NOVIA
UNIVERSITY OF APPLIED SCIENCES

How to converge a scattered IT system for improved Innovation Management

Oskar Söderlund

Bachelor's thesis in Information Technology

Vaasa 2014

BACHELOR'S THESIS

Author:	Oskar Söderlund
Degree programme:	Information Technology, Vaasa
Supervisor:	Kaj Wikman

Title: *How to converge a scattered IT system for improved Innovation Management*

Date: 20.03.2014 Number of pages: 30 Appendices: 0

Abstract

This task has been commissioned by Wärtsilä Finland Oy, Service, Innovation Management. The main weaknesses of the ways in which the idea coordinators work with an idea after it has been submitted in MyDea (Wärtsilä's Idea portal) have been pinpointed in this thesis. The thesis also includes studies of and suggestions for how the process can be faster and more convenient for the coordinators.

A prototype of a control panel has been developed that tries to solve some of the time-consuming issues and the human errors that the coordinators encounter daily in their work. The control panel is developed with VBA and VB.NET to achieve a great integration with the Microsoft Office Package that is frequently used in the office.

This thesis work has resulted in an application that can be used to speed up the work of the coordinator and can be used as a showcase for consultants or suppliers when future development or investments take place regarding the coordinator's way of working.

Language: English Key words: Innovation Management, VBA

EXAMENSARBETE

Författare:	Oskar Söderlund
Utbildningsprogram och ort:	Informationsteknik, Vasa
Handledare:	Kaj Wikman

Titel: *Hur sammanbinda ett splittrat IT system för förbättrat Innovation Management*

Datum: 20.03.2014 Sidantal: 30 Bilagor: 0

Abstrakt

Denna uppgift är given av Wärtsilä Finland Oy, Service, Innovation Management. Uppgiften går ut på att markera de huvudsakliga nackdelarna med det nuvarande sättet idékoordinatorn arbetar med en idé efter att den har publicerats i MyDea (Wärtsiläs idéportal). Examensarbetet innehåller också förslag och idéer hur processen kunde vara snabbare och mera passande för koordinatorn.

En prototyp av en kontrollpanel har också utvecklats som försöker lösa några av de tidskrävande problem och de mänskliga felen som koordinatorn träffar på under det dagliga arbetet. Kontrollpanelen har utvecklats med VBA och VB.NET för att uppnå en bra integration med Microsoft Office paketet som ofta används på kontoret.

Resultatet är en applikation som kan användas för att snabba upp koordinatorns arbete samt som ett uppvisningsexemplar för konsulter eller leverantörer när framtida utveckling eller investeringar görs gällande koordinatorns sätt att arbeta.

Språk: engelska Nyckelord: Innovation management, VBA

OPINNÄYTETYÖ

Tekijä:	Oskar Söderlund
Koulutusohjelma ja paikkakunta:	Tietotekniikka, Vaasa
Ohjaajat:	Kaj Wikman

Nimike: *Miten saada hajanainen IT-systeemi konvergoimaan, jotta Innovation Management kehittyisi*

Päivämäärä: 20.03.2014 Sivumäärä: 30 Liitteet: 0

Tiivistelmä

Tämän tehtävän on antanut Wärtsilä Finland Oy, Service, Innovation Management. Päämääränä on paikantaa olennaiset haitat ideakoordinaattorin nykyisessä työskentelytavassa, sen jälkeen kun idea on julkaistu MyDeassa (Wärtsilän ideaportaalissa). Tämä tehtävä sisältää myös ehdotuksia ja ideoita siitä, miten prosessi voisi olla nopeampi ja sopivampi koordinaattorille.

Kontrollipaneelin prototyyppiä kehitetään myös ja yritetään ratkaista osa niistä aikaavievistä ja inhimillisistä virheistä, jotka koordinaattori kohtaa jokapäiväisessä työssään. Kontrollipaneeliä kehitetään VBA:n ja VB.NET:in avulla, jotta saavuttaisiin hyvä integroituminen Microsoft Office paketin kanssa, jota usein käytetään konttorissa.

Tuloksena tulee olemaan sovellus, jota voidaan käyttää koordinaattorin työn nopeuttamisessa. Sitä voitaisiin myös käyttää näytekappaleena konsulteille ja tavaroiden toimittajille tulevaisuuden kehityksessä tai investoinneissa koskien koordinaattorin työskentelytapaa.

Kieli: Englanti Avainsanat: Innovation Management, VBA

Table of contents

Abbreviations

DOM	Document Object Model
IDM	Integrated Document Management
IE	Internet Explorer
GUI	Graphical User Interface
MS	Microsoft
RGB	Red Green Blue
SQL	Structured Query Language
URL	Uniform Resource Locator
VB.NET	Visual Basic .NET
VBA	Visual Basic for Applications

1 Introduction

1.1 Employer

Wärtsilä was founded 1834 in a little village called Wärtsilä. At first the company was a small sawmill. A few years later the sawmill was bought by Nils Ludvig Arppe. Then The Senate of Finland encouraged companies to build ironworks. Arppe invested in two blast furnaces and evolved Wärtsilä to Finland's biggest ironworks. In the mid 1920's the company was in financial trouble. The board appointed the diploma engineer Wilhelm Wahlforss as president. The plan was for Wahlforss to restructure the company. He managed to restructure the company with great success. Wärtsilä built the first galvanization plant in the Nordic countries, and the production of galvanized wire gave Wärtsilä a well deserved boost in profitability. Wahlforss also acquired several shipyards. With that move Wärtsilä became a multi-sector business. Wahlforss also anticipated the war building up in Europe and bought several engineering works. They were used for the production of various items needed in the time of crisis. After the war Finland was forced to pay war reparations to Russia. Wärtsilä became the biggest supplier of war reparations. At the end of the 1940's Wärtsilä was Finland's biggest industry and had almost 11 000 employees. Wärtsilä factories were producing licensed diesel engines but wanted to develop their own diesel engine. Wärtsilä's own built diesel engine was turned on for the first time in 1959 in Vaasa. After that Wärtsilä steered its line of business to producing diesel engines and building ships. Around 1970 more than half of Wärtsilä's business was diesel engines and ships.

Wärtsilä started to invest into diesel engine technology and experiments with inexpensive heavy fuel. The Vasa laboratory succeeded in running engines with heavy fuel and this was seen as a global milestone for the ship engine industry. With the purchase of Sulzer in 1997 Wärtsilä became the world leader in ship engines. Wärtsilä continued to expand through acquiring different companies related to ship building. With these moves Wärtsilä was able to not only supply the engine but also the whole engine room. Later Wärtsilä also began to design ships. Wärtsilä focused on product development, environmental applications and future energy technology. Having said that, Wärtsilä must always be innovative and future driven, like in the 1970's when they succeeded in running engines with heavy fuel. Wärtsilä is still today very innovative and a key player when it comes to energy efficient solutions. Environmental regulations and fuel costs make markets change

and Wärtsilä is very foresighted in this respect. Today more than half of Wärtsilä's power plants are driven by gas fuel. /17/

1.2 Innovation management within Wärtsilä

Just by looking at Wärtsilä's history it is easy to see that a company's business needs to evolve over time to stay ahead of its competition and bring the future in terms of energy production to the customers. Wärtsilä has about 19 000 inventors available all around the world. It is important that every idea can be discussed and investigated to the fullest potential. In order to do that the company must encourage its employees to open up and dare to present their ideas. An idea does not turn into an innovation before it brings added value to the company, that is, income or savings for the company. Even the smallest innovation can bring profits to the company.

But how can a large company bring all employees together from all around the world to share their ideas easily? Often the one who comes up with a great idea is not the one who is directly working in the area where the idea needs to be implemented. A service engineer, for example, might come up with something that has not occurred to the product designer.

In order to keep the employees creative and keen on discussing ideas there must be some sort of feedback to keep the interest at a high level. Ideas spawn more ideas. The company must create an environment that rewards or makes employees feel that their idea was appreciated and thought of. The interest of the creator of an idea dies out quickly if the creator feels that the idea was unnecessary and no one takes time to review the idea. Another problem with ideas is that someone might have a perfect idea in his head but thinks that it is too ridiculous to propose. Can that barrier be overrun by creating an Innovative environment? /13/

1.2.1 Idea Portal

In March 2011 Wärtsilä launched its brand new idea portal: MyDea for its employees all around the world on Wärtsilä's intranet. Employees can freely go and post the ideas that have been discussed during their last coffee break or when overhauling an engine and noticing that a tool could be more efficient. Only after a few months over 300 ideas were posted by different employees.

MyDea is the perfect way of collecting ideas on a global multicultural level, based on a couple of reasons:

- All Wärtsilä employees have access to MyDea
- The ideas are visible to everybody within Wärtsilä (except patent pending ideas)
- The ideas are stored in the same place and become a big bank of possible innovations for the future
- The idea collection is computerized and does not require a middleman to collect and make inputs into the database like a physical mailbox would require

It is the Innovative environment that a company needs to evolve its products. All ideas are treated equally and nothing gets censored. When the idea is posted it gets forwarded for expert evaluation. There are about 25 coordinators that forward ideas to experts who evaluate the ideas. The creator of the idea can always monitor the progress of the idea to see whatever action is taken. It gives the creator a rewarding feeling that it was a high value contribution to the company and it encourages to spawn more ideas. /13/

2 Present situation

2.1 MyDea

Developed by an external company and written in Silverlight, MyDea is here to stay. The proof is the large numbers of ideas posted after launch. But a large number of ideas call for a lot of coordinators and experts evaluating the ideas. The coordinators and experts need to evaluate the ideas quickly so that they don't stack up. But most important so that the interest of the idea creator doesn't fade away.

MyDea works great for the users who only post ideas. The GUI has been designed with usability in mind. It is easy to notice that it has been designed that way, because the front page of MyDea even features links like printable instructions, training videos and how to submit an idea. If the user feels that he needs training or help with MyDea, he does not have to go anywhere to get it, because the training is embedded in the application. The user is given an option to browse through all the ideas and make filters for all the attributes of the ideas. There are three different categories where the user can post the idea: Product & Solution, Operational, and Business. The user is automatically taken to a form that needs to be filled in with the basic information about the idea when the user clicks on one of the three categories. The forms are easy to understand because they feature a tooltip for every text field that guides the user through the whole process when submitting an idea. The user then submits the idea and awaits feedback or returns later to see if any progress has happened.

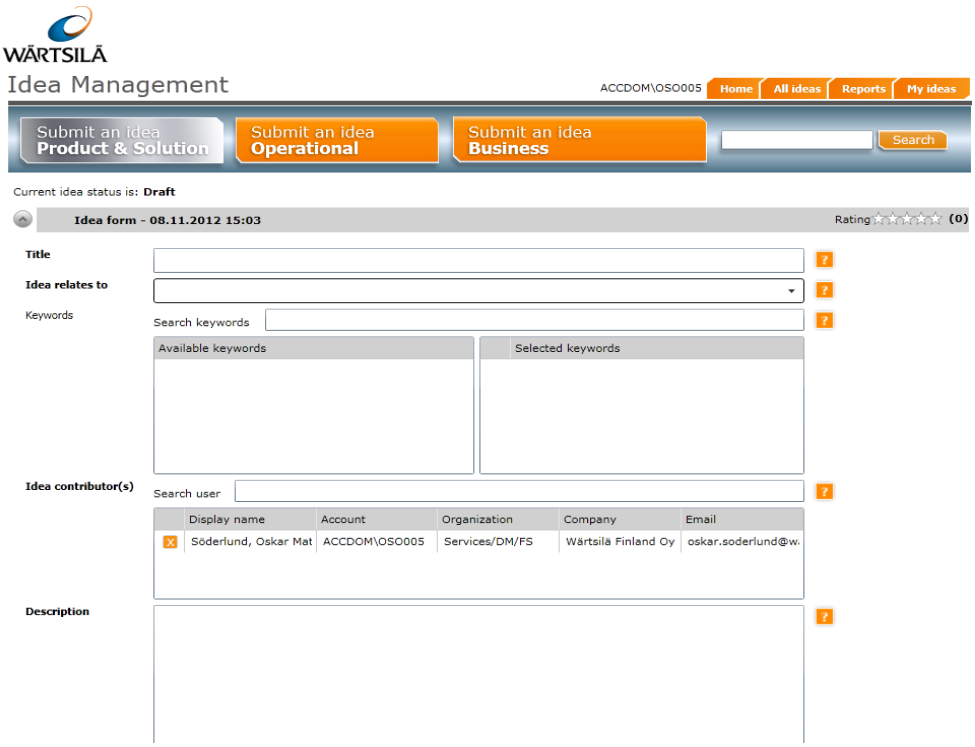

Figure 1. Mydea when user is submitting an idea.

The process is unfortunately not this easy when it comes to the part where the coordinators take over. After an idea has been submitted, one coordinator will receive a mail with information and a link to MyDea. The evaluation starts immediately in the coordinator's head when the idea is seen, according to the coordinator that I spoke to. The coordinator reads through the idea and maybe corrects some spelling errors or chooses different words or terms for some parts. With a high amount of ideas coming in for evaluation, the ability to keep track of the ideas quickly fades away for the coordinator.

In order for an idea to become an Innovation some action has to be taken. The work varies from idea to idea. But often bigger ideas need to become a project before there is a full-fledged Innovation. MyDea lacks the ability to keep track of projects and that directly means that the creator of the idea has no way of knowing at what stage the project created by his idea is. MyDea has even more and bigger flaws that directly impact the coordinators and the experts that evaluate the ideas. When looking at these it is clear that the way of working for the coordinators and experts was not accounted for when designing MyDea. The focus was put on the creators of the ideas.

2.2 The coordinator's way of working

At first the coordinator receives an email saying that a new idea has been created in his area. The email consists of a link to MyDea. He opens it and gets the first look of the idea. The evaluation process starts immediately. The coordinator that I have interviewed uses MS Excel as his master database for all ideas that he has received. There is no chance for him to keep track of all ideas otherwise. He then copies the idea number, a unique four-digit number auto created by MyDea, to his Excel file with other important data. After that is done he creates a folder on his local hard drive with the idea number as name for the folder. The purpose of this is that there is a need to work locally on the ideas before sending documents to Wärtsilä IDM. IDM is Wärtsilä's only accepted document storage application, with secure servers around Europe. After creating the folder on the local hard drive the coordinator creates in that folder an MS Power Point presentation from a template used for every idea. Then he copies various data from the Excel file to the power point presentation. From the template he copies text to put back up to MyDea in a more organized way. Before uploading the power point presentation to IDM the coordinator must create an IDM "document" in IDM. After the document is created the coordinator has the URL to the idea's IDM document. The URL is pasted in the power point presentation. Only then can the presentation be uploaded to IDM and MyDea.

Meetings occur now and then where the ideas are evaluated and their future is decided. All the ideas that have come in lately to that coordinator are brought up and discussed. For a meeting you need a meeting agenda to keep things structured and time-efficient. The coordinator prepares the meeting agenda in MS Word. The coordinator uses a slightly modified Wärtsilä standard meeting agenda. Ideas can be decided on four different options: passed, not passed, transferred and stop.

Figure 2 illustrates the process that a coordinator goes through when handling an idea before the gate meeting.

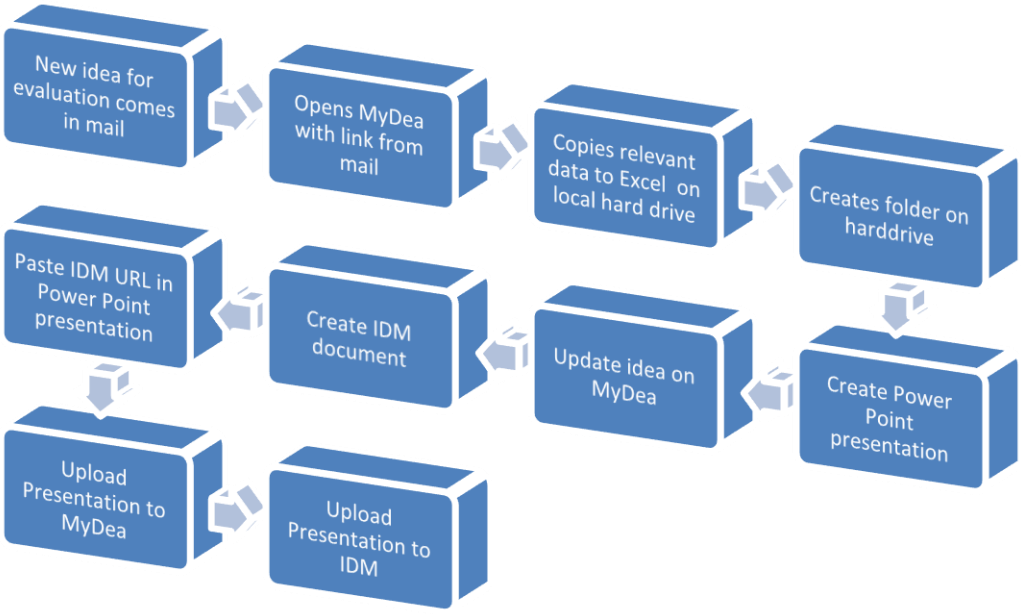

Figure 2. Process before Gate meeting.

2.3 Flaws in the present way of working

As seen in figure 2 there are many different things and applications to bounce between when handling one idea. Everything is done by hand and creates an environment for human error. The coordinator said himself in an interview that he estimates that this copying text and uploading files into the right places takes almost 90% of his working time. This is time that could be used to evaluate ideas or discuss the idea.

MyDea was created for the user who posts ideas. This is clear after looking into MyDea and evaluating the application. At first the coordinators couldn't even link to an idea. They had to launch MyDea in the browser and then provide the idea number in a search field in order to get to the idea. A request was sent to the developers so that ideas could be linked to, so that the coordinators and others could get quick access to ideas and make linking easy. This is one step in the right direction.

The coordinators can hardly work on an idea in MyDea. There is a lack of administrative functions for the coordinators in MyDea. An administrative interface would be needed that could show all the ideas that the coordinator needs to administrate. That's why the coordinators have resorted to Excel to keep track of the idea's progress. There are only some minor benefits when evaluating an idea in MyDea. Comments can be added and the

description can be changed and other small things. But the administrative side is completely left out. This forces the coordinator to have a master database with ideas in an Excel sheet where he can keep track of the ideas. The coordinator also said that you could upload one file to every idea. That sounds alright, but when the file is uploaded to MyDea there is no way of deleting, renaming or changing the file. Once it is uploaded it is stuck there forever. This could lead to some very unpleasant scenarios, if the wrong idea presentation is uploaded to another idea in MyDea.

3 Optimal situation

The optimal situation would of course be that the coordinator works 100% in MyDea. That would mean that the coordinator does not need to interact with any other applications. No transactions between other databases would be needed. The daily job could be done only within MyDea whilst the coordinator administrates the ideas. However, this requires MyDea to be completely refurbished for the coordinator's needs. Consultants and the developers of the MyDea application would need to be involved again and that would bring a very large cost, which would be way too large for Innovation Management. The coordinator was wishing for some kind of "control panel" or an "app center" in MS Excel. This panel would consist of functions that could execute the transactions that the coordinator does by hand. The coordinator uses MS Excel as a master database of the ideas that have been submitted. This would result in something that could make the coordinator's manual work faster and then more time could be spent on thinking and evaluating ideas instead of copying hyperlinks and creating presentations. The coordinator also suggested that this could also work as a prototype or showcase for future developers. Consultants or developers understand more easily what the finished product should look like if they see a working example or a graphic showcase of the process. The coordinator also stated that this automated functionality would eliminate human errors like linking to wrong presentations or ideas.

3.1 File uploading to MyDea

The current way of uploading a file to MyDea is not optimal. This statement is based on a couple of facts: according to Wärtsilä standards, documents should be stored in IDM, which has secure servers and revision handling. Moreover, the document should not be frozen in MyDea if it needs to be changed.

Another application in Wärtsilä developed by external manpower uses an interesting solution when it comes to "uploading" documents from IDM. It is called Workshop Catalogue and consists of all the Wärtsilä workshops around the world. Every workshop has its own tab for documents. The documents are not uploaded directly to the Workshop Catalogue but uploaded to IDM in a certain area used only for the Workshop documentation needed for this Workshop Catalogue. Every Workshop has its own id. When documents are uploaded to IDM, the user can enter the workshop id for the document. Once a day a script goes through the documents uploaded in the area in IDM and scans for the workshop's id and adds links under the correct workshop document tab. In this way all the documents are stored under IDM but replicated to the Workshop Catalogue.

This IDM integration could work with MyDea as well. Every idea has its own unique idea number. The documents could be uploaded to IDM with that number specified and then replicated to the right idea once every day. This would be much more efficient than the way it works right now. /16/

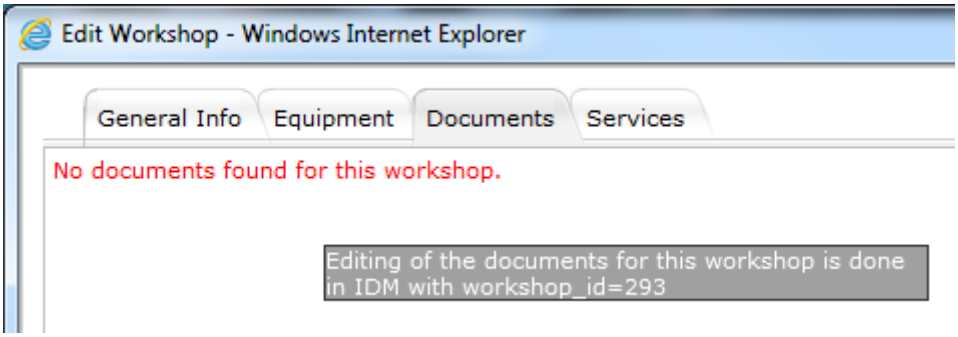

Figure 3. When in edit workshop mode a label gives information about the solution ID that the document needs to have.

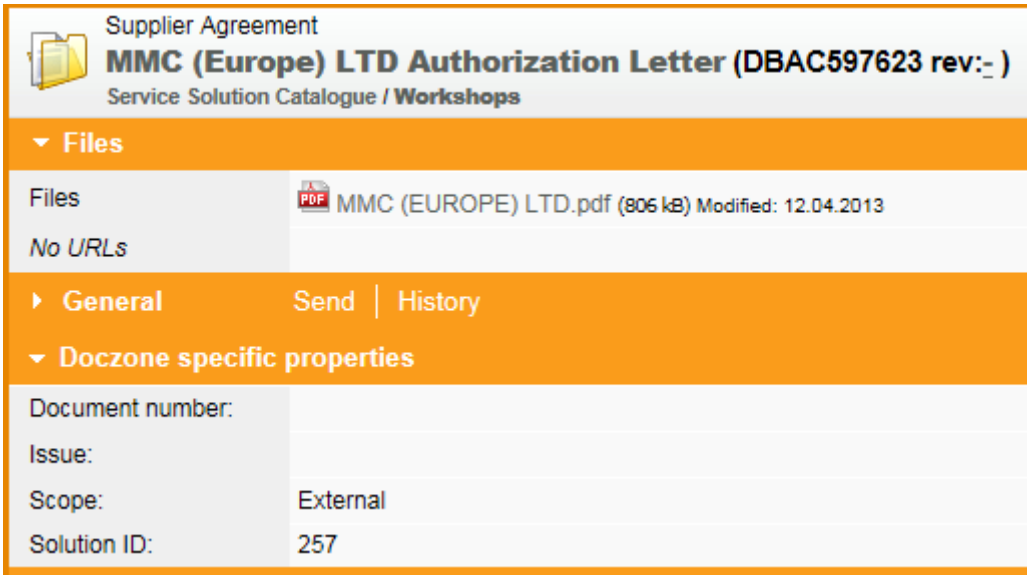

Figure 4. Document created with solution ID: 257 in IDM

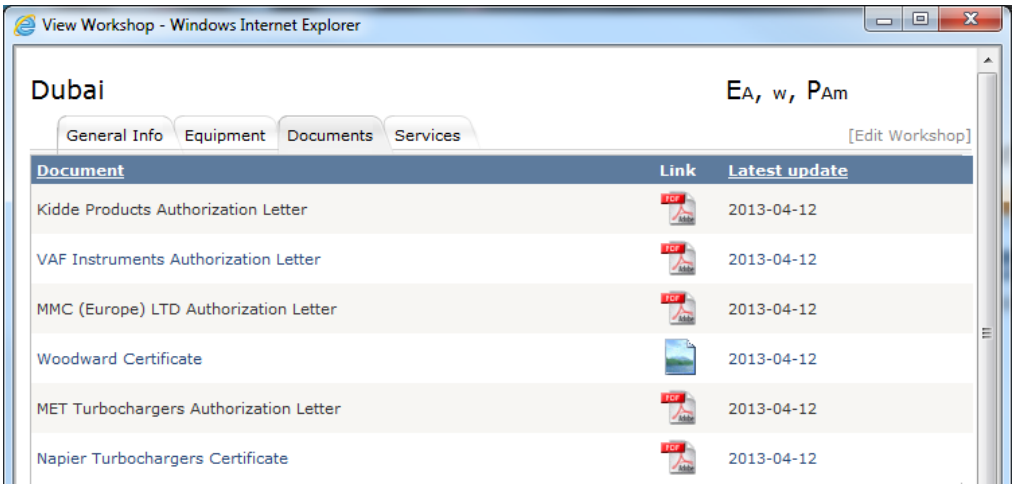

Figure 5. The document has successfully been replicated into the Workshop Catalogue under Dubai Workshop.

3.2 Developing the control panel in MS Excel with VBA

With the process given a simple table was created in MS Excel to represent a little sample of what the coordinator's Excel file might look like. This would then be used for testing the functionality of different functions in the process that will be automated by the control panel. The coordinator wanted the first version to be developed in MS Excel just to show what some short and simple programming could do to ease up his work load.

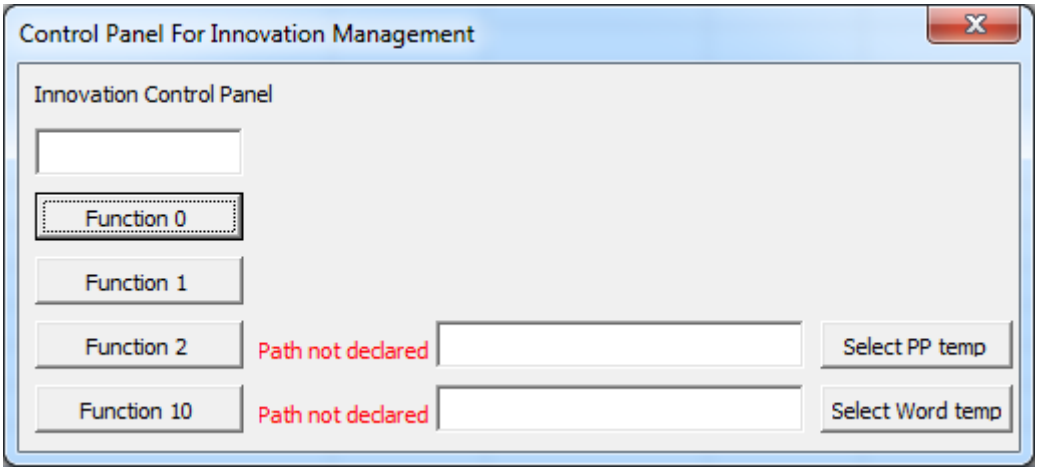

Figure 6. Early version of the control panel

With four quick and simple functions in the control panel the work could be done much faster. The strength of programming with VBA is that all MS products are heavily supported when programming. With very little effort it is easy to create integration between the different Office products.

3.2.1 Function 0

This is a simple function but a timesaver when working in Excel. When pressing the function 0 button Internet Explorer launches and opens up the idea in MyDea. The idea number must be inserted in the textbox in the upper left corner. This is possible thanks to the change that made linking to ideas easy with just adding the idea number to the URL. Writing the code for this is easy, you just create a string with the first part of the URL to MyDea. Later, the content in the textbox is added to the end of the URL string. Then an IE object can be created in the code and the IE object is set to navigate to the string that has just been created. Additional code can be added but is not necessary for the core functionality. Figure 7 shows the needed code as well as the additional code. IE.Visible is used to prevent the user from seeing Internet Explorer while the browser is loading. While hiding Internet Explorer, Application.Statusbar is given an informative string to notify the user that Internet Explorer is loading in the background. /9/

```
Private Sub FunctionButton0_Click()

    'Dim urlString1 As String already defined in the general part of the code
    urlString1 = "http://mydea.wartsila.com/#/Manage/"

    ' Create the InternetExplorer Object
    Set IE = CreateObject("InternetExplorer.Application")

    ' Popoulating strNewFolderName
    strNewFolderName = FunctionTextBox1.Text

    ' Not showing Internet Explorer before it has loaded completely
    IE.Visible = False

    ' Send the form data To URL As POST binary request
    IE.Navigate urlString1 & strNewFolderName

    ' While Internet Explorer navigates to the set path
    ' Excel shows this string in statusbar
    Application.StatusBar = "http://mydea.wartsila.com/#/Search/" & strNewFolderName

    ' Wait for Internet Explorer to load completely
    Do While IE.Busy
        Application.Wait DateAdd("s", 1, Now)
    Loop

    ' Show Internet explorer when everyting has loaded
    IE.Visible = True

    ' Clean up
    Set IE = Nothing
    Application.StatusBar = ""

End Sub
```

Figure 7. This snippet of code shows the whole inner workings of function 0

3.2.2 Function 1

This is also a very easy function, which saves time for the user and is much appreciated. If the user has inserted an idea number in the upper left textbox he might press the function 1 button to automatically create a folder on his hard drive. This is achieved with the MkDir function in VBA. The path is preset in the code. Another possibility for this would be to prompt the user a FileDialog Object. But the MkDir function is much more faster when used. The coordinator always named his folders “Idea_XXXX” where the X’s are numbers. To adapt to his system of naming the folders a string was created in the code containing “Idea_”. Adding the string and the idea number always results in a continuous name format for the folders. Some IF statements were also created to ensure that the textbox was not empty or that a folder with the same name wasn’t present in the preset folder. It also ensures that the idea number is of the right length. Figure 8 illustrates the simple use of IF statements to prevent the user from doing hasty mistakes. Figure 9 illustrates the consistent naming of the folder on the coordinator’s hard drive.

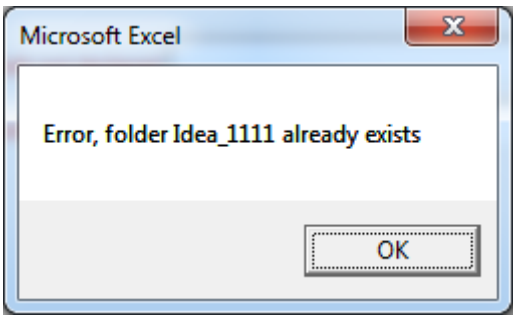

Figure 8. Message Box that pops up if the user tries to create a folder that exists.

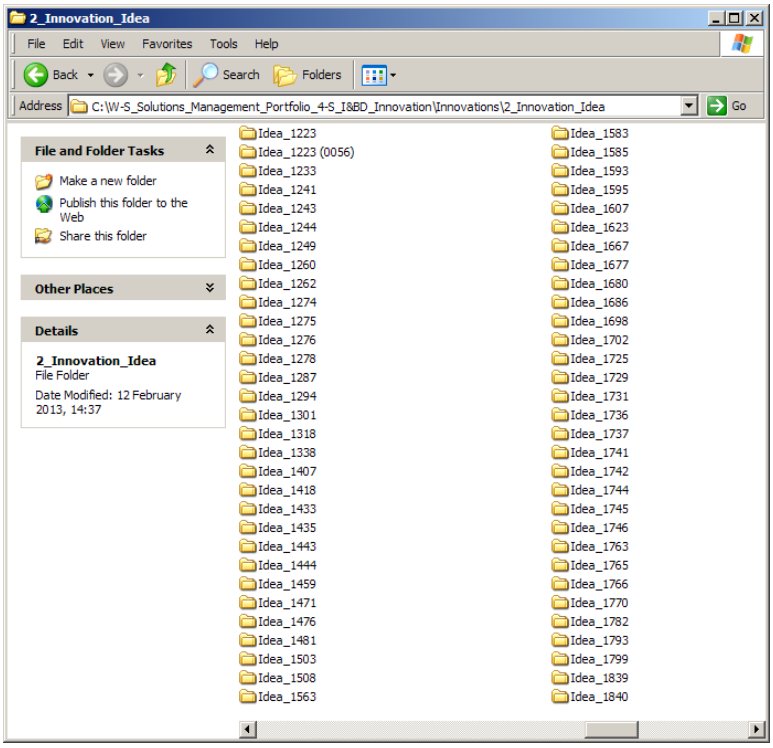

Figure 9. Example of how the coordinator's backup folder might look.

3.2.3 Function 2

Function 2 lets the user create a power point presentation automatically from the application. The coordinator that was interviewed always uses a power point presentation to present the idea's description, possible problems and benefits for Wärtsilä, benefits for the customer and other things that need to be taken into account when evaluating an idea. Ideas differ a lot but the same template for the presentation can often be used.

The presentation used as a template must be declared within the application. The "Select PP temp" button is pressed and brings up an open file dialog for the user to search for the right power point presentation template. This is achieved by using the

Application.GetOpenFilename method and, to simplify the searching for the file some parameters are set like this: ("Powerpoint Files (*.pptx), *.pptx"). This excludes all other files in the open file dialog and leaves the power point files visible. /1/

The value of this Application.GetOpenFilename is then put into the textbox beside the "Function 2" button. When the textbox changes, the label on the right also changes so that the user knows that a template is defined. The properties on the textbox are set so that you cannot manually define the path. The possibilities to change and create a fail proof template selection are endless.

When the user presses the "function 2" button with every needed data put in the system, the creation of the power point presentation will start automatically. The basic process that the application does is that it takes data from the Excel sheet and puts it into preset shapes in the power point presentation. Below is an example of how easy it is to paste data into power point after the shape in power point is known.

```
'Copy the data to the shape in power point
newPowerPoint.ActivePresentation.Slides(1).Shapes("textruta 9").TextFrame.TextRange.Paste
```

Figure 10. This snippet of code shows how text is taken from Excel to a specific shape in power point.

When the data is copied into the different shapes in the presentation, the presentation is automatically saved in the right folder on the hard drive.

3.2.4 Function 10

Function 10 is about automatic creating of meeting agendas for the regular meetings that the coordinator has to lead. Before the meeting the coordinator has to prepare the meeting agenda with all the ideas that have been submitted since the latest meeting. The meeting agenda consists of a small table with the ideas. The idea number should be hyperlink so that during the meeting when following the agenda the coordinator can click the link and automatically bring up the idea in MyDea.

The coordinator that I spoke to said that he usually copies the ideas from the Excel file and later formats the idea numbers to hyperlinks in the agenda. He said that this is not only time consuming, but it is also very easy to make human errors. The worst scenario is when

the hyperlink opens up another idea in MyDea. Things like that should not happen during a meeting with a tight time schedule.

To prevent this kind of situations the “function 10” is populating the table in the agenda automatically. In the template there is a table with the right amount of columns and with the headers already specified. By pressing the “Select Word temp” button the user is receiving an Application.GetOpenFilename popup like in “Function 2”, but this time the parameters are set to look for Word files. After the Word file is chosen, its path will populate the textbox like in “Function 2”. After that the user can click the “Function 10” button to create the agenda document. The user is prompted a popup which wants the user to enter the first idea number that goes in the table.

The VBA code then searches for the idea in his Excel master database, selects all ideas under the one specified, moves to the Word document and pastes the idea numbers in the first column. Then the code does the same thing with the headlines of the ideas.

3.3 How to use templates and find shapes

In order to usefully change or populate or somehow edit different Word and Power Point files with VBA, the applications must be declared in the VBA code. However, not only the application has to be declared. The document or presentation also has to be declared if there is going to be some sort o editing inside a Word document or a Power Point presentation. It goes even further; the slide also needs to be declared within the code. This is all done because otherwise the application would not know what to focus on and what to change and where. /6/

```
Dim newPowerPoint As PowerPoint.Application
Dim newPowerPres As PowerPoint.Presentation
Dim activeSlide As PowerPoint.Slide
Dim Rng As Range
```

Figure 11. Declaring the needed variables to later run the Power Point.

As for the specifics of what needs to be entered or changed, one must know the ID or name of the shape that needs to be changed. This can be done in a couple of ways. In Power Point there is a separate window to see all the shapes’ names in the current slide. This may be quite hard to find if you don’t know where to search. Another way of finding out the shape’s name is to create a simple macro. The basic idea of the macro is that before

running the macro the user selects the shape that the user wants to know the name of and then he runs the macro. The macro is just prompting the name of the shape to the user with a message box.

```
shapeName = ActiveWindow.Selection.ShapeRange(1).Name
MsgBox shapeName
```

Figure 12. A small snippet of the code showing how the name of the selected shape is passed on to the message box.

The tables in Word work a little bit differently. They have no set names or Id´s but they are indexed. If the word document doesn't contain a big amount of tables you could just count them, but if you are uncertain about the amount of tables in the document you could proceed with the same basic idea as in Power Point. A small macro that for example loops through all tables in the document can be created. Then you can be sure of the amount of tables.

When creating "function 10" the idea was to have the application to create a completely new table in the agenda template. It was done like that at first, which is a sustainable way of doing this process. However, a quite large amount of code is needed to get the table style to be the same as the rest of the tables in the document. Instead the table was left empty in the template with the correct style and the correct headers. With this the application can populate the table without the need of creating a table and then proceed to populate the table. Much less code needed to be written. But for an operation like this the index of the table needs to be known. /14/

```
'Set newWordTbl to be the third table in the document
Set newWordTbl = newWordDoc.Tables(3)
```

Figure 13. In this case it is the third table that will be populated by data.

3.4 Using the Macro recorder to get jumpstarted with the code

When writing VBA code there will be times of uncertainty about what to write and in what order, or complete lack of knowledge of how to tackle the challenge. A common belief is that it is only possible to record the macro and then play it if the results are satisfying. However, it will be hard for those who aren't really familiar with coding. After the macro

has been recorded it is possible to view and edit it in the VBA environment. A good eye for code is needed to edit the code into your absolute liking. /12/

The mind has to be in the right ways of thinking when recording a macro that will later be used for the code only. It is important to get the results right, but it is the way of getting the results that matters the most. Planning every step before recording the macro will save a lot of extra work later on when editing the code. Another thing that saves a lot of work when recording macros is to know the Excel shortcut and function keys. This might be good when for instance selecting all rows downwards to the last cell populated by data. CRTL+SHIFT+▼. /3/

After the macro has been recorded it should be edited in the VBA environment to fit the future needs. There are often some unnecessary lines or properties that can be removed. The macro recorder records every keystroke. If by accident a wrong key has been pressed, it will be one of the lines in the code, and this line is then easy to remove after the macro has been recorded. Below is an example of a macro recorded with the intention of creating hyperlinks automatically so that the user doesn't have to do the work by hand.

```
Sub Hyperlink2()
'
' Hyperlink2 Macro
' Recorded 11.12.13 14:15  By Oskar Söderlund

    Cells.Find(What:="7878", After:=ActiveCell, LookIn:=xlFormulas, LookAt _
        :=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, MatchCase:= _
        False, SearchFormat:=False).Activate
    activeSheet.Hyperlinks.Add Anchor:=Selection, Address:= _
        "http://wartsila.com/7878"
    Range("B6").Select
End Sub
```

Figure 14. Raw recorded macro.

Figure 14 shows the result of what happens when a macro is recorded. First the search function in Excel was used to find the idea number. When the macro was recorded the user puts in 7878 as criteria. However, as 7878 can't be used every time this script is going to run, it will have to be changed. InputBox Method could be used, a string or even some text in a textbox. In this case it will be InputBox with some informative text. Looking further into the code there is one property that needs to change to avoid trouble.

LookIn:=xlFormulas. This means that when Excel searches for the idea number it will search in the formulas in the cells rather than in the values of the cells. This could probably have been avoided when recording the macro by choosing right search options under the advanced tab in Excel's built-in search function. However, xlFormulas should be changed to xlValues to ensure that Excel does not search in the formulas. The next thing while recording the macro was that the cell being searched after got hyperlink properties added. The URL string was typed in by hand while recording. The only thing that needs to change here is the URL address; it can't be the same for every idea. But the beginning is the same for every idea so that part of the string can be kept. There are many ways to complete this address, but for this thesis work Selection.Text was used to complete the address. The cell that is selected is the one that gets processed and the value is the only thing that differs from the other URL's. The next thing that needs to be edited in the code is the Range(B6).Select. That is the result of someone recording the macro and accidentally hitting the down arrow key. Figure 9 illustrates the macro after editing. /8/ /11/

```
Cells.Find(What:=InputBox(sPrompt & "Insert the first ideanr!"), After:=ActiveCell, LookIn:=xlValues, LookAt _
    :=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, MatchCase:= _
    False, SearchFormat:=False).Activate
activeSheet.Hyperlinks.Add Anchor:=Selection, Address:= _
    "http://wartsila.com/" + Selection.Text
```

Figure 15. The same macro after editing. It can now be used in a VBA function.

4 Moving forward with the control panel

After showing the control panel built in Excel VBA to the coordinator it was clear that this kind of automation was highly appreciated. He quickly estimated that much time could be saved by using a functionality like this. After a quick discussion he presented some proposals about how to move forward.

- Make it a stand-alone .exe file
- Friendlier GUI
- More showcase/prototype oriented

The standalone .exe file is wanted because then the application is not bound any longer to one document or spreadsheet. VBA lacks the ability to compile the application into an .exe file. Therefore the project must be moved to another environment. The GUI certainly needs to be friendlier. Too often the applications in the offices or in the industries are too dull and complicated. Employees often need training to understand how to use the applications in their daily work. The applications should present themselves with a friendly GUI, and the GUI itself should guide or indirectly suggest what the user should do. Nowadays we install applications on a daily basis on our home computers, telephones, tablets and even on gaming consoles. It is very rare to see that one must read auxiliary material to get the application going. The standard should be the same in industries and offices. Some emphasis should be placed on the showcase side of the application as well.

4.1 Going from Excel VBA to VB Windows Forms Application

In order to fulfill the criteria to get the program to an executable file the environment must change. There are many programming languages that would fit the needs of this kind of application. In this case Visual Basic and Windows Forms Application looked like the best candidates for this operation.

Having the application in an .exe file includes many benefits. Firstly, the application is not bound to the Excel file, which means that the Excel file doesn’t have to be opened when running the application; it doesn’t even need to be present on the hard drive. It is also easier to get more users to use the application if they don’t have to rely on exactly the same Excel document. They can work from their own document, if they use the same layout as

the original. This is also great for showing during a conference or a meeting without showing the Excel file, as that could cause some unnecessary confusion. /4/

4.2 Refurbish the GUI

When the decision was made to move to VB.NET and Windows Forms, the first thing to do was to start with building the GUI from the scratch. This time the GUI would be more stylish and modern but not too far away from the Excel VBA version. This version should have a Wärtsilä colour themed appearance and the Wärtsilä logo incorporated in it. The background was taken from the Wärtsilä standard Power Points. An orange industrial themed background, the orange colour comes from the Wärtsilä logo. The buttons would be blue and that colour can also be found in the Wärtsilä logo.

Visual studio was chosen for the creation of the Windows Forms application, as it is easy to work in regarding the GUI. The background is easily changed within the properties panel for the form that is being worked on. Basically all properties for the form and the things belonging to the form can be changed from the properties panel.

When adding buttons and textboxes it is important to keep a consistent naming of the buttons and the textboxes to keep the coding later on as smooth as possible. Visual Studio incorporates an auto incremented naming for all different controls and other tools. But often one keeps deleting different controls, so the auto naming can feel a little inconsistent. A standardized way of naming the controls should be devised as soon as possible to avoid this inconvenience. A good example might be to name the buttons: btn + what the button does. This is nearly the same as The Hungarian notation identifier naming convention. /7/ Examples:

- btnSubmit for Submit button
- lblPrice for price label
- txtIdea for idea text box

After most of the controls have been placed in the form it is time to add some functionality to the whole system. There are no limitations regarding coming back to change the design if and when some better design ideas pop up.

4.2.1 Using Photoshop to make custom buttons

Instead of having standard buttons with text only a decision was made to make the buttons only have pictures explaining their functions. This makes a lot of sense because Wärtsilä is a global company and houses a lot of different languages. Of course the official language used is English, but a picture is understood in every language equally.

Designing the picture for the buttons in Photoshop can be done in different ways and can be very hard for one that doesn't possess the knowledge of the inner workings of Photoshop. Some artistic talent is also needed, but is not a complete must. There are many different tutorials on the Internet on the subject on how to create buttons. A quick search on Google.com on "how to create buttons in Photoshop" yields over 576 million results. These buttons are mainly for the web, but they can also be used in programs.

A good way to start is to see in what size the buttons should be. In the Visual Studio forms editor the buttons' properties can be seen in the right hand properties window. The buttons should be neither too small nor too big. In this application the buttons are a little bigger than standard Windows buttons, because the images would become too small to illustrate the purposes of the button. In Photoshop a new RGB document was created with a larger size than the buttons themselves. The rounded rectangle tool was used to create the buttons. It was important to get the selection as close as possible to the desired size of the buttons in Visual Studio to make them look as natural as possible. The radius of the corners should not be too big, as the corners will then stick out in the application later. Another important step is then to set the background color. The buttons in this application will have a blue background with a gradient overlay in linear style. Blue is inspired from the Wärtsilä logo. To make the button pop out from the background a 1 pixel stroke in a darker color was added. From this point on the buttons can be customized to one's own liking with an infinite amount of different effects. At this stage the buttons are ready to be saved as a .psd file and all the different buttons can be made from this template. Each and every button will look different and this was achieved with different images. Free icons or images can be downloaded from Internet, preferably in .png format for best quality. The images should not be much smaller than intended to avoid having to enlarge them or the icons when putting them to the button in Photoshop. Thus the image quality is kept. The image should not be copied directly from the browser to Photoshop, because when doing so there might be a risk that for some reason images with transparent background turn into black background. This is easily avoided by downloading the image to the hard drive and

then opening it with Photoshop. The images are opened with Photoshop and then inserted into the Photoshop layer that will be the finalized button. No extra effects are added because the buttons in the application itself have a “glossy” or shiny style. Then the Photoshop document is saved to .png format for highest quality. /10/

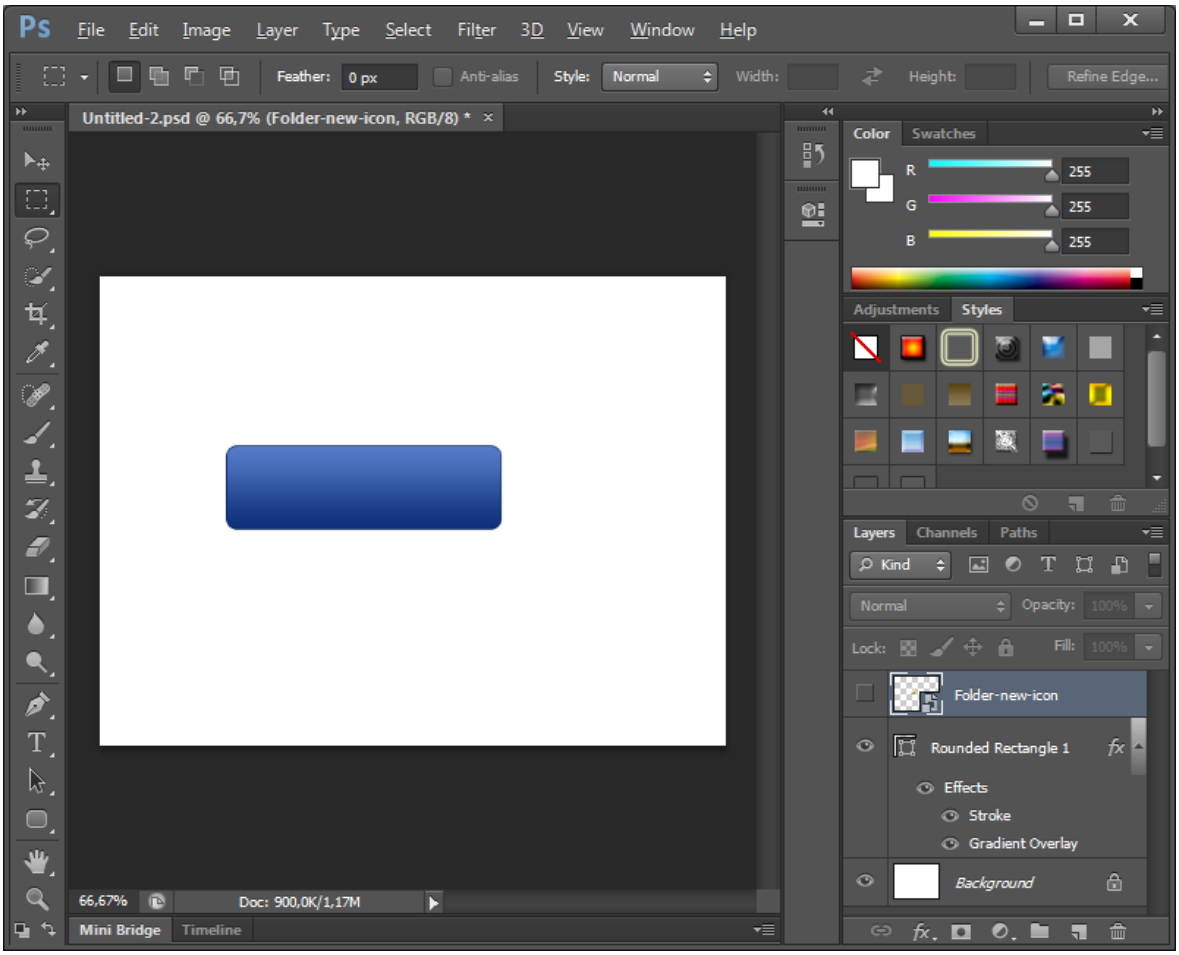

Figure 16. Photoshop with the finalized button template.

4.3 Copying the code

When most of the controls had been placed, the code was copied from the excel VBA version to the new VB.NET version in Visual studio. This was achieved by double clicking on for example a button whose task it is to execute the code. This will bring up the code window for the form that the button is placed in. When copying the code for each and every button from Excel VBA to Visual Studio there are a few things consider. Firstly, the buttons might have changed names or the textboxes have been created in the new form. To avoid this, the same naming for the buttons and textboxes can be used in the new version. It is probably not a big issue, if the form consists of many different controls. Then consider

using the “find and replace” functionality in Visual Studio. However, Visual studio will most probably tell if some naming is wrong in the code.

A thing that is certain to happen when copying the code from Excel VBA is that if the code consists of some sort of Excel automation, the references and Excel objects will be missing. In this case the application will perform Excel, Word and Power Point automation. All three will need added references and added objects in the code. The references are added from the project menu. The needed references are found within the COM tab in the add reference window. There are many different references to choose from, but a quick search on Internet makes it clear that it is Microsoft Excel Object Library that is needed for Excel automation. After the right reference has been added, a line of code needs to be entered in the general area of the code window for form1. The line is: Imports Microsoft.Office.Core. Without this line the Excel automation won’t work. A strange problem that was encountered during programming was that this did not work properly. While trying to resolve the errors another reference was added, an almost similar reference to the one mentioned above, but with a Swedish name. This could be the result of a Swedish version of Microsoft Excel on the particular pc. To avoid this it is highly recommended to use the English version of the Microsoft Office package from start to finish. /5/

The copied code in the form mentioned above will result in a lot of errors, although all the references are added. The main reason is that there is no need to declare any Excel variables in Excel VBA. In VB.NET, however, it is essential to declare the Excel variables, in order to make the Excel automation work. Application, Workbook and Worksheet are the most common objects that are needed. These are declared at the beginning of the code. In Excel VBA when automating Word or Power Point, the objects have already been defined in the code by the user so that Excel VBA knows what to take control of when the code runs. The same needs to be done for the Excel objects now when copying the code from Excel to Visual Studio. If for example Rng is declared as Range in Excel VBA, the same code in Visual studio does not know that the range will be a range in Excel. This is fixed by declaring the variable as Excel.Range in the code in Visual Studio. The same goes for all the other operations in Excel. The Illustration below shows how the code has to be modified in order for Visual Studio to know that it is in Excel that it should perform the task. /5/

```
Rng = newExcelshe.Cells.Find(What:=sIdeaNr, After:=newExcelshe.ActiveCell, LookIn:=newExcelshe.xlValues, LookAt:= _
    newExcelshe.xlWhole, SearchOrder:=newExcelshe.xlByRows, SearchDirection:=newExcelshe.xlNext, MatchCase:=False _
    , SearchFormat:=False)
```

Figure 17. The code has been modified with newExcelshe in front of different values to show Visual Studio that it is in the active Excel sheet that this code should be performed.

5 Possible ways to get data from MyDea

Even though this was out of the scope of this application, a small investigation was done in order to look at the possibilities to get and edit data in MyDea through an external application like this one developed. The coordinator briefly mentioned that he believes that the database is a Microsoft SQL server 2012.

5.1 Fetching straight from the browser

There are different ways to complete a task like this. Two main ways have been looked at. The first is a so called "Web scraping". Usually it is used by web robots roaming the Internet to scrape data out of different sites to get all sorts of information such as: weather data, price data and website change detection. What the coordinator is doing to an extent is a sort of web scraping when he copy-pastes the data from MyDea to MS Excel. It could be called Human copy-and-paste web scraping. /15/

Another way in which an application could retrieve data from MyDea would be simple Internet Explorer automation. By declaring InternetExplorer.Application as an object it is fully possible to automate Internet explorer from the code. The application can do all sorts of things with Internet Explorer, e.g. navigate to different web pages, fill in forms, press buttons, click links and fetch text from different web parts. Often on a normal web page the fields are named elements. The names can be retrieved from the web pages by inspecting the web pages' code. Inspecting the code of the web page is usually done through the browser. Inspecting the code is different for every browser. For example the Google Chrome browser´s inspect element is started by pressing the right mouse button and then selecting "inspect element". In Internet Explorer the view tab has to be pressed and then "developer tools" is selected. A quicker way to access the DOM tree is to press F12 while in the browser. This works for most of the standard browsers. /2/

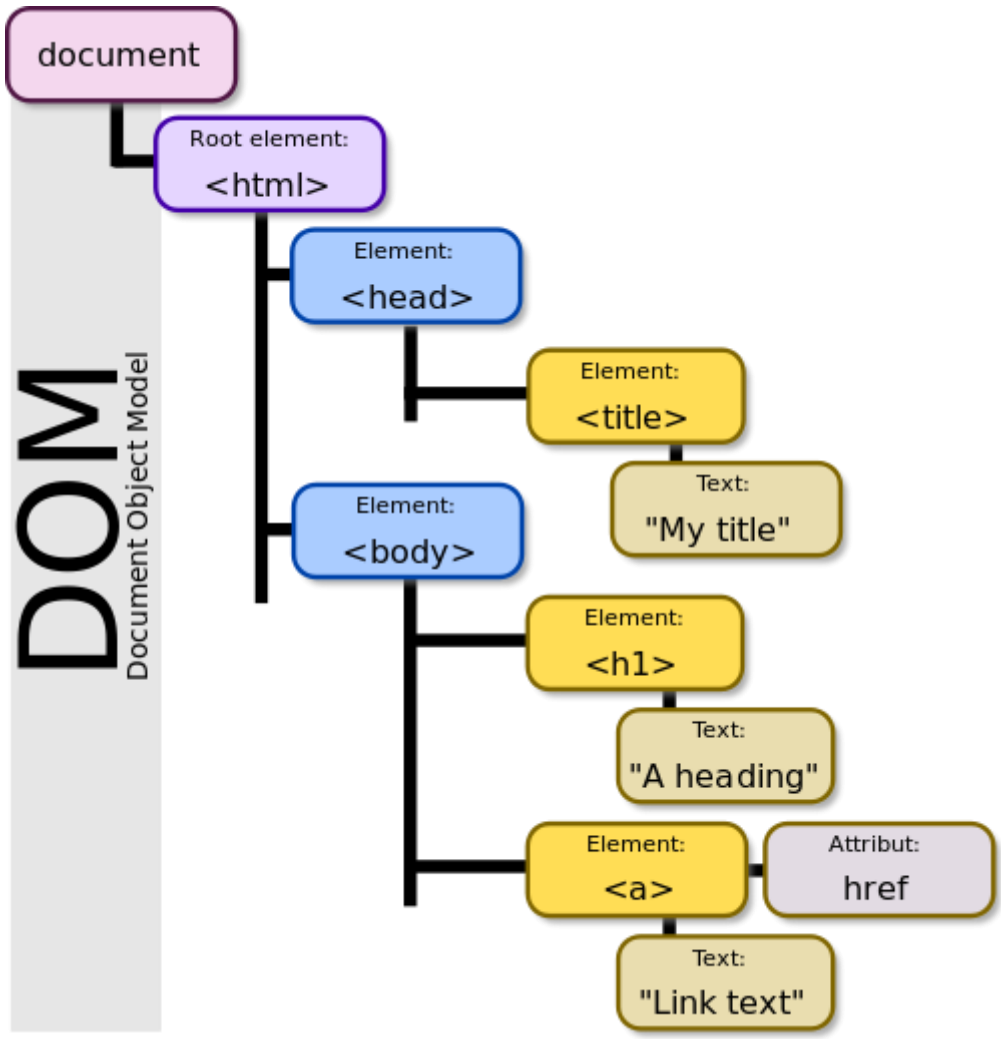

Figure 18. A simple DOM tree illustrated.

Unfortunately MyDea is developed with Silverlight, which results in the fact that the DOM tree is very lacking of useful code. The browser treats the Silverlight application as a whole object, which makes it impossible to retrieve any named elements. The DOM tree is there and the head and body elements are present. The Silverlight element is also there but there is no possibility to collapse the element and see the elements within it. Even when trying to select some text on the MyDea page with the mouse it won't let you, because all the elements are locked, except for the one that the coordinator usually copies into the Excel file.

5.2 SQL

The Wärtsilä internal documentation states that MyDea is using Microsoft SQL server 2012. This means that VBA or VB.Net could clearly retrieve data from the database. With

the other options not possible this would indeed be the proper way of sending and receiving data from MyDea. This could be achieved by having access to a properly set up user in the database and then create a connection string in the code. For just fetching data from MyDea to Excel the user would only need to be allowed to select data from the database. However, it was decided that this application should not interfere with the data stored in the database. If the case was different the coordinator could have had the possibility to get and send data directly from the application without opening the browser. This could have saved a lot of time for the coordinator. But it would have demanded extensive research and programming to make a good integration between the application and the SQL server, which would not suit my thesis work at this given time.

6 Results

The results are clear and they show that the coordinator does too much unnecessary work by hand, which could easily be automated by an application. MyDea has not been designed with the coordinator in mind. The ideas need a lot of maintenance after they have been submitted not to be buried under the large amount of ideas that come in daily. My application eases up the work and keeps a consistent way of working and storing the ideas for the coordinator. My application also automatically creates Power Point presentations to the right folder in an organized way and with very little effort required from the user. The same goes for the creation of the meeting agendas. Very little time is now needed and the risk of human errors does not exist.

A stand-alone application with the same features as described above, developed in VB.NET, was created. My VB.NET application brings more benefits due to the fact that it is a stand-alone application which is not bound to one Excel file. The VB.NET developed application is also much more stylish than the one developed in VBA Excel.

The research also shows how the document handling could be better within MyDea by using a solution that is already implemented in other applications.

6.1 Discussion

VBA and VB.NET might seem to be an insufficient and weak programming language to develop applications with. But a deep look into the languages proves that they are efficient and can perform lots of tasks and often there are many different ways of solving the tasks. It might not be so object oriented as many other languages, but writing the code takes very little time. The languages are also very well documented by Microsoft. If a solution that involves some of the MS office applications needs to be developed, VBA or VB.NET is well recommended for that task.

The automation possibilities between the Microsoft products are great when programming in VBA and VB.NET. In theory one could build an application that integrates the most common Microsoft products like: Access, Word, Power Point and Excel. The application could pass data between them and create automatic Power Point presentations with tables from Excel, with data from Access and then summarized reports in Word.

The request to build this application in VBA was brought to the table because the coordinator was using Excel. At first there was some hesitation in building the application in VBA, due to lack of knowledge of what a VBA written application can accomplish. The belief was that it could mainly do operations within Excel itself. While developing the application and researching VBA the hesitation gradually diminished, since the research showed that VBA is quite powerful and extends well beyond Excel.

It was devastating to see how much work needed to be done by hand while handling the ideas. The amount of manual copy-paste between programs was huge. These things happen when the whole process is not considered or revised before buying or developing the solution. However, there might be some coordinators that can work efficiently by keeping track of the ideas that they are responsible for in their own way. Be that as it may, there needs to be a standardized way of working with these things so that the idea is not handled differently depending on which coordinator that is responsible for the idea.

All in all MyDea is a great application and brings much value to Wärtsilä. The lack of administrative functions for the coordinator is a big flaw, which needs to be fixed in the future. How it will be fixed is up to Innovation Management itself.

7 References

1. Application.GetOpenFilename Method (Excel) http://msdn.microsoft.com/en-us/library/office/ff834966.aspx (fetched 20.11.2013 at 15:05)

2. Automate Internet Explorer http://www.jpsoftwaretech.com/excel-vba/automate-internet-explorer/ (fetched 15.1.2014 at 15:11)

3. Excel shortcut and function keys http://office.microsoft.com/en-us/excel-help/excel-shortcut-and-function-keys-HP001111659.aspx (fetched 10.12.2013 at 16:42)

4. exe http://en.wikipedia.org/wiki/EXE (fetched 19.12.2013 at 20:57)

5. How to automate Microsoft Excel from Visual Basic .NET https://support.microsoft.com/kb/301982 (fetched 13.1.2014 at 14:30)

6. How to automate PowerPoint by using Visual Basic in Office 2003, in Office XP Developer, and in Office 2000 Developer http://support.microsoft.com/kb/222929 (fetched 2.1.2014 at 13:27)

7. Hungarian notation http://en.wikipedia.org/wiki/Hungarian_notation (fetched 5.2.2014)

8. InputBox Method http://msdn.microsoft.com/en-us/library/office/aa195768(v=office.11).aspx (fetched 11.12.2013 at 16:09)

9. Navigate Internet Explorer using VBA http://www.access-programmers.co.uk/forums/showthread.php?p=993845 (fetched 20.11.2013 at 14:00)

10. Photoshop For Beginners: Creating buttons for web part 1 http://wegraphics.net/blog/tutorials/photoshop/photoshop-for-beginners-creating-buttons-for-web-part-1/ (fetched 27.12.2013 at 00:29)

11. Range.Find Method (Excel)
http://msdn.microsoft.com/en-us/library/office/ff839746.aspx (fetched 11.12.2013 at 16:12)

12. Recording a Macro to Generate Code
http://msdn.microsoft.com/en-us/library/office/ff838320.aspx (fetched 10.12.2013 at 15:12)

13. Salin M. (2011). Har du, en ide? *Wattsup, Wärtsilä staff newsletter, (3),* 18-21

14. Table Object
http://msdn.microsoft.com/en-us/library/office/aa212430(v=office.11).aspx (fetched 2.1.2014 at 19:18)

15. Web scraping
http://en.wikipedia.org/wiki/Web_scraping (fetched 15.1.2014 at 13:07)

16. Wärtsilä internal documentation.

17. Wärtsilä's history
http://www.youtube.com/watch?v=OBE-LGBMxqs (fetched 24.09.2013 at 20:00)