UNIVERSITY OF VAASA

SCHOOL OF TECHNOLOGY AND INNOVATION

AUTOMATION AND COMPUTER SCIENCE

Anton Åstrand

RE-ENGINEERING A DATABASE DRIVEN SOFTWARE TOOL

Rebuilding, automating processes and data migration

Master's thesis for the degree of Master of Science in Technology submitted for inspection, Vaasa, 17 February 2020.

Vaasa 17.02.2020

Supervisor	Ph.d. Teemu Mäenpää
Instructor	M.Sc. Johanna Ilonen

ACKNOWLEDGMENT

The research was made in collaboration with the Compliance & Certification team in Wärtsilä. Close cooperation with the members of the team was held during the whole research and it would not have been possible without them. Therefore, I would like to thank everybody who had a part in the meetings and discussions regarding the project and the research. In addition, special thanks to Johanna Ilonen who was my instructor during the thesis and gave good support throughout. A thank you also goes to my supervisor Teemu Mäenpää for his support during the whole process.

TABLE OF CONTENTS

ABBREVIATIONS AND TERMINOLOGY

EIAPP	Engine International Air Pollution Prevention
NOx	Nitrogen Oxide
SQL	Structured Query Language
IMO	International Maritime Organization
MARPOL Annex VI	MARPOL Convention Protocol
NTC	NOx Technical Code
DBMS	Database Management System
RDBMS	Relational Database Management System
NoSQL	Non SQL or non relational
DML	Data Manipulation Language
DDL	Data Definition Language
MS	Microsoft
UI	User Interface
VBA	Visual Basic for Application
IT	Information Technology
T-SQL	Transact-Standard Query Language
SQLOS	SQL Server Operating System
WFI	Wärtsilä Finland
WIT	Wärtsilä Italy
SCR	Selective Catalytic Reduction

UNIVERSITY OF VAASA
School of technology and innovation
Author: Anton Åstrand
Topic of the Thesis: Re-engineering a database driven software tool
Supervisor: Ph.d. Mäenpää, Teemu
Instructor: M.Sc. Ilonen, Johanna
Degree: Master of Science in Technology
Degree Programme: Degree Programme in Energy- and Information Technology
Major of Subject: Automation and Computer Science
Year of Entering the University: 2014
Year of Completing the Thesis: 2020 **Pages: 98**

ABSTRACT

This thesis aims to re-engineer a database driven software tool that is used to insert engine related data and generate an EIAPP Technical File that is needed for certification of marine engines to show that they comply with IMO's emission regulations specified in MARPOL Annex VI and NTC 2008. The need for an updated tool has emerged as the way of working is to be changed, from document management to content management. The current tool is also divided into two different tools, one for engines built in Italy and one for engines built in Finland, which leads to another objective that is to merge these tools into one. The tools are built-in Microsoft Access which does no longer suit the needs. Therefore, the last purpose of the research is to conduct a data migration from Microsoft Access to SQL Server.

The research was divided into theoretical and empirical research. The theoretical part first presented the theory behind software engineering and software re-engineering. Then the theory behind databases and data migration was explored to at last go through the emission regulation and certification for marine diesel engines to better understand why the tool is needed. In the empirical part, first, the existing tool and the certification process were inspected. Furthermore, the research method, the constructive research approach was discussed, that focuses on producing a construction (solution) to a real-world problem in practice. At last, a more in-depth analysis of the tool was made to propose a plan on how to re-engineer the tool, which included an implementation process plan.

The main result of this research is a re-engineered EIAPP tool that has the front-end in Microsoft Access and back-end in SQL Server. The tables have been restructured to comply with the change to only use one document number for the whole Technical File. The forms have been redesigned and processes have been automated to make the tool more reliable and efficient. The new re-engineered tool has more than 50 % fewer objects and fewer lines of code compared to the two existing tools. In addition, the research provides suggestions on how to further develop the certification process and the tool.

KEYWORDS: Software re-engineering, Database driven software, Microsoft Access, SQL Server, Data migration, EIAPP

VASA UNIVERSITET
Tekniska fakulteten
Författare: Anton Åstrand
Titel: Omstrukturering av ett databasdrivet mjukvaruverktyg
Handledare: Ph.d. Teemu Mäenpää
Instruktör: M.Sc. Johanna Ilonen
Examen: Diplomingenjör
Utbildningsprogram: Utbildningsprogrammet inom Energi- och Informationsteknik
Inriktning: Automation och datateknik
År för påbörjande vid Vasa universitet: 2014
År för färdigställande av avhandlingen: 2020 **Sidor: 98**

ABSTRAKT

Denna avhandlings syfte är att återutveckla ett databasdrivet mjukvaruverktyg som används för att sätta in motor relaterad data och generera en EIAPP Teknisk Fil som krävs för certifiering av motorer för att visa att de uppfyller och följer IMO:s utsläppsbestämmelser som anges i MARPOL:s bilaga VI och NTC 2008. Behovet av ett uppdaterat verktyg har uppkommit eftersom strukturen och arbetsättet skall ändras, från dokumenthantering till innehållshantering. Det nuvarande verktyget är också indelat i två olika verktyg, ett för motorer byggda i Italien och ett för motorer byggda i Finland, vilket leder till ett annat syfte som är att slå samman dessa verktyg till ett. Verktygen är byggda i Microsoft Access som inte längre passar behoven. Därför är det sista syftet med forskningen att utföra en datamigrering från Microsoft Access till SQL Server.

Forskningen delades in i teoretisk och empirisk forskning. Den teoretiska delen presenterade först teorin bakom mjukvaruteknik och omstrukturering (re-engineering) av mjukvara. Sedan undersöktes teorin bakom databaser och datamigrering för att till slut genomgå utsläppsreglering och certifiering av marina diesel motorer. I den empiriska delen inspekterades först det befintliga verktyget och certifieringsprocessen. Vidare diskuterades konstruktiva forsknings strategin, som fokuserar på att producera en konstruktion (lösning) till ett verkligt problem i praktiken. Till sista gjordes en mera djupgående analys av verktyget för att föreslå en plan för hur man skall omstrukturera (re-engineer) verktyget, som inkluderade en implementeringssprocessplan.

Huvudresultatet av denna forskning är ett omstrukturerat EIAPP verktyg som har frontend i Microsft Access och backend i SQL Server. Tabellerna har omstrukturerats för att uppfylla ändringen i att bara använda ett dokumentnummer för hela tekniska filen. Formerna har omarbetats och processer har automatiserats för att göra verktyget mera tillförlitligt och effektivt. Det nya omstrukturerade verktyget har mer än 50 % färre object och färre kodrader jämfört med de två befintliga verktygen. Dessutom ger forskningen förslag på hur man kan vidareutveckla certifieringsprocessen och verktyget.

KEYWORDS: Ombyggnad av programvara, databasdriven mjukvara, Microsoft Access, SQL Server, Data migration, EIAPP

1 INTRODUCTION

Software engineering and software development is a big part of today's society and new technologies are being introduced at a rapid pace (Sommerville 2016: 3). Software is according to Roger S. Pressman (2010) the most important technology in the world today and as the software engineering field is changing fast it means that older programs and tools struggle to meet the requirements, reliability and efficiency goals set today. Therefore, this thesis will present and analyze a database driven software tool and re-engineer it according to new requirements and new ways of working. The re-engineering also includes data migration to migrate the data to a new environment.

1.1 Background

The research topic is suggested by the Compliance & Certification team that is part of the Research & Development and Engineering department in Wärtsilä. They are responsible to ensure that Wärtsilä's engines and automation systems are certified, and type-approved according to customer needs and environmental requirements.

One part of the teams' work is making a so-called Technical File for EIAPP (Engine International Air Pollution Prevention) certificates. A short explanation is that the engines are being tested for NOx (Nitrogen Oxide) emissions, using reference conditions and fuels and the tests are performed according to ISO 8178 cycles. The certificates are then issued for each engine showing that the NOx level complies with the Annex VI to MARPOL 73/78.

For the creation of the Technical File document, a tool called the EIAPP Tool, has been developed around 2005 in Microsoft Access. Where Microsoft Access is a database management system that combines a graphical user interface with a database. From then on the tool has been updated and maintained based on new requirements and needs. The data in the tool is inserted into forms and stored in tables. The data is then printed in different reports that then all are merged into one file, which is the Technical File. The Technical

File document is then being sent to different classification societies that will approve or disapprove the document. The Technical File includes all the essential information that is needed to get the certificates.

The main task of the research and the project is to re-engineer the EIAPP tool, which includes data migration. Other tasks are to optimize the way of working and automating processes (if possible) of making the Technical File. As the tool is old, there are problems, and a big problem is that the tool has been divided into two tools (for the most part identical) because of Access limitations. One tool used for engines built in Finland and one for engines built in Italy. Therefore, when updates are needed, often the changes must be done in two different tools, leading to double the work. Another rework is needed as until now the tool is built around document management, this means that there are document numbers, one for each report, where one report is a subpart of the whole Technical File. The purpose now is to make it work around content management, which means to stop use document numbers for the different parts, only use one for the whole Technical File. This has been chosen due to a new system is under development for the creation of the actual Technical File document, this means that the reports in Microsoft Access will not be used anymore. Why the use of many document numbers has been done is the limitation Microsoft Access has regarding reports because it is not possible to make long reports, therefore, the Technical File has been divided into different sections that all have a separate report and a separate document number. The new reporting tool that is being developed in parallel with this project does not have the same limitations and therefore, only one document number is needed for the whole Technical File.

The data migration to SQL Server has been chosen because of the benefits that SQL Server has over Microsoft Access, but also because of future plans for the EIAPP tool. The future development plan is to go away from Microsoft Access because of its limitations and its problems and instead use a web application. Because of the different engine groups, engine types, and different emission regulations, a lot of information in the forms are dynamic and therefore, has created problems in the past in Access. All the differences between the two database software's will be discussed in the thesis.

1.2 Objectives

The objective of the research is re-engineering the existing tool, this includes changing the way of working from document management to content management, but also improving the tool by looking at if there are possibilities to automate functions when inserting data. Furthermore, another objective is to move the data from the Microsoft Access database over to a SQL Server. As this research is including both planning and implementation, the research questions are built around that and have been formed as followed:

- How to re-engineer the tool when going from document management to content management?

- Is it possible to automate more functions to speed up the work for users to insert data?

- How to realize the data migration from two Microsoft Access tools into one and from Microsoft Access to SQL Server?

There are many questions that all have the same goal of answering how to improve the tool in the best possible way. The physical outcome of the research is a re-engineered tool where the user interacts with forms in Microsoft Access and the data is stored in a SQL Server. Another outcome should be a better-made tool, where more functions are automated to speed up the work inserting data, which will make the tool more efficient and reliable.

1.3 Structure

In addition to the Introduction chapter, there will be six more chapters in the thesis that will be as followed: Theory framework, Related work, Description of the existing tools, Analyzing and planning, Results and next steps, and Conclusion and future development. The theory framework chapter will dig deeper into software engineering, software re-

engineering, databases, data migration, and emission regulation and certification for marine diesel engine, by exploring related scientific literature. The related work chapter is needed to understand what type of research there has been conducted around this topic to see where the gaps are. To be able to identify the current problems and what is needed for the new tool, a chapter about the existing tools is needed to further explain and present what the tool does and what it looks like. The planning chapter will then consist of the development plan on how to re-engineer the tool, this includes a more in-depth analysis of the tool and descriptions and discussions on what is needed to be done and in what order. The last chapters are then the result chapter where the results are discussed and the next steps and at last, the conclusion and future development chapter to conclude the thesis and suggest further development.

1.4 Company description

Wärtsilä is a Finnish company that is a global leader in complete lifecycle solutions and smart technologies for the energy and marine market. Wärtsilä maximizes the economic and environmental performance of its customer's power plants and vessels by emphasizing total efficiency, data analytics, and sustainable innovation. In 2019, Wärtsilä had approximately 19 000 employees, operated in over 80 countries with more than 200 locations all around the world. (Wärtsilä 2020.)

The company consists of two businesses; Energy Business and Marine Business. Wärtsilä Marine Business is providing integrated solutions and innovative products that are environmentally sustainable, safe, flexible, and efficient. Wärtsilä Energy Business aims at a future of 100% renewable energy. They are optimizing their customer's energy systems and future-proofing their assets. They offer energy management systems, storage, flexible power plants, and lifecycle services. (Wärtsilä 2020.)

2 THEORY FRAMEWORK

In this chapter, the theory will be studied to support the re-engineering and data migration project. First software engineering will be presented to understand the fundamentals of software development and software process models. After that software re-engineering will be studied and it will include software re-engineering processes and software re-engineering approaches. Databases are the next field that will be presented to then better explain and discuss Microsoft Access and SQL Server. Then data migration will be explored and presented to then lastly, go through emission regulation and certification for marine diesel engines to better understand what the tool is used for and why it is needed.

2.1 Software engineering

Software engineering has been described by Ronald J. Leach (2016: xxi) as "*the application of engineering techniques to develop and maintain software that runs properly and is constructed in an efficient manner*". Another description by Ian Sommerville (2016: 21) is that software engineering is an engineering discipline that is concerned with the whole software life cycle, from the system specification to maintaining it after deployment. In other words, software engineering takes into account schedule, cost, dependability issues, and the needs of software producers and customers. The techniques and methods used in software development are based on the organization, what type of software it is and the people developing it, which means that there are no universal software engineering practices that work in every company and software project. (Sommerville 2016: 24.) Furthermore, the end goal of software engineering is to write programs that are: efficient, reliable, usable, modifiable, portable, testable, reusable, maintainable and correct (Leach 2016: xxiii).

Software and software engineering is a crucial part of today's society and the current modern world cannot function without professional software systems, because for example cars, airplanes and energy all rely on complex computer systems. As the need for software will only increase, software engineering will have an essential role in meeting

the increasing needs for complex, affordable and high-quality software. (Sommerville 2016: 3; Leach 2016: xxi.)

As with any other engineering discipline, there are problems with software development and projects and the main ones are that they are delivered late and over budget (Sommerville 2016: 3). Therefore, it is very important to understand the basic concepts of software engineering and some of the problems could be avoided if the correct methods and techniques were used (Sommerville 2016: 3; Leach 2016: 1). Software engineering methods contribute to the technical aspect of how to build software. The methods include tasks such as requirement analysis, communication, design modeling, program construction, testing, and support. (Pressman 2010: 14.)

2.1.1 Software activities

A collection and sequence of software activities can also be called a software process. The software activities done in a systematic approach helps in creating the software product, where the activities strive to achieve the objective and it is used in all software processes, regardless of complexity, domain, and size. (Pressman 2010: 14; Sommerville 2016: 23.). The fundamental activities according to Sommerville (2016: 23) and Munassar and Govardhan (2010) are:

Software specification, where the engineers together with the customers or stakeholders define and specify the software that is to be produced. It should be a comprehensive and complete description of the behavior of the software and functional and non-functional requirements are defined. Use cases are often used to describe the users' interaction with the software. (Bassil 2012.)

Software development or Software design, where the planned software is designed and coded according to the software specification. Here the business requirements are realized into an executable program, in the form of, for example, a database or a website. (Bassil 2012.)

Software validation and verification, where the software product is checked to make sure it is done according to the customer's needs and requirements. Verification is the evaluation of the software after each step to check that it satisfies the conditions made at the beginning of the step. Then validation is done at the end of the development process and is the process of checking that the software satisfies the specified requirements. (Bassil 2012.)

Software evolution, where after the software is deployed it is modified and maintained according to the changing market and customer requirements. Steps here are also to correct errors, improve reliability, quality and performance. (Bassil 2012.)

These activities are complex activities and can be divided into subactivities such as requirements, validation and unit testing (Sommerville 2016:44). There are many different variations, but the fundamentals are the same and another example is from Roger S. Pressman (2010: 15), he writes about five activities: communication, planning, modeling, construction, and deployment. He describes that the use and the details of the activities differ from project to project, and dependent on the software process that was chosen for the project the activities can be used once or in iterations.

2.1.2 Software process models

As described in the last section, software processes are sets of related activities and the processes differ from company to company as there is no universal process that can be used in all software projects (Sommerville 2016: 44). Knowing that the activities described in the last section will be used during a project does not tell in what order and when the activities occur. They can occur once and in a specific order where each activity is completed before the next one or many of them can occur at the same time or they can occur many times during the same project. The order and the timing of the activities can be described by a so-called software process model, also called a software development model or software development life cycle model. (Leach 2016: 13-14.) The models are abstract high-level descriptions of software processes that help explain the approach to software development (Pressman 2010: 31; Sommerville 2016: 45).

There are many different models and most of them have been established between 1970 and 1999, especially the waterfall model, iteration model, v-shaped model, spiral model and extreme model (Munassar & Govardhan 2010). Other examples of models are the incremental model, integration and configuration model, rapid prototyping model, agile development model (Leach 2016: 14; Sommerville 2016: 45-46). These are just a few examples, therefore, all cannot be discussed further in this thesis and for this reason, two models are chosen, the waterfall model and, the integration and configuration model, as these are two common traditional models that differ from each other.

The waterfall model is a traditional and classic model of software engineering. The model is one of the oldest models and is widely used by major companies and in government projects. This model is based on planning in the early stages which leads to that it ensures design flaws before the development phase. The model works well for projects where quality concerns are a high priority because the model includes intensive documentation and planning. (Munassar & Govardhan 2010.) The name waterfall comes from the phase transition as you cascade from one phase to another. The model is plan-driven and in principle, you schedule and plan every activity before starting with the software development. (Sommerville 2016: 47.) There are many variations of the model but one example is presented below:

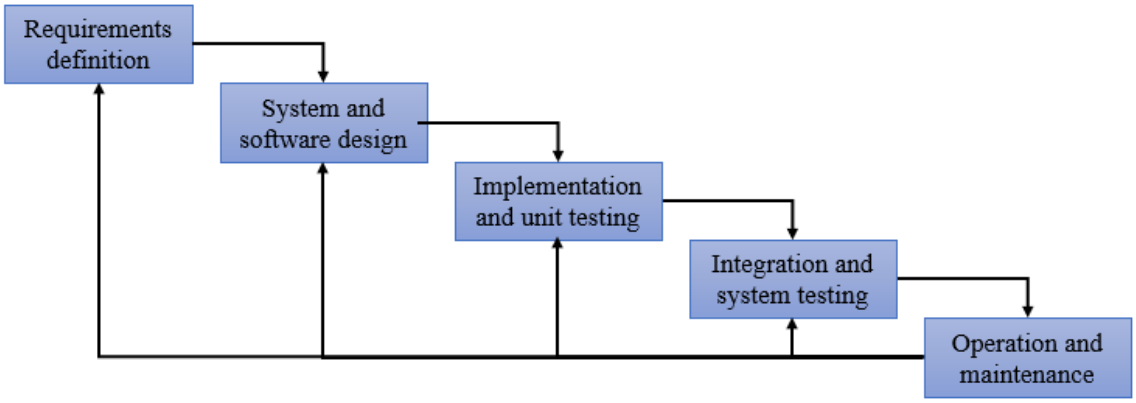

Figure 1. One example of the waterfall model based on (Sommerville 2016: 47)

The different stages seen in the figure is further explained below (Sommerville 2016: 47-48):

1. *Requirements definition* The first stage when discussions with users are held to establish the system's goals, constraints and services and from them in detail define the system specification.

2. *System and software design* The system design process form the overall system architecture, this by allocating the requirements to either software or hardware systems. The software design describes and identifies the relationships between the fundamental software system abstractions.

3. *Implementation and unit testing* This stage involve implementation of the system by making a set of programs or program units, by realizing the system design. The separate units are then verified to make sure they meet the specification.

4. *Integration and system testing* The different program units or parts are integrated to make the final system and is then tested as a complete system to check if it meets the requirements set. After the tests are done successfully the software system is delivered to the customer.

5. *Operation and maintenance* This is the last and often the longest stage in the lifecycle of the software and here the software is maintained and errors are fixed that were not found in earlier stages. This stage also includes improvements and enhancing the system if new requirements arise.

In each stage, documentation is made to document the requirements and objectives for that specific stage. At the end of each stage, a review is held to see if the requirements are met and if it is possible to proceed to the next stage. There is no overlapping of the stages and the waterfall model is based on the idea that first the requirements for the whole system are made and after the design is made before starting to code the solution. It is not prohibited to return to an earlier stage and can, therefore, make it costly to rework if issues are found in later stages of the development. (Munassar & Govardhan 2010.)

According to Munassar and Govardhan (2010), the advantages with the waterfall model are that it is easy to understand and implement and also as it has been around for a long time it is widely used and known, at least in theory. Another good part about the model is that it reinforces good habits because you define before you design and design before you code. Lastly, the model identifies milestones and deliverables. They also discuss the weaknesses of the model and firstly as the development comes late in the process, the results are seen late and therefore, management and customers can get disconcerted. Moreover, it is very expensive to make changes to documents and it is a very document-heavy model. Lastly, the model does not reflect the iterative nature of development projects and often the initial requirements made are not the most accurate. (Munassar & Govardhan 2010.)

The integration and configuration model, based on reuse-oriented software engineering is a model based on reusing software or software components. In the majority of software projects, they reuse software and since 2000, models based on software reuse have become widely used such as this model. Same as with the waterfall model, there are many variations of this process model but one example can be seen here below: (Sommerville 2016: 52.)

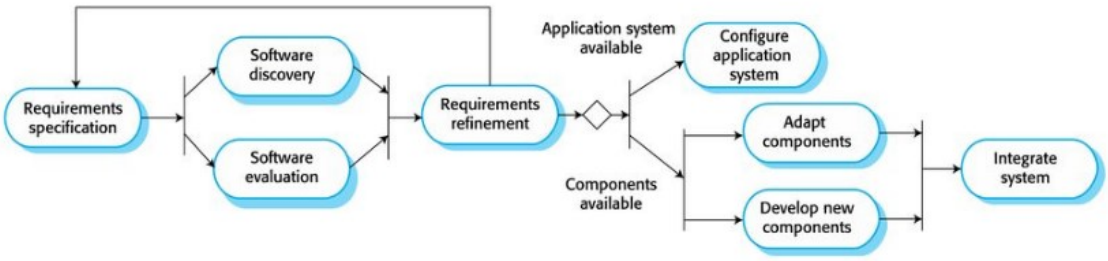

Figure 2. One example of the integration and configuration model (see Sommerville 2016: 52)

The process model is based on five stages (Sommerville 2016: 52-53):

1. *Requirement specification* Here the requirements for the whole system are suggested. They do not have to be described in detail but have the essential information and in this stage, the desirable features.

2. *Software discovery and evaluation* With the use of the requirements proposed in the first stage a search is made for components and applications fulfilling the functionalities needed. The components are then evaluated if they meet the requirements and in general if they suit the system.

3. *Requirements refinement* With the information about the reusable components and applications gathered in the previous stage, the requirements are refined. The requirements are modified to match the newfound components and the whole system specification is reformulated. If modifications cannot be done, the last stage needs to be re-entered again to find new solutions.

4. *Application system configuration* If an already made application is found that meets the requirements, it can be configured for use and be the new system.

5. *Component adaptation and integration* If no application is found in the second and third stage that meets the requirements, the individual components can be modified and new components developed to create the new system.

This model and others based on software reuse often have an advantage as it reduces the amount of software and code to be developed, this should lead to reduced costs, risks, and faster delivery of the software. However, the disadvantages are then that often compromises are made in the requirements and it can lead to software that does not fully meet the users' needs. (Sommerville 2016: 53-54.)

2.1.3 Prototyping

Prototyping has a big part in software development and is used in the design phase of the software project (Al-Husseini & Obaid 2018). A prototype is a first version of the software that is used to try out design options, demonstrate concepts, and find out more about the solution to the requirements and the problems (Sommerville 2016: 62-63). A prototype can have many different forms, it can be a presentation, a paper, or everything up to

an exact version of the future software. Nowadays, most development environments allow the developer to create some sort of prototypes, but then the prototype is usually connected to a programming language. (Sommerville 2016: 62-63; Al-Husseini & Obaid 2018.) Prototypes help the developer by allowing the users to give feedback early on in the software development, and by this find strengths and weaknesses and even get new ideas for requirements. Moreover, when the prototype is developed, it can reveal errors in the requirements made for the software. Prototyping is an essential part of user interface development as user interfaces are by nature dynamic and therefore, diagrams and textual descriptions are not enough to explain and test the user interface requirements. The objective of prototyping should be stated at the beginning of the process because a prototype can be used in many different ways. A prototype can prototype the user interface, to validate functional system requirements, or it can demonstrate the feasibility of the system but the same prototype cannot meet both objectives. (Sommerville 2016: 62-63.)

2.2 Software re-engineering

Maintaining software is about understanding the software to know what to change and how to implement new features, but for old legacy systems, this can be hard and therefore, software re-engineering is needed (Sommerville 2016: 276). Xiaohu Yang, Lu Chen, Xinyu Wand and Jerry Cristoforo (2014) describe a legacy system as a system that has been in use in an organization for a long time and the technology used is outdated. They also point out that an organization with a legacy system eventually has to evaluate options on how to go forward with the system. The options are then to either search for a vendor based system with similar functionalities or re-write the software with a new platform for example. Software re-engineering or also written software reengineering is according to Manar Majthoub, Mahmoud H. Qutqut, and Yousra Odeh (2018) the process of changing or enhancing existing software so it can be managed, reused and understood as new software. Another definition by Elliot Chikofsky and James Cross II (1990) "*reengineering is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form*". The reengineering process importance

lies in the ability to reuse and recover parts of the outdated software system. Moreover, it leads to lower maintenance costs and it sets up the system for future developments. (Majthoub, Qutqut & Odeh 2018.) Re-engineering is also known as reclamation and renovation and the reengineering process often includes other processes such as redocumentation, reverse engineering, restructuring, refactoring, forward engineering, and translation (Cikofsky & Cross II 1990; Sommerville 2016: 276).

There are more and more systems being developed but systems developed from scratch are decreasing, but the use of legacy systems is very high. The main goal of re-engineering is not to change the overall functionality of the software but the context of the new system can change, for example, the application environment or system-level hardware. Enhancements to the functionalities can be done but they should be done after the re-engineering part is completed. There are four general objectives of re-engineering: (Rosenberg n.d..)

- Preparation for functional enhancement
- Improve maintainability
- Migration
- Improve reliability

Re-engineering should not be used to enhance the functionalities of a system but it can be used to prepare the system for enhancements. Legacy systems often during the years of modifications and enhancements get difficult and expensive to change and therefore, it needs to be re-engineered for further enhancements. Furthermore, reliability and maintainability also often get critical in legacy systems and re-engineering will improve it. Lastly, as the computer industry grows at a fast rate, new software and hardware systems are being introduced and older systems can get outdated fast. Therefore, migrating to newer operating systems, hardware platforms or coding languages will improve the software. (Rosenberg n.d..) The main advantages of software re-engineering are reduced risk

and reduced cost. Because, there is a huge risk in developing new software, where system specification and development problems can occur that can lead to delays in the deployment of the software and moreover, increased costs. (Sommerville 2016: 276.)

As with the other software engineering process models discussed earlier in the thesis, the reengineering process also has many different variations and two examples will be discussed further.

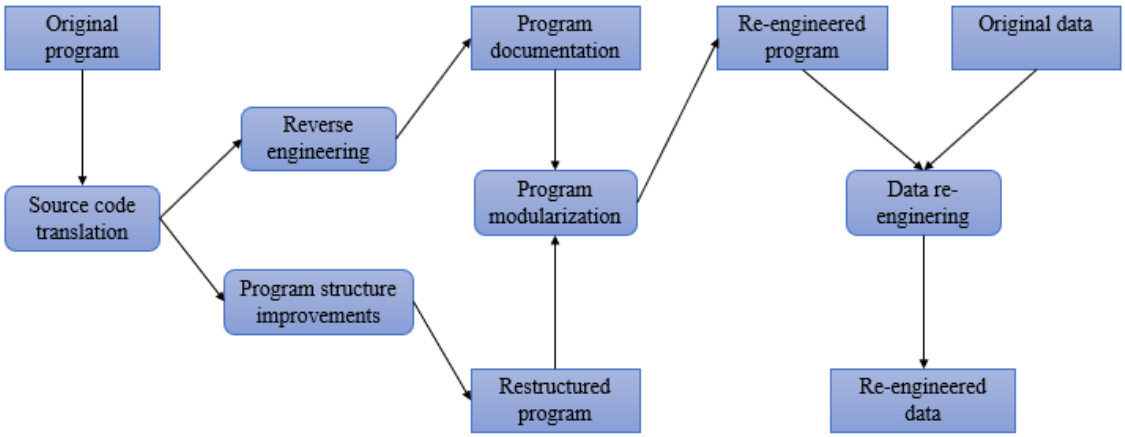

Figure 3. One example of the reengineering process (see Sommerville 2016: 277)

The model by Sommerville (2016) is based on five main activities: source code translation, reverse engineering, program structure improvement, program modularization, and data reengineering. The source code translation is about changing the coding language to a new version of the language used or a totally new one, this with the use of a translation tool. The program structure improvement activity involves analyzing and modifying the control structure of the program to make it easier to understand. Program modularization is where related parts are grouped together and redundancy is removed if possible. This stage can also involve architectural refactoring, which means that if the program is using many different data sources it can be refactored to use only a single repository. Data reengineering is where the data processed in the program is being changed to reflect the changes made during the other steps. This can mean converting existing databases to new structures and redefining database schemes. In this step, you should also clean up the data, which may mean finding and correcting mistakes and removing duplicates. Lastly, the

reverse engineering activity will be discussed later as it can be found in both example models and are a vital part of re-engineering. (Sommerville 2016: 277.)

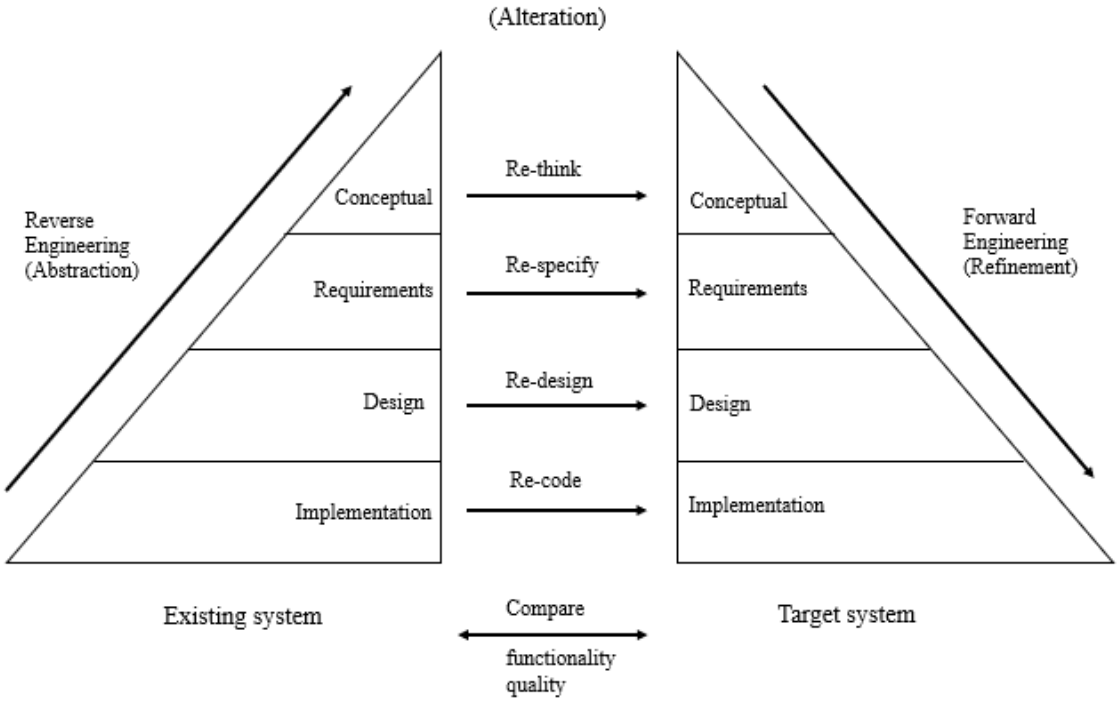

Figure 4. Another example of a traditional model of re-engineering software (see Majthoub et al. 2018, based on Rosenberg n.d.)

This second model by Majthoub et al. (2018) based on Rosenberg (n.d) covers three principles of re-engineering: abstraction, alteration, and refinement. Abstraction is a continuous increase in the abstract level of the system, this you can see in figure 4, where you start with implementation and goes to conceptual which is more abstract. This movement is called reverse engineering. The alteration is about making changes to the system without changing the degree of abstraction, this can include, modification, deletion, and addition of information, but not functionality. Refinement is the continuous decrease in the abstract level of the system, by replacing the existing system information with more detailed information. This movement is called forward engineering. (Rosenberg n.d.)

Reverse engineering is according to Cathreen Graciamary and M. Chidambaram (2018) a process where the system is analyzed to identify its components mechanisms, and their relationships with each other and with that information create representations of the system at a higher level of abstraction or in a different structure. Reverse engineering usually involves a functional system but that is not a requirement and the reverse engineering process can start from any abstraction level or at any stage of the life cycle. Reverse engineering is not about change or creating new systems, it is about the examination of the subject system. It also involves subareas and two that are widely used are, design recovery and redocumentation. The main task of reverse engineering is to recapture the structure, requirements, content, and design of the legacy system and the key objectives are to recover lost information, generate alternative views, detect side effects, and facilitate reuse. (Chikofsky & Cross II 1990; Rosenberg n.d..)

Forward engineering is the traditional forward or downward movement, from a high-level of abstraction to low-level abstraction such as the physical implementation of the system. Forward engineering is the forward movement of the standard software development process, for example, the waterfall model, and the word forward has only come to use to distinguish it from the process of reverse engineering. (Chikofsky & Cross II 1990; Rosenberg n.d..)

2.2.1 Sofware re-engineering approaches

Rosenberg (n.d.) lays out three different approaches to software re-engineering, and all of them have their own risks and benefits. The main difference between the approaches is the time of replacement from the existing system to the target system.

The first one is the **big bang approach**, which can be seen in figure 5. This approach is used when the whole system is replaced at once, this is often needed when there is a problem that needs to be solved immediately, for example, if the system needs to be migrated to another system architecture (Rosenberg n.d.).

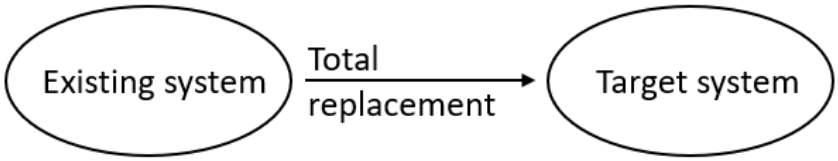

Figure 5. The big bang approach (see Rosenberg n.d)

The advantage of this approach is that the whole system is changed at once, which means that it is moved to the new environment at once. This leads to that there is no need for any interfaces between the old and the new components and there is no need to maintain two environments at once. The drawback and disadvantage this leads to are that this approach may consume a lot of resources and it can take a lot of time to produce the new system, which means it is expensive. Another major problem is that there are likely to be changes made to the old system when the new one is in development and these changes are then also needed to be done in the new system, which leads to double the work for these changes. (Rosenberg n.d..)

The second approach is the **incremental approach**, an illustration explaining this approach can be seen in figure 6. This approach is about taking the existing system and re-engineer sections of it and then add those incrementally as new versions. It brakes down the parts to re-engineer according to the sections in the existing system (Rosenberg n.d.)

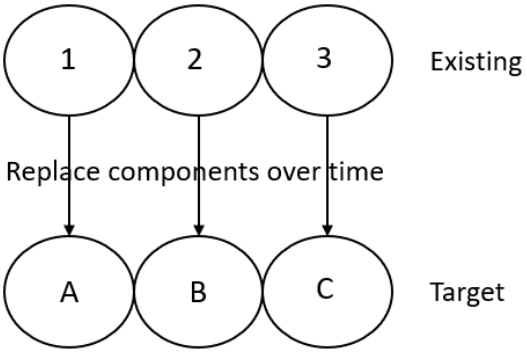

Figure 6. The incremental approach (see Rosenberg n.d)

The advantages with this approach are that components are being developed separately, this leads to that they are produced faster and errors are easier to trace during the development. Another advantage compared to the big bang approach is that changes to the old

system can be dealt with easier as changes to components, that are not re-engineered have no impact. A disadvantage with this approach is that the system probably takes longer to complete when dealing with multiple versions that all need configuration control. This approach has a lower risk than the big bang approach because the system is dealt with in sections and it is easier to monitor the risks for each component separately. (Rosenberg n.d..)

The third and last approach is the **evolutionary approach**, which is illustrated in figure 7. This approach is similar to the incremental approach in the fact that it uses sections to replace the existing system with the re-engineered one, but the difference is that the chosen sections are based on their functionality and not on the structure of the existing system (Rosenberg n.d.).

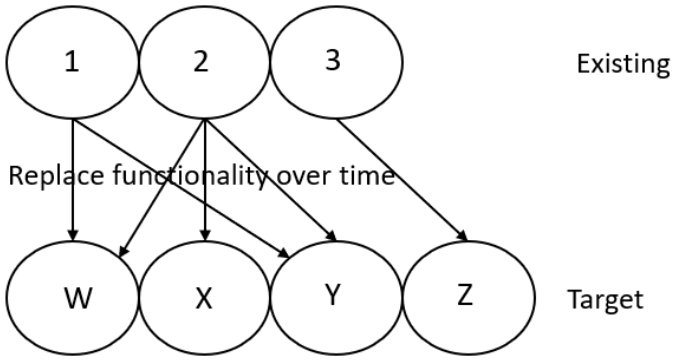

Figure 7. The evolutionary approach (see Rosenberg n.d)

The advantages of this approach are the modular design that the result of this approach gives and also the reduced scope for the components. This is a good approach for re-engineering projects where the goal is to convert to object-oriented technology. The disadvantages are that similar functionalities throughout the current system must be identified and refined to a single functional unit and interface problems can occur since the re-engineered sections are functional and not architectural. (Rosenberg n.d..)

2.3 Databases

Databases can be found everywhere in today's society and most people interact with databases daily. Activities that include databases are for example when you make flight or hotel reservations, withdraw or deposit to the bank, buy something online, or even when buying items at a supermarket. These examples are for the most part interactions that we may call traditional database applications that mainly store information that is either textual or numeric. When social media became popular they required new huge databases that could store audio clips, images, and video streams, these new types of a database system is often referred to as NoSQL (non SQL) systems or big data storage systems. There are also available many types of database systems, for example, the geographic information systems (GIS) can store and analyze satellite images, weather data, and maps. Another example is online analytical processing (OLAP) and warehouse systems that are used to support businesses in decision making by analyzing information from very large databases. (Elmasri & Navathe 2016: 3-4.)

For the purpose of this chapter, some words and things need to be discussed and defined and a general database definition that Ramez Elmasri and Shamkand B. Navathe (2016: 4) and Thomas Connolly and Carolyn Begg (2015: 52) is using is that "*a database is a collection of related data*". Data, in this case, is known facts that have an implicit meaning and can be recorded. For example, addresses, names, and phone numbers, but as written in the last paragraph it can nowadays also be images and video streams. A database management system (DBMS) is a software that controls and manages the interaction with the database. A database application is a program that can interact with the database and a database system is a collection of applications that together with the DBMS interact with the database. The common use of the meaning of a database is more restricted than the general definition and a database has the following properties: (Connolly & Begg 2015: 52; Elmasri & Navathe 2016: 4-5.)

- A database reflects the real world in some aspects, sometimes called the universe of discourse (UoD) or the miniworld. Changes to this miniworld have to be reflected in the database.

- A database is a collection of data that has some sort of inherent meaning. The data has to be logically coherent and if the data is randomly assorted it cannot be referred to as a database.

- A database is built and designed for a specific purpose and the data is populated to accommodate the purpose. The database also has an intended group of users and some preconceived applications.

This means that a database has some source where the data is obtained from and some sort of interaction with the real world. The users of the database can, for example, make a transaction, which can be that a customer buys a product or it can be an event, such as an employee is sick, which then changes the information in the database. (Elmasri & Navathe 2016: 4-5.)

A database can be used by many users and departments simultaneously and a database holds not only data but also a description of this data, often called the system catalog or metadata. The approach of a database system is very similar to the software development approach, and also a re-engineering approach, by having both an internal definition of an object and a separate external definition. The user of the database system sees only the external definition of an object and not the internal part on how the object is functioning and defined. This approach is also called data abstraction and it means that the internal object can be changed without a user noticing it as long as the external object works the same. In other words, you can add and modify fields in a database without affecting the application program, but if we remove a field from the database the application program is affected and has to be modified to cope with the change. (Connolly & Begg 2015: 63.)

Another expression that needs to be defined regarding databases is “logically related”, this refers to entities, attributes, and relationships and all these have to be identified when analyzing the information need of a database that is going to be developed. An entity can be explained as a distinct object, for example, a place, person, or a thing. An attribute is a property of the entity that describes the object in some aspects. A relationship is a connection between the entities that describe the association between them. Moreover, the

database holds these entities with attributes and the relationships between the entities, in other words, the database controls and holds data that are logically related. A popular high-level conceptual data model describing this representation is the entity-relationship model. (Connolly & Begg 2015: 63; Elmasri & Navathe 2016: 33.)

The DBMS system is as described earlier a system that lets users maintain, create, define, and control access to the database. A DBMS allows users to retrieve, insert, delete, and update data in a database, this usually through a Data manipulation language (DML). The DML can provide an inquiry facility, called a query language to the data that can be found in a central repository, thanks to the DBMS. The standard and most common query language for relational DBMS is Standard Query Language (SQL). The DBMS also allows the users of the database to define it by letting them specify the structures, data types, and the constraints on the data, often through a Data Definition Language (DDL). Lastly, the DBMS provides controlled access to the database, and depending on the DBMS it may provide; a security system, an integrated system, a concurrency control system, a recovery control system, and a user-accessible catalog. (Connolly & Begg 2015: 64; Elmasri & Navathe 2016: 6.)

There are many kinds of databases and according to Ramin Ahmadi, Bagher Rahimpour Cami and Hamid Hassanpour (2012), the differences between two databases can be broken down into three categories: Syntax difference, data model difference, and semantic difference. The syntax difference is the difference in the variant languages in the two databases, which means that if data needs to be migrated from one to the other the request should be written and sent in the source database and the output data should be stored by the language of the target database. Data model difference is as the name says the difference in the data model of the database, for example, if the database is object-oriented or relational. The third difference, the semantic difference is the difference in two database entities for example, due to inconsistency. This can be seen when there are two databases that have been developed separately but have similar data, and this problem occurs when the databases have to be connected to each other. Therefore, analysis has to be done to find the semantic differences. These three differences make it challenging to migrate data

from one database to the other, and therefore, analysis has to be done on both to successfully migrate the data. (Ahmadi, Cami & Hassanpour 2012.)

2.3.1 Microsoft Access

Microsoft Access is developed by Microsoft (MS) and is a DBMS that combines software tools and a graphical user interface with the relational Microsoft Jet Database Engine, which is the format in which the data is stored in (Tutorials Point 2018). According to Laurie Ulrich Fuller and Ken Cook (2013: 11), Access is very accessible and "*easy to use at the edges*", with that they mean that you can get much out of the software without going too deep. Microsoft Access belongs to the Microsoft Office suite of applications and is included in the editions that are professional or higher. As the name suggests, Access can work and connect directly with other sources, like applications and databases. It can understand and use many data formats, but often it uses other SQL databases that can be on servers, on the desktop, on microcomputers, or with data stored on intranet web servers or on the Internet. Moreover, you can also import and export data from or to word processing files or spreadsheets. Access is built around objects, such as tables, queries, forms, and reports. (Tutorials Point 2018.)

Tables are essential in a database as they store all data and information and from the discussion earlier, tables are the entities. All other objects in the database are heavily based on the tables. Therefore, it is very important to start the development of a database with the tables, before designing other objects. As with any other software, the requirements analysis and the designing phase is crucial, the same it is for Access projects. A well-designed table in a relational database stores data for a particular subject, like customers and products. The following figure will show an example of how a table could look like and work. (Microsoft 2019a.)

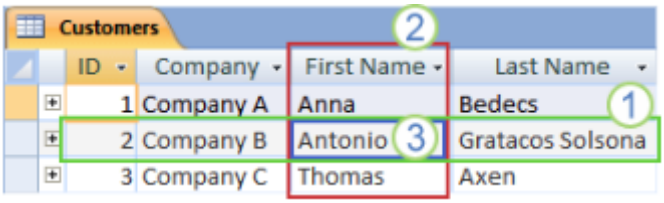

Figure 8. Example of a table in MS Access (see Microsoft 2019a)

A table in Access has **fields** (columns) and **records** (rows). Number one in the figure is referring to a record and it stores information about a particular customer in this example. The second number two is a field that stores information about one part of the table subject, in this example it is the first name of the customer. Number three is a field value and each record has a field value and in the example, it is Antonio, which is the first name of a customer from Company B. (Microsoft 2019a.)

Data types are another central part of databases and MS Access. Every field in the tables has a data type. The data types indicate what type of information the field stores, for example, is it numbers, large amounts of text or dates. The fields in a table in Access are created in the table design section and there you have to specify the data type when creating a new field. The data type of a field then determines what properties that field has, where field and table properties are attributes of the field or table that affect the appearance, characteristics, and behavior of the field or table. (Microsoft 2019a.)

Another part of the Access tables are **table relationships**, as many tables stores data about a particular subject, they often relate to another table. With relationships, you can tie together data from different tables, often with the use of **keys**. The keys are the fields that are part of a table relationship and can be a **primary key** or a **foreign key**. A table can have one primary key and the primary can be one or many fields in the table that uniquely identify each record in the table. From the example, in figure 8 the primary key are probably the ID field because that is unique for every record and with that, you can identify a particular record. A foreign key contains values that correspond to another table's primary key. For example, you could have another table called orders that have a field called customer ID that corresponds to the ID in the customer table in figure 8. (Microsoft 2019a.)

Queries are the second object that Access is built upon. A query makes it easy to add, view, change, or delete data in the database. With queries you can find specific data quickly by filtering, using specific criteria. Moreover, you can summarize and calculate data, and also automate data management tasks. The information you want to present in a report or a form can often be found in many different tables if the database is well-designed. A query can pull the data out of these different tables and put it together to then be shown in a report or a form. A query can be either an action or a request. An action query is used when there is a need for adding, changing or deleting data. A request query is for retrieving data from the database. (Microsoft 2019b.)

Forms are a database object in Access that can be used to create a user interface (UI) for a database application. There are two types of forms that can be made, either bound forms or unbound forms. The bound form is connected directly to a data source, such as a query or a table, and is used to display, enter or edit data from the source. An unbound form is a form that is not connected to any source and can instead contain command buttons, text, labels, or other controls that help you operate the application. (Microsoft 2019c.)

Reports are the fourth and final database object that is used to summarize, view, and format information in the MS Access database. With a simple report you can, for example, show a summary of the total sales across different regions. The reports are usually used to present information from the database and as with the forms the reports can be bound or unbound. The reports can run at any time and they reflect the data in the database in real-time. They are generally used to be printed out but can also be viewed on the screen, sent as an attachment in an email or exported to another program. (Microsoft 2019d.)

If there is a need to automate processes and make more complicated applications in Access, programming is needed and can be done by either Access **macros** or Visual Basic for Application (**VBA**) code. For example, if you need a command button that opens a report, that can be done by programming, using the property “OnClick” from the command button. The OnClick property is an event that will call and run a macro or VBA code if the user clicks on the button. The decision between using macros or VBA code is

based on two concerns: the functionality that you want and the security. VBA code is needed if there is a need to use built-in functions or own made functions. Furthermore, if there is a need of creating or manipulating objects or manipulating records one at a time, VBA code is needed, but if there is no need to do the things mentioned, macros can be used. If security is a concern, macros are better because VBA code can be used to create code that harms the users' computer or it can compromise the security of the data. (Microsoft 2019e.)

Moreover, another important part of Microsoft Access is **modules**. Modules are used to add functionalities to the database by programming in VBA. A module is a collection of statements, procedures, and declarations that are stored together. There are two types of modules, one is the class module and the other is the standard module. The class module is modules attached to reports or forms and contains specific procedures for the object it is attached to. Standard modules are then not attached to any object and contain general procedures. (Microsoft 2019g.)

A macro is a tool that can be used to add functionality and automate processes in the reports, forms, and controls. A macro in Access can be made using the Macro Builder without using any code and it can be seen as a simplified programming language. When creating a macro, you select actions from a drop-down list to create an action list on what that macro should do and in what order. The macro can then be associated with the event of a button, for example, the OnClick event is triggered every time the button is clicked and the macro is being executed. Access also has a function to convert macros to VBA modules. You can convert both global macros and macros that are attached to forms or reports. The macros attached to forms or reports that are converted will be added to the class module of that object. The class modules are part of the objects and will be moved or copied with the object. For beginners, the function to convert macros to VBA can be used to learn VBA, but for others, VBA can be used to make more complex applications with own made functionalities. A Visual Basic Editor is built in Access and is used to write the VBA code. (Microsoft 2019e.)

2.3.2 Microsoft SQL Server

Microsoft SQL is a relational database management system (RDBMS) developed by Microsoft. It supports a wide range of business intelligence, transaction processing, and analytics applications in information technology (IT) environments. According to Margaret Rouse, Adam Hughes and Craig Stedman (2019), Microsoft SQL Server is together with Oracle Database and IBM's DB2 one of the leading database technologies. From the name it already is clear but Microsoft SQL Server is built on top of the SQL. Microsft has its own implementation of SQL, called Transact-SQL (T-SQL), which adds a set of proprietary programming extensions to standard SQL. (Rouse, Hughes & Stedman 2019.)

As SQL Server uses the RDBMS technologies, it is built around table structures with rows and columns and it connects related data in the different tables together. The SQL Server Database engine is the core part of the SQL Server. It controls security, processing, and data storage and it includes two other engines, a relational engine, and a storage engine. The relational engine processes queries and commands and the storage engine manages the database parts, such as tables, pages, files, data buffers, indexes, and transactions. The underlying layer from the Database Engine contains the SQL Server Operating System (SQLOS). The purpose of the SQLOS is to handle low-level functions, such as I/O management, memory, job scheduling, and avoiding conflicting functions by locking data. Above the Database Engine is a network interface layer, it uses Microsoft's Tabular Data Stream protocol to handle response interactions with the database, and facilitate requests. Lastly, at the user level, the database administrators and developers write the T-SQL code to modify and build the database structures, implement security, manipulate data, and back up the database. (Rouse, Hughes & Stedman 2019.)

Microsoft has bundled up SQL Server with a variety of tools, such as data management, analytics, and business intelligence tools. The data analysis offering includes SQL Server Analysis Services, and SQL Server Reporting Services but now also in the newer version Machine Learning Service technology. The Analysis Services is an analytical engine that processes data to be used by business intelligence and data visualization tools. The Reporting Services is supporting the delivery and creation of business intelligence reports.

Microsoft provides SQL Server in four primary editions that are providing different levels of services. Two of the editions are free, an Express edition and a full-featured Developer edition. Where the Express edition is for smaller database projects with up to 10 GB of disk storage space and the Developer edition is for database development and testing. The non-free editions are the Standard edition and an Enterprise edition. These both are for larger applications and the Enterprise edition includes all of SQL Server's features, and the Standard is including a partial feature set and is limiting the processor and memory that can be configured in the database server. (Rouse, Hughes & Stedman 2019.)

The SQL Server was first developed in the 1980s by the former Sybase Inc., it was then developed for Unix systems. Microsoft came into the picture in the later 1980s and the first Microsoft SQL Server version was released in 1989. In 1994 Microsoft took over all development for SQL Server for their own operating system, and in 1996, Sybase renamed their version to Adaptive Server Enterprise, which meant that from then on Microsoft was the only one using and developing SQL Server. Ten versions of SQL Server have been released between 1995 and 2016, but since then SQL Server 2017 has been released and it is stated that SQL Server 2019 should be released in late 2019. (Rouse, Hughes & Stedman 2019.)

2.3.3 Differences between Microsoft Access and Microsoft SQL Server

It is important to know the differences in the databases in which the data migration is being executed. First off MS Access cannot hold more than 255 concurrent users and has a size limit of 2 GB, while SQL Server can hold more users, and have a larger capacity. SQL Server does also minimize the memory requirements when more users are added than Access. Another difference and benefit of using SQL Server are that in SQL Server you can dynamically backup the data while the database is used and therefore, the users do not have to exit the database to back it up. SQL Server does also has higher scalability and performance than Access because, for example, queries are being processed in parallel, which makes it faster. Furthermore, SQL Server has better security than Access because SQL Server uses a trusted connection that is integrated with Windows system security, and therefore, using the best of both security systems to provide integrated access

to the database and the network. Lastly, SQL Server does automatically recover the database if the operating system crashed, this in a matter of minutes and with no need of a database administrator. (Microsoft 2019f.)

Data types are another thing that has to be compared when doing data migration between two different databases. The naming of different data types differs a lot between Access and SQL Server but how they actually work does not differ much. For example, a large number in Access is called "bigint" in SQL Server, another example, the data type double number in Access is called "float" in SQL Server. But a big difference in how a data type is working is the Access data type Yes/No, which in SQL Server is a bit, that is 0 or 1, where 0 is "No" and 1 is "Yes". This difference has to be considered when migrating data between the databases. (Microsoft 2019h.)

2.4 Data migration

According to Johny Morris (2012: 7), the definition of data migration "*is the selection, preparation, extraction, transformation and permanent movement of appropriate data that is the right quality to the right place at the right time and the decommissioning of legacy data stores*". This definition highlights the importance of data quality and planning before migrating data, because, it can be that the existing setting of the data in the current environment is not working in the new target environment. Therefore, when moving permanently to a new environment it is important to know before how to maintain good quality after the migration. Stated by Chidananda Gouda, Sudarshan Patil, Anil Kumar, Guru Prasad, and Sai Madhavi (2016) data migration can be applied to any area where we work with data, such as file systems, information systems, databases, storage types, etc.

The need for data migration often occurs when new systems are being introduced that changes the environment of the existing system (Oracle 2011). The reason for the data migration is often that the existing system needs to be upgraded to meet the industry re-

quirements. Often the data migration is database migration, which means that data is migrated from one database to another, often the source and the structure of the current and the target database are different. Other reasons for data migration are that for example, saving measures, investments to IT services, and change of company policy. Therefore, the development of data migration and database migration tools have emerged lately. (Elamparithi & Anuratha 2015; Gouda et al. 2016.) The ultimate aim of data migration is to improve performance and deliver a competitive advantage (Oracle 2011).

A lot of the literature regarding data migration is about legacy migration, which can lead to a very expensive task. Therefore, it is very important for organizations to make it as cost-effective as possible by simplifying the migration process. The process of legacy migration often includes research areas, such as reverse engineering, scheme mapping, business reengineering, translation, and application development. The lifecycle procedures of legacy migration are that before the migration, you need to plan, assess and prepare. This can include assessing software, network, and hardware readiness for the migration. Another task is to clean up the legacy system, by consolidating resources and eliminating useless data. During the migration, you have to prototype, pilot, and deploy the migration, by using different tools to model and simulate the migration and resolve issues before committing, and it is important to track the migration. Lastly, after the migration is it important to manage and maintain the new environment. (Elamparithi & Anuratha 2015.)

As discussed earlier reverse engineering is a technique used in software engineering, and especially in re-engineering. This technique can also be used for databases and is called database reverse engineering. It is the first step in the migration process where the source (existing) database is analyzed to identify its components and their dependencies. Schema information, which is showing the entities and their relationship, is used to understand the source database design and structure. (Elamparithi & Anuratha 2015.) The migration strategy is similar to re-engineering approaches, where there are two types of migration approaches: trickle migration and big bang migration. The big bang approach works the same as for re-engineering. The whole migration is done in one small window and the

whole migration is done at the same time. The trickle migration is an incremental approach where the migration is done over a longer time and the two systems work in parallel while the migration is done in phases. (Oracle 2011.)

According to Lalitha, Lalithakumari, and Surekha (2016), data migration tools are very important in the process of migrating data between databases. They also mention that organizations and users often start with a MS Access database but then as the organizations or the database grows the need for a more efficient database is needed. In recent years cloud-based applications have grown and to support that many have migrated to NoSQL databases, such as MongoDB, which was the most popular in 2016 (Lalitha, Lalithakumari & Surekha 2016). NoSQL means not only SQL and is a nonstructured, nonrelational database compared to the traditional relational database of SQL Server for example. A tool for doing the data migration is very useful and for example, data migration between an MS Access file and MySQL (which are both relational databases) can be done by creating the same tables in MySQL and then by knowing the differences in the data types make the correct data types in MySQL and then query over the data. Having a tool that does all this makes it more efficient and with better quality. (Lalitha, Lalithakumari & Surekha 2016.)

To be able to do the data migration successfully a data migration process model has to be used according to Florian Matthes, Cristopher Schulz and Klaus Haller (2011). Their process model has four main stages that are initialization, development, testing, and cut-over, which then contains fourteen different phases, here only the main stages will be explained in short. In the initialization stage you set up the organization and infrastructure, then in the development, you develop the data migration programs needed. The testing stage then validates the stability, correctness, and execution time of both the migration programs and the actual data migration. Then the last step the cut-over is where you execute the data migration and switch over to the target application. (Matthes, Schulz & Haller 2011.)

2.5 Emission regulation and certification for marine diesel engines

This chapter is about emission regulation and certification for marine diesel engines, this because the EIAPP (Engine International Air Pollution Prevention) tool is built to support the certification of marine engines and to better understand the tool this chapter is needed to explain in general why the certification is needed and what is needed to get it. To be able to re-engineer the tool and to migrate the data between the tools it is important to understand the certification process and emission regulation in general because many of the terms and explanations in this chapter will be used later in the thesis when the existing tool is described.

The International Maritime Organization (IMO) is a United Nations specialized agency that is responsible for the security and safety of shipping and for the prevention of atmospheric pollution by ships. They are a global standard-setting authority that has the main role to create a regulatory framework for the maritime industry, that is effective and fair and is adopted and implemented universally. The regulations make sure that shipping companies cannot cut and compromise when it comes to environmental, security, and safety performance. This also leads to and encourages efficiency and innovation. (IMO 2019.)

It was in the late 1980s that IMO started to work on the prevention of emissions from ships. Before that, they had been working more with visible sources of ship pollution, like oil spills from ship accidents. But as more research and scientific information showed the long term risks that exhaust gases had on the environment and human health IMO changed their approach. In 1997 IMO added a new regulation called "*Regulations for the Prevention of Air Pollution from Ships*", to the MARPOL Annex VI. There they also added the "*Technical Code on Control of Emissions of Nitrogen Oxides from Marin Diesel Engines (NOx Technical Code)",* to the mandatory part under MARPOL Annex VI. The MARPOL Annex VI entered into force on the 19th of May 2005. (IMO 2017: 1.)

The Technical Code is for the prevention of NOx emissions and the purpose of the code is to specify the requirements for the survey, testing, and certification of marine diesel

engines to make sure that they comply with the NOx limits of the regulations from Annex VI. The code and regulations apply to all engines that are designed or installed on a ship that has a power output of more than 130 kilowatts (kW). Important words and their meanings are EIAPP and a Technical File. As written before EIAPP stands for Engine International Air Pollution Prevention and an EIAPP certificate is the certificate that relates to the NOx emissions. A Technical File is a record that contains the information regarding needed details, such as the parameters of settings and components that can influence the NOx emission of the engine, this in accordance with the Code. (IMO 2017: 81-82.)

Another important part is the engine family and the engine group. Engines that have similar design and emission characteristics can be presented by an engine family or an engine group. The difference between these is that engine groups are used for groups of engines that have similar design and specification but individual engine modifications and adjustments are allowed after the testbed measurements, where for engine families this is not allowed. For both approaches, one engine is chosen to be the parent engine. The parent engine is the engine in the group with the highest NOx emissions. All other engines in the engine family or group are called member engines and their documentation is based on the parent engine and therefore the approval process of the member engines is much faster than for the parent engine. (IMO 2017: 92-95.)

Lastly, dependent on when the ship is built and where the ship is going to sail the NOx limits differ and for this, there are after 1 January 2000 three tiers for this: Tier 1, Tier 2, and Tier 3. Tier 1 is for ships built between 1 January 2000 and 1 January 2011. Tier 2 is for ships that are built on or after 1 January 2011. Last, Tier 3 is for ships constructed on or after 1 January 2016 and is operating in the United States Caribbean Sea Emission Control Area or the North American Emission Control Area. The difference is that for each higher tier the NOx limits are lower than the previous, this to make sure the newer ships emit lower NOx and also for engine manufacturers to innovate and get the NOx emissions lower. (IMO 2017: 22.)

3 RELATED WORK

Both software re-engineering and data migration have been explained in the previous chapter. Therefore, this chapter will work as a literature review of some of the existing research papers regarding software re-engineering and data migration. Software re-engineering has been researched in the last three decades from the early 1990s when it emerged when users needed to shift their software solutions to web-based solutions (Chikofsky & Cross II 1990; Müller, Jahnke, Smith, Storey, Tilley & Wong 2000; Majthoub et al. 2018). According to Abdelsalam Maatuk, M. Akhtar, Ali and Nick Rossiter (2011) database re-engineering and database migration are the same thing and often used together because database re-engineering almost always includes data migration of some sort.

Moreover, Xiaohu Yang, Lu Chen, Xinyu Wang and Jerry Cristoforo (2005) have made a research paper on a dual-spiral reengineering model for a legacy system. The model works around the two systems, the legacy one and the target one. The functionalities, not the modules are moved from the legacy system to the target system one by one in steps, as in a spiral model. In short, a spiral model is a model where each loop in the spiral is one phase of the software process, for example, one loop can be system feasibility and the next one is requirements definition (Sommerville 2016: 48). Furthermore, Yang et al. (2005) used the software engineering spiral model and made it for re-engineering where for the whole process the functionality in the existing system is in a decremental pattern, and in the target system in an incremental pattern.

Another research made by A. Cathreen Graciamary and M. Chidambaram (2018) is about an effective approach to improve the performance of re-engineering. They propose a framework called Efficient and Enhanced Software Re-engineering Mechanism. There are four steps in the process, first, the existing system is analyzed in terms of accessing time and storage size of the software application. Then the solutions that solve the initial problem within the implemented technology in the existing system are analyzed. After that, the feasibility of the solution to solve the existing problem is checked. The last step is then to compare the re-engineered system with the old one in terms of memory usage

and access time. Their conclusions are that their proposed framework improves the quality of service and it increases the reliability and efficiency of the software application. Also, another notion they make is that due to the fast development of the computer industry, changes are often needed in software and hardware, and new software development is much risky than re-engineering and old system. This also shows how important re-engineering is today to minimize the risks and to save money.

There is a lot of research made for data migration but as for any research made in the subfields of software engineering, there is a lot of variations and there are no general rules or theories that work for every data migration project. One paper written by Philip Howard (2014) from Bloor Research named “Data Migration Customer Survey” is based on a survey looking at data migration. First of Howard mentions that data migration is a critical function of IT because everything from a change of application or database or even migrating data between versions of applications or databases is data migration. The survey showed that only 62% of data migration projects were finished on budget and on time, which is the biggest problem with data migration. The survey also showed that with the use of best methods and practices data migration can be successfully achieved.

The result of the survey is seven key best practices in a data migration project. The first one is the use of a data profiling tool during the project, both before and during. Data profiling is the process of understanding the source data, structure and relationships. The second best practice is to use a data cleansing tool, or else it can lead to poor data quality. The third one is the use of a data integration tool and from the survey, it showed that if not using one it is 50/50 chance of success. The fourth one is to adopt a methodology that has been tested. The fifth practice is that if the data migration is part of a bigger project, the data migration should be treated completely separately and independently, by having its own budget and testing. The sixth one is that companies should develop in-house competence of data migration instead of relying on resources from outside the company. The seventh and last best practice the paper presents is that the business has to be involved and engaged throughout all stages of the project including data migration. These seven best practices had risen up during the survey and organizations that adopted all the practices had a significantly better success rate.

A paper named “Testing & Quality Assurance in Data Migration Projects” by Florian Matthes, Christopher Schulz and Klaus Haller (2011) discusses the importance of risk mitigation and testing to deliver data migration projects on time and on budget. Their risk model consists of three levels: the business level, the IT management level, and the data migration level. The top-level, the business level risks are risks often articulated by the customers and the three most relevant business risks are profitability, reputation, and regulation. The second level, the IT management level, has a more technical focus and the most relevant risks are data or information loss, target application stability, cut-over aborts, extended downtime, project budget overruns, and delays. The last level, the data migration project risks are risks like, data migration program risks, that can be data corruption risks or stability risks, then the migration run risks that can be execution time risk and last the infrastructure risks that can be interference risks or dimensioning risks. (Matthes, Schulz & Haller 2011.)

The testing-based quality assurance also presented by Matthes, Schulz & Haller (2011) aims at finding problems or faults in the data migration program, in the data itself, or in the underlying infrastructure. According to them, there are two categories of testing techniques, the first one is migration run tests and the second one is data validation. Migration run tests are designed to ensure that the data migration programs run smoothly and are well-working. There are two subcategories and they are partial- and full migration run tests. The full migration run tests are for testing the migration programs and measure the execution time of the overall migration, identical to the one done for the final cut-over. While the partial migration run test is to test speed-up tests with for example fewer data to make sure everything is working as it should. Then the data validation consists of four aspects: semantical correctness, completeness, interoperability with other applications, and consistency on the data and structure level. Subcategories to the data validation are three tests where the first one is an appearance test, that focuses on the appearance of objects on the GUI level of the source and the target application. The second, the processability test is another test that processes the migrated data and ensures it works in the new environment. The third test, the integration test, is to make sure that if one application changes that the other still works, as the data migration represents a big change, therefore,

the functions of the new application has to be tested with the migrated data. (Matthes, Schulz & Haller 2011.)

In data migration, more practical researches were found compared to software re-engineering. One example is a paper made by Sushma Velimeneti (2016) called “Data Migration from Legacy Systems to Modern Database”. Her task was to migrate the data from mainframe databases to a common repository under SQL Server for a big healthcare carrier in the United States. She used the iterative development lifecycle process for the employment of the data migration and a big factor in that model is to involve the business owners and users and to follow the data migration methodology. The result of the data migration boosted the application performance by 65% and it reduced the downtime of the systems. When having the data in the same place it also improved the reliability and security of the application. Her paper again shows how important it is to use the methods and frameworks for data migration and also it also showed the benefits of the migration.

A research paper by Mario Bernhart, Andreas Mauczka, Michael Fiedler, Stefan Strobl and Thomas Grechenig (2012) called “Incremental Reengineering and Migration of a 40 Year Old Airport Operations System” did combine data migration with re-engineering. They described the experiences and challenges by using the incremental re-engineering approach when migrating an old airport operations system. The system was a legacy system using old technologies that were hard to maintain. The challenge when dealing with a complex airport system is that downtime of the system is not wanted, therefore the incremental approach was chosen over the big bang approach. They used this approach to step by step go from the old system to a new re-engineered system. They followed the re-engineering steps of planning, analysis, user interface design, and technical design. Their conclusions of the project were that using an incremental approach may come with large overhead but the risk with the big bang approach is often unacceptable. A critical success factor according to them was the strong focus on the user interface design and live testing with actual user interaction with actual production data and that was highly valuable and found many issues that would not have been found otherwise. (Bernhart et al. 2012.)

Lastly, as said earlier the research made in the software re-engineering field is very theory-based, in which models, frameworks and processes are primarily discussed (Yang et al. 2005; Graciamary & Chidambaram 2018). During the literature review, fewer case studies or constructive researches were found than theoretical ones, therefore, the conclusion that there is a need for more practical researches. For data migration, the research made is more practical than for software re-engineering but also here it is concluded that there is a need for more constructive research that presents the whole software project, which includes theory, models, and methods but also a practical part where the implementation and the results are explained. Therefore, this thesis will be a constructive research that focuses on a real-life problem that combines software- and database re-engineering with data migration. Constructive research will be discussed and presented in the "Planning" chapter and the next chapter is a description of the existing tools.

4 DESCRIPTION OF THE EXISTING TOOLS

This chapter will describe the current tools and the way of working, this includes a description of the process that is used to be able to get the Engine International Air Pollution Prevention (EIAPP) certificate for the marine diesel engines produced by Wärtsilä. This chapter can, therefore, be seen as the reverse engineering phase of the re-engineering, as the main point is to understand the current tool from a low level of abstraction to a high level and this chapter will result in new documentation of the tool. In other words, this chapter will describe with text and figures what the tool looks like, what it is used for, and what the components are and their relationships, this according to the reverse engineering principles. But before going more into detail on the tool we first have to discuss the whole certification process, to then better understand the purpose and the need for the EIAPP tool that Wärtsilä has developed back in 2005.

4.1 EIAPP certification process

This chapter will describe in general and on a high level how the whole EIAPP certification process is working. The whole process is in place according to the regulations and needs coming from the IMO – MARPOL Annex VI, described in the theoretical framework. It is also important to know that the engine group approach has been chosen by Wärtsilä over the family group approach. This thesis does only focus on the high level and will not go into details of the internal process. The process is different both if the engine is an IMO Tier 2 or IMO Tier 3 or if the engine is a parent or a member. The process that is going to be presented is the process only for the Compliance & Certification team as it is this team that uses the EIAPP Tool and is responsible for developing the tool.

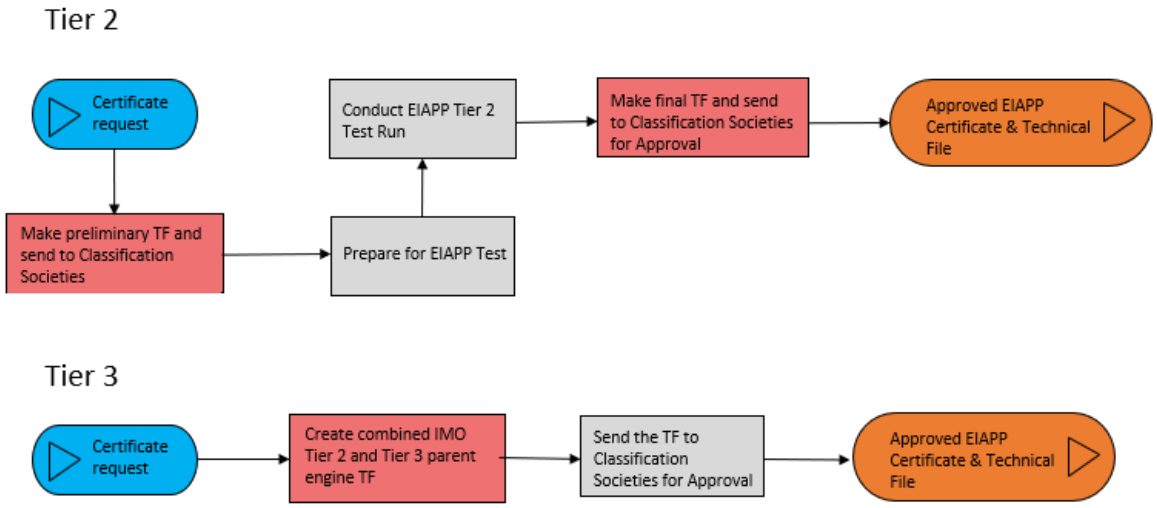

Figure 9. The EIAPP parent engine certification process.

The above process is for the IMO Tier 2 and the IMO Tier 3 parent engine certifications. The red color boxes are steps in the process where the EIAPP tool is used. The difference between the two processes is that the IMO Tier 3 does not need any EIAPP test because the Technical File is built upon an IMO Tier 2 parent engine, therefore, the IMO Tier 3 parent Technical File is a combination between an IMO Tier 2 and an IMO Tier 3 engine. As stated these steps are just general and on a high level, for example, the step “Prepare for EIAPP Test” is containing sub-steps.

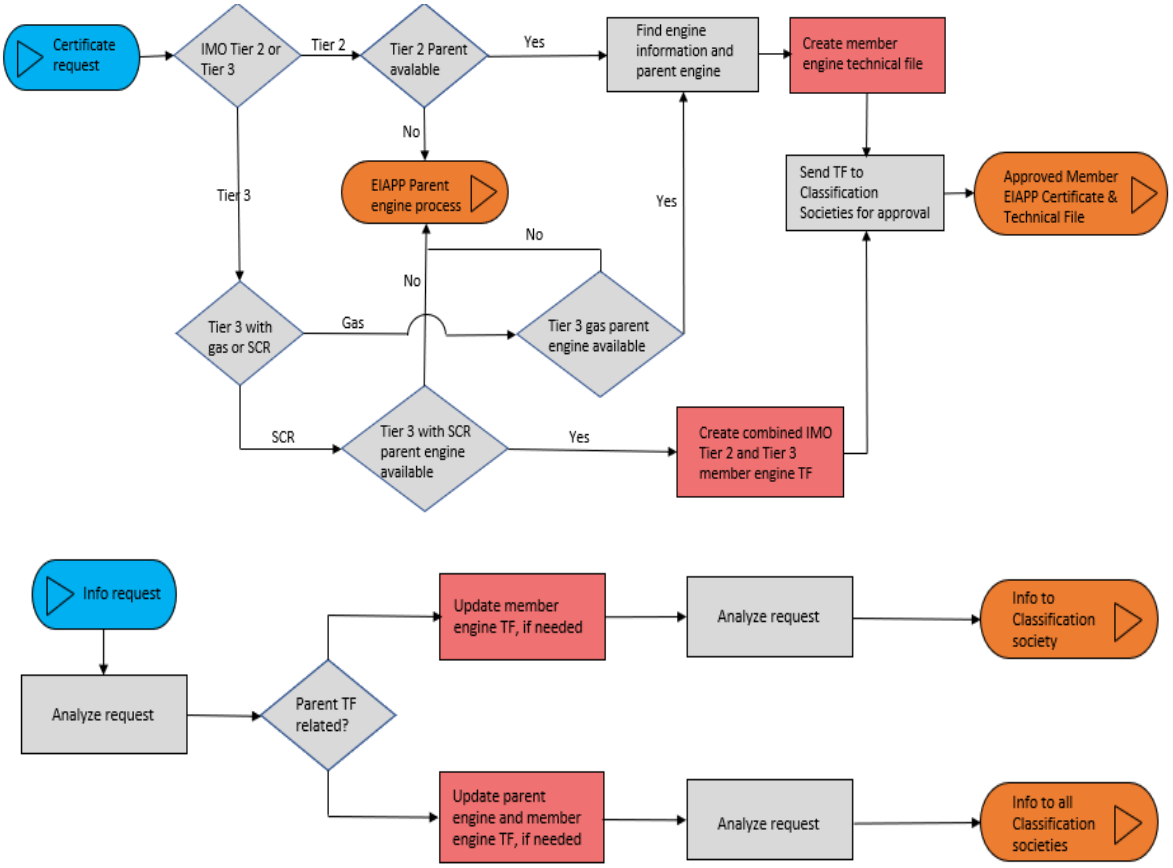

Figure 10. The EIAPP member engine certification process

The EIAPP member engine certification process shows how Wärtsilä has solved the Tier 3 regulations, this can be seen in the upper part of the figure. If the engine runs on gas then it is the same procedures as for a Tier 2 engine and a gas engine complies with the Tier 3 NOx limits. But if the engine runs on diesel, the engine needs a Selective Catalytic Reduction (SCR) unit, which is a unit connected to the engine to remove particles and to lower NOx emission to comply with the IMO Tier 3 NOx limits. The idea as can be seen for that type of an engine configuration is that it can be a Tier 2 engine that then has an SCR, and therefore, it can be a Tier 3 with a combined Tier 2 and Tier 3 Technical File. If not a parent engine is found for the specific member engine then the process in figure 9 has to be conducted to create a parent engine for the specific engine group.

The first option described in the top of the figure is when a new parent is needed but there can also arrive an info request, seen in option two in the lower part of the figure. The info

request tells that there has been a change in an engine. First, it has to be checked if the request concerns member or parent engines. Then the scope of the change has to be evaluated according to the MARPOL Annex VI regulations to check if an update is needed in the Technical Files. If an update is needed that is only member engine-related then only that specific member engine's Technical File has to be updated and sent to the society it was sent to originally for a new approval. If an update is concerning a parent engine, then also all member engine Technical Files have to be updated and the updated parent engine Technical File has to be sent to all classification societies.

4.2 The EIAPP Tool

The EIAPP Tool is a tool built by the Compliance & Certification team in Wärtsilä to faster and more efficiently make the Technical Files for the EIAPP certification purpose. The EIAPP Tool is mainly used by the certification engineers in the team and they are the ones responsible for making the Technical File for EIAPP certification. As written in the last chapter 4.1 about the certification process, the tool comes into the picture when the Technical File is needed. Before this tool was created Microsoft Word was used to manually create the Technical Files for each engine and this took a lot of time, therefore, the tool was needed to save time and more efficiently and reliably make the Technical File.

The tool is built in Microsoft Access and it is using the database objects tables, forms, queries, modules, and reports for the making and storing of the data needed for the Technical File for both parent engines and member engines. The user is interacting with the forms by inserting data and from the forms, they can preview and print reports that are part of the Technical File. The forms and reports are unbound objects, which as stated earlier mean that the objects are not directly connected to any source. Programming of the objects is made in VBA and no macros are used. To better explain the tool a figure will be shown that explains how the tool is built from inserting data to getting out the Technical File.

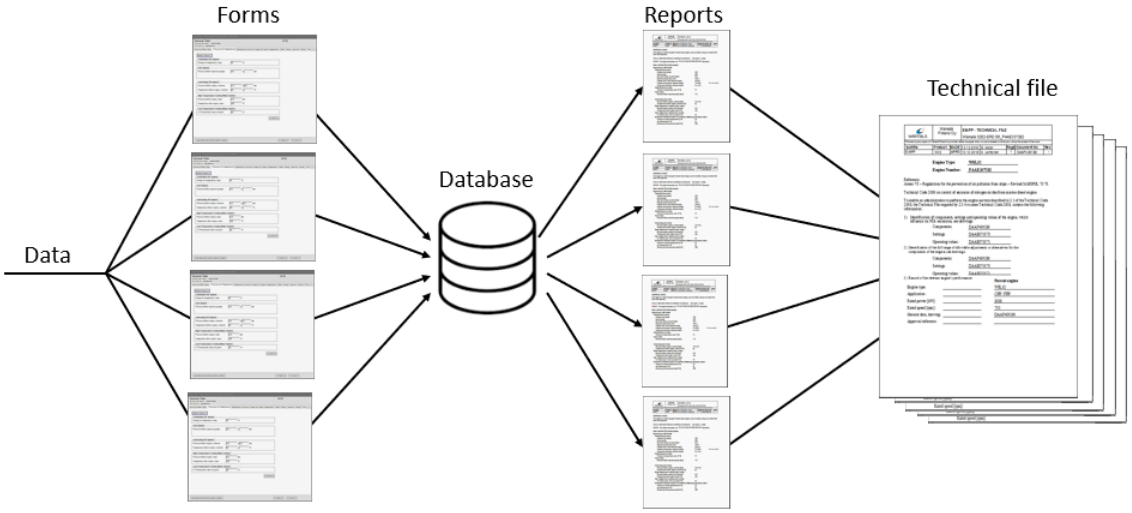

Figure 11. Example of the Technical File creation process in the tool

The figure shows how the tool is built, where the user is inserting data into many different forms that then through VBA and SQL code saves the data to the Microsoft Jet Database Engine. The user can then print and preview the reports that queries data from the database. The reports are built around the forms, in which almost every report is built upon a specific form and the information that was inserted there. When putting all the reports together you get the Technical File that contains all necessary information regarding the specific engine and can then be sent to classification societies for approval.

The forms and reports are unbound which means that for every form and report there is a class module behind them that contains the procedures for the specific object. Standard modules are then used for non-specific procedures and for storing the SQL statements for every table. Almost every table has a standard module containing the SQL statements for the tables, for example, statements that save or updates data in the specific table. In other words when users interact with the forms and reports their specific class module's code is being executed and as the reports and forms are interacting with the tables that contain the data, the functions in the standard modules are called to query the information, using SQL code.

As written in the Introduction chapter a problem is that there are two tools that have the same functionalities and they are for the most parts identical. One tool is for engines built

in Finland and the other for engines built in Italy and for that purpose the thesis will use abbreviations WFI (Wärtsilä Finland) and WIT (Wärtsilä Italy) to separate the tools. A VBA code was used made by Allen Browne (2008) to count the number of lines of code in the class modules behind forms and reports and the stand-alone modules. Furthermore, a query was used to count the number of forms, reports, tables, and stand-alone modules. The tools are both living, which means that they are maintained and new functionalities are added at times, this means that the number of objects and lines of code will change often and therefore this is just showing the difference between the tools. The function made by Browne was inserted into the two tools and the result is seen below:

Table 1. Lines of code behind the different modules on 9.8.2019

Lines of code	**WFI**	**WIT**	**Total**
Stand-alone modules	43977	42140	86117
Form modules	50066	47990	98056
Report modules	7818	7610	15428
Total	101861	97740	199601

Table 2. Number of objects on 9.8.2019

Number of objects	**WFI**	**WIT**	**Total**
Stand-alone modules	145	135	280
Forms	78	82	160
Reports	39	39	78
Tables	151	134	285
Total	413	390	803

The stand-alone modules are the standard modules containing, for example, SQL code for the tables. From the result, you can see that the number of objects is quite similar but

a more in-depth analysis of the tables and their differences will be presented in chapter 5. This also shows the other problem when having two almost identical tools and that it is the huge amount of code and objects the two tools have combined. If changes are needed that concern both tools then the changes have to be done twice, which can lead to quality problems.

As the idea and most parts are identical in the two tools, it is chosen that for this chapter the tool for engines built in Finland will be used for the examples and figures. There are a lot of buttons, functions, and forms in the tool and therefore, it is chosen to only explain the most important ones. To start to describe the tool a picture of the start page will be shown to then easier explain how the tool is used and also what it looks like, as the design is a big part in software development.

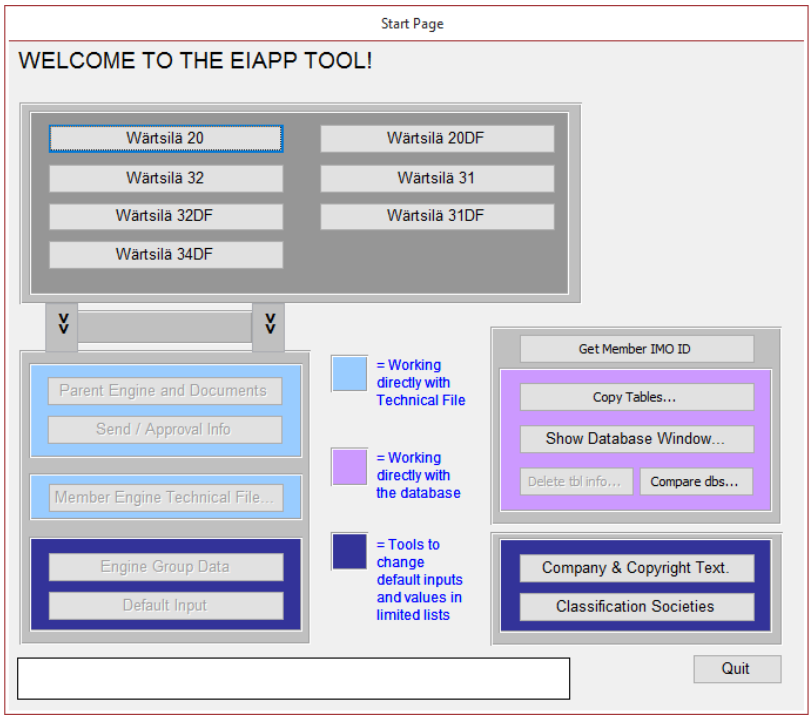

Figure 12. The start page of the WFI EIAPP Tool.

Figure 12 shows what the start page looks like for the WFI EIAPP tool. Please note that from here and on if there are white rectangles in the figures, that is for hiding information and is not seen in the real tool. This is a form that is unbound and the buttons and text (called labels in Access) that can be seen are controls that are interacting with the VBA

code. On the top part of the start page, there are buttons with the different engine types for engines built in Finland. For the user to go further in the tool, they first have to select an engine type. Every engine type button is connected to an OnClick event that starts to run VBA code that will enable the grey buttons, seen on the left. When they are enabled the text will turn to black and after that they are clickable.

The colors and their explanation can be seen in the middle of the form and first the pink one, those are functions that the developers use to interact with the tables and the function behind the “Copy Tables” button will be explained later. The two blue buttons on the bottom right are functions that are not dependent on the engine type, rather on the whole tool. There the user can add or delete the classifications societies that are used all over the tool and also some text that will be seen in the reports can be changed there. The two buttons on the bottom left are functions that are dependent on the engine type and they both will open up different forms where the user can insert engine type specific data that are then used when creating the parent engine and the Technical File. The information inserted there will show up in for example drop-down lists in the tool after they are inserted for the different engine types.

From this form, it is also good to explain the high-level process of using the tool. The main parts the user will use are the sections behind the buttons in the light blue area. These are all corresponding to the EIAPP certification process explained in chapter 4.1. In the parent section, the user creates the parent engines and their corresponding Technical File. When the file is created and sent to the classification societies the user goes into the “Send / Approval Info” section where they can insert what Technical File is sent to which society and when. When the Technical Files are approved they then insert the approval info in the same section. When that information is inserted the user now can start to make member engines for that specific engine group with the newly approved parent engine Technical File. Before the approval info is inserted for the parent engine it is not possible to insert member engines in the specific engine group.

Both the Access database (back-end) and forms and reports (front-end) is in the same Access file. Therefore, if new features are needed or problems need to be solved, the

developers have taken a copy of the file to their own computer and made the changes there. When the changes are done, they have used the “Copy Tables” function to copy all data from the file on the server to the new version they have on their computer. This is done because if users have used the file after the developer has copied their file they need to get the latest data into the tables in the new version. After that, they have then copied over the new version of the tool where the problem is solved or new features are added to the server. This is not a good way of working but it has worked well as there have usually only been one developer in the team. The next part of the tool that will be looked at is the parent engine section, which is found if the user clicks on the “Parent Engine and Documents” button found on the start page. A new form is then opened and can be seen in figure 13:

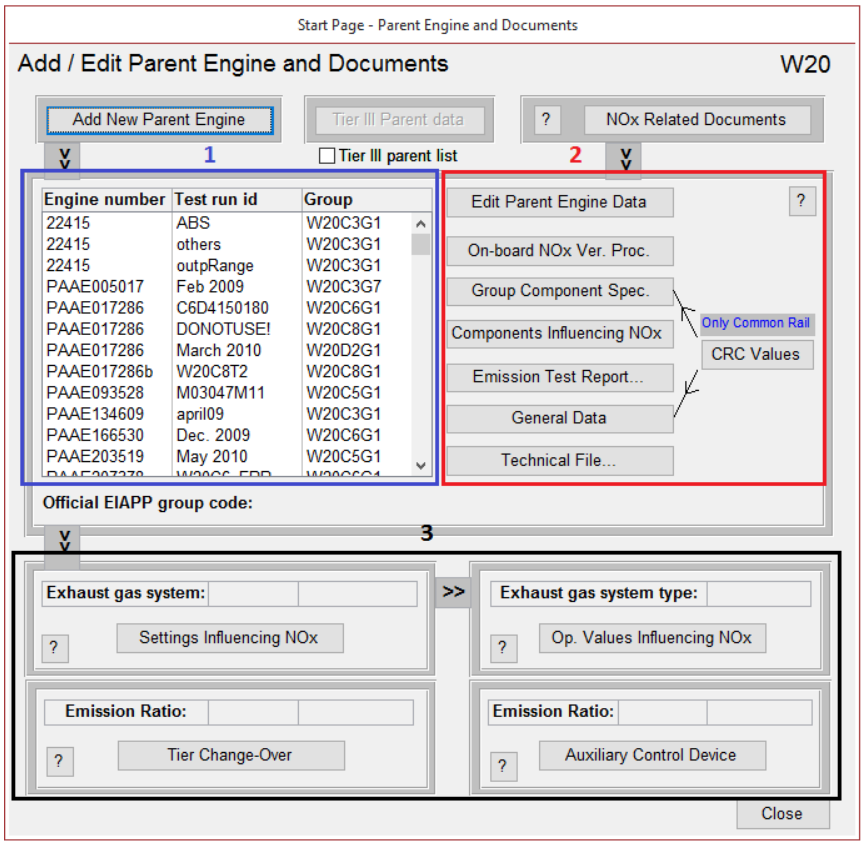

Figure 13. The first page in the Parent engine section

This part of the tool is the most important, as it is here in the subsections where the user will insert all data that is needed for the Technical File for the different parent engines.

The color boxes are there now only to better explain. First, the list that is found in the first blue box. The list is containing all parents for the engine type Wärtsilä 20 (W20) in this example and the engine type that the user is working on can be found in the upper right corner. The information in the list contains the Engine number, Test run id, and Group, and the user can click on any parent engine in the list to select it. Furthermore, Engine number and Test run id are the primary keys in most tables containing parent data. The button on the top left is used to open the form where a parent is created and after the parent is saved in that form it will show up in the list. Then the user can add further information to that parent. The "Edit" parent button that is found in the second red box is used to edit the parent engine data, for example, the engine number.

When a new parent is added, the user needs to start to add all information that is needed for the Technical File and as explained earlier the whole tool works around many reports with their own document number. The buttons in the red box except the edit parent button are opening forms where the user will first add the document specific data which can be seen in figure 15 and after the data needed for that specific document is added. All these functions in the red box are parent engine specific, which means that every parent engine has its own documents. The buttons in the third black box on the bottom of the form are also different documents but those are not parent engine specific, for example, the "Settings influencing NOx" is exhaust gas system specific, which means all parents with the same system have the same document.

It is important to understand that all these different sections and documents are all built on the regulations from IMO - MARPOL Annex VI. Every marine engine manufacturer then must on their own interpret the regulations and build their own Technical File. This means that there is no correct way on how to build the TF, and this is only how Wärtsilä has built it, but one goal of this thesis is to change the way of working from many document numbers to only one for the whole TF. The next important part from the document number perspective is the "NOx Related Documents" section that can be found in the upper right part of the Parent form (figure 13) and is presented in figure 14.

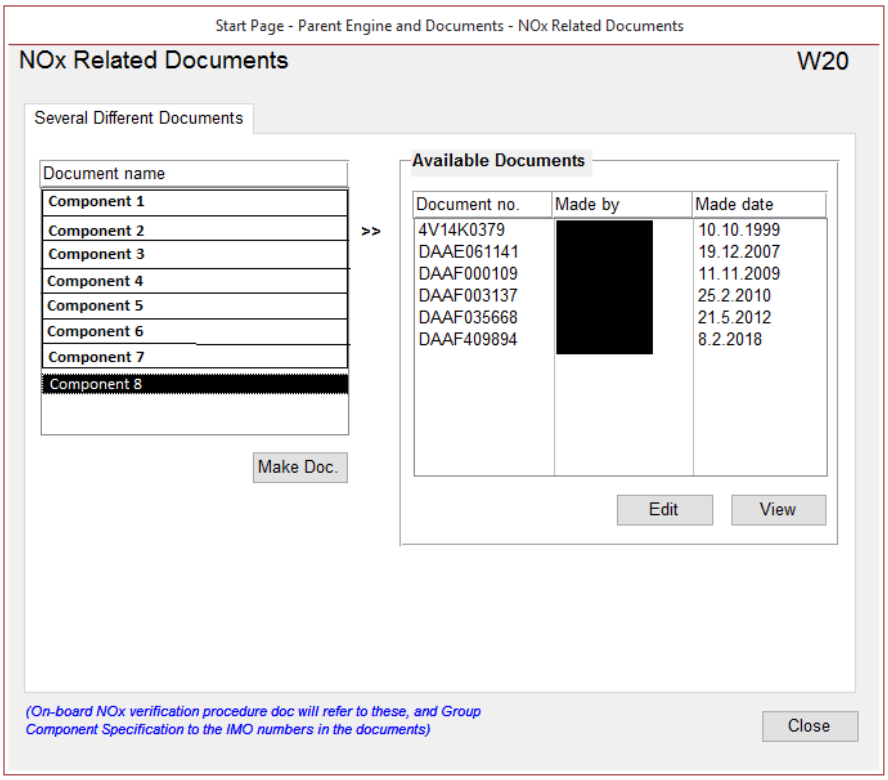

Figure 14. The NOx Related Documents section of the tool.

The components cannot be shown but that is not important. This section is engine type specific, for example, for W20, every parent has these same documents. In the figure, an example is seen for one of the NOx critical components (Component 8). There are 6 documents that are referring to this component. The documents for these different components for the engine types often hold a picture of the component and its IMO number, which means a number that can be seen on the component to make sure it is the right one. If we then go back to figure 13 and the Parent section form, there the form behind the button “On-board NOx Ver. Proc.” is where the user then chooses what component document that specific parent engine needs. For example, the user has to choose between the 6 document numbers for Component 8 and choose the correct one for the specific parent engine. This is the principle on how the tool works, that the user inserts data in some places and in other the user chooses from drop-down lists the correct information. The next thing that can be found all over the tool is the form where the user inserts the document-specific data, such as the document number.

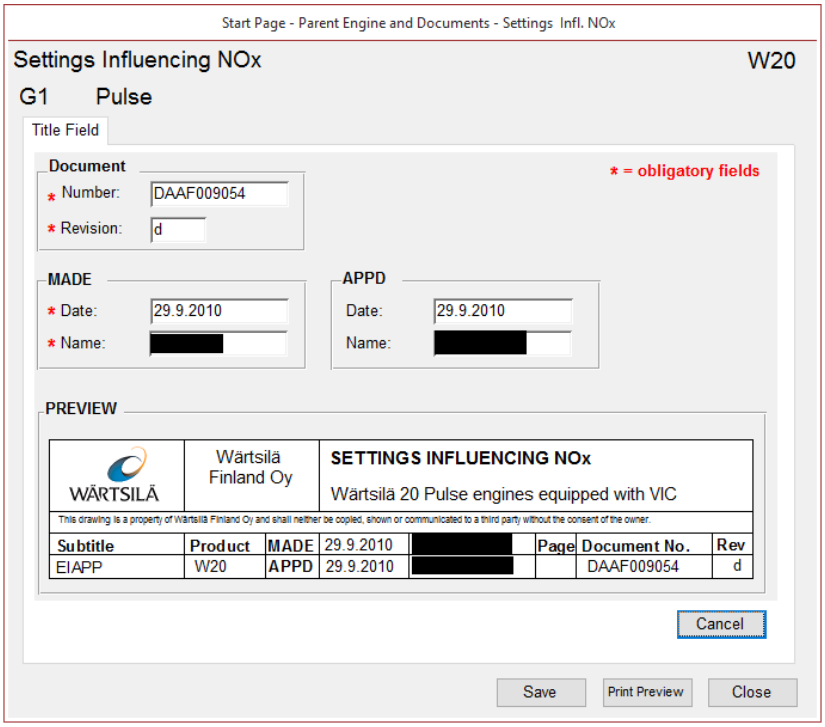

Figure 15. Example of the form where document specific data is inserted.

This example comes from the “Settings Influencing NOx” section and is where the user inserts the document-specific data. As can be seen in the middle of the form is where the user inserts a document number, revision, and who made and approved the document and when. The “Title Field” form is a sub-form that is inserted in the “Settings Influencing NOx” form, this because the same title field form is used all over the tool, therefore, a sub-form is used. The lower part of the form is the preview of how the header will look like for the settings influencing NOx report, where the document information is inserted after the user has inserted it. The next function that is important is the “Print Preview” that can be seen on the button in the bottom right corner of the form. This button will show the user the preview of what this report will look like.

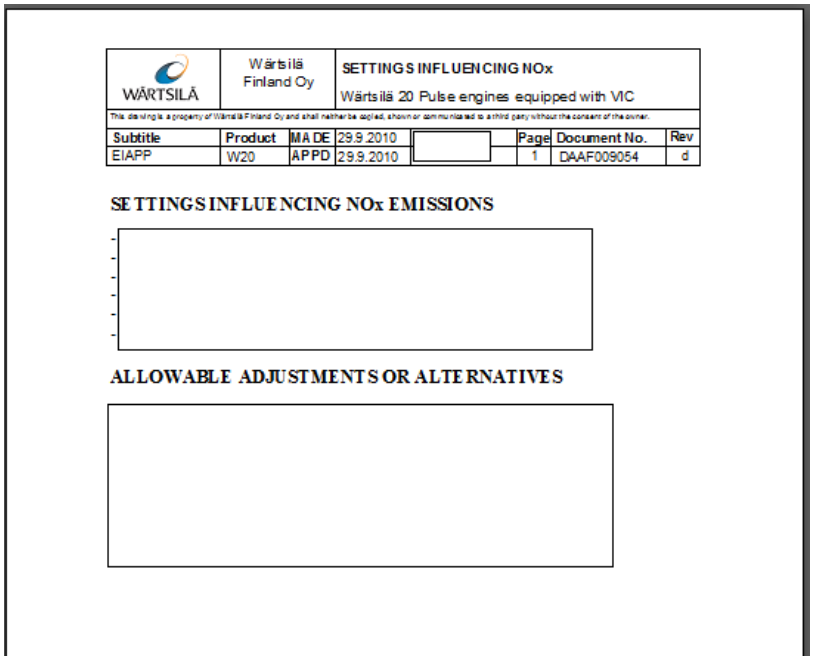

Figure 16. Example of the report Settings Influencing NOx.

This is how the settings influencing NOx report look like for one specific parent engine. The report is dynamic in the way that information changes depending on properties and the configuration of the parent engine. The top header is information coming from the database and is based on the information seen in figure 15. This specific report is all based on the engine configuration and the properties of it and no other information was inserted in the settings influencing NOx form, except the document specific data. This is how many reports work in the tool where only the report specific (header information) is inserted and then the report can be printed according to and dependent on the engine specific configuration and properties inserted when creating the parent engine. The whole Technical File is then a document that contains all these different reports that can be printed separately or together in the tool.

Lastly, in the parent section (figure 13), the creation of the Technical File will be explained. First, the user inserts the Technical File document specific data, in the same way as any other report in the tool. After that, the user goes to the print form for the Technical File and it can be seen in figure 17.

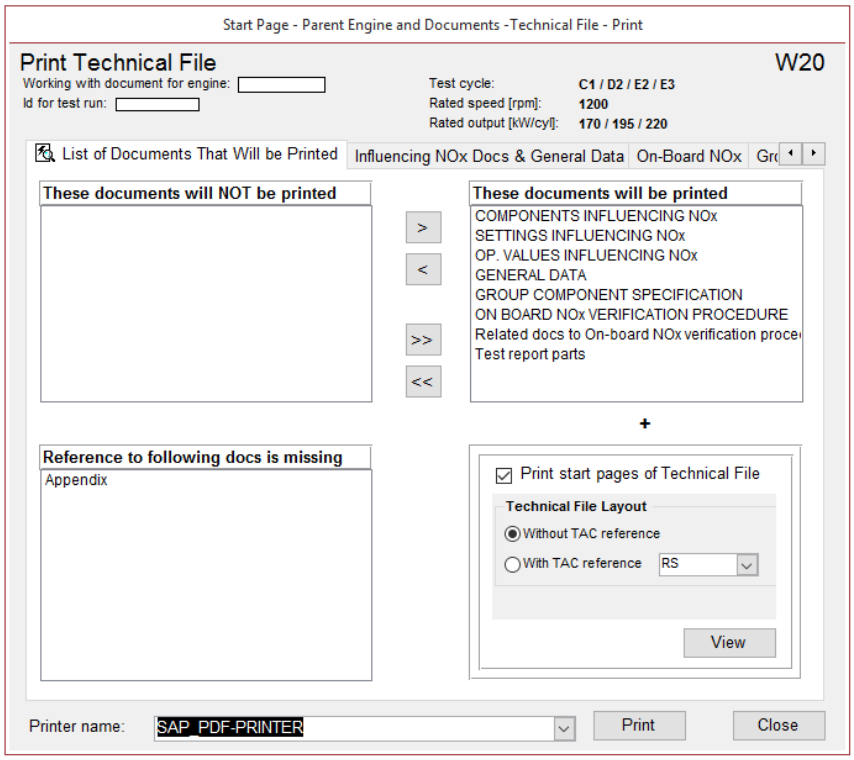

Figure 17. The Technical File printing form in the tool.

In this form, the user can see what reports will be printed in the list box on the top right that together makes the Technical File. In the tabs seen in the top, the user can also more in detail check exactly what documents and document numbers that will be printed. In the box in the right lower part of the form, a checkbox can be seen regarding the start pages of the Technical File. The start pages are a crucial part of the Technical File where the most important parent engine specific information is. The user can then, for example, print a paper copy of all the reports or print it in pdf format. After printing, the Technical File is one file or one paper copy that can be combined and then sent to the classification societies for approval.

The next part of the certification process is as described earlier to insert the sending information and then after the Technical Files are approved by the classification society, the approval info. When the parent part is completed, the user can move onto the next part, the member engine section. The member engine section is where the users can insert the member engines connected to a parent Technical File. The way it works is that the most important member engine information is added, the information that is needed for

the first pages, the so-called start pages, that was discussed in the previous paragraph. The start pages can then be printed for each member engine and the certification engineers then manually takes the correct parent engine Technical File and replaces the start pages for the member engines. Because the parent engine Technical File contains the whole engine group's configuration and information, it means that for each member engine, only the parent Technical File information is needed, because they belong to the same group.

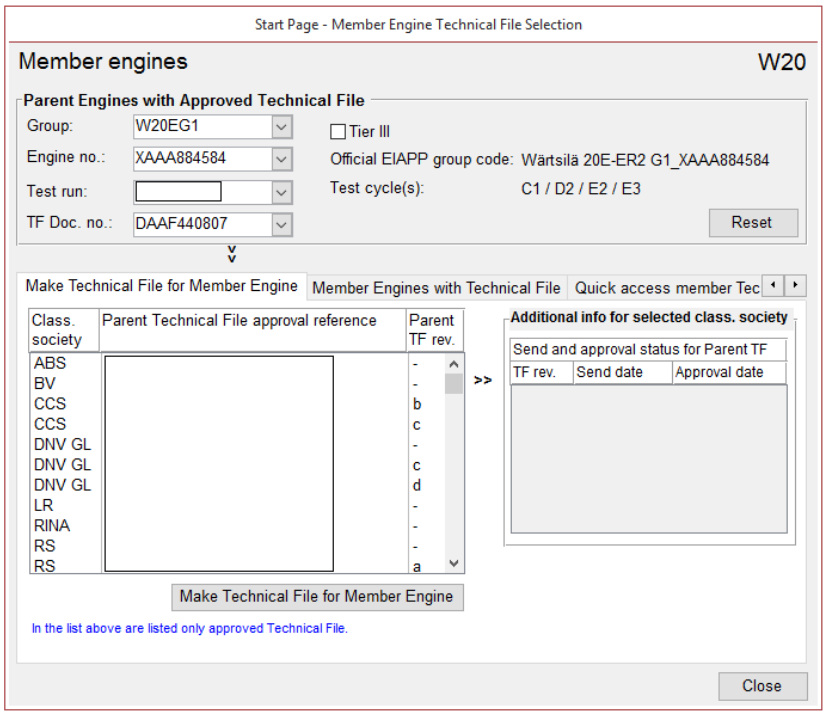

Figure 18. The member engine section in the tool

In the form seen in figure 18, the user creates member engine Technical Files for Tier 2 engines. If Tier 3 is needed the user has to click on the checkbox found in the top middle part of the form that takes the user to a new form. The Tier 3 form for member engines is mainly the same but as Tier 3 member engines have both a Tier 3 and a Tier 2 parent it is a bit different but the main idea is the same. The user first has to select the parent engine and Technical File in the four drop-down lists in the top left. After that, the list box in the bottom left will be populated with all the approved Technical Files from the different classification societies. Then the user has to click on one of the rows in the list to select a specific approved document to then click on the make Technical File for member engine.

When then creating a Technical File for a member engine the user gets to a new form where the user inserts the member engine specific data needed for the start pages of the Technical File such as document specific data and, for example, engine number, engine configuration, application, and test cycles. In the second tab seen in the middle of the form called “Member Engines with Technical File” is where all the member engines can be found for the selected parent and Technical File. From there the user can view, edit and print the start pages of the member engine Technical File and then manually take the correct parent Technical File and append it to the start pages of the member engine. The member engine Technical File is then sent to the classification society for approval.

4.3 Summary

In this chapter, the certification process was described to better understand the whole process and why and when the EIAPP Tool is needed. Then the most important parts of the tool were explained. As could be seen in the count of forms, and reports there are a lot of them and therefore, only the ones that are needed to be understood to understand the tool were chosen and explained further. The two main parts of the tool are the parent engine section and the member engine section. The parent engine section is built around different documents that together make the Technical File. The user inserts the needed data for the different documents and then through the printing function prints out the different documents before sending the whole Technical File to the different classification societies for approval. When a parent engine is approved the tool is then used for making the member engines in the same group as the approved parent engine Technical File.

The need for re-engineering emerged mainly because of the change in requirements as the reports will no longer be printed in the tool, which leads to that no more than one document number is needed. As the tool has to be changed according to that requirement it is also good to take the opportunity to re-design, restructure, and clean up the code and data at the same time. Because there are a lot of forms, every one of them will not be looked at in detail but the main parts and the parts that need to be re-designed or re-structured according to the new requirements will be analyzed and changed accordingly.

5 ANALYZING AND PLANNING

In the theory framework (chapter 2) it was discussed that the software processes and models need to be adapted based on project needs. In this project, a combination of processes and models will be used. The first part of the reverse engineering was done in chapter 4. In this chapter, it will continue by examining the tool more in detail and documenting the findings and with the new requirements plan on how to re-engineer the tool. This by planning design changes and examining the tables to know what to change when the tool will not work around the different documents and document numbers anymore. The development will be done by taking a copy of the WFI tool as a starting point and from there develop and implement the changes according to the findings in this chapter. The WFI Tool was chosen as it is the tool that has been developed and maintained the most and the previous chapter 4 showed that the WFI tool had more objects. From now on when talking about the development tool it will be the new tool that is being re-engineered.

5.1 Research method

The research method used in this thesis is the constructive research approach. According to Kari Lukka (2003), constructive research is an approach that focuses on producing innovative constructions to solve real-world problems in practice. Constructions can be for example diagrams, models, mathematical algorithms, computer languages, and new medicines. (Lukka 2000; Lukka 2003.) The process and the core features of a constructive research consist of seven main steps specified by Kari Lukka (2000; 2003):

1. **Find a relevant problem that has the potential for theoretical contribution**. The selection of the topic is the most important one, the same as for any research.

2. **Review the potential long-term research cooperation with the target company.** The best solution is to have the researcher a member of the project team devoted to the project to solve the problem.

3. **Acquire a deep understanding of the topic field, both theoretically and practically**. In this step, the researcher has to both review and understand the prior theory in the field of study, but also understand the organization and the problem in depth.

4. **Establish a solution idea and develop the solving construction**. This is the most critical step in the process because if no construction to solve the problem can be designed then there is no point in going further with the research and the project.

5. **Implement the construction and test how it works**. In this step, the solution is implemented and the first level practical tests of the design are viewed as one of the key characteristics.

6. **Reflect on the scope of the applicability of the solution**. In this step, the researcher together with the organization reflect on the learning process the project has given. The main goal of this step is to analyze the results of the process and its preconditions.

7. **Analyze and identify the theoretical contribution**. This is an important part of the project from an academic view as here the researcher has to explicate the theoretical contribution of the project.

The need for constructive research is as Kari Lukka (2003) mentions that many organizations tend to show fatigue of always being a target to interviews, observations, and surveys. There is a lot of academic research that is not practical and is not constructive and many starts to wonder what they actually get out of the academic analyses. Therefore, constructive research is important as it works around two-way communication because of the importance of teamwork in the empirical parts of the research. Furthermore, as the task of the researcher is to bring related theoretical knowledge to the process it helps organizations to overcome problems which otherwise would maybe not be done because of lack of resources. (Lukka 2003.)

The first step of the constructive research, to find a relevant problem is already done and is the foundation of the thesis. The main problem is how to re-engineer the tool and how to do the data migration. Those are part of the research questions in this research. The second step of the constructive research process, to review the long-term research cooperation with the organizations was also done prior to this thesis as it was the organization that proposed the topic. They will get two different things out of this project, first, a re-engineered tool and secondly, documentation. This leads to the third step in the process, to acquire a deep understanding of the field, and this has been done in the theoretical framework chapter in this thesis and also in the chapter about the existing processes and the existing tool, this to understand the practical part of the problem. This chapter will continue on the third step to even more in-depth discuss the problem and this chapter will also work as the fourth step, about establishing a solution to the problems. A table was made to further explain where and when each step has and will be concluded.

Table 3. Summary of the where and when the steps have been and will be conducted.

Steps	Where and when?	Description
Step 1: Find a relevant problem	Pre research	The problem was suggested by the target company and is the foundation of the research
Step 2: Review the long-term research cooperation	Pre research	Also done pre-research with the target company
Step 3: Aquire a deep understanding of the topic field	Chapter 2, 4 and 5	The theoretical understanding was established in chapter 2 and the practical understanding in chapter 4 and 5
Step 4: Establish a solution idea and develop the construction	Chapter 5	This chapter will plan on how to solve the problems and work as a plan for the implementation
Step 5: Implement the construction and test it	Results in chapter 6	Implementation will be done out of the research but the results will be shown in chapter 6
Step 6: Reflect on the scope of the solution	Chapter 6 and 7	At the end of chapter 6, a reflection will be made together with the organization to reflect on the learning process
Step 7: Identify the theoretical contribution	Chapter 6 and 7	Also at the end of chapter 6, the theoretical contribution will be analyzed and identified

5.2 Requirements

The first new requirements were gathered in the initial phase of the project when the topic was specified. The people involved in the initial discussions were the stakeholders and users of the tool, which are the certification engineers in the Certification & Compliance team. The new requirements for the tool are the ones that will guide the re-engineering work needed. The first requirement discussed in the Introduction chapter and throughout the thesis is that the existing tool works around document numbers and each report and document has its own document number. Now as the reports will be printed in a separate new reporting tool, where only one document number is needed for the whole Technical File. Based on this requirement, the tables in the database will be analyzed to see where the existing document numbers for each report is stored and in what way. Based on the result of the analysis, a plan will be made on how to change the needed tables. The biggest change is needed if the document numbers are primary keys in tables and that will lead to the need for table re-structuring to change to the new way of working and the new requirement.

Another requirement is to check if there are possibilities to automate processes in the tool to speed up the work of inserting data. For example, if possible, automate parts of the data insertion in the tool. Another possibility is inside the tool if there are processes that can be done more automatically or faster, for example, inserting data into tables in forms. This will be discussed with the certification engineers that work with the tool to first understand where all data is coming from and then see if there are possibilities to automate it.

A third requirement is to re-design parts of the tool. Firstly, change the parts affected by the requirement to only have one document number for the whole EIAPP Technical File. Secondly, to make the EIAPP Tool more intuitive and user-friendly. This can be done by, for example, looking at buttons, checkboxes and namings to see that are everything understandable or could something be designed easier.

5.3 Re-designing

In this chapter the steps in re-designing the tool will be discussed, this includes table analysis, re-designing of forms, and automating processes. This chapter will result in a plan on how to re-design tables and forms.

5.3.1 Table analysis

The table analysis will be done in two steps. The first step is to analyze the difference between the WFI and the WIT tool. First, it is examined if the WFI and WIT tools contain different tables. Secondly, all the mutual tables are examined for structure differences, like attribute names and attribute orders. Thirdly to check if there are differences in the properties of the mutual fields, for example, the data type or field size. With the result of the comparison, a plan on how to harmonize the WFI and WIT tool and when to do the changes is made.

The result of the first step analysis is:

- 25 tables that only WFI tool have
- 6 tables that only WIT tool have
- 123 mutual tables
- 76 mutual tables that have data fields with different field sizes
- 18 mutual tables with differences in fields

This result of the first step table analysis shows that there are quite a lot of differences. The biggest difference is the total of 31 tables that are only found in the WFI or in the WIT tool. A second big difference is the mutual tables with different fields. These two differences have to be investigated in regard to where the data is used in the different tables, this can be done by looking at the VBA code. If tables are found in the WIT tool

that is for WIT engine type-specific information, the structure has to be copied to the new development tool. If there are differences in the way of working with the data from the tables then it has to be chosen one of the two solutions and change the other accordingly.

The second analysis step needed is to see where document numbers are stored and if they are primary keys or not. This to then set up a plan on how to change it.

The result of the document number analysis:

- 33 different document numbers found
    - 10 document numbers directly related to the parent Technical File
    - 17 document numbers related to components in “NOx Related Documents”
    - 1 for the member Technical File
    - 5 other document numbers

The two existing tools have the same structure and the result is the same for both tools. The five other document numbers are documents not found in the tool, only referenced in the tool and will, therefore, also be used in the new tool. The 10 document numbers directly related to the parent Technical File are found to not be primary keys in tables, which leads to no need for changes in the form structure except for deleting the part where the document specific data is inserted. The only document number that will be used from the 10 related to the parent Technical File is the actual document number for the Technical File. The 17 document numbers related to components in “NOx Related Documents” (see figure 14) are all primary keys in the different tables containing components data and will lead to a need for design changes both in the tables and in the corresponding forms.

How the component documents work in the existing tools has to be changed. The current way of working is now:

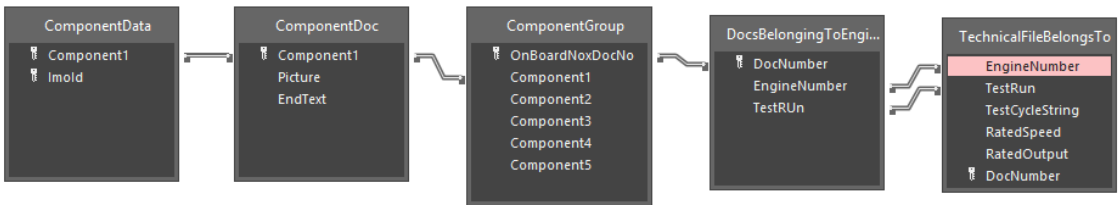

Figure 19. Example of how component tables work in the existing tool.

In figure 19 an example of how one component table structure is working now in the existing tools. For each component, there are two tables containing the component-specific data, one with the IMO id and one with the picture and text both related to a document number that is "Component1" in this example. Then each component is related to a component group table that contains the component document number in that group and the onboard NOx document number. The onboard NOx document number is then related to a parent in the table "DocsBelongingToEngine" and then that parent engine is connected to a Technical File in the table "TechnicalFileBelongsTo". Summary of this is that now for one component five tables are used to connect the component-specific data to a Technical File. For the new tool, the idea is to have three tables, where one is the same as in the existing tool, the "TechnicalFileBelongsTo" table.

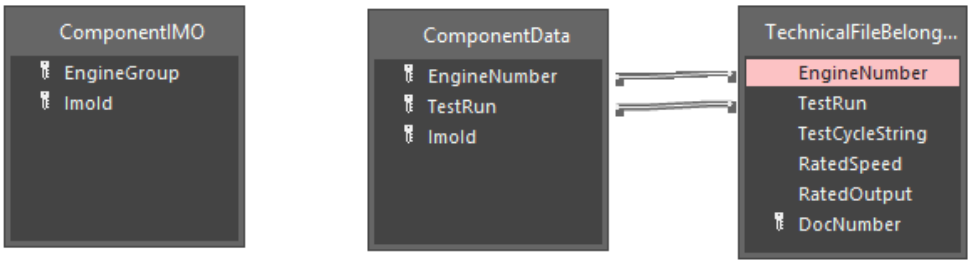

Figure 20. Example of how components will work in the new tool

In figure 20 can be seen the idea of how the components will work in the new tool. The "ComponentIMO" table will hold all IMO numbers there are for the specific component in regards to the engine group. The user then has to choose the correct IMO numbers

when inserting the parent engine data and that will then be saved in the table “ComponentData” that holds the engine number, test run and IMO id. Then there is the same relation to the Technical File table to connect the IMO id for a specific component to a parent engine and then to the parent engine’s Technical File. The picture and the text found in the existing tools will be moved over to the new reporting tool, which is out of scope in this thesis. To sum up, the new tool will have three tables instead of five that the existing tools have to hold the component-specific data. This same idea will be used for all the components in the tool.

5.3.2 Re-designing forms

The main purpose of this part in the re-engineering process is to make the tool more easy to use. This by re-designing forms and parts of forms in close cooperation with the users. The forms in the tool are dynamic, which means that for the different engine types they can look differently. The different engine types have different properties and it can lead to that different information is needed for the Technical Files. This means that when the WIT data is merged into the development tool, we manually have to check and change the forms according to the needs of the “new” engine types. This will be done in the latter part of the project when the design changes regarding the requirements for the tool are ready. This to avoid double work.

The requirement to only have one document number for the Technical File opens up a new possibility for a different grouping in the tool. Because up to this point the data is grouped according to the different reports, see figure 13, and for new users the sections “Group Component Spec.” or “On-board NOx Ver. Proc” does not tell much about what information is needed there. When the document numbers are not needed anymore we can re-group the data according to the content itself, for example, one section for all data regarding IMO numbers, and one section for data regarding turbocharger. This change also opens up for deleting parts of the parent section, for example, all the sections in the third black box in figure 13 can be deleted because there the user only inserts document specific data and that will not be used anymore.

The table changes regarding components discussed in chapter 5.3.1 will also lead to changes needed in the forms. The idea for the new tool is that the user can insert all IMO id:s to each engine group in the engine default data section found on the start page of the tool. All the current IMO id:s will be taken from the existing tools and in the new tool, only when there are new IMO id:s taken into use it has to be inserted. Then when inserting IMO id:s in the parent engine section, the user gets for every component a drop-down list containing all IMO id:s for that engine type and that component, and can then choose the correct ones. This is the biggest change in regards to re-designing the tool.

5.3.3 Automating processes

This step in the re-engineering process is to analyze if there are possibilities to automate processes in the tool. Automation, if done correctly, will speed up the time needed to insert data into the tool and also increase the quality and reliability, by eliminating human errors. There are two ways to automate processes in the tool. The first one is to automate how the information is inserted, for example, if the users insert a lot of information from the same source, that could be done automatically. The second way is to automate or speed up tasks inside the tool if there are found places where the insertion of data could be done faster, in another way or more convenient.

To be able to understand and examine the possibilities to automate processes in the tool, informal discussions and meetings were held with the users of the tool. All the sections in the tool were discussed to understand from where the user takes the information. It fast became clear that the information inserted in the tool comes from a lot of different sources, both from files but also from experts in different parts of the company. Therefore, it is chosen that there is no point in trying to automate the process of inserting data into the tool at this point as there is no single source that is used for a lot of data.

The discussions and meetings then focused more on the inside, the UI of the tool to see if there are possibilities there to automate tasks. The thing that came up was tables in the forms. An example of a table in a form can be seen in figure 21.

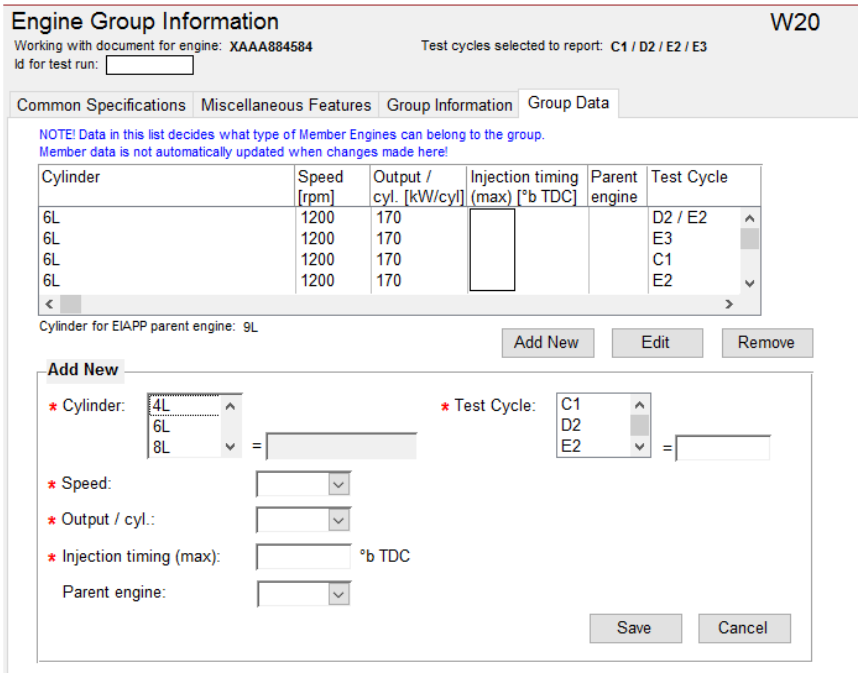

Figure 21. A table in a form where the user inserts engine group data

To not confuse, this is a table in a form and has nothing to do with a table in the database but works in a similar way. Here the user inserts engine group data to a parent engine. This means a combination between the cylinder configurations, speeds, output / cyl, and test cycles. From these combinations, the user can then later make member engines. It shows what engines can be found in the group. If there are a lot of combinations of engines in the group, the user has to insert a lot of data, one engine per row. Here automation can help the user to insert the data faster by that the user chooses what combination is needed and then it automatically creates a row for each combination by a function made in the VBA code.

5.4 Data migration planning

The next step in this analyzing and planning phase is to take a look at the data migration and how it will be done. There are two different data migration stages in this project. The first one is to migrate the data from the two MS Access tools into one. The second one is to migrate the data from the MS Access tool to a SQL Server. There are also two sub-

steps in the first migration. To be able to re-design and change the development tool to work with the WIT engines. The data has to be migrated at an early stage to make sure all data is present to be able to debug and test correctly. The last and final migration will be done when everything is ready as the two tools will live during the project.

5.4.1 MS Access data migration

To be able to do this first stage of data migration between the two MS Access databases we will reuse the function already discussed in chapter 4, the "Copy Tables" function. The function goes through all the tables found in the database that the user chooses to copy the tables from, then for each table that can be found in both tools the data will first be deleted from the tool that uses the function and then the data from the other tool will be copied to the table. This function can be reused to get the data from the WFI Tool to the development tool. The function has then to be modified to make get the WIT data to the tool. It has to be modified to not delete the data in the tool but to append the data in the tables. These two functions have to be coded and used at the beginning of the project to be able to get all data for all engine types to the development tool. The data quality and reliability of the function will be monitored and tested to make sure all data is merged.

5.4.2 Data migration between MS Access and SQL Server

The good part about the second step data migration between Access and SQL Server is that both are developed by Microsoft. Therefore, there are tools that can be used to migrate the data that have been developed by Microsoft. The tool that will be used is the Microsoft SQL Server Migration Assistant (SSMA) (Microsoft 2019f). According to Microsoft (2019f), you have to follow six steps to successfully migrate, and those are:

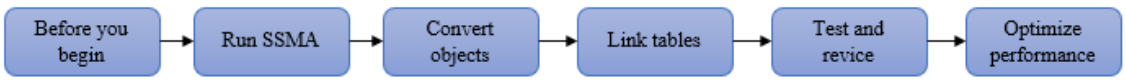

Figure 22. The migration steps needed for a successful migration according to Microsoft (2019f)

The “before you begin” step is about knowing the SQL Server benefits and choosing what SQL Server option will be used. The SQL Server benefits were presented in chapter 2.3.3 and the SQL Server is provided by Wärtsilä. Steps that have to be done before running SSMA are to check and make sure each table has primary keys and at least one index. Where an index is a data structure that can be used to improve the speed of data retrieval. The primary and foreign key relationships have to be checked to make sure they are based on fields that have the same data type and size. Microsoft Access has a data type called attachment and if a table uses that it will not be migrated so those fields have to be removed if they are used. The last step that needs to be taken is to close the Access database before migration and make sure to back up the database. (Microsoft 2019f.)

The SSMA tool will migrate the selected tables and queries but will not migrate reports, forms, macros, and VBA modules. A SQL Server Metadata Explorer will display the SQL Server objects and Access database objects to review the content of both databases and this connection is saved in the migration file and can be used if additional objects have to be migrated in the future. The SSMA can be downloaded from the Internet. When running SSMA, the beginning instructions will show basic information, such as the Access database and SQL Server location, connection information, objects to migrate, and if you want to create linked tables. (Microsoft 2019f.)

The third step to convert objects will not happen right away. The SSMA will provide a list of all objects that can be migrated and you have to decide and select what you want to migrate to the SQL Server. The objects that can be migrated are; tables and columns, select queries without parameters, primary and foreign keys, indexes and default values. The SSMA assessment report should be used to get and show the conversion results, including warnings, errors, the time estimate for the migration, informational messages, and individual error corrections steps that have to be taken to move the objects. The conversion takes the Access metadata object definition and converts them into equivalent Transact-SQL syntax and then loads that information to the project. (Microsoft 2019f.)

The next step in the process is the linking of tables between Access and SQL Server. Microsoft prefers you to use and install the latest version of the SQL Server OLE DB and

ODBC drivers instead of using the shipped native SQL Server drivers from Windows. By linking the data back to Access and by having the permissions set by the SQL Server administrator you can view, edit, and query the data in Access. (Microsoft 2019f.)

The last two steps in the process are to test, revise and optimize the performance. The test and revise step makes sure that everything is working and that the migration went successfully. This by using the tool to check that it works as it should and that all selected tables have been merged with the correct data. You can also do a lot to optimize the new solution by checking where you run what type of queries for example. Small and read-only queries are best to run in the Access client and long and read/write queries are best to run on the server. (Microsoft 2019f.)

5.5 Implementation process

In the theoretical framework, a lot of models, processes, and approaches were presented and this chapter will discuss what was chosen and why. The re-engineering model chosen was the model seen in figure 4 where the reverse engineering has been done in this and in the previous chapters and the forward engineering will be in between this and the next result chapter. It is also chosen that the re-engineered tool and the final data migration will be done by the big bang approach because the new requirements make it impossible to do it in incremental or evolutionary steps. When removing the use of multiple document numbers, the Technical Files cannot be printed in the tool anymore, therefore, the big bang approach is needed.

This leads to a problem discussed in the theory that as the two so-called live tools will still be used and maintained during the whole development of the new tool, there can come up changes that then have to be implemented in both the live tools and the development tool. This is a drawback of the big bang approach but it cannot be prevented and has to be dealt with. To minimize the risk of this problem it is chosen that only big urgent changes will be implemented in the live tools, other changes are to be developed only in the new tool.

Next, we will discuss the process that will be used in the implementation phase. What steps will be done and in what order.

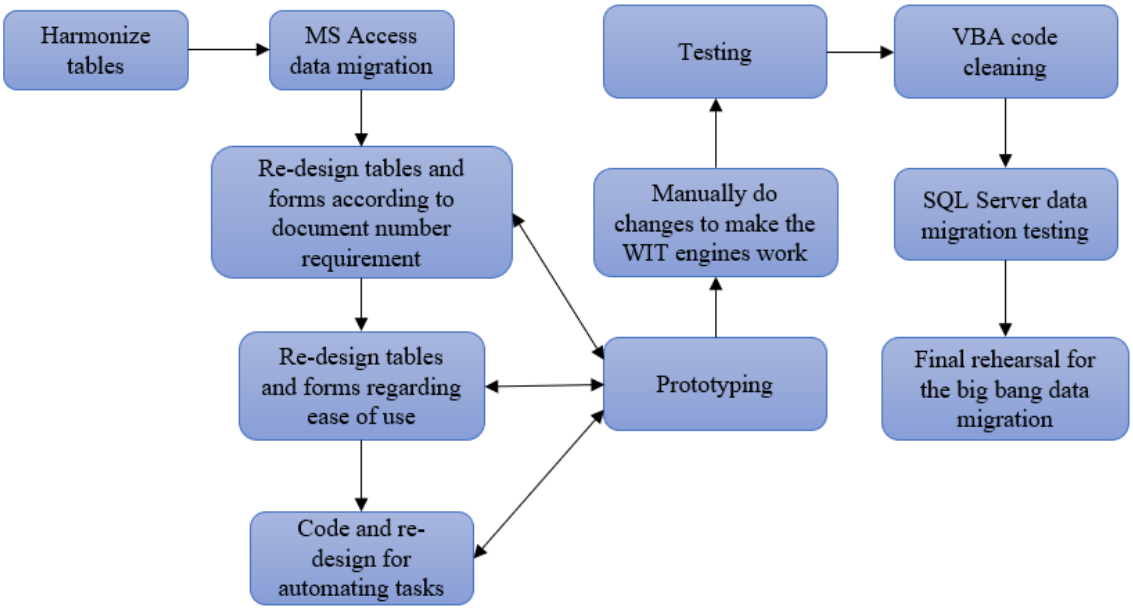

Figure 23. The implementation process plan.

This process is built upon the tasks discussed in this chapter 5. Where the first step is to harmonize the tables in the two tools to be able to do the data migration into only one tool. When that is done the first data migration will be done to help re-designing the tool when data from both existing tools are present. Then to the re-designing part that is split into three sub-parts, one regarding the document numbers, one regarding ease of use, and one regarding the automating of tasks. These three steps will be done to make prototypes in iterations and the prototypes will be tested and tried by the users to get feedback and find errors. This iteration will be done until the best solutions are found and the final prototype will be used going forward. The good part about Access is that it is easy to make working prototypes or at least visual prototypes with the use of its built-in tools.

Furthermore, the final changes will be done to make the WIT engines work in the tool, by comparing the information in forms in the development tool with the WIT tool and make code changes when needed. This because there are many forms that rely on the engine type to be able to show the correct information. Then tests will be done to make

sure the tool is working and the correct information is being shown for the different engine types. After that, VBA code cleaning will be done to clean up the code in the tool as it has lived for a long time, therefore, there are parts that are not used anymore. Then a migration from MS Access to the SQL Server will be done to test that it works and that all steps are taken to make the final migration as smooth as possible. Lastly, a rehearsal will be made before the final data migration to see that everything is working according to the plan. The final migration does also has an implementation process in place to make sure the correct tasks are done in the correct order.

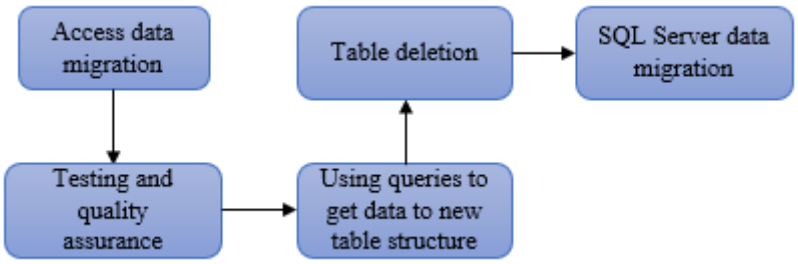

Figure 24. The tasks that will be done during the final migration

The final migration will be done when the new re-engineered tool and the new reporting tool are ready. The two existing tools will be taken out of use for a day or two to make sure no data is edited or added during the migration to get all up-to-date information to the new tool and to the SQL Server. The final migration contains as stated earlier, two migration steps. First, the migration to get all data from the two existing tools to the new tool and the other to get the data from Access to SQL Server. In between the migration steps, testing will be done to make sure all data is being migrated. For the new table structures that could not be implemented in the existing tools, queries will be used to migrate the data from the old table structure to the new ones. After all of the data is in the correct tables, the tables that will not be used anymore will be deleted. Furthermore, when the steps are done in the Access tool, the SQL Server migration will be conducted, following the steps presented in the previous chapter 5.4.2.

5.6 Summary

This chapter started with a description of what research method that will be used. The constructive research approach was chosen because of the nature of this thesis. This thesis takes a real-world problem and tries to solve it, using theory and gathered knowledge of the current situation. The seven-step process by Lukka (2000;2003) will and have been used during this research. This chapter has worked as the fourth step in the process, the most critical one, to establish a solution idea. This plan consists of a more in-depth analysis of the existing tools and a plan on how to re-engineer the tool. In between this chapter and the next chapter 6, the plan will be executed by using the implementation processes discussed and the next chapter 7 will present the result of the implementation.

6 RESULTS

This chapter will discuss and explain the results of the implementation phase. The implementation process in figure 23 has been followed successfully. The results that will be presented further are re-designing results and data migration results. At the end of this chapter, the 6th and 7th step of the constructive research approach will be discussed to see what was learned during the process. Due to the time constraint and that the new reporting tool was not ready the final migration will not be in the scope of the result but will be discussed in chapter 6.5 regarding the next steps.

6.1 MS Access data migration results

The first step migration to migrate the data from the two existing tools to the development tool was done by using the already made "Copy tables" function to migrate the WFI data and another modified version of it to append the information from the WIT tool. Before the migration could be done the tables were harmonized in both tools to make sure the migration would proceed successfully. The tables presented in the first step analysis in chapter 5.3.1 were the tables in focus.

Almost all of the 31 tables that could only be found in one of the tools were tables that were used because of differences in data needed between the different engine types. As the WFI tool was used for the base of the development, the structure of the five WIT tables that were used only in the WIT tool could be copied straight to the development tool. The one table that could not be copied straight was a table called "ParentDataUndependentOfTestRun". Investigation results showed that the WIT tool had split parent engine data related to the test run into two tables, where the WFI tool only used one. It was chosen to use the WFI way of working and therefore, the two tables in the WIT tool were merged together. This was done by changing the structure of the table "ParentDataCanChangeForTestRun" to make it the same as in the WFI tool and then use a query to get the data from the table "ParentDataUndependentOfTestRun". This was done in the live WIT tool to make the final migration easier when more data is already harmonized

in the two live tools. To make the change work in the live WIT tool, the VBA code also had to be changed to only save the parent engine data into the “ParentDataCanChange-ForTestRun” table, for this the WFI code was used for an example to make it work the same in the WIT tool.

The 76 tables that had data fields with different field sizes discussed in chapter 5.3.1 were harmonized by taking the bigger field size and changing the other tool’s field size according to that. Furthermore, the analysis of the 18 mutual tables also discussed in chapter 5.3.1 that had differences in fields showed that also here the fields that could be found only in one of the tools had the information needed for the engine types in that tool. Therefore, the table structure in the development tool had to be changed according to the differences. To accommodate the new fields, the VBA code had to be changed, for example, changes were needed in the code for saving and updating the data in the table to save the new data to the correct field and also changes in the forms were needed where the new data is to be inserted. To make sure all changes were made correctly, testing was done in the testing phase that was concluded in the latter part of the implementation process (figure 23). After all the tables in the two existing tools were harmonized, the two migration functions could be used and were used successfully and migrated all data to the development tool for further development. The same data were used during the whole project and the final data will be moved during the final migration. To sum up, as written earlier the WFI tool that was used as a base for the development had 151 tables (see chapter 4.2) and with adding the new 5 tables from WIT that was exclusive for that tool, the final amount of tables in the development tool after the migration was 156 tables. This amount did change during the implementation phase due to new tables needed for the document number requirement.

6.2 Re-designing and restructuring results

The re-designing and restructuring of tables and forms was a big part of this project. It was conducted using prototyping and prototypes were tested and used by the users to get feedback.

6.2.1 Table restructuring results

The table restructuring done followed the plan made in chapter 5.3.1, where the biggest change needed was for the components influencing NOx. The hard part about restructuring tables in the development tool is how to migrate the old data to the new structure. Small changes could be made already in the two live tools. But bigger changes were made only in the development tool, for example, the component table changes, they were only done in the development tool. For bigger changes, queries were developed in the development tool to get the correct data from the old table structures to the new tables. Using queries is a risk because they have to be used when the final migration is being executed to move the up-to-date data but it is the only option when doing a big bang migration. This adds another step in the final migration where errors can occur but that risk was chosen over the other option to change the table structures in the two live tools and in the development tool.

Furthermore, after the table restructuring and forms redesigning had been made, the tables were again investigated to find the tables that will not be needed anymore in the new tool. The corresponding standard module to the tables can also be deleted because they hold the SQL code to save and retrieve data from the tables. The result of the investigation was 46 tables and 40 standard modules that were not needed anymore, mainly tables containing different document numbers. The table and standard module names were written down and were deleted during the final migration rehearsal to make sure the tool still worked and finally the tables will be deleted when the final migration is being executed.

6.2.2 Forms redesigning results

The redesign of forms will have the most impact on the user of the tool. There were done a lot of small and bigger changes to the forms in the tool and in this chapter the biggest ones will be presented. The first change the user will notice is when they open the tool and sees the start page. The main change needed for the start page was to accommodate all engine types and the result of the reengineered start page is:

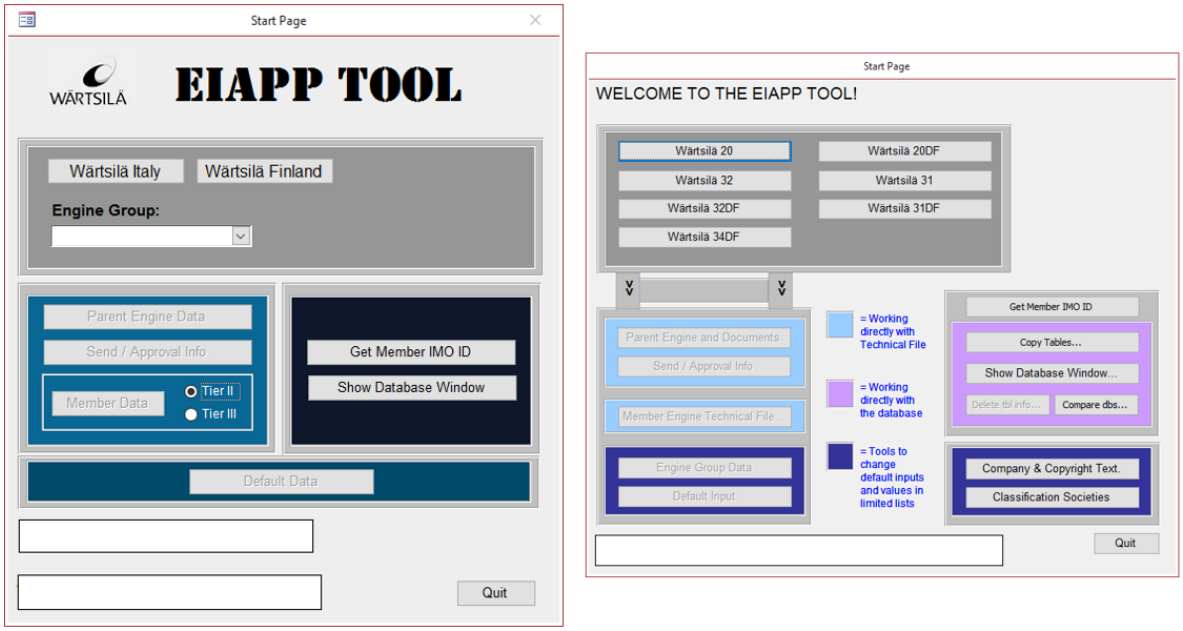

Figure 25. The new start page on the left compared to the old one on the right

The colors of the new start page have been changed according to new Wärtsilä color schemes. In the top grey box can be seen how the engine types have been integrated. The user first chooses either Wärtsilä Italy or Wärtsilä Finland. Dependent on the selected location, the engine group list will be populated with the correct engine groups and the user selects the wanted engine group from the “Engine Group” combo box. The other parts of the new start page have been done to make the tool easier to use by, for example, choosing tier before moving to member data, this can be compared to the old way where the user moved between the tiers in the member forms with the use of a checkbox that could be hard to see. All default data can now be changed in the default data section compared to the four buttons found in the old tool in the dark blue boxes. The default data section will be looked at next.

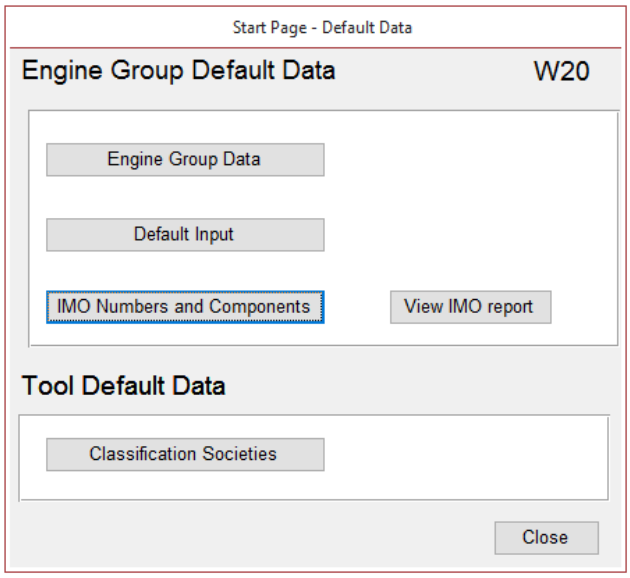

Figure 26. New engine group default data form

The new form has taken the “buttons” from the old start page and merged everything regarding default data into a new form to not have the start page crowded with buttons. Behind the “View IMO report” can be found the only report in the new tool. It is a newly developed report that includes all components and their IMO id:s regarding an engine type, in this example for W20. When a user clicks on the “IMO Numbers and Components” button the user will move to a re-engineered form based on the form seen in figure 14. The difference is that the whole list box with the available documents was removed and the user only has to choose a component and then click on a button to add or edit information for that component. For each component, the user can add or edit IMO id:s. The difference from the old tool is that in the old tool one component had many documents and each document held one or a combination of IMO id:s and for each new IMO id or combination of IMO id:s a new document had to be created. Now the user has a list for each IMO id that the component can have and later in the parent section the user selects what IMO id the parent engine has. The next section covers the new re-engineered parent section.

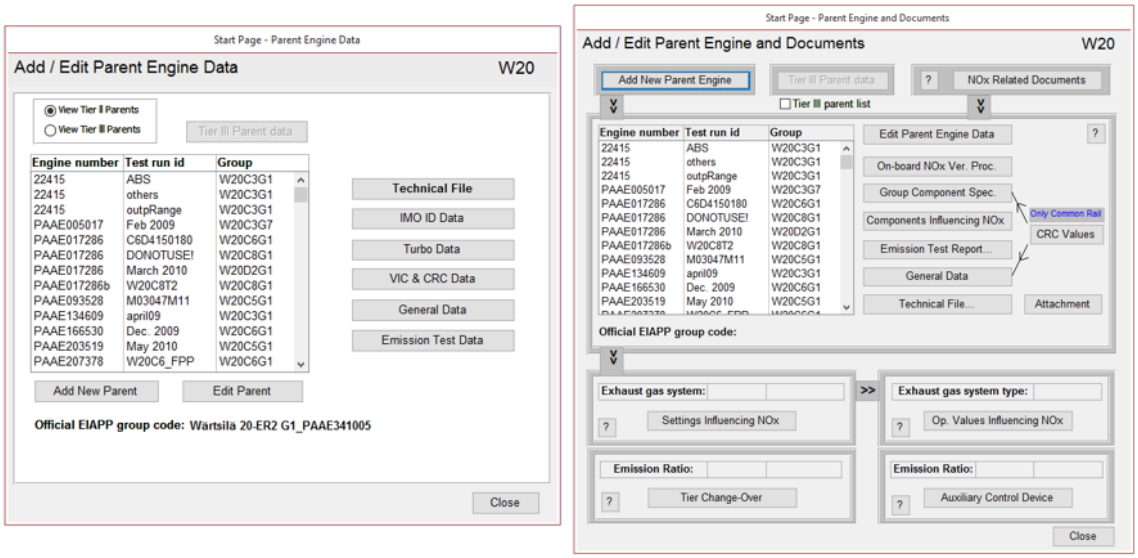

Figure 27. The new parent form on the left compared to the old on the right

The old parent form was very crowded with buttons because of all the different reports. The new parent form is re-engineered on the base of the content of the data instead of documents. The information behind the buttons seen on the right side of the list containing parents in the old tool has in the new tool been grouped according to the content, for example, all turbocharger data in one form. The buttons and data on the bottom part of the old tool are not included in the new tool as that was only information used for document specific information. Other changes are for example, how the user can change tier, from using a checkbox to now using a selection box. The edit and add parent buttons are now moved closer to each other and are more visible. In the old tool, the “Technical File” button was in the bottom of the other buttons but now it is moved to the top for better visibility and to showcase that the other information is inserted in the Technical File. The next and last form that will be covered is behind the button “IMO ID Data” and that is connected to the components influencing NOx.

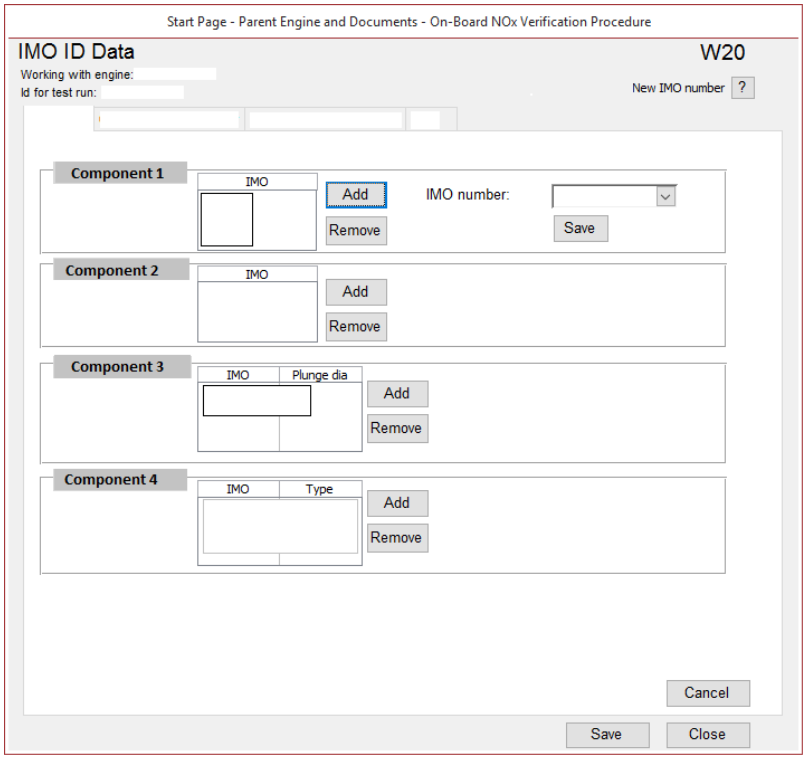

Figure 28. The new form where IMO ID:s are added to a parent engine

In the old tool in this form, the user chose the document number that had the correct IMO id:s for the specific parent engine and for the specific component. Now in the new tool, the user chooses directly the correct IMO id:s to each component. In the figure for component 1 can be seen how an IMO id is added. In the "IMO number" combo box seen on the top right part of the form in regard to component 1 will all IMO id:s be found that have been added in the default data section to this specific component 1 and regarding the engine type W20 in this example. This makes it a lot easier when there is a need for multiple IMO id:s for a component, where now the user only chooses the needed ones compared to in the old tool where a document was needed for each IMO id and each combination.

The discussed form changes are the biggest changes made regarding re-designing. All changes made were either because of the document number change, easy of use or for the migration of all data into one tool. With the use of prototyping and close cooperation with the users, a good result of the re-engineering was guaranteed.

6.2.3 Automating tasks results

During the planning phase (chapter 5) and the implementation phase, only one task was found that could be automated. The one task found was when inserting data into the group data tables found in the emission test data section. Before the data were inserted line by line and the new idea was to insert when possible many lines in the table at once by choosing a combination of data to be inserted, the result can be seen in figure 29 and it can be compared to figure 21.

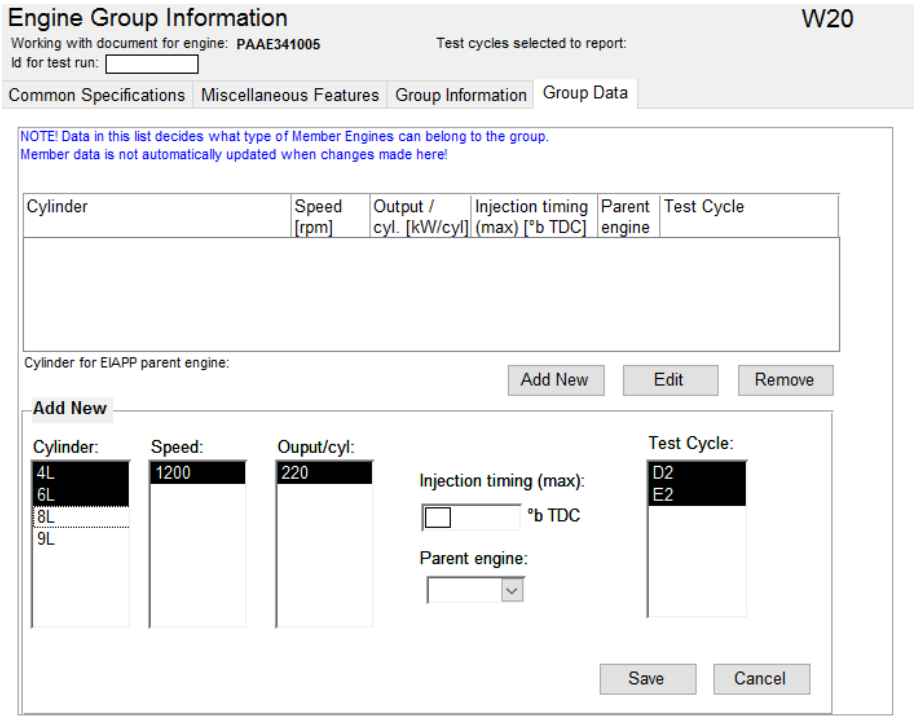

Figure 29. The new way of working in inserting engine group data in the new tool

The figure shows the result of automating the inserting of data in the engine group data table. Here the user can still insert one line at a time but also as seen in the example choose many options, and in this example with two cylinder configurations selected and two test cycles selected, the result will be four lines in the table. This will speed up the work for creating parent engines with a lot of combinations, when now if the injection timing is the same, many combinations can be inserted at the same time.

6.3 Final migration rehearsal results

The final migration rehearsal followed the final migration process plan made in chapter 5.5. The two functions made to copy the information from the two existing tools to the development tool were first used. Some small problems were found when trying to get the WFI data to the development tool due to table changes made during the development. After the problems were solved the data could be merged over to the development tool. After the data was successfully migrated, the queries could be used to get the correct component data to the new table structures. For this, a VBA function was created to use all the queries automatically. This to avoid human errors when using the queries manually. After the tables and modules that were earlier found to not be needed anymore were deleted the function to count lines and objects could be used again to now compare the re-engineered tool with the existing tools.

Table 4. Table comparing the new tool to the existing tools regarding lines of code

Lines of code	WFI	WIT	WFI + WIT	New Tool	Change Total(%)	Change from WFI (%)
Stand-alone modules	43977	42140	86117	43663	-49,30 %	-0,71 %
Form modules	50066	47990	98056	41901	-57,27 %	-16,31 %
Report modules	7818	7610	15428	59	-99,62 %	-99,25 %
Total	**101861**	**97740**	**199601**	**85623**	**-57,10 %**	**-15,94 %**

In the table, it can be seen the number of code lines in the new table and two different comparisons made. The “Change Total (%)” compares the new tool to the combined total of the existing tools (column WFI + WIT) and the last column compares the new tool to the WFI tool because the WFI tool was used as the base for the new tool. The biggest difference will, of course, be in the report modules because the new tool only has one report. The total amount of code in the new tool is decreased by **57,1 %** compared to the combined existing tool and it has decreased by **15,83 %** compared to the WFI tool.

Table 5. Table comparing the new tool to the existing tools regarding objects

Number of	WFI	WIT	WFI + WIT	New Tool	Change Total (%)	Change from WFI (%)
Stand-alone modules	145	135	280	138	-50,71 %	-4,83 %
Forms	78	82	160	78	-51,25 %	0,00 %
Reports	39	39	78	1	-98,72 %	-97,44 %
Tables	151	134	285	145	-49,12 %	-3,97 %
Total	**413**	**390**	**803**	**362**	**-54,92 %**	**-12,35 %**

The object comparison between the new tool and the combined total and the WFI tool have similar results as the lines of code comparison. Here the total amount of objects in the new tool compared to the total objects in the two existing tools has decreased by **54,92 %.** Both table 4 and table 5 showcases the big difference it will be for especially the developers when dealing with **50 %** fewer lines of code and objects compared to the old two tools combined.

6.4 Reflection and theoretical contribution

The last two steps in the constructive research are about reflecting on the solution and identifying the theoretical contribution. As written in the theoretical framework, re-engineering has and is mainly used for legacy systems that use outdated technology and the goal of re-engineering is to re-write the software with a new platform. The results presented in this chapter show the importance of software re-engineering also in the field where the existing software is still fully working and uses up-to-date technology but in a need of a rebuild to fulfill new requirements.

The results show that the use of the constructive research approach is good for this kind of project and problem that may not have been done otherwise due to lack of resources because software re-engineering takes a lot of time, but still a lot less than developing new software. The seven-step constructive research approach was followed and it gave a good guideline on what to do and in what order. It lifts up the importance of close cooperation between the researcher and the organization and that is also very important in the software re-engineering field. With the use of constructive research and general software re-engineering models and processes, the main problems of software engineering and software re-engineering discussed in chapters 2 and 3 about projects being delivered late and over budget can be minimized.

Even though the tool is not yet taken into use, the result regarding the successful migration into one single tool and the reduced amount of code and objects in the new tool will make it easier to maintain and develop the tool for the developers. The new tool will not only reduce the time needed for the developers it will also reduce the time for the users to create the Technical Files. The task to automate processes was a task that was thought to have a bigger impact on the new tool before initiating the project. The results show that it could have had a bigger role if the information needed to be inserted in the tool was more streamlined.

The biggest theoretical contribution of this thesis is the showcase of what constructive research can accomplish in software engineering and software re-engineering. It also shows that it works for software that not necessarily have to be re-engineered but re-engineering will help both the user and the developers. The new improved tool is easier, more reliable and more efficient to use by the users and it is saving a lot of time for the developers to maintain and update it compared to the old one. This thesis does again show that for software engineering and software re-engineering projects there are no general models or processes that can be used by all projects. In this project, a lot of different and modified models and processes have been used.

The project has followed the re-engineering model seen in figure 4 by Majthoub et al. (2018). Where the reverse engineering phase has been followed in this research by taking

the existing tool and examined it from different angles, beginning from a low abstraction level and moving up to a high abstraction level. The implementation phase then followed the forward engineering phase which is the same as for other software engineering developments. Then the research paper itself has followed the constructive research approach and the project implementation and data migration have followed the big bang approach.

6.5 Next steps

As stated at the beginning of this chapter 6, the final data migration could not be done during the thesis, because the new reporting tool that will replace the Access reports and generate the Technical File is not ready, and that is required to be taken into use at the same time as the new tool. This means that the tool will not be taken into use and no result of how the tool was received will be gathered. Therefore, in this chapter, the next steps will be discussed.

When the new reporting tool is ready to be taken into use the final migration can be done. The final migration will follow the plan established in chapter 5.5. Because the migration will be done by another developer, a more detailed document has to be created that explains every step that has to be taken. The document has to also explain where problems can occur and how to solve them. One big drawback now when the reporting tool is not ready is that all three tools have to be maintained in parallel, which means that if there are updates in the forms in the existing tools, the same update has to be done in the new tool.

The big bang approach will be used for the final migration as explained before. This means that the developer has to choose a day or two when the current live tools are taken out of use, to get the up-to-date data to the SQL Server. When the migration is finished the developer has to test the new tool together with the new reporting tool to make sure they work together. Lastly, the tools will be taken into use and the real testing will be done as the user starts to use the tools. For the following weeks after the migration, the developer has to be ready to maintain and fix problems that can occur.

7 CONCLUSIONS AND FUTURE DEVELOPMENT

Overall this research was a software re-engineering project that included data migration and software development. The aim of the thesis was to re-engineer the EIAPP tool according to new requirements and at the same time improve the tool by looking at possibilities to automate processes and at last migrate the data to a new database platform. This research covered the whole development process from planning to implementation and execution.

The first research question was how to re-engineer the tool when going from document management to content management. This was accomplished by analyzing the existing tools and gather information on where the current document numbers were used and how. The result of the analysis showed that most of the document numbers were not used as primary keys in tables, which leads to no need in changing the table structures. The document numbers that were used for primary keys were regarding components influencing NOx. The forms and tables regarding these components had to be re-engineered. The result was a new table structure that connected the IMO numbers to each engine type and when creating parent engines the IMO numbers for each component were connected to the specific parent engine.

Furthermore, another result of the re-designing of forms regarding the document number requirement was a new way of grouping where the information was inserted in the tool. In the old tool, the data was inserted into forms that all had their own report with its own document number. Now the result of the re-engineering is that the data is grouped according to its content so for example, all turbocharger data is inserted in one form and all IMO numbers in another form. This makes the tool easier to understand for new users and easier to use.

Another research question was if it is possible to automate functions in the tool to speed up the work needed by the users to insert data. Meetings were held with the certification engineers that use the tool to understand from where they take the data before inserting it into the tool. It fast became clear that the data is taken from a lot of different places and

there were no streamlined sources that could be used to automatically insert some data. Therefore, the focus changed towards inside the tool to see if there are possibilities to automate processes there. One place was found where the user inserted engine group data into a table in a form, this was done one line at a time but the inserted data was found to be very similar and could often be inserted in combinations. The result was a new way of inserting the data where the users could choose a combination of input values and then it automatically inserted all the rows at once, one for each combination.

The third and last research question was how to do the data migration. There were two different data migration that had to be done. One to get the data from the two existing tools to one single tool and then the other to get the data from Microsoft Access to SQL Server. Before being able to migrate the data, an analysis of the tables had to be done to understand the differences between the two existing tools to be able to harmonize the tables in the two tools to easier migrate the data. The result showed that most differences were because of the difference in engine types between the tools.

The migration from the two old tools was conducted by using an already made function in the existing tools and a modified function of that one was coded. The function has been used in the old tool to copy over data from one version to another. This same function was used to get the WFI data to the new tool but to be able to get the WIT data, another function had to be created that appended the WIT data. When all data was migrated to the development tool, queries were used to get the component data to the new table structure. The 46 tables and 40 standard modules that were not needed anymore could then be deleted before the second migration was conducted. The Microsoft SQL Server Migration Assistant (SSMA) was then used to migrate the data from Microsoft Access to the SQL Server.

The final result of the re-engineering is a reduction of code lines from 199 601 (WFI and WIT together) to 85 623, which is a reduction of 57,1 %. If compared to only the WFI tool that was used as the base for the new tool the reduction was 15,94 %. The reduction in objects from both tools to the new tool is from 803 to 362, which is a reduction of 54,92 % and compared to the WFI it is a reduction of 12,35 %. While the changes in the

forms are the biggest change for the user, the reduction of objects and lines of code is the biggest change for the developers.

The hard part about the project was because the big bang approach had to be used for the re-design and data migration. This meant that if changes had to be done to the two live tools during the development the changes also had to be done in the development tool and that took extra time. Another hard part of the research and the development was that as a developer you have to understand exactly how the tool is used to be able to understand how to improve it and how to change it. Therefore the close cooperation with the users was very important and the prototyping helped to get their feedback and get new ideas. Without this kind of research, the re-engineering of the tool would have been very hard, because this research gathered theory and a deep understanding of how the existing processes and tools work. Without that knowledge and close cooperation with the users, the result would probably not have been the same.

Further development proposals would be to streamline the data that is needed for the Technical File. This could include a streamline of the process of doing the EIAPP tests to save the data in a way that the tool could automatically retrieve the data without needing a user in between to insert the data. It is understandable that all information cannot be automated but if more data would be in streamlined sources they could be retrieved automatically. This would make the time needed for the certification engineers to insert data a lot less and human errors in inserting data could be prevented. This could be a whole thesis on its own to understand exactly where all data is taken from and try to streamline it and this would also need a lot of cooperation between the different teams in the company and would not be possible only by the Compliance & Certification team.

REFERENCES

Ahmadi, Ramin. Cami, Bagher Rahimpour & Hassanpour, Hamid (2012). Automatic Data Migration between Two Databases with Different Structure. International Journal of Applied Information Systems (IJAIS). Foundation of Computer Science FCS, New York, USA. Vol. 3, No. 3, July 2012. ISSN: 2249-0868

Al-Husseini, Khansaa Azeez Obayes & Obaid, Ali Hamzah (2018). *Usage of Prototyping in Software Testing.* Multi-Knowledge Electronic Comprehensive Journal For Education And Science Publications (MECSJ). Issue 14, November 2018. Available at: https://www.researchgate.net/publication/329454078_USAGE_OF_PROTOTYPING_IN_SOFTWARE_TESTING

Bassil, Youssef (2012). *A Simulation Model for the Waterfall Software Development Life Cycle*. International Journal of Engineering & Technology (iJET). Vol. 2, No. 5, 2012. ISSN: 2049-3444

Bernhart, Mario. Mauczka, Andreas. Fiedler, Michael. Strobl, Stefan & Grechenig, Thomas (2012). Incremental Reengineering and Migration of a 40 Year Old Airport Operations System. 28th IEEE International Conference on Software Maintenance (ICSM). Trento, 2012, pp. 503-510. doi: 10.1109/ICSM.2012.6405313

Browne, Allen (2008). Count lines (VBA code). [online]. [30.7.2019]. Available at: http://allenbrowne.com/vba-CountLines.html

Chikofsky, Elliot J & Cross II, James H. (1990). *Reverse Engineering and Design Recovery: A Taxonomy*. IEEE Software, Vol. 7, Issue 1, pages 13-17, January 1990. Available at: https://ieeexplore.ieee.org/document/43044

Connolly, Thomas & Begg, Carolyn (2015). *Database Systems: A Practical Approach to Design, Implementation and Management*. 6th Ed. England: Pearson Education Limited. 1328 p. ISBN 13: 978-1-292-06118-4

Elamparithi, M. & Anuratha, V. (2015). *A Review on Database Migration Strategies, Techniques and Tools*. World Journal of Computer Application and Technology 3(3): 41-48, 2015. Available at: https://www.researchgate.net/publication/299534668_A_Review_on_Database_Migration_Strategies_Techniques_and_Tools

Elmasri, Ramez & Navathe, Shamkant B. (2016). Fundamentals of Database System. 7th Ed. United States of America: Pearson Education, Inc. 1242 p. ISBN-13: 978-0-13-397077-7

Gouda, Chidananda. Patil, Sudarshan. Kumar, Anil. Prasad, Guru & Madhavi, Sai (2016). *Database Migration Tool*. International Journal of Computational Engineering Research (IJCER). Vol 06, Issue 05, May 2016. ISSN (e): 2250-3005

Graciamary, A. Cathreen & Chidambaram, M. (2018). *EESRM: An Effective Approach to Improve the Performance of Software Re-Engineering*. International Journal of Applied Engineering Research. Vol. 12, Number 6 (2018). pp. 3648-3654. Research India Publications. ISSN: 0973-4562

Howard, Philip (2014). *Data Migration Customer Survey*. An Original Research Paper by Bloor Research. Available ay: https://www.bloorresearch.com/research/data-migration-customer-survey/

IMO (2017). *Marpol Annex VI and NTC 2018 with guidelines for implementation*. 2017 Ed. United Kingdom: CPI Group Ltd. 422 p. ISBN 978-92-801-1658-8

IMO (2019). *Introduction to IMO*. International Maritime Organization. About IMO. [online]. [25.9.2019]. Available at: http://www.imo.org/en/About/Pages/Default.aspx

Lalitha, R. Lalithakumari, G & Surekha, Y. (2016). *Classical Data Migration Technique in Multi-Database Systems (SQL and NOSQL)*. International Journal of Computer Science and Information Technologies. Vol. 7, 2016. pp. 2472-2475. ISSN:0975-9646

Leach, Ronald J (2016). *Introduction to Software Engineering*. 2nd Ed. Florida: Taylor & Francis Group. 390 p. ISBN 978-1-4987-0528-8.

Lukka, Kari (2000). *The key issues of applying the constructive approach to field research*. In: Management Expertise for the New Millennium. Editor: Reponen, T, pp. 113-128. Turku School of Economics and Business Administration. Available at: https://www.researchgate.net/publication/281549256_The_key_issues_of_applying_the_constructive_approach_to_field_research

Lukka, Kari (2003). *The Constructive Research Approach. In: Case study research in logistics*. Editors: Ojala, L. Hilmola, O-P, pp. 83-101. Publication of the Turku School of Economics and Business Administration. Available at: https://www.researchgate.net/publication/247817908_The_Constructive_Research_Approach

Maatuk, Abdelsalam. Ali, Akhtar & Rossiter, Nick (2011). *Re-engineering relational databases: The way forward*. Proceedings of the 2nd International Conference on Intelligent Semantic Web-Services and Applications, ISWSA 2011, Amman, Jordan, April 18-20, 2011. Available at: https://www.researchgate.net/publication/220717114_Re-engineering_relational_databases_The_way_forward

Majthoub, Manar. Qutqut, Mahmoud H & Odeh, Yousra. (2018). *Software Re-engineering. An Overview*. The 8th International Conference on Computer Science and Information Technology (CSIT 2018), July 2018. Available at: https://www.researchgate.net/publication/326696263_Software_Re-engineering_An_Overview

Matthes, Florian. Schulz, Christopher. Haller, Klaus (2011). *Testing & Quality Assurance in Data Migration Projects*. 27th IEEE International Conference on Software Maintenance (ICSM'11), Williamsburg, VA. 25-30 September 2011. Available at: https://ieeexplore.ieee.org/document/6080811

Microsoft (2019a). *Introduction to tables*. [online]. [13.7.2019]. Available at: https://support.office.com/en-us/article/introduction-to-tables-78ff21ea-2f76-4fb0-8af6-c318d1ee0ea7

Microsoft (2019b). *Introduction to queries.* [online]. [17.7.2019]. Available at: https://support.office.com/en-us/article/introduction-to-queries-a9739a09-d3ff-4f36-8ac3-5760249fb65c

Microsoft (2019c). *Introduction to forms.* [online]. [17.7.2019]. Available at: https://support.office.com/en-us/article/introduction-to-forms-e8d47343-c937-44e8-a80f-b6a83a1fa3ae

Microsoft (2019d). *Introduction to reports.* [online]. [17.7.2019]. Available at: https://support.office.com/en-us/article/introduction-to-reports-in-access-e0869f59-7536-4d19-8e05-7158dcd3681c

Microsoft (2019e). *Introduction to Access programming.* [online]. [19.7.2019]. at online: https://support.office.com/en-us/article/introduction-to-access-programming-92eb616b-3204-4121-9277-70649e33be4f

Microsoft (2019f). *Migrate and Access database to SQL Server.* [online]. [23.7.2019]. Available at: https://support.office.com/en-us/article/migrate-an-access-database-to-sql-server-7bac0438-498a-4f53-b17b-cc22fc42c979

Microsoft (2019g). *Database basics*. [online]. [23.7.2019]. Available at: https://support.office.com/en-us/article/database-basics-a849ac16-07c7-4a31-9948-3c8c94a7c204

Microsoft (2019h). *Comparing Access and SQL Server data types*. [online]. [30.7.2019]. Available at: https://support.office.com/en-us/article/comparing-access-and-sql-server-data-types-9188f41d-6c0e-4733-9d20-d08916f50bd2

Morris, Johny (2012). *Practical Data Migration*. 2nd Ed. United Kingdom: BCS Learning & Development Ltd. 248 p. ISBN: 978-1-906124-84-7

Munassar, Nabil Mohammed Ali & Govardhan, A. (2010). *A Comparison Between Five Models Of Software Engineering.* IJSI International Journal of Computer Science Issues, Vol. 7, Issue 5 (September 2010). p. 94-101. ISSN (Online): 1694-0814.

Müller, Hausi A. Jahnke, Jens H. Smith, Dennis B. Storey, Margaret-Anne. Tilley, Scott R & Wong, Kenny (2000). *Reverse Engineering: A Roadmap*. ICSE'00 Proceedings of the Conference on The Future of Software Engineering. pp 47-60, 11 June 2000. Available at: https://www.researchgate.net/publication/2885971_Reverse_Engineering_A_Roadmap

Oracle (2011). *Successful Data Migration*. An Oracle White Paper. Available at: https://www.oracle.com/technetwork/middleware/oedq/successful-data-migration-wp-1555708.pdf

Pressman, Roger S (2010). *Software Engineering: A Practioner's Approach*. 7th Ed. New York: McGraw-Hill. 895 p. ISBN 978-0-07-337597-7.

Rosenberg, Linda H. (n.d.). *Software Re-engineering*. Software Assurance Technology Center. Goddard Space Flight Center, NASA. [online]. [1.7.2019]. Available at: https://www.semanticscholar.org/paper/Software-Re-engineering-Rosenberg/693114c2f50a63f449cbb27b7680ace6624dae23

Rouse, Margaret. Hughes, Adam & Stedman, Craig (2019). *Microsoft SQL Server*. Microsoft Ignite 2017 conference coverage. TechTarget. [online]. [20.7.2019]. Available at: https://searchsqlserver.techtarget.com/definition/SQL-Server

Sommerville, Ian (2016). *Software Engineering*. 10th Ed. England: Pearson Education Limited. 810 p. ISBN 978-1-292-09613.1.

Tutorials Point (2018). *MS Access*. Tutorials Point Pvt. Ltd. Available at: https://www.tutorialspoint.com/ms_access/ms_access_tutorial.pdf

Ulrich Fuller, Laurie & Cook, Ken (2013). *Microsoft Access 2013 For Dummies*. New Jersey. John Wiley & Sons, Inc. 436 p. ISBN 978-1-118-56851-4

Velimeneti, Sushma (2016). *Data Migration from Legacy Systems to Modern Database*. Culminating of Mechanical and Manufacturing Engineering. St. Cloud State University. Paper 54. Available at: https://pdfs.semanticscholar.org/3ade/2ef39cbd3b0b72722107be52bd5ee0bf5041.pdf

Wärtsilä (2020). *About Wärtsilä: This is Wärtsilä*. [online]. [10.2.2020]. Available at: https://www.wartsila.com/about

Yang, Xiaohu. Chen, Lu. Wang, Xinyu & Cristoforo, Jerry (2005). *A Dual-Spiral Reengineering Model for Legacy Systems*. TENCON 2005. 2005 IEEE Region Conference. Available at: https://www.researchgate.net/publication/224280752_A_Dual-Spiral_Reengineering_Model_for_Legacy_System