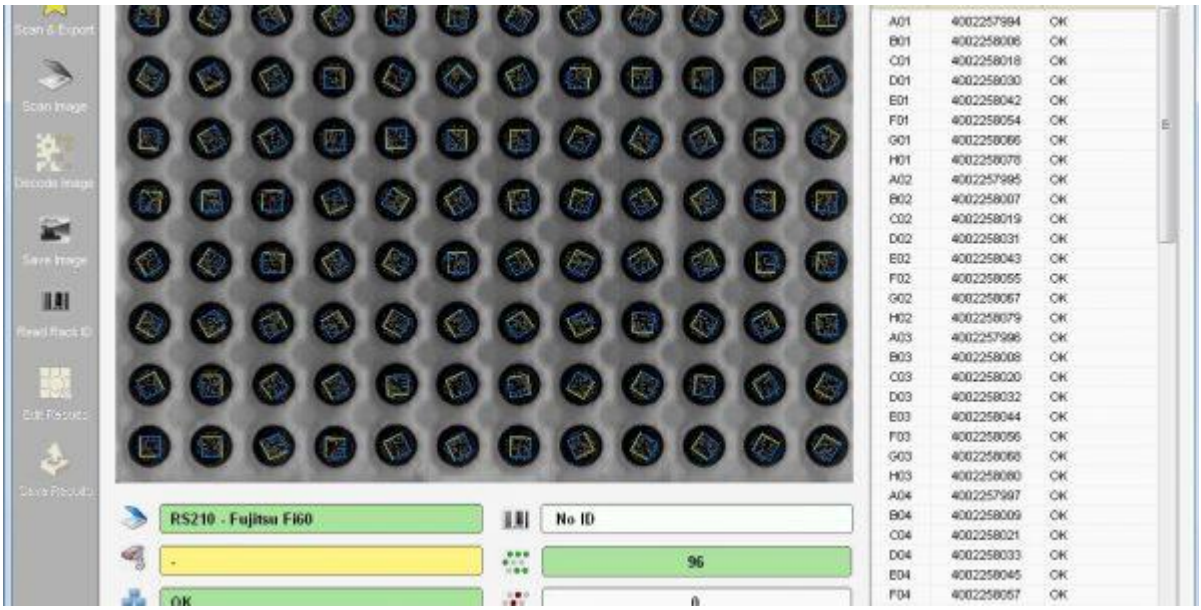



Figure 4-1: snapshot of the Micronic Traxcer software that is operated in conjunction with the Micronic scanner, which scans and manages the file scans of the barcoded boxes from Micronic

In Figure 4-2, a snapshot of the FreeLIMS software can be seen, where an example shows how the storage containers management is implemented and how to assign locations using the FreeLIMS software. A hierarchical structure in the form of a tree view is shown that lists the boxes or racks on the left and tube information of the boxes or racks on the right-hand side. In this snapshot, the software seems to manage liquid samples and indicate where the box is filled with a sample, and where the box has empty spaces, which can be filled by the employees. [40]

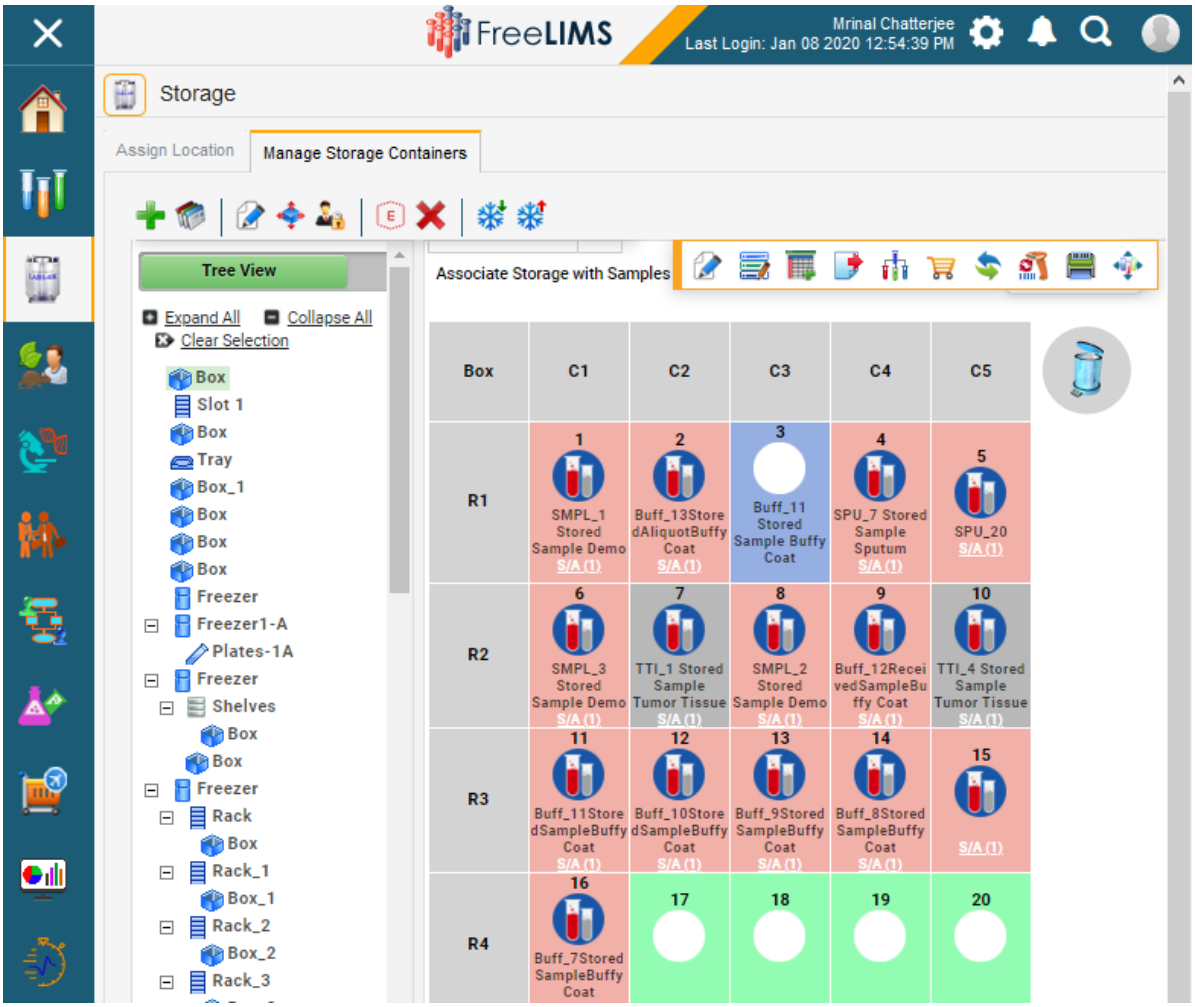


Figure 4-2: snapshot of the FreeLIMS software that manages liquid samples and shows a tree view for managing storage containers

While both applications can be of use for the users, the switching and cross referencing of applications from the datasets that are existent in files and those recording the files is additional work for the employees of the biobank.

The Micronic Traxcer software is purely a data generator, which scans the barcodes of the samples and the plate and generates a file with the placements of the tubes and the barcode of the box or plate. Conversely, the software from FreeLIMS is a management software, which has the capabilities to create sample structures and show storage in specialized views, and much more. However, the graphic

user interface is better suited for scheduling and processing lab values than the management of samples. This is an instance, where the software just lacks features, which are essential to the biobank and the use cases, and supports the idea of the users having to adapt to the workflow of the software and not the software conforming to the workflow of the users, which is the advantage of a inhouse development of a sample management system.

Although the biobanking field is quite a young field, it is rapidly expanding. While it is truly a growing and developing field, it is still a niche market, if the global view of such organizations is considered. The similarities and differences of LIMS and sample management systems only really concern the biobanks, which have to have other tools at their disposal, when dealing with a high number of samples.

The way in which sample quality and samples themselves are viewed is changing every year, recent worldwide pandemic developments in sample acquisition included. This might just be an area of research, which will improve the way medical research is done at the moment, because both samples and research are inherently linked together, because one cannot have one without the other.

# 5 Conclusion and Future Work

This section is going to reflect on the two main topics or the thesis, summarize the key points and recommend future adaptations or related work that could be done to further improve the existing projects.

The completed work encompassed both a sample management system for fresh frozen samples and a hybrid development model for developing software at the Biobank Graz, which is part of the Medical University of Graz and deals with the collection, storage and management of human samples for research purposes.

Regarding the hybrid development model, the research aimed to identify whether a hybrid methodology could work in a waterfall only organizations with specifically set limitations. Based on the critical analysis and reflection on this topic, it can be concluded that the custom methodology is effective in developing software in such circumstances and representations world-wide have shown that users tend to gravitate towards adaptable and adjustable systems.

To shortly summarize, the custom hybrid development model is a hybrid waterfall-agile model, which is identical to the waterfall model up until the implementation phase of the development, where the agile practices adapt and change further steps in the methodology to customize the development to match the requirements of the organization. These agile practices primarily were part of Extreme Programming methodologies, with the caveat that practices needed to be achievable for a team with only one developer assigned to this project.

This latter distinction made the choosing process of the agile practices much easier, as some requirements for the application of the practices could not be met. Because of that reason, some practices were not included in the final methodology.

Applying this custom methodology resulted in increased time spent on development tasks, improvements in the quality of resulting code and the productivity of the developer. The inter-work communication evolved and created a better working relationship with the users and the customers, who could trial and error the code, even in prototype stages.

While the hybrid methodology only brought positive results for the development of the project, there are two things that need to be addressed: First, the custom approach to the development of software is nothing new, as these methodologies have existed for years and have been implemented from companies worldwide. Second, the main goal for the thesis regarding the methodologies was the question on whether a hybrid methodology with agile practices could be developed and used by only one developer working on this project and whether this could work in a waterfall only organization.

This goal is accomplished, as the project has been implemented with many subtasks and changing requirements by only one developer with great success.

Future adaptations of the hybrid methodology could be possible depending on if the dynamic in the organization changes. The hybrid methodology created for this thesis meant to work as a proof of concept that the developer could work on his or her own on this project, with the developer being the central point of contact between all the groups in the organization. In the case of the dynamic changing, the management will decide the next steps regarding the development process and how projects should be approached, i.e. which development methodology should be adhered to.

The project implemented with the hybrid methodology mentioned above, is the sample management system developed for fresh frozen samples. The final form of the project was created to record samples, find samples according to research questions, manage incoming and outgoing samples, track samples every step of the way and deliver key figures for the employees of the biobank.

Regarding the implementation of the sample management system, the research aimed to realize a custom management model to make it possible to abandon the old legacy systems and start fresh with a more advanced management system.

This was accomplished through a central sample management web interface, an Android app for the recording of the data sets, multiple iterations of inventory software and a central database to store the data sets centrally. These separate parts of the system work in tandem with each other and are updated, should the data in the database or the database itself be adjusted.

The created system can tell the employees, which events occurred during the recording of sample data, i.e. from the handover of the sample to the employee until it reaches the final storage container. These features are the reason for the increase in sample quality. An important lesson learned: The reason being that a sample without data is useless, and can be thrown out, because the sample would have lost all its value, should the data not be connected to the sample and its storage place.

There were factors that made the development problematic, such as a high percentage of tubes, i.e. the containers of the samples, having a handwritten label and no barcode. This meant finding a solution for samples and how to track these according the newest guidelines, plus how to track the samples in the same way an employee would handle and track tubes with a barcode. Another factor that resulted in the scope of the project getting larger, was the inherent understanding that the storage information of samples and tubes could not be trusted according to the legacy sample management system, which meant having to program a sample inventory program to inventory all the samples, to see which samples were placed where in the storage tanks.

These are only some of the hurdles in the development of the sample management system, however with the hybrid methodology and agile practices, these hurdles and problems were dealt with to deliver a sample management system, where many of the tasks can be done by the employees themselves without having to ask the developer for their support in using the developed systems.

Future work must be addressed as well. There are some parts of the management system that are still in development. One such part would be the administrative graphic user interface for advanced users of the software, so that they can adjust the dropdowns and predetermined choices for each of the studies and cohorts. Furthermore, something similar should be implemented for the admins of the software, to keep an eye on the functions of the software, plus should bigger parts need to be fixed, there should be a central management system available for this system, to manage administrative tasks from the IT side.

Another part for future adaptations would include an updated connection to the wireless local area network. Right now, the data is encrypted and sent via a wireless connection to a separate server, from where the data is retrieved, checked and imported into the database. The way in which the Android tablets connect to the network and upload data is still very complicated. It could function easier should the company invest in a different kind of wireless area network. This would make the data transmissions easier to handle and the systems' susceptibility to errors decrease.

Added to that, this could further improve the data exchange between the servers and the tablets, which are responsible for the generation of the dropdowns for the tablet software. While developing the tablet software, the constant desynchronization and unavailability of current data due to the security of the servers had to be handled with care and precision, where the developer estimated the extent of synchronization, which was available, and how to achieve the optimal version for the users to utilize.

Finally, the tablet software in its entirety still is not fully deployed for the users. The functionality and main use cases have been programmed according to the specifications, though the required functionality for the new standardization of biobanking has not yet become part of the data acquisition process. These changes for the tablet include the timestamp generation in the background and the import of the datasets into the database, simultaneously to the data generated by the files. In the future that quality aspect needs to be part of the XML file, which is transferred from the tablet using the local wireless are network to the servers.

Connected to that, the backup strategy for the sample acquisition process is not fully realized. The planned backup strategy intends to use the same process for data transmission for the backups. That would mean transferring the saved datasets once a day to the servers to add another layer of

protection against data loss. Decisions on whether the backups are only file based and saved as files to the server, or if the files should be saved differently is still not finalized, whereas the database and everything else around it is saved daily for data protection purposes.

While future innovations are planned and surely will become part of the newer system preferences, the research of this thesis has spanned from theoretical aspects of planning and developing software to actually creating a product that is being used by employees of the biobank at the Medical University of Graz. These biobanks being organizations, which collect various types of biological samples and must have a flexible and adaptable sample management in place to be able to track and manage all these samples, without losing any data.

Through this work, the adaptability of hybrid models has been investigated and shown to be appliable in the medical engineering and IT setting. At the same time, a sample management system for fresh frozen samples got implemented, to give the users the ability to manage and handle samples according to high quality and security standards. This ensures that the data acquisition during the biobank procedures is seamless and that the quality of the samples does not degrade over time, while keeping the costs at a minimum.

The quality and security standards are partly responsible for the tracking of samples needed, by keeping track of the sample activities or events (e.g. changes in temperatures, changes in storage containers, etc.), managing of datasets according to international standards and making sure that the confidentiality of the data and their connection to datasets is preserved.

Furthermore, the GDPR concerns are nonexistent, due to the pseudonymized datasets being shown to the users of the software and should any names of employees be part of the dataset, these have been similarly pseudonymized for the security of everyone involved.

To sum up, the main idea of this thesis has been accomplished and while there are still parts that can be improved and adapted over time, the developed functionality cannot be found at any other similar company to manage sample collections and their datasets in such a way.

# I. Listing of All Figures

# II. References


[1] "About Biobank Graz." https://biobank.medunigraz.at/about-biobank-graz/ (accessed Dec. 03, 2020).

[2] G. J. Race, G. W. Tillery, and P. A. Dysert, "A History of Pathology and Laboratory Medicine at Baylor University Medical Center," *Bayl. Univ. Med. Cent. Proc.*, vol. 17, no. 1, pp. 42–55, Jan. 2004, doi: 10.1080/08998280.2004.11927956.

[3] J. B. Vaught and M. K. Henderson, "Biological sample collection, processing, storage and information management," p. 20.

[4] M. Cornes *et al.*, "European survey on preanalytical sample handling – Part 1: How do European laboratories monitor the preanalytical phase? On behalf of the European Federation of Clinical Chemistry and Laboratory Medicine (EFLM) Working Group for the Preanalytical Pha," *Biochem. Medica*, vol. 29, no. 2, pp. 322–333, Jun. 2019, doi: 10.11613/BM.2019.020704.

[5] D. J. Emge, "Cryosectioning Techniques: The Basics." www.feinberg.northwestern.edu/research/docs/cores/mhpl/TissueFreezing.pdf (accessed Aug. 08, 2016).

[6] "Frozen section procedure," *Wikipedia*. Sep. 01, 2020, Accessed: Dec. 02, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Frozen_section_procedure&oldid=976157863.

[7] "What is a Laboratory Information Management System (LIMS)? - Definition from Techopedia," *Techopedia.com*. http://www.techopedia.com/definition/8085/laboratory-information-management-system-lims (accessed Nov. 27, 2020).

[8] G. A. Gibbon, "A brief history of LIMS," *Lab. Autom. Inf. Manag.*, vol. 32, no. 1, pp. 1–5, May 1996, doi: 10.1016/1381-141X(95)00024-K.

[9] "Sample Management Systems," *Micronic*. https://micronic.com/product-category/sample-management-systems/ (accessed Nov. 27, 2020).

[10] "Micronic's is commited to serving scientists - Get to know us," *Micronic*. https://micronic.com/about-micronic/ (accessed Nov. 27, 2020).

[11] "Laboratory Sample Management System," *WhiteSci*, Apr. 01, 2014. https://whitesci.co.za/brand/micronic/laboratory-sample-management-system/ (accessed Nov. 27, 2020).

[12] "Free LIMS | Lab Management Software | FreeLIMS." https://freelims.org/ (accessed Nov. 27, 2020).

[13] "Laboratory Information Management Systems (LIMS) | Advantages for genomics labs." https://www.illumina.com/informatics/sample-experiment-management/lims.html (accessed Nov. 30, 2020).

[14] "Manifesto for Agile Software Development." https://agilemanifesto.org/iso/en/manifesto.html (accessed Oct. 03, 2020).

[15] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *J. Syst. Softw.*, vol. 85, no. 6, pp. 1213–1221, Jun. 2012, doi: 10.1016/j.jss.2012.02.033.

[16] M. Kuhrmann *et al.*, "Hybrid Software Development Approaches in Practice: A European Perspective," *IEEE Softw.*, vol. 36, no. 4, pp. 20–31, Jul. 2019, doi: 10.1109/MS.2018.110161245.

[17] "How to Make Agile and Waterfall Methodology Work Together," *ReQtest*, Dec. 05, 2016. https://reqtest.com/agile-blog/agile-waterfall-hybrid-methodology-2/ (accessed Apr. 26, 2020).

[18] S. Ajmal and S. Ali, "AGILE-WATERFALL HYBRID MODEL FOR SOFTWARE DEVELOPMENT PROCESSES," p. 6, 2016.

[19] M. Z. Nafchi, H. Zulzalil, and T. J. Gandomani, "On the current agile assessment methods and approaches," in *2014 8th. Malaysian Software Engineering Conference (MySEC)*, Sep. 2014, pp. 251–254, doi: 10.1109/MySec.2014.6986023.

[20] V. Anes, A. Abreu, and R. Santos, “A New Risk Assessment Approach for Agile Projects,” in *2020 International Young Engineers Forum (YEF-ECE)*, Jul. 2020, pp. 67–72, doi: 10.1109/YEF-ECE49388.2020.9171808.
[21] M. T. Sletholt, J. Hannay, D. Pfahl, H. C. Benestad, and H. P. Langtangen, “A literature review of agile practices and their effects in scientific software development,” in *Proceeding of the 4th international workshop on Software engineering for computational science and engineering - SECSE ’11*, Waikiki, Honolulu, HI, USA, 2011, p. 1, doi: 10.1145/1985782.1985784.
[22] R. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*, 1st ed. Upper Saddle River, NJ: Prentice Hall, 2008.
[23] R. C. Martin, *Agile Software Development, Principles, Patterns, and Practices*, 01 Auflage. Upper Sadle River, NJ: Pearson Ptr, 2011.
[24] “Honeywell Completes Acquisition Of Hand Held Products, Inc.” https://www.fieldtechnologiesonline.com/doc/honeywell-completes-acquisition-of-hand-held-0001 (accessed Nov. 12, 2020).
[25] “Hand Held Products,” *Wikipedia*. Aug. 21, 2020, Accessed: Nov. 12, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Hand_Held_Products&oldid=974172732.
[26] “Hand Held Products Inc - Company Profile and News,” *Bloomberg.com*. https://www.bloomberg.com/profile/company/25990Z:US (accessed Nov. 12, 2020).
[27] “History of software engineering,” *Wikipedia*. Sep. 18, 2020, Accessed: Nov. 10, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=History_of_software_engineering&oldid=979082731.
[28] “The Software Crisis.” http://www.chris-kimble.com/Courses/World_Med_MBA/Software_Crisis.html (accessed Nov. 11, 2020).
[29] “Infographic: A Brief History of Software Development Methodologies,” *Intetics*. https://intetics.com/blog/a-brief-history-of-software-development-methodologies (accessed Nov. 10, 2020).
[30] I. Sacolick, “What is agile methodology? Modern software development explained,” *InfoWorld*, Feb. 25, 2020. https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html (accessed Oct. 31, 2020).
[31] “What is Waterfall Model in SDLC? Advantages & Disadvantages.” https://www.guru99.com/what-is-sdlc-or-waterfall-model.html (accessed Oct. 31, 2020).
[32] “What is Legacy Software and Legacy Systems - Overview.” https://www.stromasys.com/2016/07/an-overview-of-legacy-software-and-legacy-systems/ (accessed Nov. 12, 2020).
[33] “Asus Zenpad Z380M-6A024A 20,3 cm Tablet-PC dunkelgrau: Amazon.de: Computer & Zubehör.” https://www.amazon.de/Z380M-6A024A-Tablet-PC-MediaTek-Graphics-dunkelgrau/dp/B01HIH1H8Q (accessed Nov. 12, 2020).
[34] “Definition of warm ischemia time - NCI Dictionary of Cancer Terms - National Cancer Institute,” Feb. 02, 2011. https://www.cancer.gov/publications/dictionaries/cancer-terms/def/warm-ischemia-time (accessed Nov. 06, 2020).
[35] “Definition of cold ischemia time - NCI Dictionary of Cancer Terms - National Cancer Institute,” Feb. 02, 2011. https://www.cancer.gov/publications/dictionaries/cancer-terms/def/cold-ischemia-time (accessed Nov. 06, 2020).
[36] “Empowering Agile,” *Empowering Agile*. https://www.empoweringagile.com (accessed Nov. 22, 2020).
[37] “‘Enabling Business Agility’ - Das Programm - Agile World 2020,” *Agile World*. https://agileworld.de/programm/ (accessed Nov. 22, 2020).
[38] “best quotes about software documentation.” http://www.linfo.org/q_documentation.html (accessed Nov. 19, 2020).
[39] “Tracxer-04-RS210-scan-no-single-tube-read-600x300.jpg (JPEG-Grafik, 600 × 300 Pixel).” https://micronic.com/wp-content/uploads/2020/08/Tracxer-04-RS210-scan-no-single-tube-read-600x300.jpg (accessed Nov. 27, 2020).

[40] “Biobanking LIMS Software | Biobank Management Software.” https://freelims.org/industries/biobanking-lims-software.html (accessed Dec. 05, 2020).