YRKESHÖGSKOLAN
NOVIA

Optimization of Documentation Processes and Tools

Johan Engblom

Bachelor's thesis for Wärtsilä Finland Oy

Produktionsekonomi

Vasa 2022

EXAMENSARBETE

Författare: Johan Engblom

Utbildning och ort: Produktionsekonomi, Vasa

Handledare: Leif Backlund

Titel: Optimering av Dokumentations Verktyg och Processer

Datum: 02.04.2023 Sidantal: 41 Bilagor: 7

Abstrakt

Målet med denna studie var att undersöka fördelarna med att optimera en utvald del av dokumentation-arbetsflödet vid Wärtsilä Finland Oy. I detta examensarbete innebär det att utveckla och implementera ett verktyg i form av ett program som automatiserar uppgifter som tidigare har gjorts manuellt av arbetare. Detta verktyg har utvecklats med att minska på mänskliga fel samtidigt som arbetseffektiviteten och kvaliteten ökar i åtanke.

Denna studie fokuserar huvudsakligen på att förbättra arbetsflödet inom Wärtsiläs interna dokumentationsrelaterade mjukvara vid namn WSHS. Målet med denna studie uppnåddes men hjälp av en blandning av kvantitativa och kvalitativa metoder, forskning i vad optimering är, diskussioner med kollegor samt problemlösning på basis av min arbetserfarenhet. Resultatet från implementering av programmet i arbetsprocessen var väldigt positivt, vilket visar att optimering av utvalda delar i en arbetsprocess kan ha en betydande inverkan på tidshantering och effektivitet med avseende på aktivt arbete.

Språk: svenska
Nyckelord: optimering, automatisering, arbetseffektivitet, kvalitativ/kvantitativ forskning

BACHELOR'S THESIS

Author: Johan Engblom
Degree Programme: Industrial Management and Engineering
Supervisor(s): Leif Backlund

Title: Optimization of Documentation Tools and Processes

Date 02.04.2023 Number of pages: 41 Appendices: 7

Abstract

The goal of this study was to research the benefits of optimizing a select part of the documentation work process at Wärtsilä Finland Oy. To elaborate, this means developing and implementing a tool in the form of a program that automates tasks that previously have been done manually by workers. This tool was developed with decreasing human errors while increasing the work-efficiency and quality in mind.

This study focuses mainly on improving the workflow within Wärtsilä´s internal documentation-related software by the name of WSHS. To achieve the goal of the study, a mix of quantitative and qualitative methods were used, researching the aspects of optimization, discussing with colleagues, and problem-solving with my own work experience in mind. The results from implementing the program into the work process ended up being very positive which goes to show that optimization of select parts in a work process can make a significant impact on time management and effectiveness regarding active work.

Language: English
Keywords: optimization, automation, work efficiency, qualitative/quantitative research

Table of contents

List of abbreviations

BOM	Bill of material
WSHS	Wärtsilä Specification Handling System
VBA	Visual Basic for Applications
QOL	Quality of Life
ERP	Enterprise Resource Planning
SPC	Spare Part Catalogue

1 Introduction

This thesis was done in collaboration with Wärtsilä Finland Oy to research the benefits of optimizing select stages in a work process in order to increase effectiveness and quality.

Wärtsilä as a company is one of the global leaders when it comes to solutions in marine and energy markets. With a strong emphasis on innovation within sustainable technology and lifecycle solutions, Wärtsilä are as of 2023 operating in 79 different countries with over 17 500 employees. (Wärtsilä Finland Oy, 2023)

The research data is based on optimization of a select part in the documentation process of engines at Wärtsilä Finland Oy. This means that the time taken to complete certain tasks regarding data processing is decreased, while the efficiency and quality of the work is retained, but also increased. There are a lot of ways this can be done, but I chose the route of utilizing VBA programming and partial automation. I will describe and explain the different methods used more thoroughly in the theoretical part of this thesis. The practical implementation of these methods, how it was planned and executed will be covered in the empirical part.

While this thesis was done for Wärtsilä, it can also be viewed as a testament to how optimization of only select parts in a longer work process can increase effectiveness and amount of work done which enables workers to spend more time on more crucial tasks and less on manual work where there is room for human error.

1.1 Background

The part in the work process that I have chosen to optimize is related to different material numbers connected to the same spare part numbers when it comes to documentation and data handling of engines at Wärtsilä.

Every engine has got a data structure containing all of its parts. This data structure is called a “Bill of Material-structure” and has different levels which you can go down depending on what you are looking for. When processing the BOM-structure of an engine you send the data through a couple of checkpoints. When the data is processed in these checkpoints, several lists of information regarding potential errors in the engine’s BOM-structure are compiled based upon a comparison between “master data” that already exists in an internal database and what the BOM-data structure is containing.

These lists must be manually checked for every single engine BOM-structure that is processed to make sure everything is correct. This process can take up to two hours from my own experience depending on the errors that are encountered.

In this thesis, the focus lies on optimizing one of the checkpoints in which you are given a compiled list of all spare part numbers that are connected to different material numbers. As there can only be one connection to a material number in the end you would have to go through this list manually and check every single one of the connections manually to figure out which of the different material numbers is the latest and correct one for the spare part number in question.

I started working on a solution to decrease the time spent checking the aforementioned duplicate errors already during the summer of 2022 while working at Wärtsilä but only as a personal, small quality-of-life project. I presented the results of the work I had done and received good feedback which led to the idea of making this my final thesis subject and continuing the development of these improvements to the work process by making it more sophisticated while perhaps creating something that is suitable to be a permanent addition to the work process in the future.

1.2 Objective

The objective of this study is to create a working toolset in the form of a program that is used to optimize the workflow and eliminate tedious manual work that takes away active work time from other things while also being very prone to human errors. This program is to be made available to everyone working with said tasks. In this case, the task is checking the previously mentioned connections between a spare part number and multiple material numbers.

The program will eliminate unnecessary spending of active work time on manually going through these compiled lists of potential errors in the engine's BOM-structure while retaining a high quality of work and increasing the number of engines that go through the duplicate check per day/week/month. The program has to be good enough to truly replace the manual process of checking each row in the compiled lists while ensuring that mistakes will not be made.

The programs' functionality will be based on the database of existing spare part numbers and material numbers at Wärtsilä and run on Visual Basic for Applications code. Updating the tool will also utilize this database but this will be described more in-depth in the empirical part.

The general hypothesis of this study is that developing the program for this select part of the documentation work process at Wärtsilä will have a positive result.

1.2.1 General program requirements

The following is the general list of both soft- and hard requirements for the program.

Hard requirements:

- Simple to use without extra training or courses.
- Effective enough to truly replace the manual counterpart.
- Simple to update and configure further depending on the area of usage.
- Universal online-based access for all users.
- When updated it is updated for all users at once (cloud/online based).
- Universally compatible with most engine models and their data structures.
- Program should be based on VBA programming.
- A simple-to-use interface that clearly states connections between spare numbers and material numbers.

Soft requirements:

- Integration with Teamcenter and/or SAP for automatic updating.
- Automatically executed when opening the duplicate error report for an engine.
- Automatic updating when new material numbers or spare part numbers are released.

2 Theory

In this part of the thesis, I will explain the theoretical elements of the study. This means that I will include information and theory regarding the different technologies and methods utilized to reach the thesis' goal.

2.1 Typical engine data structure

To start, we should take a look at what a typical engine data structure looks like. This will enhance the reading experience and understanding of the context of this thesis. The following is a picture of a typical engine BOM data structure opened up in a program called Teamcenter. (See appendix 1)

In the picture, the different parts and components of the engine are presented on the left in a tree structure; each main component contains the belonging sub-component(s) and their content. To the right, different columns contain information such as the parts' spare part numbers. This data is sent over to WSHS (internal Wärtsilä software) where the data is processed by the worker by sending it through the aforementioned checkpoints.

The following are figures picturing the amount of engine data structures that are processed per month. Each figure depicts one engine type and in this case, the types in question are W20, W32 and W46F. The figures are based upon data in Wärtsilä's Polarion which is a software to keep track of engines and modules that are in need of documentation

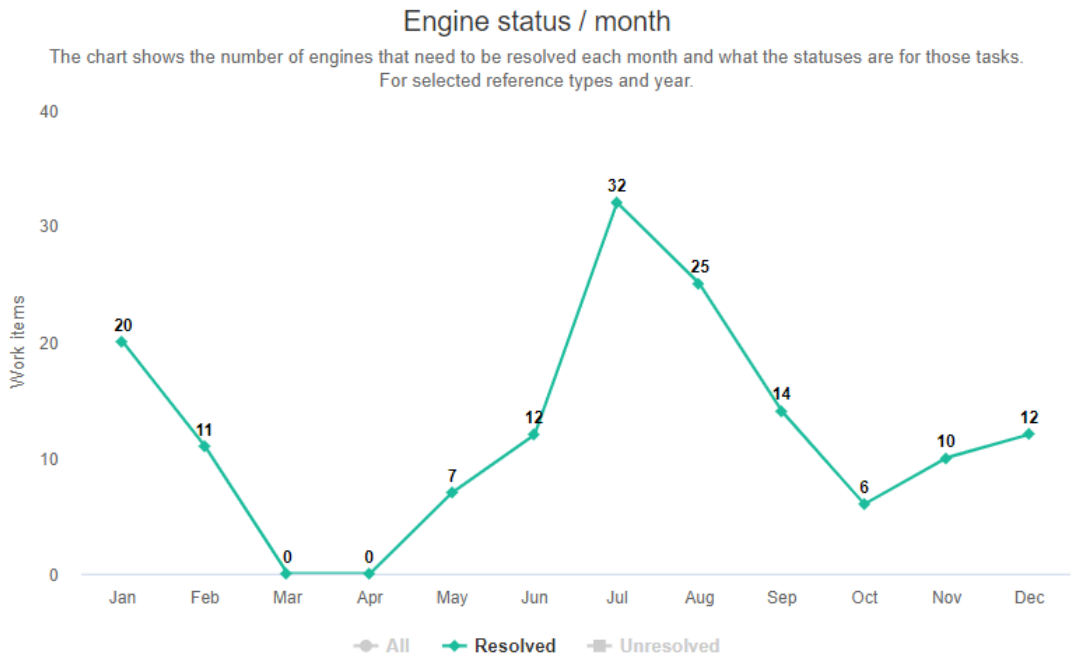


Figure 1. Graph showing the amount of completed W20 engines per month in the year 2022. (Wärtsilä Polarion 2023)

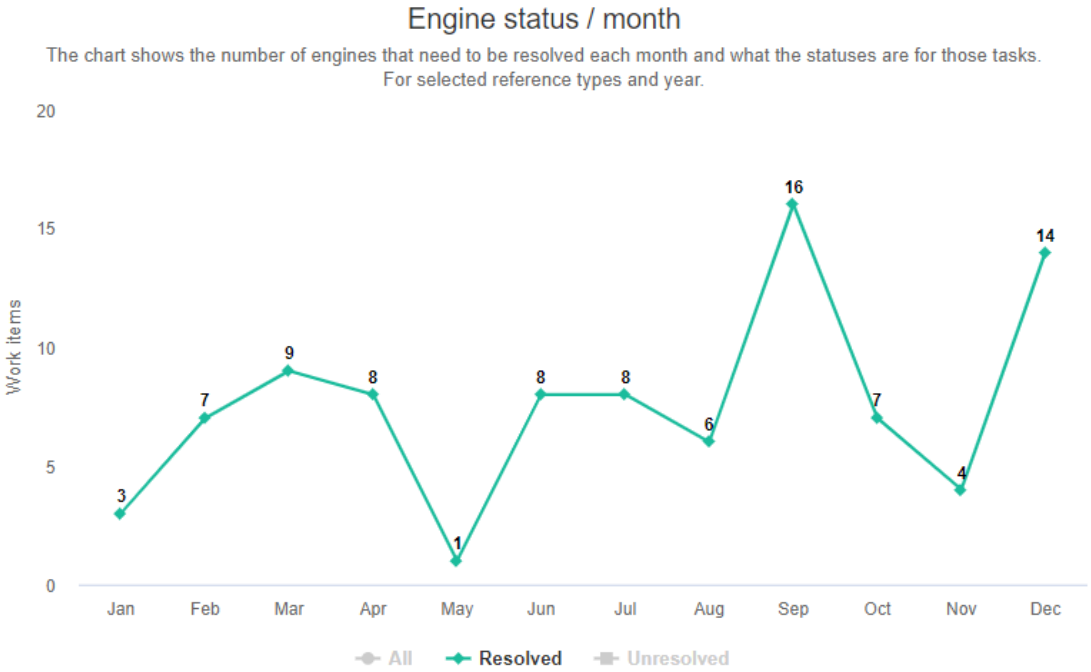


Figure 2. Graph showing the amount of completed W32 engines per month in the year 2022. (Wärtsilä Polarion 2023)

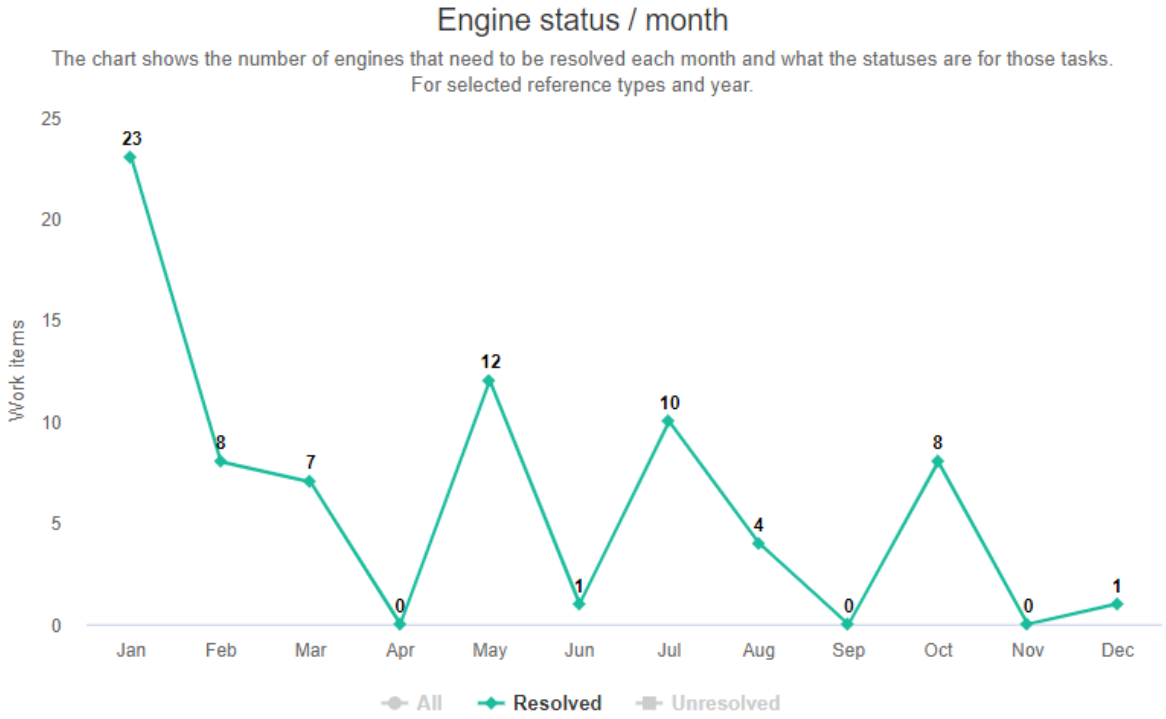


Figure 3. Graph showing the amount of completed W46F engines per month in the year 2022. (Wärtsilä Polarion 2023)

2.2 Visual Basic for Applications

Visual Basic for Applications or VBA is a programming language implemented into the Office Suite by Microsoft. It is commonly used to develop custom applications, functions or automating tasks within the Office suite. Almost every thinkable action within an Office application can be programmed using VBA. The programming language itself is a very powerful tool when it comes to manipulating data, performing calculations, or automation. Users familiar with VBA programming can easily develop applications or create macros within applications related to the office suite regarding solutions for repetitive tasks, general QOL automation, or data analysis. (Microsoft Corporation, 2022)

2.3 Optimization

Optimization is a word with a very broad meaning depending on the area of usage. Generally, it is used to describe principles and methods of quantitative problem-solving across a multitude of subjects, such as economics, mathematics, physics and engineering, etc. (Wright, 2023)

However, in this thesis, the word optimization refers to minimizing the amount of active work spent on manually verifying duplicate error reports, as well as human errors regarding documentation and data handling of engines at Wärtsilä.

3 Research method

When choosing a method to conduct a study you are often left with a choice. In this case, it is between conducting a qualitative and quantitative study. However, in some cases, it is hard to decide whether a study should be conducted by only going down either route. In this case, it is good to know how each of the different studies is defined, what they imply, and what methods are utilized. (Blomkvist, 2015)

To distinguish between qualitative and quantitative methods there are several approaches. Generally, it is depending on which kind of knowledge a researcher is aiming for and how the knowledge will be achieved meaning by what means. Quantitative is a word referring to the properties of something such as size, amount, or extent. These properties are usually expressed in a numerical way. This means that said properties can be explained with a numerical value which is also called quantitative data. Qualitative is a word also referring to the properties of something but instead the nature or quality. qualitative data is a way to describe non-numerical data such as images or words which makes it easily distinguishable from quantitative data. In other words, assigning numerical data to something that is qualitative is not possible. So to summarize, quantitative data is used to

express numerical values of something like amount or size while qualitative data is used to express the nature or quality of something. Both types of data are needed to conduct studies but it is the type of data that is collected that determines whether it is qualitative or quantitative.

Alternatively, quantitative and qualitative methods can be distinguished by taking a look at how collected data is processed. Quantitative methods often consist of statistical methods to analyze collected data while qualitative methods consist of verbal or non-numerical ways of analysis. Finally, it all comes back to the type of data that is collected, not how it is processed. (Säfsten, 2020)

The research method in this thesis consists of both quantitative and qualitative elements. To elaborate, the following list of the different stages of the study summarizes the research method in a short but concise way.

- Interviews with colleagues in order to specify which functions are needed from a tool like this.
- Realization of the tool by creating a program using VBA programming.
- Validation by temporarily implementing the tool in the work process in a series of experiments.
- Data collection to test the hypothesis of increased effectiveness in the workflow.
- Analysis and evaluation to quantitatively determine the efficiency of the tool.

4 Results

In this part of my thesis, I will explain how the tool was developed and implemented. I will also present, discuss and analyze the results of the experiment conducted regarding the implementation of the program. There are more than part of the work process that could use optimization in some form, but for this study, I ended up focusing on only one. The checkpoint in question checks the BOM-structure for duplicates regarding connections between material numbers and spare part numbers.
I will present and explain how the checkpoint was optimized through the development of the tool in the form of a program. I will cover how the program was created and its' functionality in a more in-depth fashion. Furthermore, the checkpoints' functionality will be explained to contextualize the relevancy of developing this program.

Lastly, I will also present my own calculations regarding the most optimal usage of the program based on the data collected during the experiment and existing data regarding the number of engines processed per week/month, etc.

This part will also cover how the tool was implemented with ease-of-access regarding updates and everyday usage.

4.1 Analysis of hard- and soft-requirements

Before doing anything practical, I made a list of both hard - and soft requirements for the program. This is essential when creating something that will be used by multiple people to make sure the result is the most optimal and that all possible features are included.

To clearly state the difference between hard and soft requirements; hard requirements are must-have features that cannot be ignored or skipped as they will function as the fundamental base of the program on which you can build and develop it further. These hard requirements should and will be taken into consideration for each added feature for the respective part of the toolset.

Soft requirements are features that would add a lot of “QOL” or quality of life for the user. However, these requirements are not necessarily needed as they do not affect the initial function of the program. They might also be a challenge to implement or simply not implementable at all.

The list of requirements was compiled by interviewing coworkers and by taking away what I, in my personal experience, felt would be a good feature. The list will be split up into different parts to ensure specific features/requirements are listed under the relevant part of the toolset besides a general list of requirements for the program as a whole.

4.2 Duplicate Spare Part Numbers/Material numbers

The first checkpoint when processing the BOM data of an engine is called “Duplicate SPN/Matno’s”. Here the data is checked for duplicate spare part numbers or material numbers. This means that there might be e.g., two spare part numbers connected to one material number or vice versa because of updates to either one. When launching this checkpoint the user is presented with a report that lists all spare part and material numbers that are potential duplicates. (See appendix 2)

Usually, this list is pretty long as all the old spare part numbers or material numbers that have not been removed after getting updated to a new one are still displayed with their updated counterpart under the same SPN. This results in the worker having to manually go through the whole list and check which spare part number or material number that is the correct one since the latest update.

When conducting the manual check of these duplicate SPN:s or material numbers the worker is looking for replacement chains. A replacement chain is a way of showing changes made to a material number meaning the latest version will always be found at the end of a “chain” consisting of all the versions of said material number, works kind of like a family tree and they are found in various databases in the form of a data structure. When an SPN or material number is updated it is added to a replacement chain so that the latest version easily can be spotted when viewing the replacement chain in various databases. However, the replacement chains cannot be viewed in WSHS and there are cases where a new number has not been added to the replacement chain even though it is updated and this usually requires further investigation. There are also cases where two parts may have the same SPN but in these cases, it is not required to dig through replacement chains.

For the duplicate error report, the idea of automatic cross-referencing became the base of the program. This became a possibility as you can export the error report to excel format. (See appendix 3)

In the picture, the data is displayed in an exported form in an Excel spreadsheet. This data will later be copied into another spreadsheet where the automatic cross referencing will occur when running the program.

4.3 Building the program and functionality

To build the base for automatic cross-checking, a list of most material numbers and their replacements was extracted from Teamcenter. It would be way too time-consuming to manually add each of the numbers. This is the list the tool will reference when checking the exported report from WSHS. (See appendix 4)

The picture only displays a small part of the list but it consists of over 5000 rows of material numbers and their replacements. The replacement material number is always listed in the right column and the old one in the left.

After this was done the actual creation of the program began by first creating an easy-to-understand layout on another sheet within the same workbook. (See appendix 5)

In the picture, the input area is found to the left. The data we want to paste here is found in the “Shortsection” and “Edition” columns when exporting the error report to excel.

To the right we find the output area. Here the old material number, the latest replacement material number, and the respective SPN will be listed when the program has run.

Instructions on how to use the program are also included to the right of the layout but not visible in the picture. (See appendix 6)

4.4 Programming in VBA

The programming part, also probably the toughest challenge of this thesis is where we bring the program to life.

When starting out, I had to go back to my list of hard- and soft requirements. As mentioned before, this will be the base of all functionality and it has to be reflected in the code. I chose to build the program within excel using VBA programming as this would be the easiest. This is because the export options regarding the error reports are limited and excel seemed to be the alternative that made the most sense.

4.4.1 Automatic cross-checking

The main function of this program is the automatic cross-checking. The user will input exported report error data whereof it is cross-checked against the list of old material numbers and their replacements that already was added on another sheet within the same spreadsheet.

I will refer to the “main sheet” and “data sheet”. (See appendix 5.) The main sheet is the sheet where you paste the exported data, run the program and see the output. The data sheet is the 5000+ row long list of material numbers with their replacements. Other references I will use are “Input column(s)” and “Output column(s)”. Input column(s) are related to the input area and Output column(s) to the output area.

Now, the input data usually contains multiple material numbers for one spare part number when it is pasted into the input area. We need a way to differentiate the old material numbers and the latest material numbers, or as I will refer to them: “latest replacements”.

This problem was solved by using a lookup function for both the old and latest material numbers.

Code example 1. Visual Basic Programming, VLookup function

```
Dim Row As Long
Dim Clm As Long
Table1 = Sheet3.Range("B2:B100")
Table2 = Sheet2.Range("B2:C10000")
Row = Sheet3.Range("E2").Row
Clm = Sheet3.Range("E2").Column
For Each i In Table1
 Sheet3.Cells(Row, Clm).FormulaR1C1 = Application.WorksheetFunction.VLookup(i, Table2, 2,
False)
 Row = Row + 1
Next i
```

The first function loops through the first 100 rows of the input B column or material numbers of the input data on the main sheet and checks each value against both the "Materialnumber" and "Replaced By" columns on the data sheet in the workbook. The data sheet is the long aforementioned exported list of material numbers and their replacements. If a value (material number) that is found in the input data is also found "Replaced By" column on the data sheet it will be listed in the output E column (Latest Replacement) on the main sheets' output area.

Code example 2. Visual Basic Programming, VLookup function

```
Dim Row2 As Long
Dim Clm2 As Long
Table1 = Sheet3.Range("B2:B100")
Table2 = Sheet2.Range("B2:C10000")
Row2 = Sheet3.Range("D2").Row
Clm2 = Sheet3.Range("D2").Column
For Each r In Table1
 Sheet3.Cells(Row2, Clm2).FormulaR1C1 = Application.WorksheetFunction.VLookup(r, Table2, 1,
False)
 Row2 = Row2 + 1
Next r
```

The second function works the same way, but instead, it will return all matching values found in the "Materialnumber" column on the data sheet in the output D column (Old materialnumber) on the main sheets' output area.

Now both the old material number and its' replacement will be correctly listed next to each other in the output area on the main sheet.

4.4.2 Failsafe for the automatic cross-checking

There is a chance that the exported list does not contain every single material number and its' replacement. Also, material numbers are updated every now and then meaning the new ones won't be included in the list initially.

However, to ensure that the user is alarmed if this would be the case I implemented a function that compares the output columns D (Old materialnumber) and E (Latest Replacement) against input column B (Material number). This code will run after the initial automatic cross-check.

Code example 3. Visual Basic Programming, Column comparison

```
Dim ws As Worksheet
   Dim columnB As Range, columnD As Range, columnE As Range
   Dim cellB As Range, cellD As Range, cellE As Range
   Dim lastRow As Long
   Dim match As Boolean

   Set ws = ThisWorkbook.Sheets("Duplicates")
   Set columnB = ws.Range("B2:B100")
   Set columnD = ws.Range("D2:D100")
   Set columnE = ws.Range("E2:E100")
   lastRow = ws.Cells(ws.Rows.Count, "D").End(xlUp).Row

   For Each cellB In columnB
      match = False
      For Each cellD In columnD
         If cellB.value = cellD.value Then
            match = True
            Exit For
         End If
      Next cellD
      For Each cellE In columnE
         If cellB.value = cellE.value Then
            match = True
            Exit For
         End If
      Next cellE
      If match = False Then
         ws.Cells(lastRow + 1, 4) = cellB.value
         lastRow = lastRow + 1
      End If
   Next cellB
```

If a value in input column B is not found in either output columns D or E it will be listed on the first empty row in output column D after the cross-check has run and all the found matches already are listed. This makes sure that the user will immediately notice if a material number is found in neither column on the data sheet and prompt a manual check.

4.4.3 Spare part number & material number connections

As I have mentioned earlier, each spare part number is connected to one or more material numbers. When the error report data is pasted into the input area, the user will immediately notice that at least two or more material numbers are listed next to the same spare part number. Once the data has been processed by the program, the user should be able to easily differentiate between which SPN is connected to which material numbers.

To make this possible, I added a function that populates the first 100 rows with a formula that checks for a match between the output D column and input B column.

Code example 4. Visual Basic Programming, Formula population

```
Dim x As Long
  For x = 2 To 100
    Cells(x, "F").Formula = "=IFERROR(INDEX(A" & x & ":A100, MATCH(D" & x & ", B" & x & ":B100,
0)), """")"
    If Cells(x, "F").value = "" Then Cells(x, "F").Clear
  Next x
```

This occurs after the automatic cross-check and the failsafe have run. (See appendix 7.) If a match is found, the corresponding value in the input A column on the same row is returned in output column F meaning the SPN that is connected to the material number found in output column D. I chose to use output column D for this function instead of E as there might be new material numbers which will not return a match. However, by using the values in output column D for this there will most likely be a match every time as these

values are the old material numbers. This function will also clear all cells with no match to make sure there aren't a bunch of cells with an error code in output column F.

4.4.4 Color coding & error handling

When it comes to error handling, ease of use and alerting a user of something then color coding is probably the smoothest way.

There might be cases where a replacement material number is found on the data sheet, but not in the input data, in other words, there is a "replacement for the latest replacement". To elaborate, there are some material numbers that are yet to be added to a replacement chain and therefore won't show up in the error report even though they replace something in it. These material numbers exist in the long, exported list on the data sheet meaning the automatic cross-checking will find them and list them as a replacement to a material number that already has been listed as a replacement for another one.

To alert the user if this is the case, I created a function that loops through all values in output column E (latest replacements) and compares them to the values found in input column B. If there are values(material numbers) in output column E that are not found in input column B then these are highlighted in green. To simplify, if a material number is highlighted in green it means that it was not found in the input data but in the list of material numbers and their replacements. (See appendix 7.) If this is the case, the user should add the replacement chain manually to the list on the datasheet.

Code example 5. Visual Basic Programming, Green Color-coding function

```
Dim rng1 As Range, rng2 As Range, cell As Range
Set rng1 = Range("E2:E100")
Set rng2 = Range("B2:B100")

For Each cell In rng1
   If cell.value <> "" And WorksheetFunction.CountIf(rng2, cell.value) = 0 Then
      cell.Interior.Color = RGB(0, 255, 0)
   End If
Next cell
```

There are also cases where a replacement material number is not found after running the program. To alert the user of this I added a function that loops through the output D and E columns checking if the current cell is in the D column, the current cell is not blank and if the cell next to it is blank (No replacement has been listed in the E column). If these conditions are met, then the cell in the output D column will be highlighted in red and the text “Not found” will be added to the cell on the same row in output E column. (See appendix 7.)

Code example 6. Visual Basic Programming, Red Color-coding function

```
Dim rng3 As Range, cellred As Range
Set rng3 = Range("D2:E100")

For Each cellred In rng3
   If cellred.Column = 4 And cellred.value <> "" And cellred.Offset(0, 1).value = "" Then
      cellred.Interior.Color = RGB(255, 0, 0)
      cellred.Offset(0, 1).value = "Not found"
   End If

Next cellred
```

Now the user will be alerted of oddities and errors in a easy and noticeable way.

4.4.5 QOL and additional features

The programs' output should be easy to decipher and it should also look organized. Meaning the output should consist of a concise list consisting of all the info the user should need.

To ensure that the list is organized and concise with no blank spaces or empty rows I added a function that simply deletes all blank cells in the output area on the main sheet and shifts the rows up.

Code example 7. Visual Basic Programming, Empty cell deletion and shifting

```
With Worksheets("Duplicates").Range("D2:F100")
   If WorksheetFunction.CountA(.Cells) > 0 Then .SpecialCells(xlCellTypeBlanks).Delete
Shift:=xlShiftUp
End With
```

I also added a QOL function that tells the user whether all the input data was processed correctly or not. This took the form of two counters, one that counts the amount of unique spare part numbers found in the input area and one for the output area. As the same spare part number might be listed more than once the counters use a formula that only counts unique values and ignores iterations of the same value in a column.

If both these counters show the same amount of SPN:s in the input and output areas then the user will know that everything was processed correctly. To make them work however, I had to add a couple of functions that convert the input A column and out F column to number format.

Code example 8. Visual Basic Programming, Formatting functions

```
With Worksheets("Duplicates").Range("A2:A100")
   .NumberFormat = "0"
   .value = .value
End With

With Worksheets("Duplicates").Range("F2:F100")
   .NumberFormat = "0"
   .value = .value
End With

With Worksheets("Duplicates").Range("D2:E100")
   .NumberFormat = "@"
   .value = .value
End With
```

4.4.6 Reset function

Finally, to reset the main sheet after processing an exported error report I added a couple of functions. One clears all the contents of both the input and output area while the other clears all formatting, color coding, etc.

Code example 9. Visual Basic Programming, Function to clear formatting and contents

```
Sub Reset()

   Worksheets("Duplicates").Range("A2:B100").ClearContents
   Worksheets("Duplicates").Range("D2:F100").ClearContents
Set dict = Nothing

   Dim rng As Range
   Set rng = Range("D2:E100")
   rng.Interior.Color = xlNone
   rng.Font.Bold = False
   rng.Font.Italic = False
   rng.Font.Underline = xlUnderlineStyleNone

MsgBox "Ready"
End Sub
```

4.4.7 The final program

Once I felt comfortable that all the functions would seamlessly work together I created a sub in VBA that contained all the functions as one program. I had previously tested each function one by one in a “test sub” to ensure they worked as intended. The final step was creating buttons that activate the program and the reset function.

For visualization of when the program has run and the data has been processed utilizing all the functions described, see appendix 7.

4.4.8 Updating and ease of access

When it comes to updating the program, the original plan was to make it easy. This is possible now due to the program running on VBA code with added comments in the code to make sure anyone taking a look at the code will understand what each function does. However, updating is not only about updating the code or functionality in the program. New material numbers are released every so often and replace old ones. If a new material number is found in the duplicate error report, it will be colored red in the output area. (See appendix 7).

When this is the case, the user has to check the replacement chain manually, but only once. The user can add the replacement chain to the data sheet columns accordingly (which material number replaces which). Once this is done the program will recognize the replacement chain and process the data properly.

Another thing that was crucial for the program is that it is accessible easily by anyone working with documentation and data handling of engines at Wärtsiläs’ Content Management department. To make this possible, the departments’ internal Microsoft Teams channels are used. You can open the workbook through the Microsoft Teams application and the program, all list or code updates are accessible to everyone as the workbook is stored online. This creates a seamless way of usage and updating.

4.5 Implementation of the program

To implement the program into the work process, I had to make sure that it works as intended but also that it is worth using. By this, I mean that surely there must be cases when the duplicate error report is very short and the difference in time between using the program and doing a manual check of each duplicate is very little. Also, the program should work with different models of engines.

To research when the program is actually worth using and if it works universally for different engine models me and a colleague at Wärtsilä compiled a list of random engines. Once our list was done which ended up being 20 different engines of varying models, the experiment started. The models ended up being W20, W32 and W46F.

4.5.1 Testing the program

To start off, both I and my colleague opened the duplicate error report of the same engine. Once that was done, one of us went through the list manually while the other exported the list and used the program to go through it. We did this for all 20 engines during the course of a full workday. Every time we started processing the error report either manually or using the program, we also started a timer.

When either one of us was finished the timer was stopped, but using the lap function we could keep it going and later add the difference between the times taken to complete the error report manually or using the program.

The results of this experiment will be presented and discussed later in the thesis.

4.6 The implementation experiment result

Table 1. Collected data during the experiment.

Engine #	Start time (hh:mm)	End time (Manual) (hh:mm:ss)	End time (Tool) (hh:mm:ss)	Time spent (Manual) (hh:mm:ss)	Time Spent (Tool) (hh:mm:ss)	Amount of rows in report
1	08:45	08:47:04	08:45:29	00:02:04	00:00:29	14
2	08:52	08:52:46	08:52:25	00:00:46	00:00:25	6
3	08:56	08:58:24	08:56:29	00:02:24	00:00:29	19
4	09:04	09:04:55	09:04:28	00:00:55	00:00:28	8
5	09:35	09:40:27	09:35:21	00:05:27	00:00:21	30
6	10:15	10:16:11	10:15:17	00:01:11	00:00:17	10
7	11:01	11:03:27	11:01:20	00:02:27	00:00:20	14
8	13:06	13:07:43	13:06:18	00:01:43	00:00:18	13
9	13:14	13:17:29	13:14:19	00:03:29	00:00:19	17
10	13:31	13:36:38	13:31:46	00:05:38	00:00:46	46
11	13:53	13:59:04	13:53:41	00:06:04	00:00:41	83
12	14:20	14:21:13	14:20:26	00:01:13	00:00:26	21
13	14:36	14:38:32	14:36:33	00:02:32	00:00:33	19
14	14:44	14:45:46	14:44:26	00:01:46	00:00:26	11
15	14:49	14:51:10	14:49:24	00:02:10	00:00:24	30
16	14:53	14:55:18	14:53:27	00:02:18	00:00:27	24
17	14:59	15:01:26	14:59:25	00:02:26	00:00:25	20
18	15:08	15:09:47	15:08:22	00:01:47	00:00:22	15
19	15:12	15:14:14	15:12:26	00:02:14	00:00:26	28
		Average time spent per engine		**00:02:33**	**00:00:26**	23

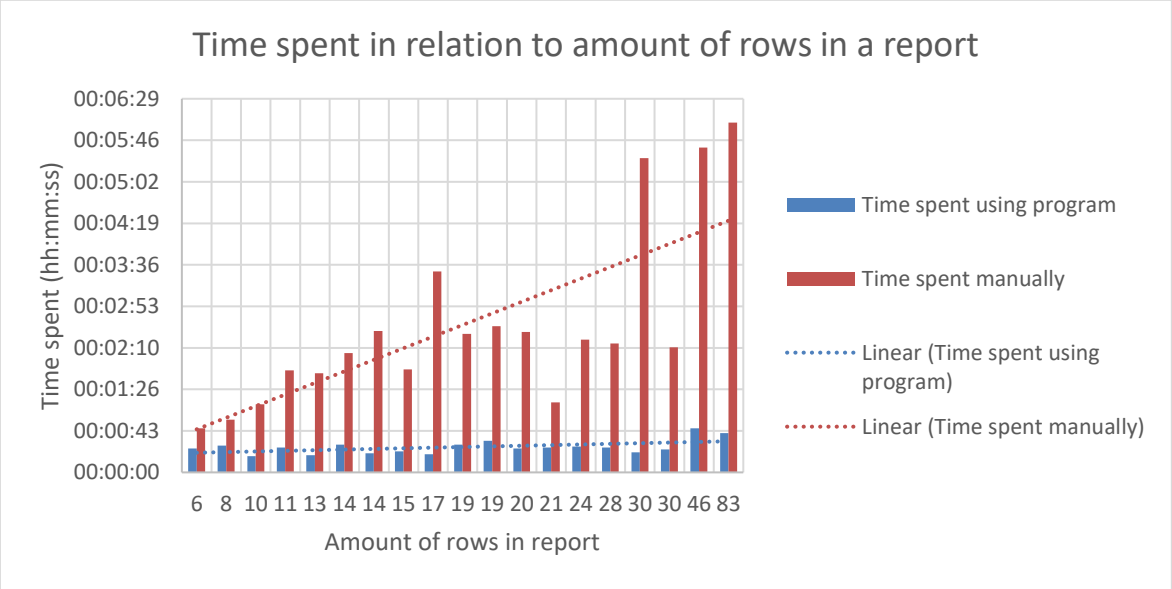


Figure 4. Graph showing time spent using the program and manually in relation to the amount of rows in a duplicate report

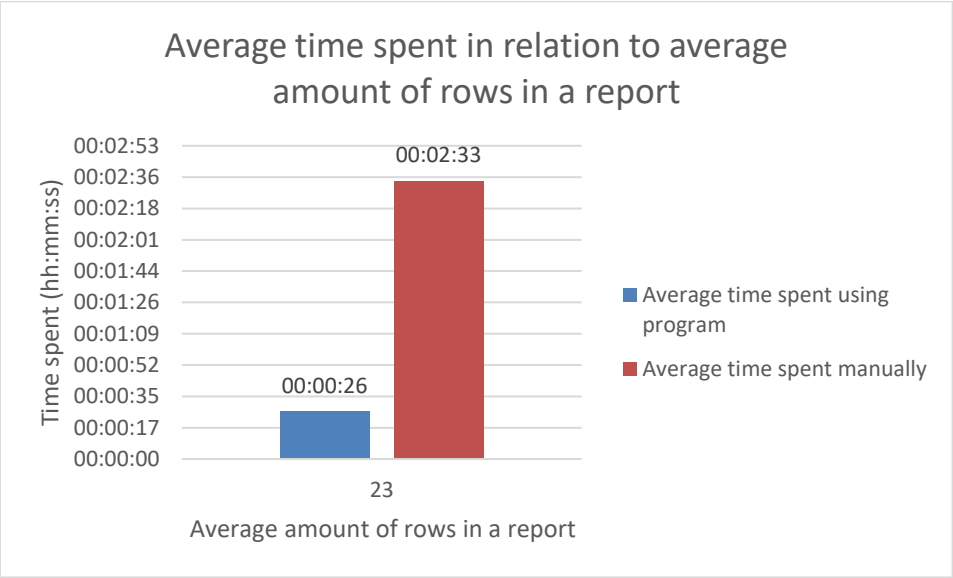


Figure 5. Graph showing average time spent using the program and manually in relation to the average amount of rows in a duplicate report.

4.7 General results and calculations

In this chapter, the hypothetical results of using the program in 2022 will be presented. The results are based upon calculated averages and data from Wärtsilä's Polarion.

Table 2. Table containing data presented in figures 1, 2 & 3 with extra calculations.

Month	Amount of compl. W20	Amount of compl. W32	Amount of compl. W46F	Total across all engine types
January	20	3	23	**46**
February	11	7	8	**26**
March	0	9	7	**16**
April	0	8	0	**8**
May	7	1	12	**20**
June	12	8	1	**21**
July	32	8	10	**50**
August	25	6	4	**35**
September	14	16	0	**30**
Oktober	6	7	8	**21**
November	10	4	0	**14**
December	12	14	1	**27**
Total amount completed	**149**	**91**	**74**	**314**
Average amount completed per	**12**	**8**	**6**	**26**

Table 3. Calculations regarding time spent on duplicate reports manually in 2022 utilizing data from table 2 and calculated averages from table 1.

Engine type	W20	W32	W46F	Total
Total time spent on dupl. Report (hh:mm:ss)	06:20:52	03:52:37	03:09:09	13:22:38
Average time spent per month (hh:mm:ss)	00:31:44	00:19:23	00:15:46	01:06:53
Average time spent per week (hh:mm:ss)	00:07:19	00:04:28	00:03:38	00:15:26

Table 4. Calculations regarding time spent on duplicate reports hypothetically using the program in 2022 utilizing data from table 2 and calculated averages from table 1.

Engine type	W20	W32	W46F	Total
Total time spent using program (hh:mm:ss)	01:05:37	00:40:04	00:32:35	02:18:16
Average time spent using program per month (hh:mm:ss)	00:05:28	00:03:20	00:02:43	00:11:31
Average time spent using program per week (hh:mm:ss)	00:01:16	00:00:46	00:00:38	00:02:40

Table 5. Calculations regarding time saved hypothetically using the program in 2022 utilizing data from table 2 and calculated averages from table 1.

Engine type	W20	W32	W46F	Total
Total time saved using program (hh:mm:ss)	05:15:15	03:12:32	02:36:34	11:04:21
Time saved using program per month (hh:mm:ss)	00:26:16	00:16:03	00:13:03	00:55:22
Time saved using program per week (hh:mm:ss)	00:06:04	00:03:42	00:03:01	00:12:47

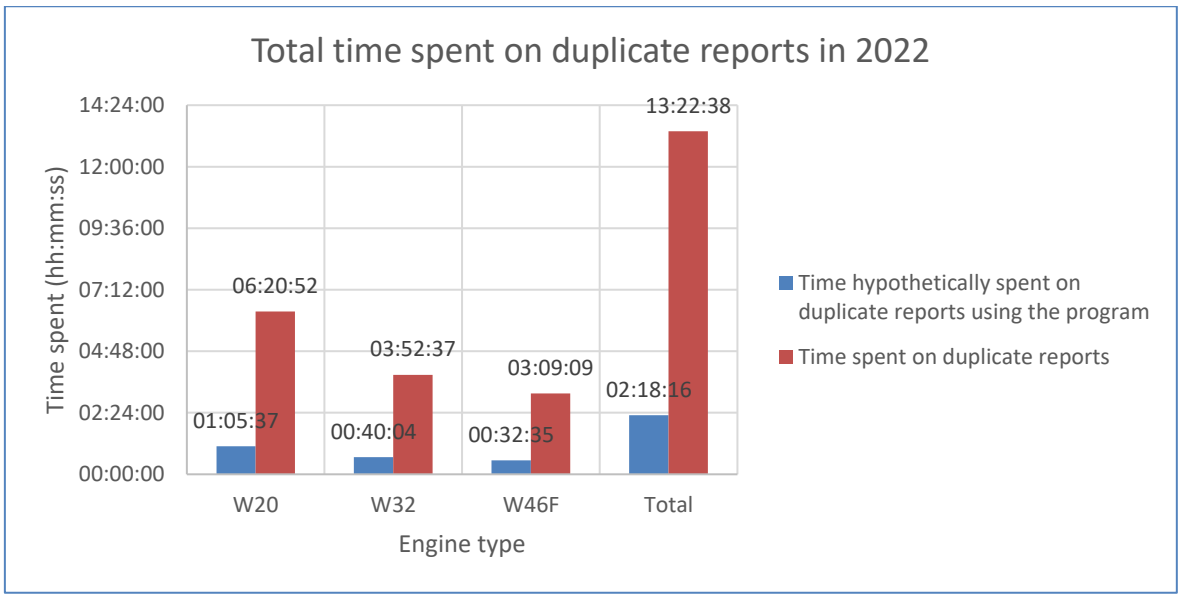


Figure 6. Graph showing the difference in time spent on duplicate reports manually and hypothetically using the program in 2022 utilizing data calculations from tables 3, and 4 and calculated averages from table 1.

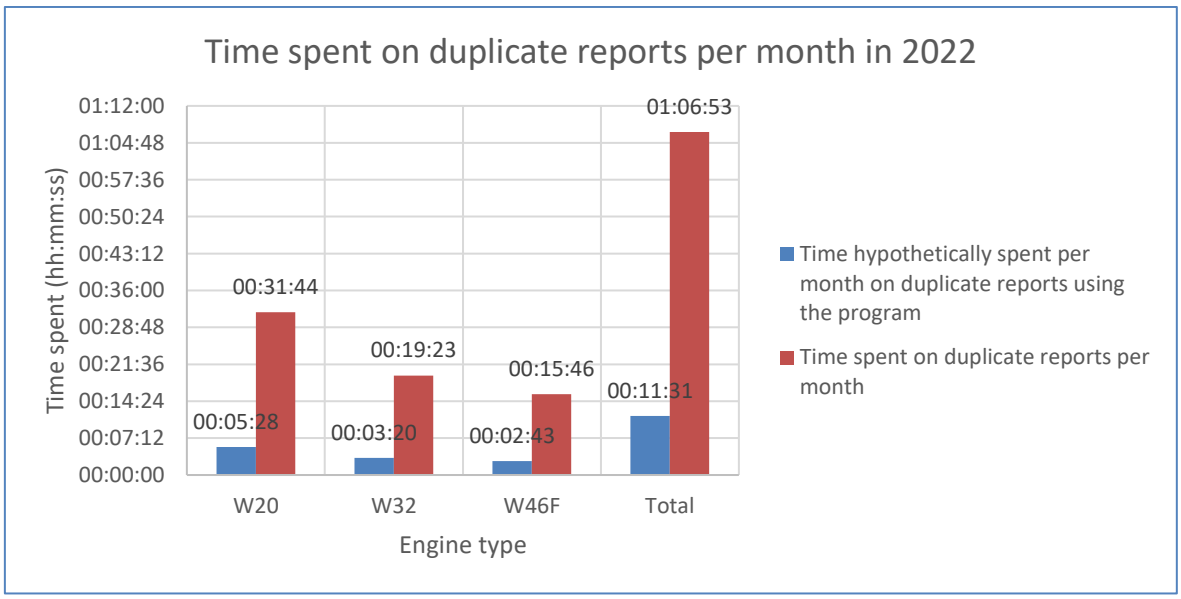


Figure 7. Graph showing the difference in time spent per month on duplicate reports manually and hypothetically using the program in 2022 utilizing calculations from tables 3, 4 and calculated averages from table 1.

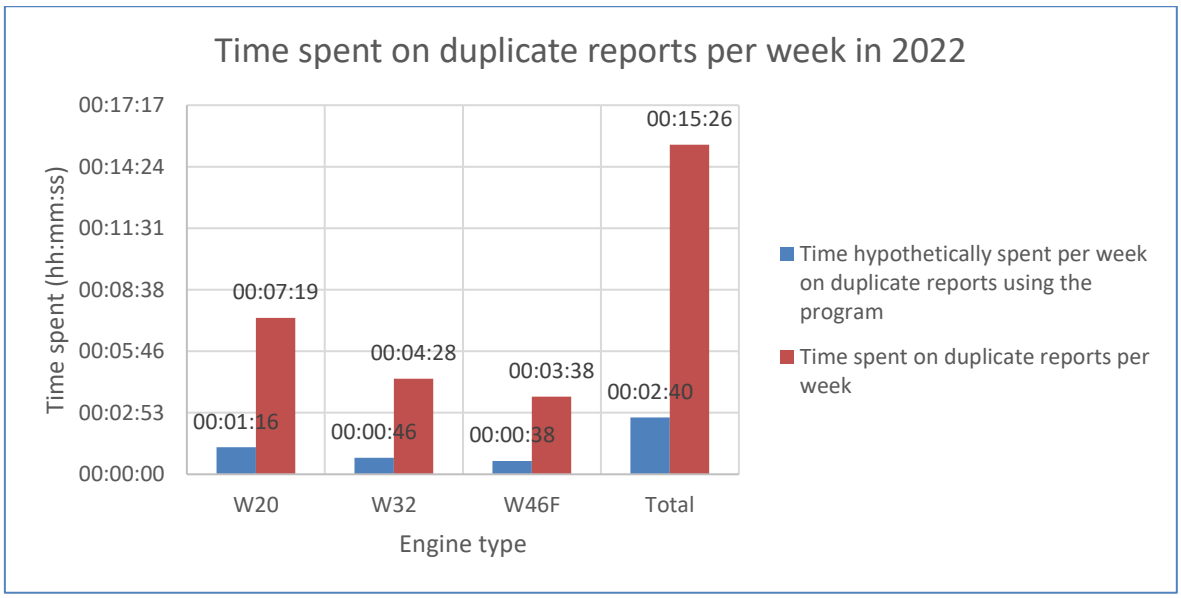


Figure 8. Graph showing difference in time spent per week on duplicate reports manually and hypothetically using the program in 2022 utilizing calculations from tables 3, 4 and calculated averages from table 1.

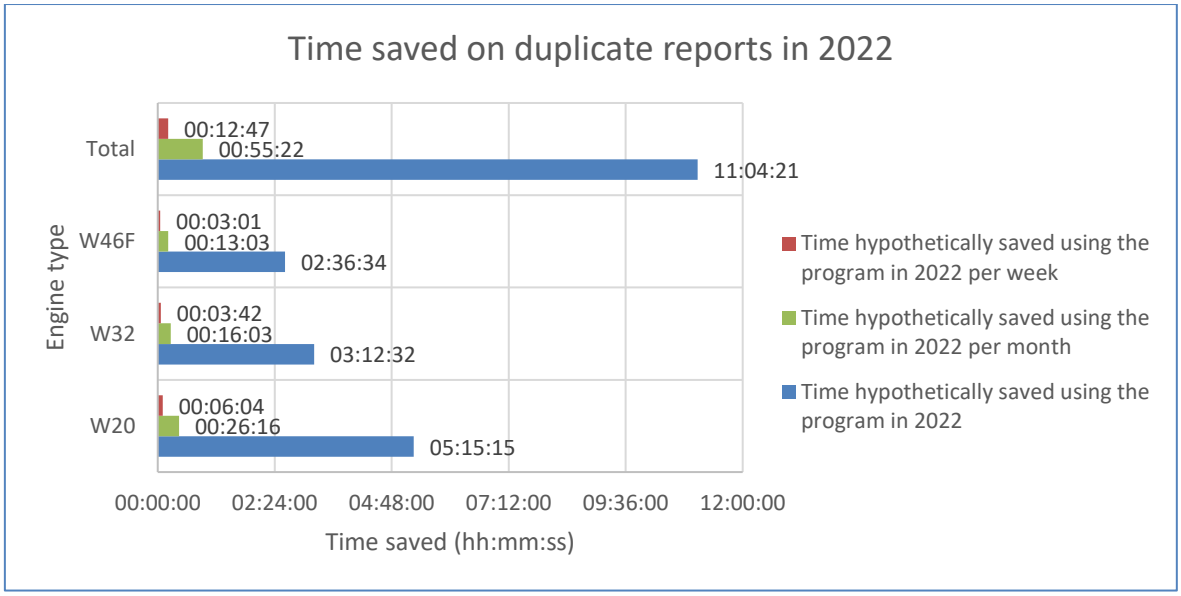


Figure 9. Graph showing time hypothetically saved using the program on duplicate reports in 2022 utilizing calculations from table 5.

5 Discussion

I didn't exactly know what I was expecting beforehand except that the program would probably be faster, but not by how much. The results are probably a little bit skewed due to using a timer that manually had to be started and stopped. However, it only comes down to seconds regarding the timer but as the difference between using the program and doing the work manually only differs in minutes it is worth mentioning.

Another thing worth mentioning is that the program will always take the same amount of time to run as the number of rows the program goes through is set to 100. This means no matter the length of the duplicate error reports the program will always check 100 rows. That means that the only factor affecting the time taken using the program is how fast the user can export the error report, paste the data and execute the program by clicking the button. If one were to disregard the time taken to export and execute then the amount of time taken using the program would be constant.

As one can clearly see in the figures, using the program is faster in every case. But the program also passively deals with another issue aside from time spent on manually checking the rows; when manually checking the rows the room for human error is a lot bigger as you have to remember which row you are on and make sure you are not skipping any rows by mistake. Using the program eliminates this problem.

When calculating the time hypothetically spent or saved in 2022 I only took into consideration the calculated average amount of time spent per engine from the implementation experiment and the number of engines processed for respective engine type and in total. There are however other factors that will affect the number of engines that are processed per week. Month or year. These factors include number of engines that are ordered and other errors in the work process (the duplicate report is only one of many steps). These factors are not something that can be changed as it is impossible to foresee what the future holds.

Overall the result is positive and very much shows that optimization is an effective method to cut active work time spent on tasks that could be automated.

6 Conclusion

To summarize, I managed to optimize a part of the documentation work process at Wärtsilä by planning, creating, and implementing a program that eliminated the risk of human error and unnecessary spending of active worktime on a task that can vary greatly in longevity.

Based on the results, I can say that this thesis is proof of how optimizing select stages in a longer work process can increase the effectiveness and quality of the work. The results in this thesis are only the results of optimizing one step in the work process, imagine if every single step you could think of was optimized. The result would be even better then.

However, as always there is room for improvement regarding the program and the way it was planned, created, and implemented. One example is creating a function that only checks the amount rows that are pasted into the input instead of 100 every time. Another example would be implementing the program into WSHS as a base function.

The previously mentioned soft requirements listed for the program didn’t make it into the final version as they proved to be way too challenging at this time. However, maybe they will be included in the future.

7 References

Blomkvist, P. &. (2015). In *Method for engineering students Degree Projects Using the 4-phase Model* (p. 61). Lund: Studentliteratur AB.

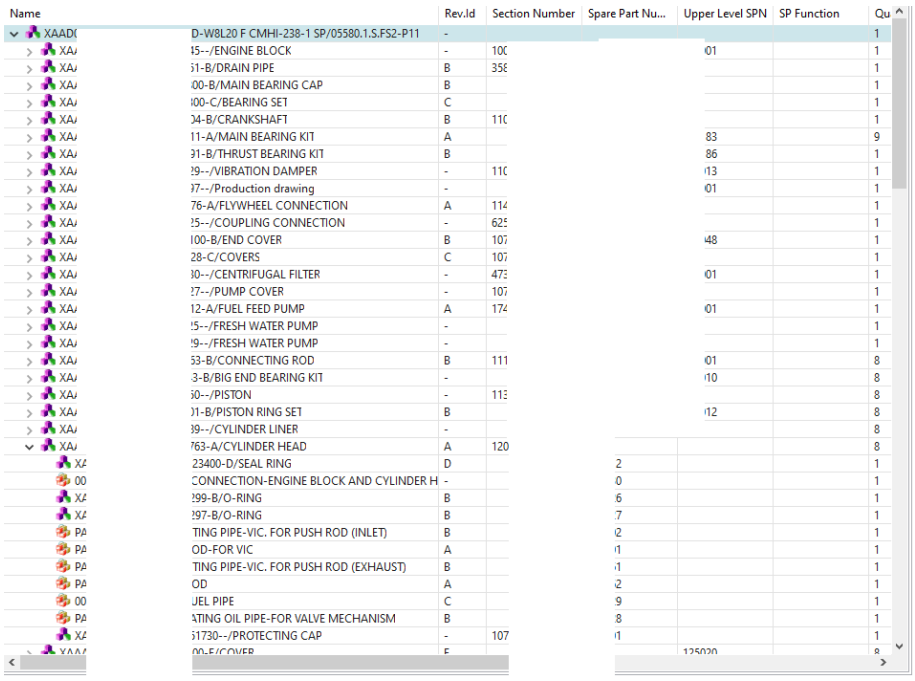
Microsoft Corporation. (2022, August 27). *VBA programming in Office*. Retrieved from Microsoft Learn: https://learn.microsoft.com/en-us/office/vba/api/overview/#vba-programming-in-office

Säfsten, K. &. (2020). In *Research Methodology For Engineers and Other Problem-Solvers* (pp. 37-39). Lund: Studentliteratur AB.

Wright, S. J. (2023, February 11). *optimization*. Retrieved from Encyclopedia Britannica: https://www.britannica.com/science/optimization

Wärtsilä Finland Oy. (2023, March 6). *About: This is Wärtsilä*. Retrieved from https://www.wartsila.com/about

8 Appendices



Name		Rev.Id	Section Number	Spare Part Nu...	Upper Level SPN	SP Function	Qu
XAAD(	D-W8L20 F CMHI-238-1 SP/05580.1.S.FS2-P11	-					1
XA/	15--/ENGINE BLOCK	-	10(		01		1
XA/	51-B/DRAIN PIPE	B	35				1
XA/	00-B/MAIN BEARING CAP	B					1
XA/	00-C/BEARING SET	C					1
XA/	04-B/CRANKSHAFT	B	11(				1
XA/	11-A/MAIN BEARING KIT	A			83		9
XA/	91-B/THRUST BEARING KIT	B			86		1
XA/	19--/VIBRATION DAMPER	-	11(		13		1
XA/	97--/Production drawing	-			01		1
XA/	76-A/FLYWHEEL CONNECTION	A	114				1
XA/	15--/COUPLING CONNECTION	-	625				1
XA/	100-B/END COVER	B	107		48		1
XA/	28-C/COVERS	C	107				1
XA/	10--/CENTRIFUGAL FILTER	-	473		01		1
XA/	17--/PUMP COVER	-	107				1
XA/	12-A/FUEL FEED PUMP	A	174		01		1
XA/	15--/FRESH WATER PUMP	-					1
XA/	19--/FRESH WATER PUMP	-					1
XA/	53-B/CONNECTING ROD	B	111		01		8
XA/	3-B/BIG END BEARING KIT	-			10		8
XA/	10--/PISTON	-	113				8
XA/	01-B/PISTON RING SET	B			12		8
XA/	19--/CYLINDER LINER	-					8
XA/	763-A/CYLINDER HEAD	A	120				8
XA	23400-D/SEAL RING	D		2			1
00	CONNECTION-ENGINE BLOCK AND CYLINDER H	-		0			1
XA	99-B/O-RING	B		6			1
XA	97-B/O-RING	B		7			1
PA	TING PIPE-VIC. FOR PUSH ROD (INLET)	B		2			1
PA	OD-FOR VIC	A		1			1
PA	TING PIPE-VIC. FOR PUSH ROD (EXHAUST)	B		1			1
PA	OD	A		2			1
00	JEL PIPE	C		9			1
PA	ATING OIL PIPE-FOR VALVE MECHANISM	B		8			1
XA	51730--/PROTECTING CAP	-	107	1			1
XAA	00-E/COVER	E			125020		8

Appendix 1: Typical engine BOM-data structure

WSHS: Duplicate SPN/Matno's in BoM

Note! This search does not support SHARED part numbers or part numbers at engine level!

EngineNumber XAAC409040

Partnumber	Materialnumber	Fixed section	Module SPL	Module
145055	PAA	145-	165	PAB
145055	0033	145-	165	PAB
145060	0033	145-	165	PAB
145060	PAA	145-	165	PAB
156074	0042	156-	156	PAA
156074	PAA	156-	156	PAB
165050	003	165-	16	PAB
165050	003	165-	16	PAB
165190	PAA	165-	16	PAB
165190	6363	165-	35	PAB
167043	003	167-	35	PAA
167043	003	167-	12	PAA
182021	005	182-	18	PAB
182021	PAA	182-	18	PAB
182066	003	182-	18	PAB
182066	003	182-	18	PAB
352011	PAA	352-	35	PAA
352011	PAA	352-	10	PAB

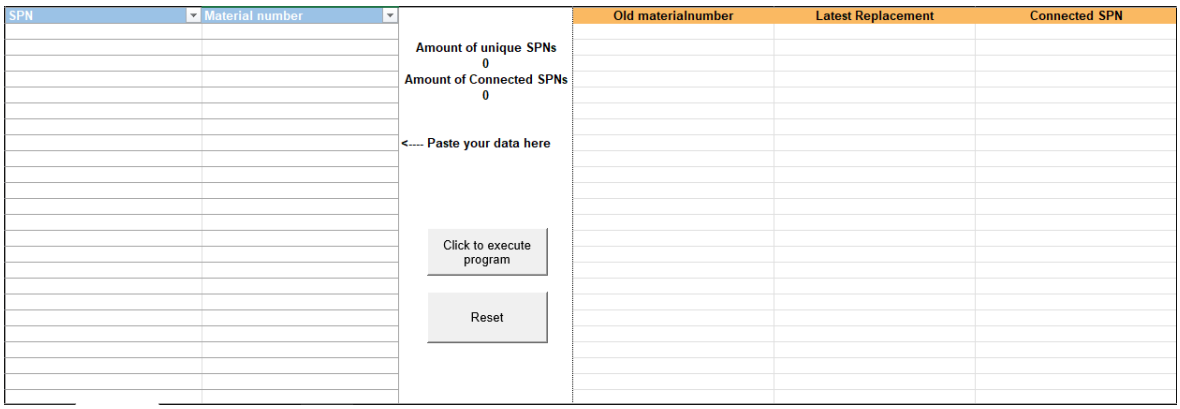
Appendix 2: Example of duplicate error report in WSHS.

XAAC409040		
ShortSection	Edition	Desc
145	PAA	145-0
145	0033	145-0
145	0033	145-0
145	PAA	145-0
156	0042	156-0
156	PAA	156-0
165	0033	165-0
165	0033	165-0
165	PAA	165-0
165	6363	165-0
167	0033	167-0
167	0033	167-0
182	0052	182-0
182	PAA	182-0
182	0033	182-0
182	0033	182-0
352	PAA	352-0
352	PAA	352-0

Appendix 3: Example of exported duplicate error report in Excel format.

5258	XA	PA
5259	XA	PA
5260	XA	PA
5261	XA	PA
5262	PA	PA
5263	PA	PA
5264	00	PA
5265	00	PA
5266	00	PA
5267	209	PA
5268	00	PA

Appendix 4: Screenshot of a part of the list containing old material numbers and their replacements.



Appendix 5: Screenshot of the programs layout in Excel

CLICK "OPEN IN DEKSTOP APP" IF YOU ARE VIEWING THIS THROUGH THE TEAMS TAB, OTHERWISE THE PROGRAM WONT WORK

- Export the duplicate report to Excel in WSHS

- Copy & paste the contents of the first two columns (Short section, edition) excluding the headers into the A and B columns on this page.

- Click the "Execute program" button

The materialnumbers in the "Old materialnumber" column are the redudant ones

The materialnumbers in the "Latest Replacement" column are the latest ones in the replacement chain

The Spare part numbers in the "Connected SPN" tab are the Spare part numbers connected to the material numbers in the afformentioned column D and E

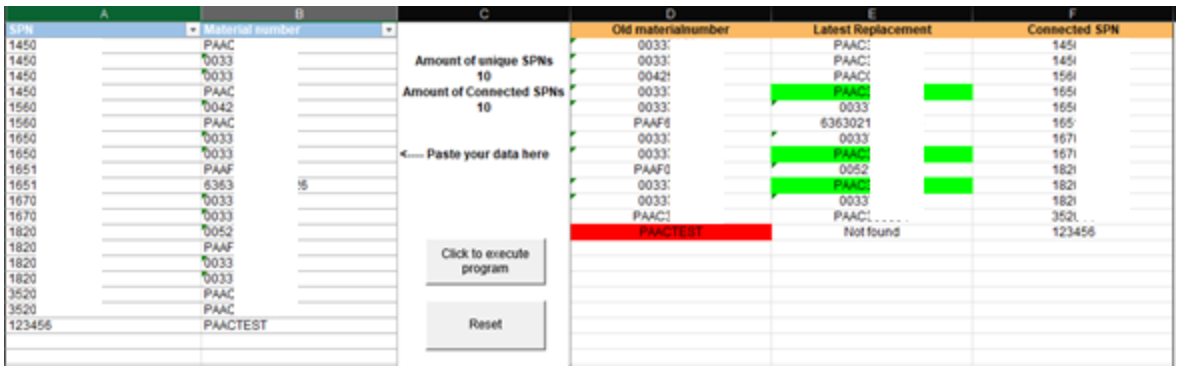
The two counters; "Amount of unique SPNs" and "Amount of Connected SPNs" makes sure the same amount of unique SPNs in the input are outputted aswell aka the data was processed correctly. If these numbers differ, check manually which of the SPNs in the input that are missing in the output

Check the color-legend above if any cells are highlighted in either colors

If you have to check something manually, remeber to add the old and new materialnumber to the list on the second sheet called "List of replacements" accordingly. After doing this, run the program with the updated list.

- To reset, click the "Reset" button

Appendix 6: Screenshot of provided instructions for the program.



A	B	C	D	E	F
SPN	Material number		Old materialnumber	Latest Replacement	Connected SPN
1450	PAAC		0033	PAAC	145
1450	0033	Amount of unique SPNs	0033	PAAC	145
1450	0033	10	0042	PAAC	156
1450	PAAC	Amount of Connected SPNs	0033	PAAC	165
1560	0042	10	0033	0033	165
1560	PAAC		PAAF6	6363021	165
1650	0033		0033	0033	167
1650	0033	<---- Paste your data here	0033	PAAC	167
1651	PAAF		PAAF0	0052	182
1651	6363		0033	PAAC	182
1670	0033		0033	0033	182
1670	0033		PAAC	PAAC	352
1820	0052	Click to execute program	PAACTEST	Not found	123456
1820	PAAF				
1820	0033				
1820	0033	Reset			
3520	PAAC				
3520	PAAC				
123456	PAACTEST				

Appendix 7: Screenshot of when the program has ran and the data has been processed.