

Expansion of Sharepoint department portal with self-developed web part

Per Martin Klougart Mortensen

Kgs. Lyngby 2007
IMM - B. Eng. – 2007 - 10

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Summary

The title of my examination project is: “Expansion of Sharepoint department portal with self-developed web part”. It is the “second version” of the department portal I developed during by training period in MAN Diesel A/S.

In my training period I observed that the possibilities to customize table/list controls in Microsoft Sharepoint are limited. In the current Sharepoint versions no controls which make the user capable of searching directly in a list exist. One is more or less force to use the build-in search control which searches on the site itself and therefore not suitable for exactly this purpose.

With this problem in mind, I analyze in my project the possibilities of developing my own web part, which satisfy the user’s functional requirement of searching in a list.

In this project I will develop a web part to the department portal, which e.g. will make a FAQ (Frequently Ask Question) list more practical in Sharepoint.

As a supplement to the development of the web part I will furthermore analyze the possibility of using regression testing on the web part.

I will use the development process Unified Process and the software development tool Visual Studio 2005 with programming language C#, plus ASP.NET and Windows Sharepoint Service to develop the web part.

Keywords: Sharepoint, web part, Regression test

Resumé

Titlen på mit eksamensprojekt hedder: "Udvidelse af Sharepoint afdelingsportal med egen udviklede web part". Det er "anden version" af den afdelingsportal, jeg har udviklet for MAN Diesel A/S i min praktikperiode.

I min praktikperiode observerede jeg at mulighederne i Microsoft Sharepoint for at tilpasse en tabels/listes kontroller er begrænset. I de nuværende Sharepoint udgaver findes der f.eks. ikke en kontrol, som gør brugeren i stand til at søge direkte i en liste. Man er derfor tvunget til at bruge den indbyggede Sharepoint søge kontrol som findes på selve siden, og den er ikke særlig velegnet til lige præcis dette formål.

På baggrund af denne observation undersøger jeg i mit projekt mulighederne for at udvikle egne web parts som opfylder brugerens funktionelle krav.

I projektet vil jeg udvikle en web part til afdelingens web portal, som fx gør en FAQ (Frequently Ask Question) liste mere brugbar i Sharepoint.

Som supplement til udviklingen af en web part undersøger jeg i mit projekt ligeledes muligheden for at bruge regressionstest til at teste web parten.

Jeg benytter mig af udviklingsprocessen Unified Process og software udviklings programmet Visual Studio 2005 med programmeringssproget C#, samt ASP.NET og Windows Sharepoint Service til at udvikle web parten.

Nøgleord: Sharepoint, Web part, Regressionstest

Preface

This project is my leaving project at IMM DTU before fulfilling the requirements for acquiring the B.Eng. degree in engineering within the field of Information Technology. The task of this exam project has been made in cooperation with MAN Diesel A/S.

This project describes the development of a web part for MAN Diesel A/S department 9580's Sharepoint web portal and the use of regression testing on my web part.

This project consists of a report and a CD with the code for the web part which were written during the period 15.January-23.Marts 2007.

Lyngby, Marts 2007

Per Martin Klougart Mortensen

Acknowledgements

Throughout this project I have had the pleasure to discuss my project with many people. I want to thank all these people and particular the following.

- **Peter Falster** – my DTU adviser who guided me about the structure and content of this report, while reminding me of the delivery dates.
- **Jens Chemnitz** – my company adviser who participated in the early phases of the project with identifying the requirements of the web part.
- **Torkil Pedersen** – who guided me in the use of the UP development process, and for his many fine inputs on how I could improve my web part.
- **Lars Nikolajsen** – who always were available for questions and packed with good suggestions, when I needed it, during the implementation.
- **Per Klougart Mortensen (Senior)** – who read and commented my report and helped me reformulate and rewrite my sometimes cryptic and bad phrases and sentence.

Table of contents

SUMMARY	I
RESUMÉ	III
PREFACE	V
ACKNOWLEDGEMENTS	VII
TABLE OF CONTENTS	9
1 INTRODUCTION	13
1.1 Project Description.....	13
1.2 Project Scope	14
1.3 Project Delimitation	15
1.4 Abbreviations.....	16
1.5 Document Outline	17
2 PROJECT PLANNING	20
2.1 Development Process.....	20
2.2 Unified Process	20
2.3 Project Plan.....	24
2.4 Summary.....	24
3 USER REQUIREMENT SPECIFICATION	26
3.1 System Requirements.....	26
3.2 Use-Case Model	27
3.3 Supplementary Specification.....	31
3.4 Iteration Plan.....	32
3.5 Summary.....	32
4 TECHNOLOGIES	34
4.1 Sharepoint.....	34
4.2 Development Tools	42
4.3 Summary.....	43

5	ANALYSIS	45
5.1	Approaches	45
5.2	Use Cases.....	47
5.3	Conceptual-Model	61
5.4	Summary	62
6	DESIGN	64
6.1	From Analysis to Design.....	64
6.2	Patterns	64
6.3	Interaction Diagram.....	65
6.4	Class Diagram.....	67
6.5	Logical Architecture	68
6.6	Summary	69
7	IMPLEMENTATION	71
7.1	Implementing the Design	71
7.2	Summary	82
8	TEST	84
8.1	Purpose.....	84
8.2	VS05.NET TS Test Tools.....	84
8.3	Unit Test.....	85
8.4	Search Performance.....	87
8.5	Summary	89
9	DEPLOYMENT	91
9.1	Adding the Search Web Part to Sharepoint.....	91
9.2	Presentation of the Search Web Part.....	92
10	CONCLUSION	99
10.1	Purpose.....	99
10.2	Summing up.....	99
10.3	Evaluation	100
10.4	Perspective	101
10.5	Future Improvements	101

	11
<hr/>	
11 LITERATURE LIST	103
APPENDIX A	105
Guidance for Use Case Template	105
APPENDIX B	109
Interaction diagram Use Case 2	109
Interaction diagram Use Case 3	110
Interaction diagram Use Case 4	111
Interaction diagram Use Case 5	112
Interaction diagram Use Case 6	113
APPENDIX C	114
Test	114
APPENDIX D	115
Contracts	115

Chapter **1**

Introduction

1 Introduction

1.1 Project Description

MAN Diesel A/S has for the last couple of months been working on replacing the corporation's old intranet with a new and more up-to-date intranet solution. To do this MAN Diesel A/S has decided to use *Microsoft Office Sharepoint Portal Server 2003*, which is a Microsoft product that allows the creation of enterprise web portal solutions.

As with all web portals, it contains a number of web sites, which with the help of Sharepoint is very easy made. However the main reason MAN Diesel A/S has chosen Sharepoint to create their new corporation intranet with, is due to the fact that with the product comes a number of web applications also called web parts. Some of these web parts allow companies to keep track of documents, projects, tasks etc. Each web part can be configured to have certain behaviours.

The fact that Sharepoint provides these free web parts saves MAN Diesel A/S a lot of time and resources that otherwise would have been used on developing their own solutions.

Although Sharepoint in many ways provide MAN Diesel A/S with a much better alternative to the old intranet there are situations where some of the web parts aren't enough to solve a specific problem. In this situation it can be necessary to customize or create your own web part to solve the problem.

The purpose of this project is to create a web part which allows users to search directly in a list, which is one of the many web parts which Sharepoint provide.

A list is a web part, which can show data, i.e. list of projects e.g. which gives an overview of all ongoing projects in MAN Diesel A/S department 9580. Furthermore a list has a number of controls, which allow users to add, delete, edit, sort items and subscribe to an entire list among other.

As seen on figure 1, the project list contains a number of columns. Each columns contain some data which in this case show the status of a project, the phase the project is-in, which person who has the primary responsibility, the projects number and a link to the project web site.









9580 Projects					
Number	Project Site	Manager 9580	Status	Phase	Edit
Status : In Progress (6)					
20011	PIMS	MD-MAN\jic	In Progress	Design	
20006	CAD integration	MD-MAN\hsn	In Progress	Design	
20021	9580 Portal	MD-MAN\pmm	In Progress	Deployment	
20001	4 stroke load correction	MD-MAN\stj	In Progress	Development	
20051	QUARK Project Site	MD-MAN\gmm	In Progress	Development	
20015	OPERA project site	MD-MAN\jhn	In Progress	Definition	
Status : Yet to Start (2)					
20008	COM2	MD-MAN\tkp	Yet to Start	Request	
20056	IMAN-SAP Modification of interface	MD-MAN\ham	Yet to Start	Definition	
Add new item					

Figure 1 – Project list.

Although a list provides all these nice features it has come to our attention that when a list grows beyond a certain level, when users add items, it becomes increasingly difficult for a user to keep track of items.

Take a “Frequently Ask Question” list, such a list can contain hundred of items/answers and if a user has to find a specific item in this list it will take a long time. This is particular problematic since many of the lists eventually will grow beyond a size that is manageable for the user. Sharepoint does however provide a way to search on the portal, but this functionality is not very helpful, since the Sharepoint search often create a lot of “garbage” results, which you then need to filter from the result you actual can use.

Besides finding a solution to searching in a list, the purpose of this project is to look into regression testing techniques and use it on my web part. Regression testing is a method to capture regression bugs which occur during the implementation process.

In my report I have decided to make a section about Sharepoint since I have experienced that to completely understand the issue concerning lists, it might be necessary to have worked with Sharepoint before. This is why the report contains section 4.1 Sharepoint to give a more comprehensive introduction to Sharepoint.

1.2 Project Scope

The end result of this project consist of two elements; the first element, which is primary for this project, is a web part which can search in a Sharepoint list and give users a more friendly approach to finding items in the list.

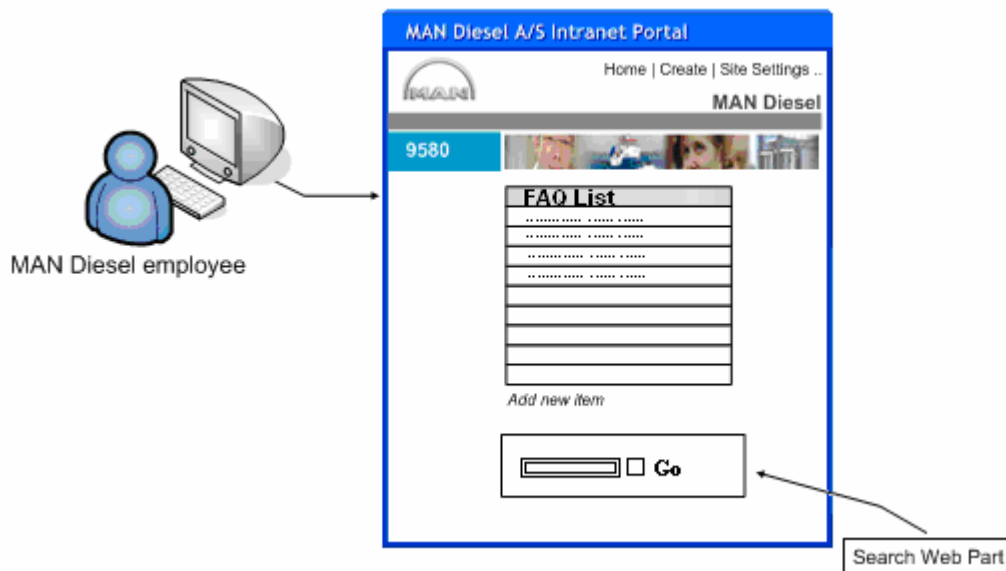


Figure 2 – Environment of Search web part.

The second element of this project is to examine regression testing techniques and to use some of them on my web part.

Although I take some space in the report to describe the Unified Process, it is not performed as a study itself, I merely write this section in chapter 2 - Project Planning to explain the reason for choosing this development process.

1.3 Project Delimitation

The main limitation for this project is the given time-frame. This fact will of course have influence on the final result. Therefore when gathering requirements for the web part I will determine which are most crucial for the project. With the most important requirements in mind I will create a functional prototype and gradually describe the other requirements in greater details and implement these on the prototype. More about this matter in chapter 2 - Project Planning.

Since regression testing can be quite a big topic of it own and the development of the web part is primary for this project, MAN Diesel A/S department 9580 and I have decided to delimit this part of the project. The task will be to look into Visual Studio 2005 Team Suite's test environment to see if the test tools provided can be used to uncover regression bugs.

1.4 Abbreviations

MD	-	MAN Diesel A/S
UP	-	Unified Process
DEPT 9580	-	Department 9580
FAQ	-	Frequently Ask Question
VS05.NET TS	-	Visual Studio .NET 2005 Team Suite
UML	-	Unified Modeling Language
SPS	-	Sharepoint Portal Server
WSS	-	Windows Sharepoint Services
CAML	-	Collaborative Application Markup Language

1.5 Document Outline

This project is organized as followed;

Abstract

Summarize the topic of this project.

1. Introduction

Introduces the reader to the project and explains the background for the user requirement specification found in chapter 3.

2. Project Planning

Introduces the reader to the development process I use in my project and a sketched plan for the project.

3. User Requirement Specification

Gives the reader an overview of functional and non-functional requirements for the project and provides the reader with a risk-list and iteration plan.

4. Technologies

Gives the reader a little introduction to Sharepoint and a description of the development tool I use.

5. Analysis

Each of the Use Cases is analyzed and possible solutions for creating a web part are explored. A conceptual model is created which provide reader with a high level abstract of the system.

6. Design

Use Case goes through realization and a design model is establish by creating interaction diagrams, a class diagram and a package diagram.

7. Implementation

Software classes are described in detail.

8. Test

Regression test tools in VS05.NET TS are explored and the test types are used on the web part. A performance test is made on the web part.

9. Deployment

This chapter is used to present the web part and a description of how the web part is added to the Sharepoint environment is included.

10. Conclusion

The project is summarized, future improvements and the perspective of the web part is analyzed.

11. Literature list

Contains reference to papers and literature.

Appendix

Contains additional text and information to the project.

CHAPTER 2

Project Planning

2 Project Planning

2.1 Development Process

In the earliest days of software development the process of developing a software solution only contained two steps; Write some code and fix the problems in the code, also known as - The code-and-fix model. As software solutions became increasingly bigger and more complicated this model made it clear that planning and preparation tasks in the early phases were needed. It became increasingly important from the start to define and describe a project in a well-defined software development process. Today there exist numerous of software development processes.

For this project I have chosen to build my development process upon the Unified Process (UP) or Rational Unified Process (RUP), the only distinctive between the two processes lies in the fact whether you are using Rational software or not. As I do not use Rational software for this project, I use the expression Unified Process or UP throughout the report.

2.2 Unified Process

The UP is a well-known iterative and incremental software development process which is Use Case driven and relies a great deal on the Unified Modeling Language (UML). However to fully define and understand it, one should think of it as an extensible framework which contains a number of artifacts that can describe disciplines of a project [3].

The UP is not a framework which one should followed out-of-the-box, it is more a framework which users can customize and adapt to their project, a best practices guide [1, page 18]&[2]. Neither is it my intention to follow all the workflow steps in UP step by step. At the end of the day this would only generate a lot of material which would add little of no value to my project.

2.2.1 Iterative and evolutionary development

The basic idea behind iterative and evolutionary development is developing a software system incrementally. Development is organized into series of small projects called iterations, where each of these iterations includes its own requirement, analysis, design, implementation and testing discipline.

The philosophy behind this idea lies in the fact that project evolves and changes during the development process because of unforeseen events. Consider the user requirements they often change during a project, due to unforeseen events, to reflect this reality it is

necessary with a development process that is flexible. There is no idea in trying to define all requirements and analyze them before moving on to the next discipline (design) in a project when it is highly likely that changes to the requirement will occur.

Such an approach where one discipline must be completed before moving on to a new discipline can cause significant problems later on in the project. Especially the fact that testing is left late in a project this can cause unexpected bugs or unforeseen risks to threaten the deadline or the entire project.

Therefore software systems should be incremented in smaller pieces according to iterative development [1, page 19] where the most important requirements are designed, implemented and tested first. This will also give the developer early on in the project an idea of significant risk and whether to continue the project. The possibility of implementing functionality in later iterations, if it isn't possible within the given time-frame, gives a flexibility that the Waterfall Model can't provide.

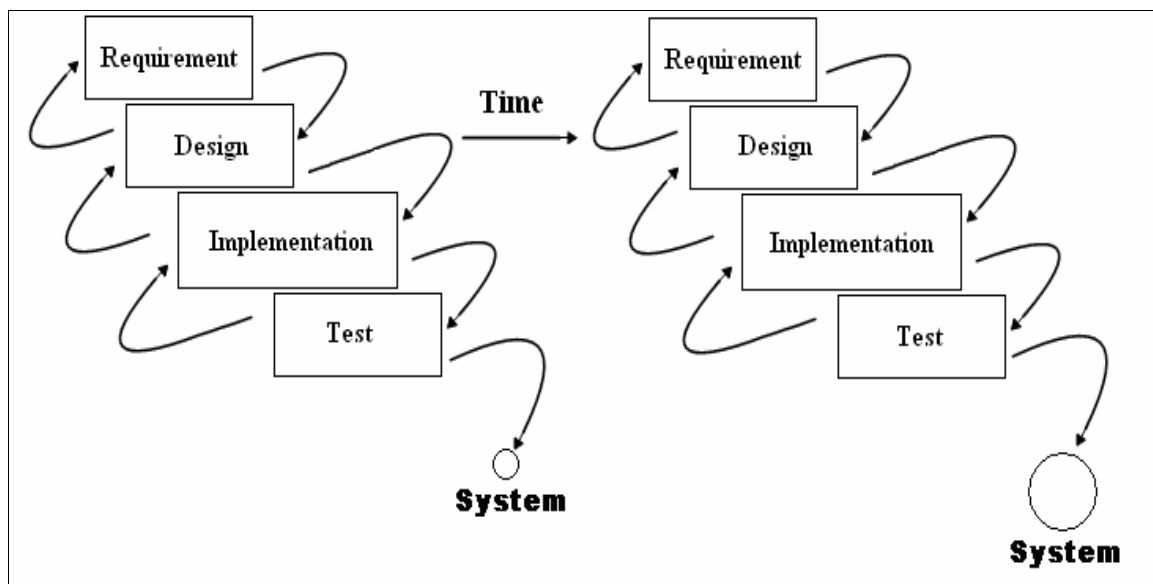


Figure 3 - Iterative development

After each of the iteration a review is made where latest changes to the project is updated. From the prototype created in the first iteration, a new iteration can begin where some of the changes to the project or lower risk requirement can be implemented, see figure 4. It is worth adding that these feedbacks/reviews are what separate evolutionary development from incremental development [1, page 19].

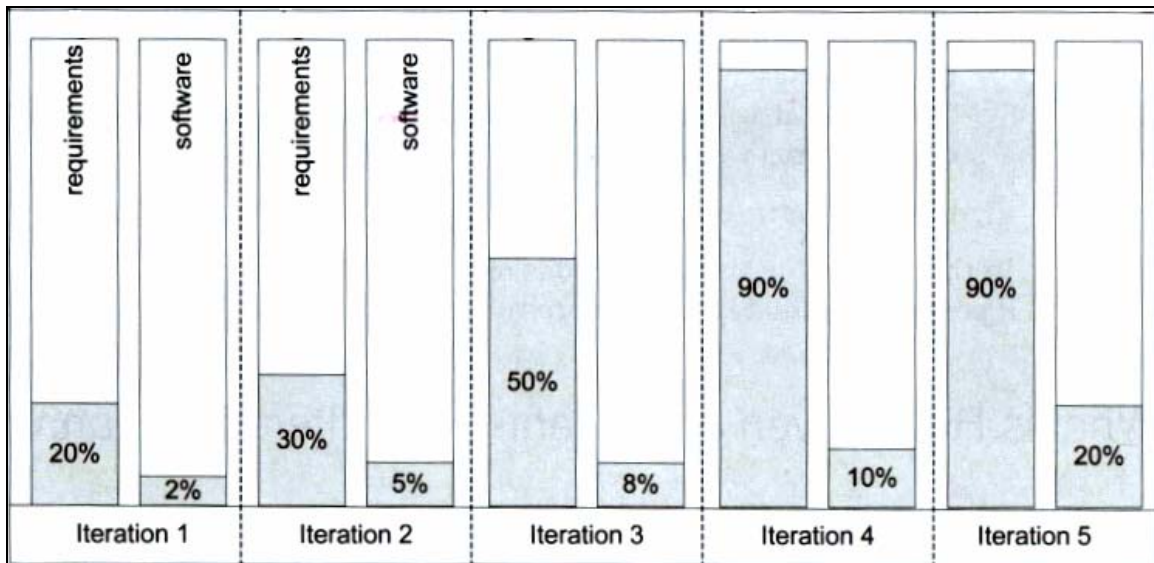


Figure 4 - Requirements evolve over iterations [1, page 28]

As with the Waterfall Model, UP contains a number of disciplines. However apart from this detail, in the UP all disciplines are more or less performed at the same time as seen on figure 5. Although some disciplines gets more attention in certain phases than others.

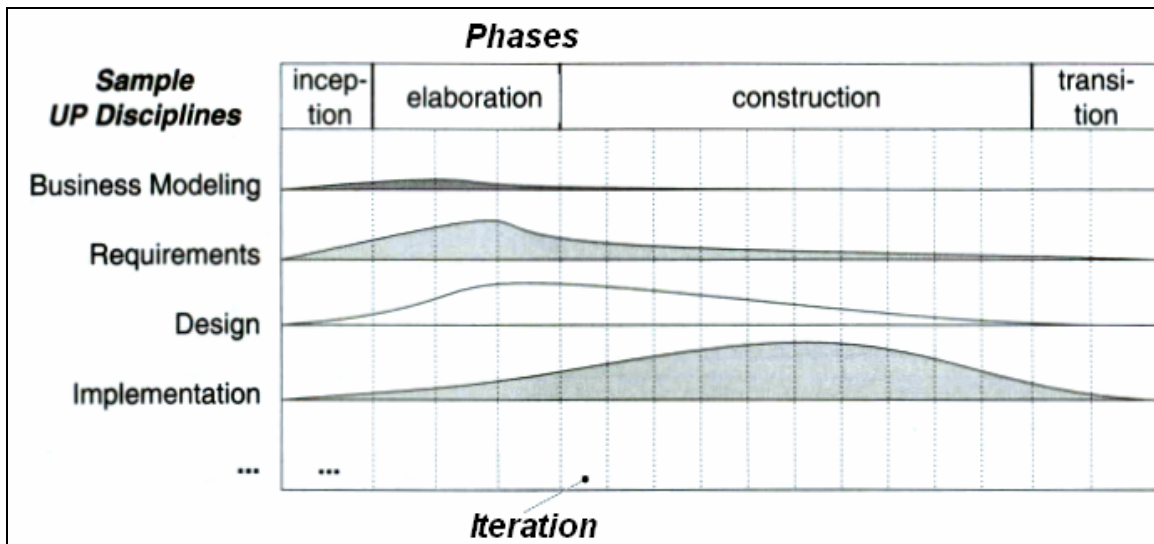


Figure 5 - UP disciplines [1, page 36]

It is my ambition to adapt the iterative and evolutionary methods which is implemented in the UP to my project plan, because as Martin Fowler state [1, page 17]:

“You should use iterative development only on projects that you want to succeed”

2.2.2 Unified Process phases

In an UP project all work and iterations is organized into four phases [1, page 33] & [2].

Inception is the phase where the project scope is established, where potential risk is estimated and where the most important requirements are captured by critical evaluating Use Cases. Furthermore an early architecture and estimation of the plan for the entire project is sketched.

Elaboration is the phase which perhaps is the most important one, where a project goes from being a low cost risk project to a high cost risk project with the possibility of major bureaucracy inactivity. In this phase a better understanding of risks is established which leads to a more stable architecture through the implementation of a prototype that exposes the top technical risk of the project. Furthermore a more reliable project plan is sketched which give a better idea of the amount of iterations it takes to complete the project.

Construction is normally the longest phase because it is where the remaining Use Cases and other requirements are described, implemented and tested in a series of iterations where each of the iteration brings a new release.

Transition is the last phase before the final production release. This phase mainly goes with testing, doing minor adjustment and ensuring that software is available to users.

Due to the fact that my project has a short time-frame on 10 weeks the phases will be rather small, and it is likely that not all requirements will be implemented which means that the project will only parse through the three first phases – Inception, Elaboration and Construction.

2.2.3 UML & Use Cases

The Unified Modeling Language is a standard diagramming notation which it used to visualize a reality that otherwise would be difficult to understand. Models drawn from the notation of UML help specify and construct a software system.

Use Cases are an UML notation which is ideal for finding functional requirements with. Furthermore it provides a useful tool for the software developer to help explain users sometimes complicated areas of a system under development.

2.3 Project Plan

The dates on the project plan below illustrates when phases of the project should be finished, however changes can still occur after this date. It's merely my estimation to keep the project on right track. The project starts 15 January 2007 and ends 23 March 2007.

During the project I must arrange at least one meeting with my DTU adviser which I intend to hold around halfway through the project.

Although the objectives of this project are defined before the project start and a project plan is more or less established I intend to use some of the first week on writing report about these matters.

Phases	Action	Date	Week
User Requirement Specification	Requirements are defined and reviewed.	02. February 2007	3
Analysis & Design	An analysis of problems is created and a system design is proposed. Analysis and design are reviewed.	09. February 2007	4
Implementation	The system design is implemented and code is reviewed	02. March 2007	7
Test	Test of the implementation should be finished.	09. March 2007	8
Deployment	System should be deployable and user guide created.	16. March 2007	9
Report	Report must be completed and ready for delivery.	23. March 2007	10
Meetings	Action	Date	Week
Colloquium	Talk to DTU adviser.	12. February 2007	5

2.4 Summary

In this chapter I stated that I will be working with the Unified Process. Furthermore I used the chapter to explain the principles behind the UP, which is an iterative and evolutionary process, and I establish a plan for the project.

CHAPTER 3

User Requirement Specification

3 User Requirement Specification

3.1 System Requirements

When defining the requirements of a system it is common [1, page 57] to divide them into two categories; Functional requirements or Non-functional requirements. Where functional requirements describe the behaviour of a system, i.e. a calculation or some other task, non-functional requirements describe every other type of requirements.

In the UP requirements are categorized according to the FURPS+ model. However it is optional which system of requirement-categorization one uses in an UP project. I intend to use most of the categories in the FURPS+ model.

In the FURPS+, according to [1, page 65] & [4], the 'F' category describe the functional requirements and the remaining "URPS+" categories describe the non-functional requirements.

- **Functional requirements**
 - **Functional**
- **Non-functional requirement**
 - **Usability**
 - **Reliability**
 - **Performance**
 - **Supportability**
 - **+**
 - **Implementation**
 - **Interface**
 - **Operations**
 - **Packaging**
 - **Legal**

I intend to describe the functional requirements with the Use Case Model and the supplementary requirements in section 3.3 Supplementary specification.

3.2 Use-Case Model

3.2.1 Identification of Use Cases

Seven Use Cases has been identified; six in the Search domain and one in the Admin domain.

The search domain consists of a main Use Case which is extended by five other Use Cases. '**Search in List**' is extended by '**Search Pattern**', '**Clear display**', '**Search in All Columns**', '**Search in Attached File**' and '**Select Columns to Search in**', because these Use Cases depend on what happens in the '**Search in List**' Use Case.

- **Search in List**
 - Make user capable of searching in a list – one column per. search.
- **Search Pattern**
 - Allow user to type more inputs in the search field.
- **Clear**
 - Clears the filter after a search and exposes all items again.
- **Search in Attached File**
 - Allows user to search in a file which is attached to an item.
- **Search in All Columns**
 - Allow user to search in all columns.
- **Select Columns to Search in**
 - Allow user to search in an undefined numbers of columns.

The admin domain consists of one Use Case – '**Setup Search**'.

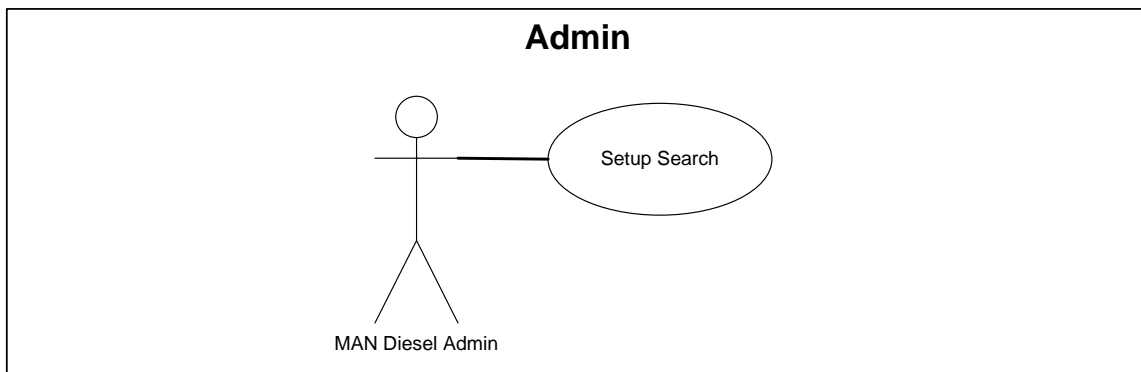
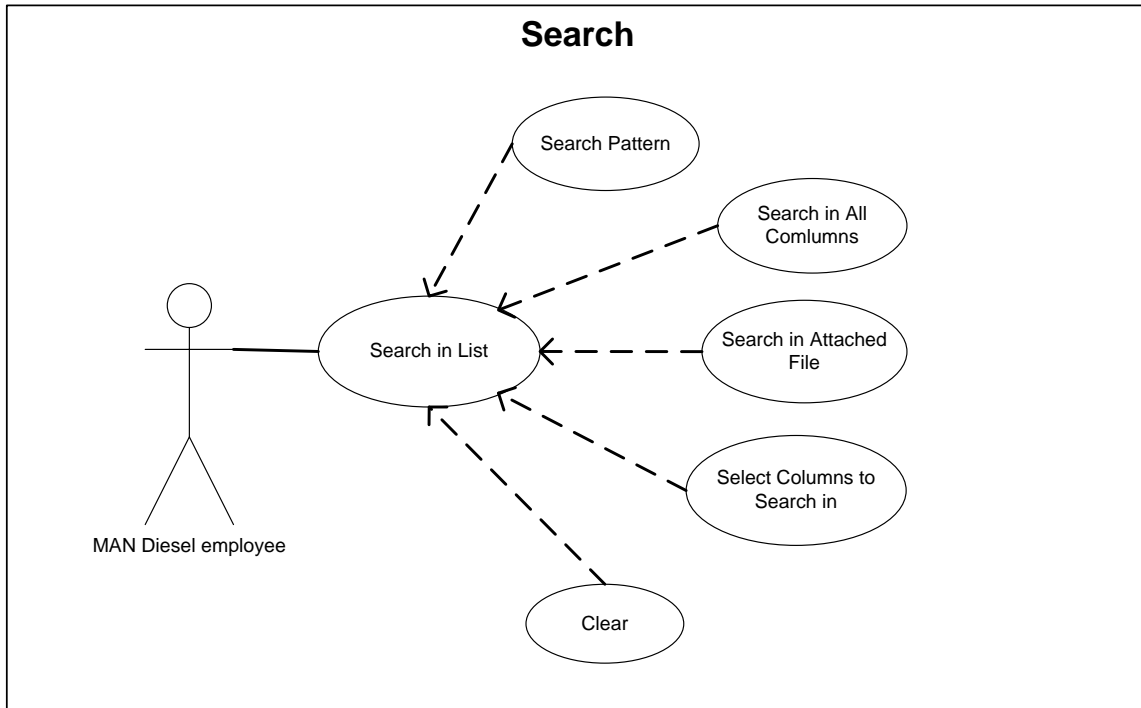
- **Setup Search**
 - Admin setup the search web part so it works on a Sharepoint list.

Use Cases are analyzed in greater detail in chapter 5 – Analysis.

3.2.2 Identification of actors

I have identified two actors in the system. The first actor is the MD employee who needs to be able to search, to find an item in a list, in a more friendly approach and the MD Admin who setup and configures the web part.

3.2.3 Use case diagram



3.2.4 Use Case ranking

To get an idea of which Use Cases to implement first I rank the requirements. I have rated the requirements after the following ratings:

1. Key requirement.
2. General requirement.
3. Nice-to-have requirement.

Use Case ID	Requirement	Rank	Risk	Comments
UC1	Search in List	1	High	This Use Case is the most important one since it is the basic core of the system and will exposes potential high risk.
UC2	Search Pattern	1	High	This Use Case is important because it will enable searching with multiple inputs. The risk involved implementing this feature I consider to be high because it is related with some uncertainties that can have major affect on the solution.
UC3	Search in Attached File	2	High	This Use Case enables the user to search in attached files and is considered to be an advanced feature. The risk implementing this feature is considered high because of similar uncertainties, as the 'Search Pattern' Use Case.
UC4	Clear	3	Low	This Use Case is a nice-to-have feature which will clear the filter after a search automatically. This Use Case shouldn't possess any risk and therefore low.
UC5	Setup Search	1	Medium	This Use Case is important because it is close related to 'Search in List'; admin needs to setup the web part before the employees can use it. Risk is considered moderate and shouldn't cause big problems.

Use Case ID	Requirement	Rank	Risk	Comments
UC6	Search in All Columns	2	Medium	This Use Case is considered to be a general requirement, which is an advanced feature on the search that will allow searching in all columns. The risk involved implementing this feature is considered medium.
UC7	Select Columns to Search in	3	Medium	This Use Case is a nice to have feature which allow admin to select particular columns. This risk is considered moderate, since implementing ' Search in All Columns ' should have exposed the most critical risk in this context.

3.3 Supplementary Specification

3.3.1 Usability

The term *Usability* implies how easy the tool is to use. When looking at the idea behind developing a Search web part it was to make the process of finding an item in a list more user-friendly. The answer for this problem was provided by the implementation of the search web part. However the web part itself must be easy to use, the user should without much knowledge intuitive master the web part.

3.3.2 Reliability

The term *Reliability* implies how stable a system is. The web part must work no matter what Sharepoint list it is chosen to search in and the search results must be accurate.

3.3.3 Performance

The term *Performance* implies how well a system performs. User shouldn't wait to long for a response from the web part.

3.3.4 Supportability

The term *Supportability* implies how easy it is too modify or maintain the typical usage or change scenarios of the web part. The web part must be able to work on other Sharepoint list without the need for additional programming.

3.3.5 Implementation

The term *Implementation* in this context implies to everything that surround the coding. The web part will be developed with Visual Studio 2005 .NET Team Suite and in C#. The platform for the development of web parts is the Windows Sharepoint Services and coding and comments must be in English. More about Windows Sharepoint Services in section 4.1 Sharepoint.

3.3.6 Interface

The term *Interface* in this context implies to which external item a system must interact with. Considering that a Sharepoint list can have attached file to it, these can be considered external items that the Search web part must be able to interact with.

3.3.7 Design

The term *Design* in this context implies which constrains there is in designing a system. Which in the context of the web part, means that the look and feel of the web part should fit into the environment of the Sharepoint site? Buttons and colours shouldn't be entirely opposite of the theme of the Sharepoint sites.

3.4 Iteration Plan

The iteration plan only tries to describe a single iteration in advance. The first iteration starts after the Inception phase. As the project evolves from iteration to iteration new iterations will be added in the iteration plan. During the project changes to the plan might occur because of new insight, some requirement may change rank. The iteration plan should be seen as the iterations it has taken to implement features of the search web part.

Iteration	Iteration goal	Date
1	Create the prototype of the search web part. It involves implementing UC1, UC2 & UC5.	29. January 2007 – 9. February 2007
2	Implement some of the advanced features. UC3 & UC6	12. February 2007 – 27. February 2007
3	Implement the nice-to-have features. UC4	2. March 2007 – 4. March 2007

3.5 Summary

In this chapter I established the requirements for the search web part. Each of the requirements were rank according to risk involved implementing them and how important they are for the project.

CHAPTER 4

Technologies

4 Technologies

4.1 Sharepoint

4.1.1 What is Sharepoint?

It is a term that can refer to three Microsoft products: Sharepoint 2001, Sharepoint 2003 and Sharepoint 2007. The product that I have been working with is Sharepoint 2003. I therefore intend to explain what Sharepoint is from the perspective of the Sharepoint 2003 release.

Sharepoint 2003 or **Microsoft Office Sharepoint Portal Server 2003** is a portal based collaboration and document management system that is based on the Windows Sharepoint Services (version 2) platform, which is a free Windows server component.

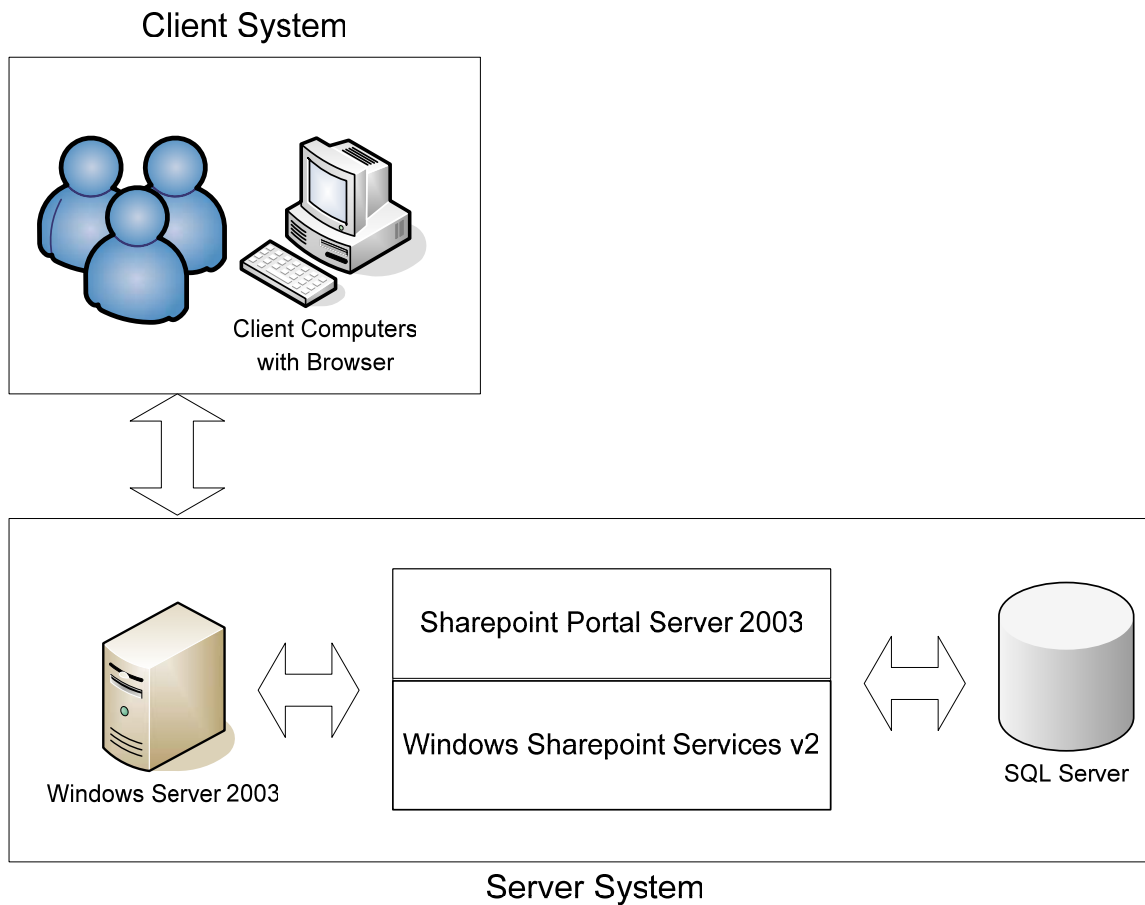


Figure 6 – The Sharepoint 2003 environment

Basically what Sharepoint does is that it provides a tool for companies to easy and quickly establish their own web portal, on the World Wide Web, which can provide all kind of services for its visitors.

Sharepoint 2003 consists of two components: Sharepoint Portal Server 2003 (SPS) and Windows Sharepoint Services v2 (WSS). Both components provide a collection of services for Microsoft Windows Server 2003.

It is not my intention to explain all the services in WSS and SPS but just to give a short introduction to some of the most important ones. More can be found on the following web sites.

WSS - <http://www.microsoft.com/technet/windowsserver/sharepoint/v2/default.mspx>.

SPS - <http://www.microsoft.com/office/sharepoint/prodinfo/default.mspx>.

4.1.2 Windows Sharepoint Services v2

One of the services that WSS provides is the possibility of creating web part pages.

Web part page & web part

A web part page is an average web site which contains small independent web applications.

With WSS come a number of ready-to-use web parts such as a list, document library, discussion board, calendar and survey.

Sharepoint allow the users to very quickly create a web site by simply dragging and dropping web parts onto the web site without the user having any programming experience what so ever. In this way is it possible for the user to create an advanced web site. A web part might be a calendar which show a users appointment, another might be a graph showing sale figures, a third could be a web part showing a list of news, it could even be a web part showing today's weather.

The best way to understand how web parts works, if not familiar with this phenomenon, is taken a visit to www.google.dk. Google has something called a Personal site where it is possible to add a lot of web part to your own personal site. The following URL would normally show the personal site <http://www.google.dk/ig?hl=da>.



Figure 7 – Google and web parts

Every web part in Sharepoint has a tool pane with some tool parts which in design mode allow the user to modify the appearance of the web part. Height and width of the web part among other can be changed without the user having to be a programmer.

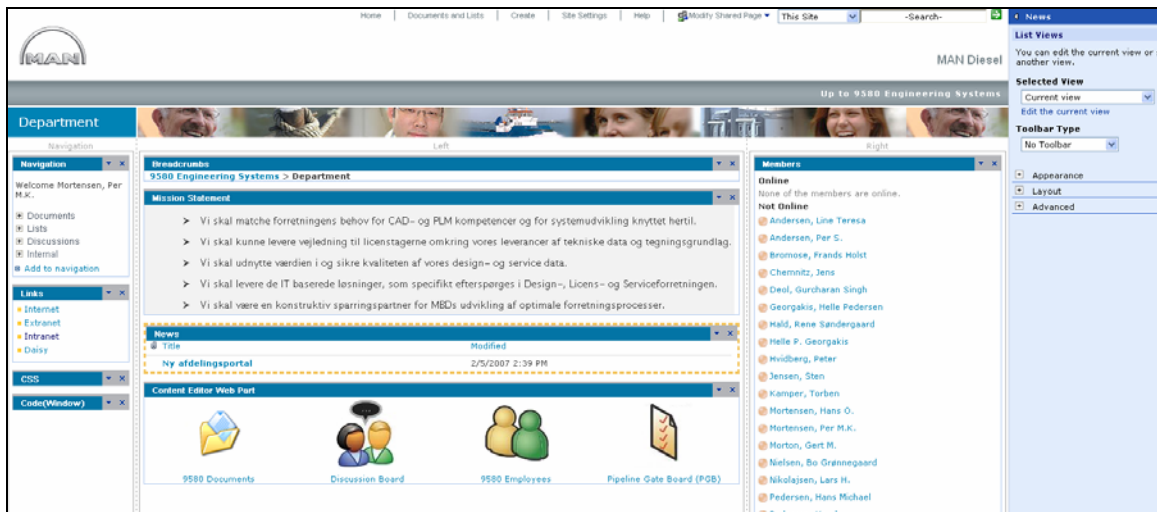


Figure 8 – In design mode, the tool pane in the right side of the screen.



Figure 9 – Tool pane of news list/web-part

List

A list is a web part which basically is used to show some data. In Sharepoint a number of pre-defined list have been made.

- Link
- Announcement
- Contact
- Event
- Task
- Issue

Tasks						
Title	Assigned To	Status	Priority	Due Date	% Complete	Edit
Test1		Not Started	(1) High	04-12-2006 11:30	20%	
Test2 ! NEW		Not Started	(2) Normal			
Test3 ! NEW		Not Started	(2) Normal			

[Add new item](#)

Figure 10 - Task List

On figure 10 is a task list which not shockingly can be used to keep track of tasks. The task list contain a number of columns where it is possible to see what priority the task has, what status it has, which date the task is due to, how much of it is completed and so on.

Besides the basic features of adding, deleting and editing an item, a list can also have multiple views. Figure 11 shows the page where all the views for a specific list can be

found. A view can be changed so that it sorts the list in a special way or filter the list so only some items appear in the list. It is practical with all these views because you can have a view where only your tasks appears, another view which show all items and a third view which only show high priority tasks.

Use the Events list to keep informed of upcoming meetings, deadlines, and other important events.

New Item | Filter | Edit in Datasheet

Title	Location	Begin
rtrtrr		1/29/2007 12:00 AM
?ToolPaneView=2		1/29/2007 12:00 AM
hund og kat		1/29/2007 12:00 AM
hund og mus		1/29/2007 12:00 AM
bnhfgf		2/6/2007 12:00 AM

Figure 11 – Page which show all the task list views

When you want to create a new view in the task list you go to the page which shows all the task views and press on the link ‘Modify settings and columns’ which can be found under ‘actions’ – see figure 11. This will provide the user with a site where views, columns, list permissions and many other things can be customized.

Customize 9580 Projects
 Use this page to change the design of the list, such as its name, security settings, and columns. You can also create or change views of the list.
[Go Back to "9580 Projects"](#)

General Settings
 General settings of this list include its name, description, and security. Current general settings of this list:

Title:	9580 Projects
Web Address:	http://inside.manbw.dk/EngineeringSystems/9580Projects/Lists/Projects/AllItems.aspx
Description:	List of 9580 Projects
On Quick Launch Bar:	Yes
Attachments enabled:	Yes
Content Approval Required:	No

- Change general settings
- Save list as template
- Change permissions for this list
- Delete this list
- Select a portal area for this list

Columns
 A column stores information about each item in the list. Columns currently in this list:

Column (click to edit)	Type	Required
Number	Single line of text	
Name	Single line of text	✓
Status	Choice	✓
Phase	Choice	
Manager 9580	Lookup	
Manager MD	Lookup	
System	Lookup	
Domain	Lookup	
Start	Date and Time	
End	Date and Time	
Budget	Currency	
Project Site	Hyperlink or Picture	

- Add a new column
- Change the order of the fields

Views
 A view of a list allows you to see a particular selection of items or to see the items sorted in a particular order. Views currently configured for this list:

View (click to edit)	Default View
All Items	✓
Front Page	

- Create a new view

Figure 12 – Customization site for columns and views

In the Views section two views appear (figure 12): All Items and Front Page view. It is possible to create a new view on this site.

By pressing on one of the created views I will get a new site where the selected view can be modified. All views is made up of a SPview class which among other control how the view sort, filter, group and which columns appear on the list view.

9580 Projects: Edit View
To customize this view further, use a Web page editor compatible with Windows SharePoint Services.

Name
Type a name for this view of the list. Make the name descriptive, such as "Sorted by Author", so that site visitors will know what to expect when they click this link. [Show me more information.](#)

Columns
Select or clear the check box next to each column you want to show or hide in this view. To specify the order of the columns, select a number in the **Position from left** box.

Display	Column Name	Position from Left
<input checked="" type="checkbox"/>	Number	1
<input checked="" type="checkbox"/>	Project Site	2
<input checked="" type="checkbox"/>	Manager 9580	3
<input checked="" type="checkbox"/>	Status	4
<input checked="" type="checkbox"/>	Phase	5
<input checked="" type="checkbox"/>	Edit (link to edit item)	6

Sort
Select up to two columns to determine the order in which the items in the view are displayed. [Show me more information.](#)

First sort by the column:
None

Show items in ascending order (A, B, C, or 1, 2, 3)

Show items in descending order (C, B, A, or 3, 2, 1)

Then sort by the column:
None

Show items in ascending order (A, B, C, or 1, 2, 3)

Show items in descending order (C, B, A, or 3, 2, 1)

Filter
Show all of the items in this view, or display a subset of the items by using filters. To filter on a column based on the current date or the current user of the site, type **[Today]** or **[Me]** as the column value. [Show me more information.](#)

Show all items in this view

Show items only when the following is true:

Show the items when column: Status

is equal to: Yet to Start

And Or

When column: Status

is equal to:

Figure 13 – Modified a specific view

4.1.3 Sharepoint Portal Server 2003

SPS is built on top of the WSS which means that all features in WSS are available in the SPS. However SPS provides some additional features which can not be found in WSS.

The main purpose of the SPS is to create the portal and to connect the web part pages which are created with the WSS.

Some of the features that SPS provide are the possibility of having personal sites, searching functionalities that allow to search on sites that resides outside the site from which the search was invoked and a Single Sign On service which allow web parts to automatically sign on to its enterprise application without prompting the user for password.

4.1.4 Collaborative Application Markup Language

Collaborative Application Markup Language (CAML) is a XML based language which is used by Sharepoint to define all aspect of a Sharepoint site from link structure to available web parts. [6]

It is the CAML language which is used to present data in Sharepoint, through the use of query-strings against Sharepoint list data, so that items in the list can be found and displayed dynamically based on a variety of criteria's.

A CAML query could look this way;

```
<Where>
  <Or>
    <Contain>
      <FieldRef Name="Title" />
      <Value Type "test" />
    </Contain>
    <Contain>
      <FieldRef Name="Title" />
      <Value Type "test2" />
    </Contain>
  </Or>
</Where>
```

Such a query-string would filter the list so that the items in the column 'Title' with name 'test' and 'test2' would be displayed.

More about CAML and CAML queries can be found on the following sites.
http://en.wikipedia.org/wiki/Collaborative_Application_Markup_Language
<http://msdn2.microsoft.com/en-us/library/ms467521.aspx>

4.2 Development Tools

For this project I intend to use the development tool called Microsoft Visual Studio 2005 .NET Team Suite (VS05.NET TS) which is an environment for creating windows and web-applications that is executed on the Microsoft .NET Framework 2.0.

VS05.NET TS environment allow the user to code in programming language like C#, C++, Visual Basic or J#.

For this project I use C# since it is the preferred language in MD DEPT 9580.

When I use VS05.NET TS it is because it provides an easy way to create a HTML user interface and because it has a good test environment.

Instead of spending a lot of my time on creating the user interface, VS05.NET TS environment allow me to very easy drag the needed controls on to my web part and spend more time on other issues concerning the search web part.

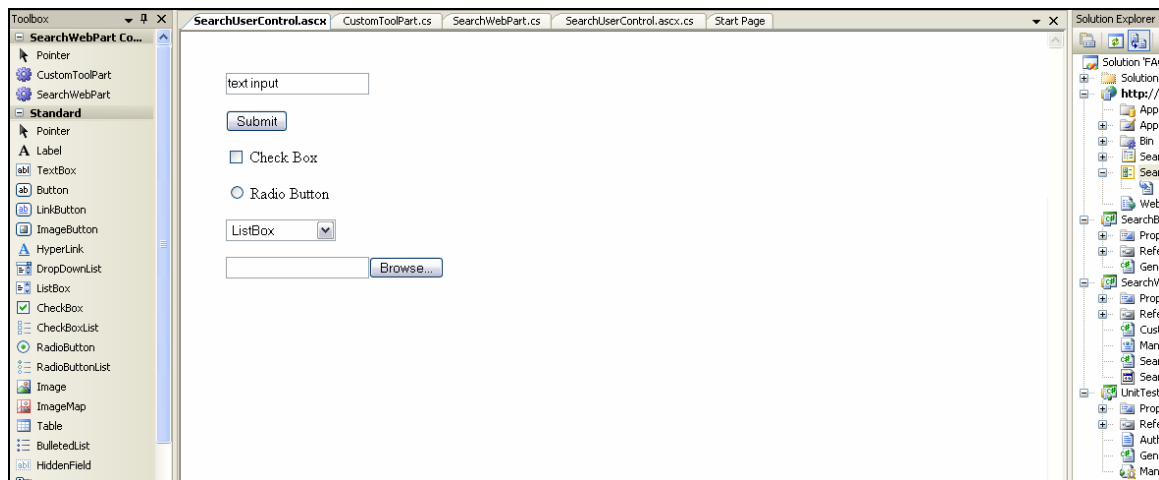


Figure 14 – Creating user interface

The test environment in VS05.NET TS has integrated a number of test types such as unit, load, web and manual test.

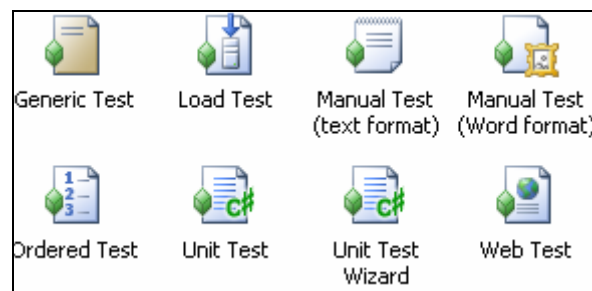


Figure 15 – Test types

4.3 Summary

I used this chapter to give a short introduction to Sharepoint since the issue I deal with in my project can be complicated to understand if the reader has no experience with Sharepoint. A short introduction to the development tool I intend to use in my project can be found.

CHAPTER 5

Analysis

5 Analysis

5.1 Approaches

I see three possible approaches for creating the search web part. Each of these approaches has its strengths and weaknesses. However I will describe each of them to find the right approach for this project.

I have chosen to define the three approaches in the following three categories;

- *Data grid*
- *Connection*
- *View*

5.1.1 Data grid

This approach was the first I heard of since it was used in an earlier project in DEPT 9580. The idea behind the data grid approach is to retrieve the data from the Sharepoint list and create your own data grid where the data can be manipulated without having to be concerned about how Sharepoint works.

As I mention in the chapter 4 – Technologies, all aspect of a Sharepoint site is made up of CAML. CAML is a language based on XML elements. So all I need is to get the XML for the list, I want to search in, and sort or filter this data and then show it in my own data grid.

The main advantages behind this approach lies in the fact that you are not affected by the limits of the Sharepoint architecture. You can create your own data grid which you can do what ever you want with.

The biggest drawback is that it is time-consuming to make a data grid which can do half of what the existing Sharepoint list can do.

5.1.2 Connection

This approach evolves using a technique in Sharepoint where you can connect web-parts/lists with other web-parts/lists. The idea behind the connection approach is to create a search web part which uses a connection interface that is capable of connecting to the Sharepoint list. There are four connection types;

- ICell - Provider provides a single value to Consumer
- IRow - Provider provides a single or multiple rows of values to the Consumer.
- IList - Provider provides an entire list to the Consumer.

- IFilter Provider provides a filter value to the Consumer.

The search web part will act as a provider web part which provides information to the Sharepoint list which is an 'IRow' consumer.

The main advantages are that the original Sharepoint list to present data is used and all of the features that comes with it.

The biggest drawback is that there exist different connections types and that these connection types can not be changed on the consumer Sharepoint list. This will affect how advanced the search web part can become.

5.1.3 View

This approach evolves using the Sharepoint list views. During the creation of a view it is possible to filter, sort and group items in a list. What happens is that when users make changes to the view a query-string is generated that sets the values of the views (SPview) query property [5]. The SPView class represents a view of the data contained in a list on a Sharepoint site.

It is possible to programmatically write to this query property and perform the same actions as when user creates CAML queries through the dialog interaction.

The main advantage of the view approach is that it is possible to work with the original Sharepoint list and use all its features.

The biggest drawback is that the search web part will only work on the page which displays all the views since the view's query is access from this site.

5.1.4 Choosing the right approach

The data grid approach would have worked fine and been a good solution. However it was very early on in the project decided that a potential solution should use the existing Sharepoint lists. The time used on creating a data grid, which would give the same features as a Sharepoint list, would only have taken away time from what was consider the scope of this project.

An approach with connecting web parts with web parts I decided to abandon after a couple of weeks, this solution turn out to be complicated and full of problems due to way Sharepoint works. A potential solution would have been too small, and not very user-friendly, since it would only have been possible to search in one column.

The approach I have decided to go a head with is the view approach. It will allow me to create a search web part that allow user to dynamically edit the *SPView.Query* property.

The solution will be using the original Sharepoint lists to display results after a search and fulfil the users requirement of a solution with allow the user to search in all columns and in attached files.

5.2 Use Cases

In chapter 3 – User Requirement Specification I establish a number of Use Cases. To analyze these Use Cases I follow the template which DEPT 9580 has developed for making Use Cases. The guidance for the template can be found in Appendix A.

5.2.1 UC1 - Search in List

Use Case ID: Domain	UC1 - Search in List Search		
Created By:	PMM	Last Updated By:	PMM
Date Created:	23. January 2007	Date Last Updated:	06. February 2007

Actors:	MAN Diesel employee
Description:	This Use Case represents the basic core of the Search web part. The web part should be able to search in a custom made list (FAQ list etc.). The search web part should only work on one list and the core search itself is in one column. An important detail is that search results should be displayed in the Sharepoint list since this is preferred by user.
Trigger:	User needs to find a specific item in the list.
Pre-conditions:	Web Part is visible
Post-conditions:	
Normal Flow:	<ol style="list-style-type: none"> 1. User enter search input in the search text box 2. User selects a column to search in. 3. User hits the search button 4. System generates a query-string 5. System update Sharepoint view query. 6. The Sharepoint list is filtered, and user gets search results.
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	High
Frequency of Use:	Daily
Business Rules:	
Special Requirements:	See section 3.3 Supplementary specification
Assumptions:	Search web part must be attached to a Sharepoint list.
Notes and Issues:	

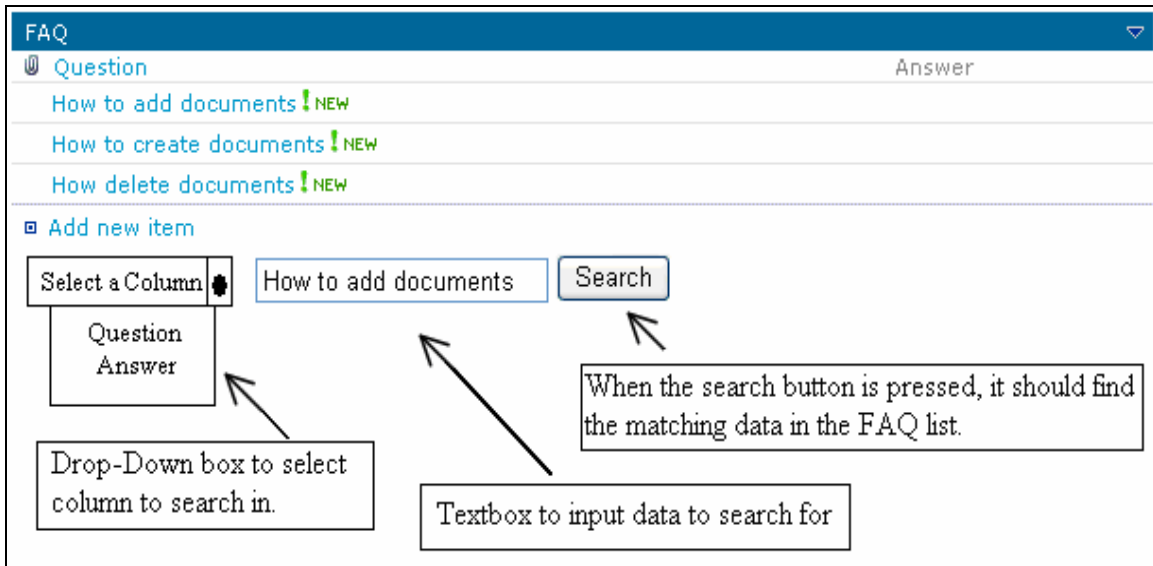


Figure 16 – How I imagine the interface of the simple search

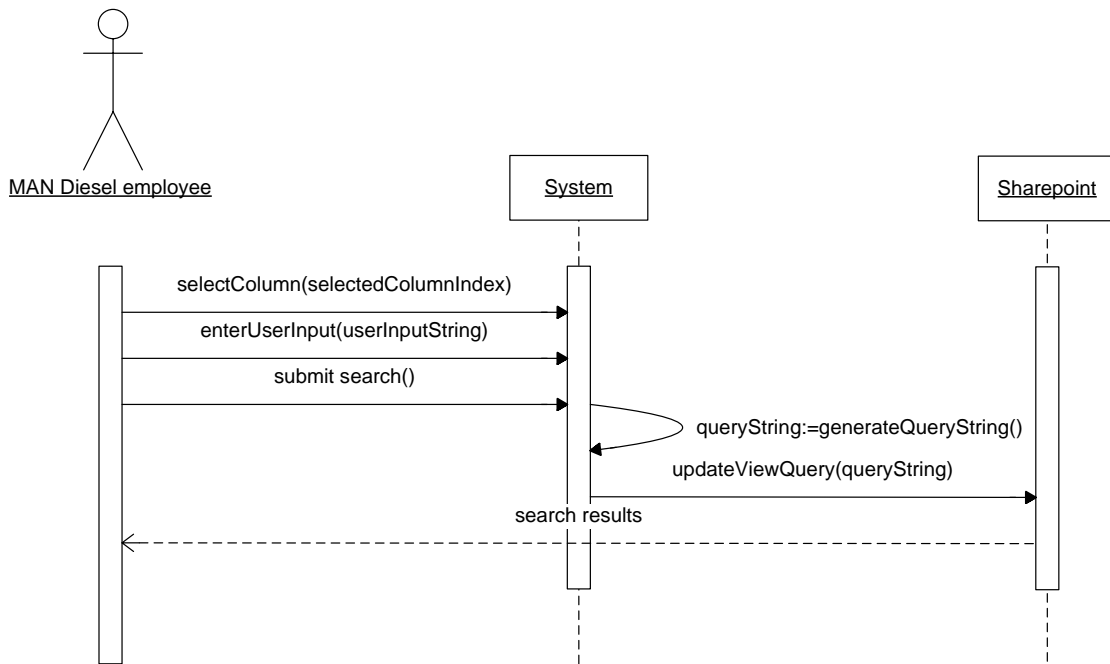


Figure 17 – System sequence diagram of Use Case 1

The system sequence diagram show input and output events related to the system.

5.2.2 UC2 - Search Pattern

Use Case ID: Domain	UC2 - Search Pattern Search		
Created By:	PMM	Last Updated By:	PMM
Date Created:	26. January 2007	Date Last Updated:	06. February 2007

Actors:	MAN Diesel employee
Description:	When entering data in the search textbox it shouldn't be necessary to fill-in the hold word to get a match. Selecting a search condition in a drop-down list should allow the user to retrieve search results that contain parts of the word or start with it. Furthermore it should be possible to enter more than one search input in the search textbox. By separating the words with the '+' sign more inputs can be added to search text box.
Trigger:	User needs to have more than one search input or be able to search on parts of the word
Pre-conditions:	Web Part is visible
Post-conditions:	
Normal Flow:	<ol style="list-style-type: none"> 1. User enter multiple search inputs by using the '+' sign. 2. User selects a column. 3. User selects a search condition, deciding which user input should be retrieve. 4. User hits the search button. 5. System split the user input string, so it can be used to generate a valid query-string. 6. System generates a query-string 7. System update Sharepoint view query. 8. The Sharepoint list is filtered, and user gets search results.
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	High
Frequency of Use:	Daily
Business Rules:	
Special Requirements:	See section 3.3 Supplementary specification
Assumptions:	Search web part must be attached to a Sharepoint list.
Notes and Issues:	

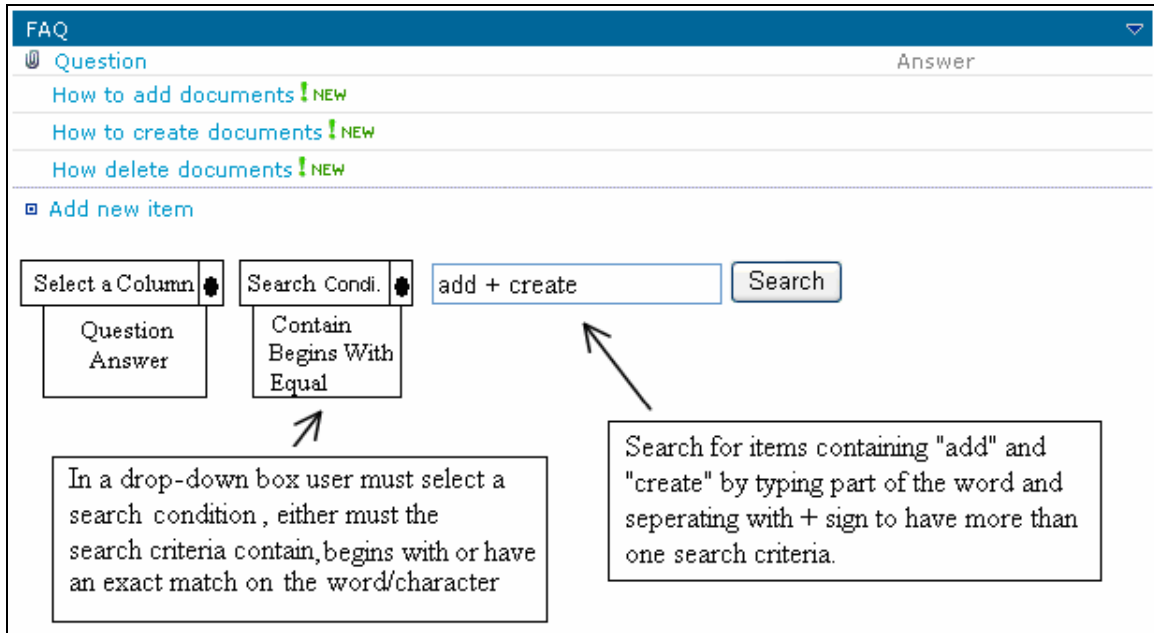


Figure 18 – How I imagine the interface when search pattern is added

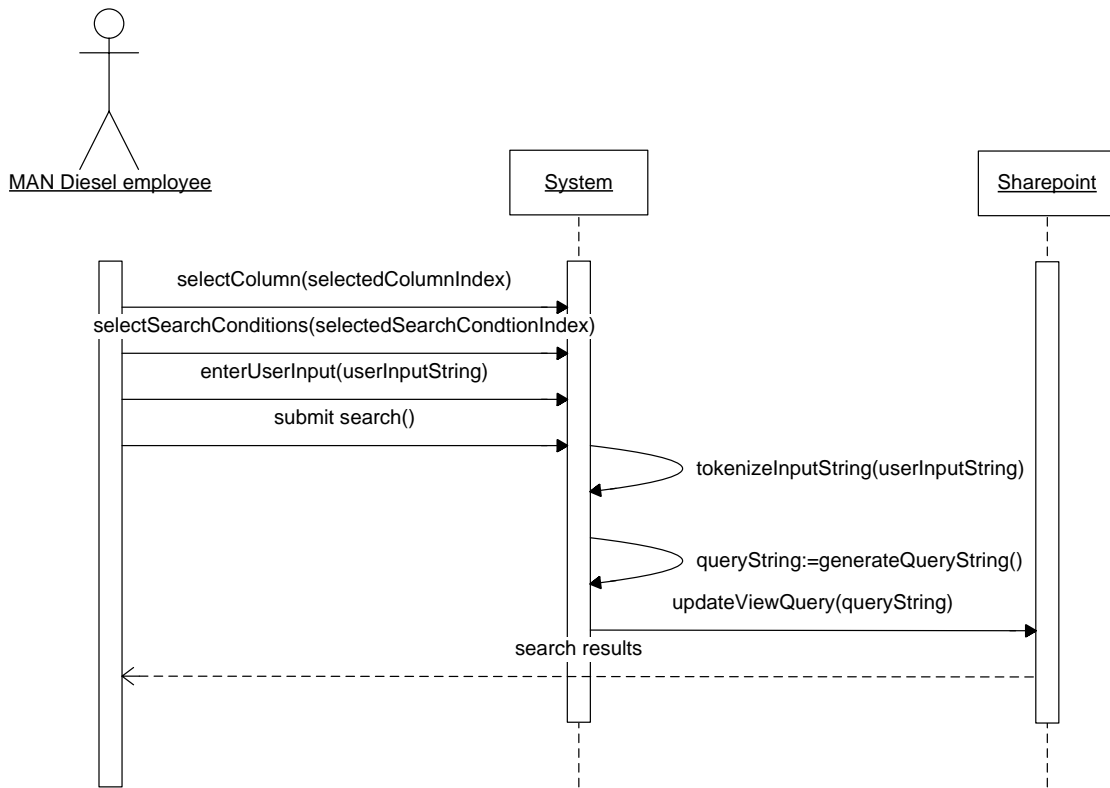


Figure 19 – System sequence diagram of Use Case 2

5.2.3 UC3 - Search in Attached File

Use Case ID: Domain	UC3 – Search in Attached File Search		
Created By:	PMM	Last Updated By:	PMM
Date Created:	10. February 2007	Date Last Updated:	18. February 2007

Actors:	MAN Diesel employee
Description:	A Sharepoint list can have attached files. It should be possible to search in these files.
Trigger:	User need to search in files.
Pre-conditions:	Web Part is visible
Post-conditions:	
Normal Flow:	<ol style="list-style-type: none"> 1. User enter search input in the search text box 2. User checks the small box below the search box, to search in the attached files. 3. User hits the search button. 4. System split the user input string so it can be used to generate a valid query-string, however only if user-input contained multiple search inputs. 5. System generates a query-string 6. System update Sharepoint view query. 7. The Sharepoint list is filtered, and user gets search results.
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	Medium
Frequency of Use:	Daily
Business Rules:	
Special Requirements:	See section 3.3 Supplementary specification
Assumptions:	Search web part must be attached to a Sharepoint list.
Notes and Issues:	

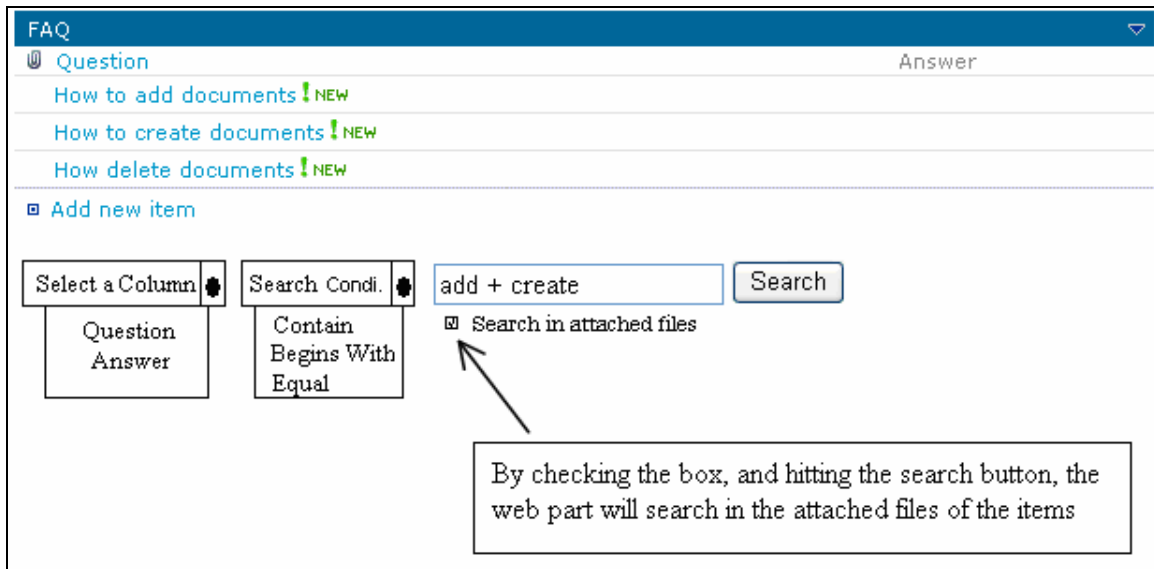


Figure 20 – How I imagine the interface when search in attached files is added.

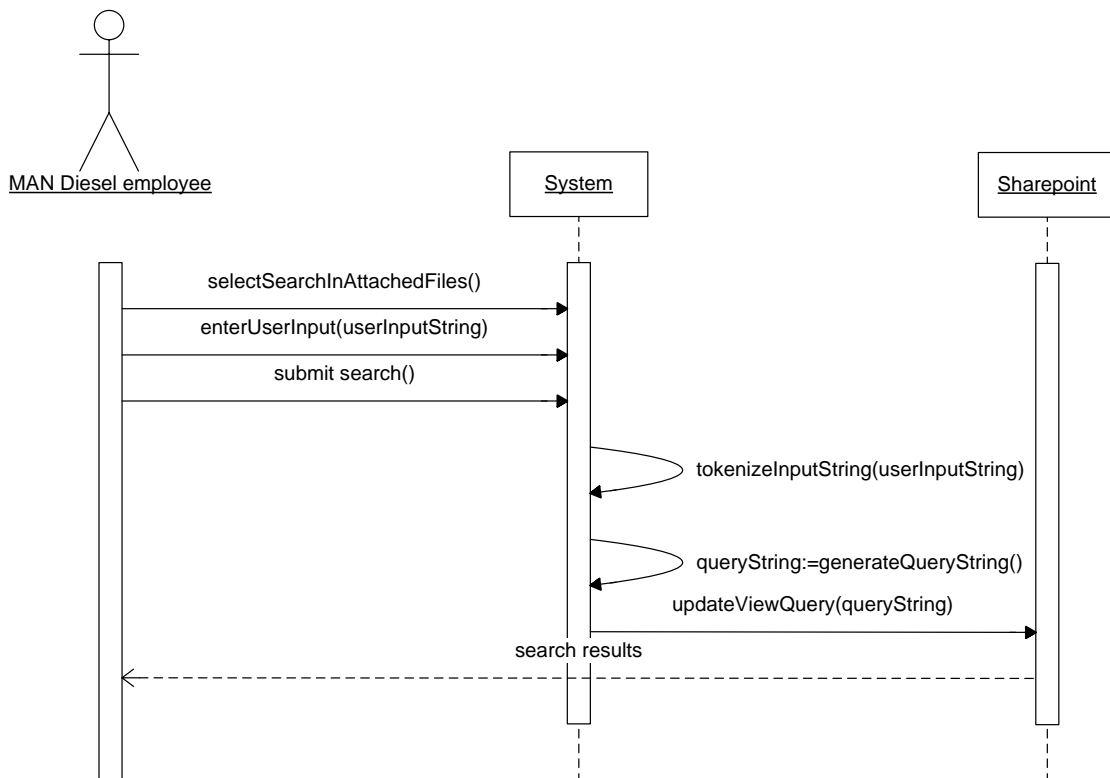


Figure 21 – System sequence diagram of Use Case 3

5.2.4 UC4 - Clear

Use Case ID: Domain	UC4 – Clear Search		
Created By:	PMM	Last Updated By:	PMM
Date Created:	24. February 2007	Date Last Updated:	28. February 2007

Actors:	MAN Diesel employee
Description:	When searching in the Sharepoint list what really happens is that a filter is created, therefore a button that clears the filter on the view and allows user to see all items in the list again is needed.
Trigger:	To perform a new search user must clear the lists filter
Pre-conditions:	Web Part is visible
Post-conditions:	
Normal Flow:	<ol style="list-style-type: none"> 1. User hits the clear button 2. System generates a query-string that is empty which clears the filter on the Sharepoint view. 3. System update view query. 4. The Sharepoint list filter is cleared, and all items in the list are visible.
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	Low
Frequency of Use:	Daily
Business Rules:	
Special Requirements:	See section 3.3 Supplementary specification
Assumptions:	Search web part must be attached to a Sharepoint list.
Notes and Issues:	

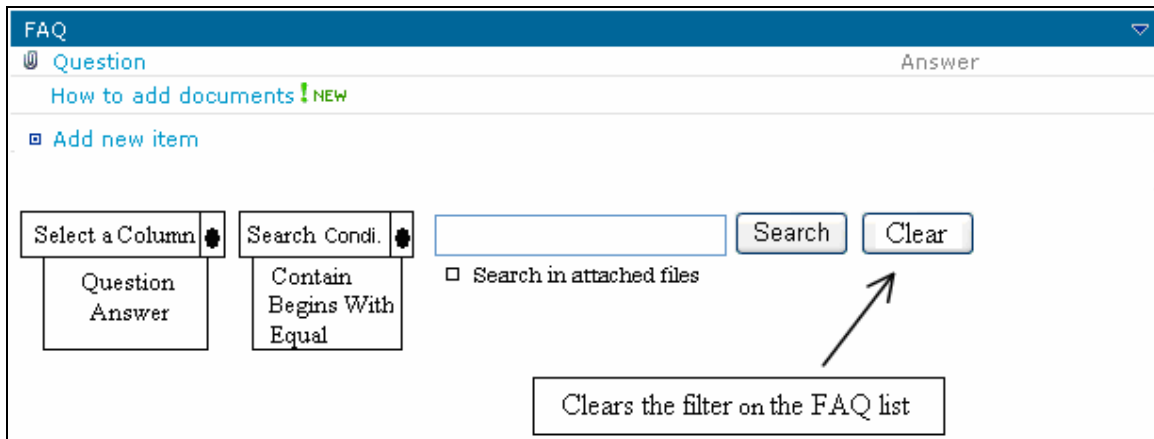


Figure 22 – How I imagine the interface when search the clear button is added

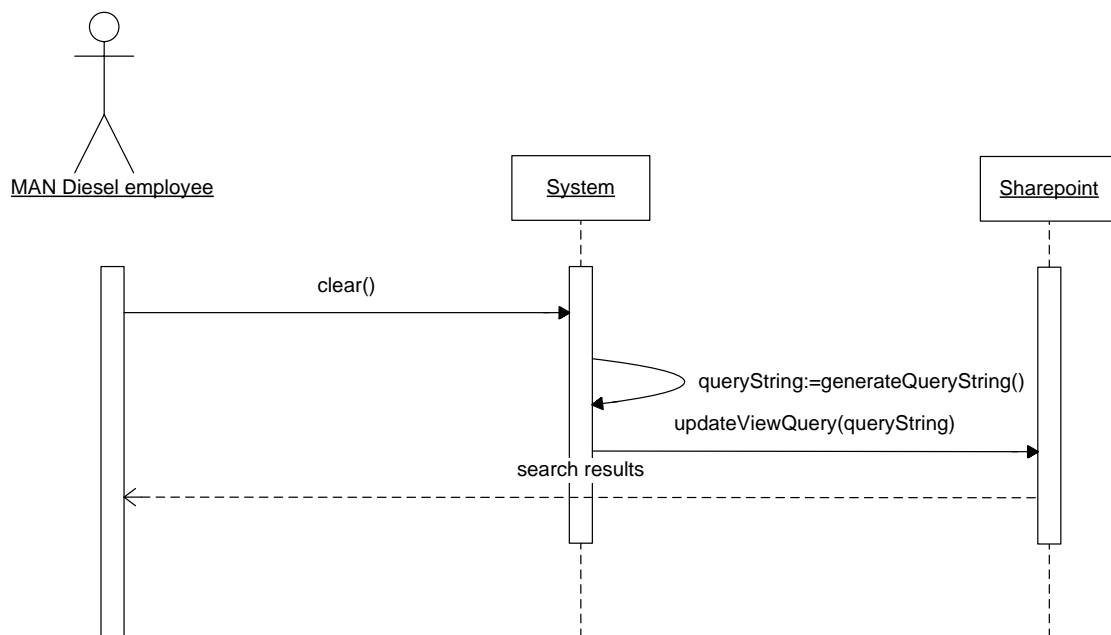
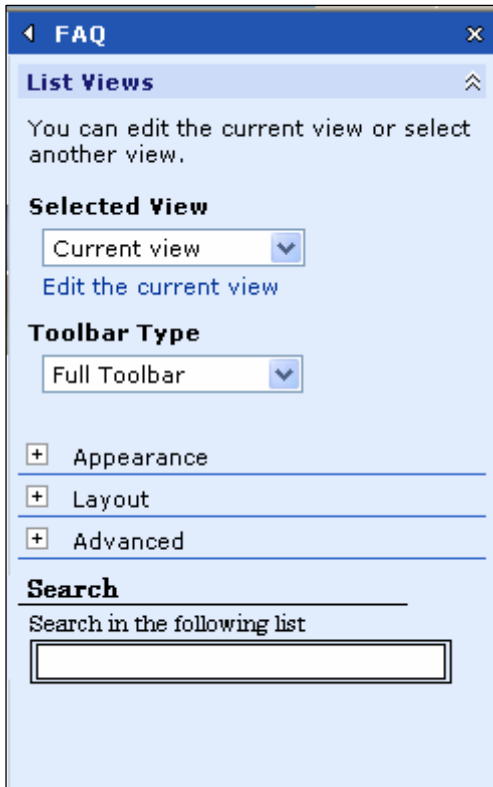


Figure 23 – System sequence diagram of Use Case 4

5.2.5 UC5 – Setup Search

Use Case ID: Domain	UC5 – Setup Search Admin		
Created By:	PMM	Last Updated By:	PMM
Date Created:	26. January 2007	Date Last Updated:	06/02/2007

Actors:	MAN Diesel Admin
Description:	To setup the search web part admin must give the name of the list to the web part, which admin will be searching in. Furthermore a view must be created called “Search” in the Sharepoint list and the appropriate user rights to filter a list must be given to the employee.
Trigger:	User needs to search in some custom made list
Pre-conditions:	
Post-conditions:	
Normal Flow:	<ol style="list-style-type: none"> 1. User enters the name of the list in the tool pane. 2. User saves data in the tool pane. 3. System retrieves lists from Sharepoint view site. 4. System validates the list name. 5. Entered list name exist. System fills the drop-down list.
Alternative Flows:	5a. Entered list name doesn't exist. Message printed to user.
Exceptions:	
Includes:	
Priority:	High
Frequency of Use:	Rare
Business Rules:	
Special Requirements:	See section 3.3 Supplementary specification
Assumptions:	Search web part is added to the Virtual Server Gallery in Sharepoint.
Notes and Issues:	



In Sharepoint a tool pane will emerge if you choose “Modified Shared Web Part”, in this tool pane a textbox should be added so Admin can select which list the search should be working on.

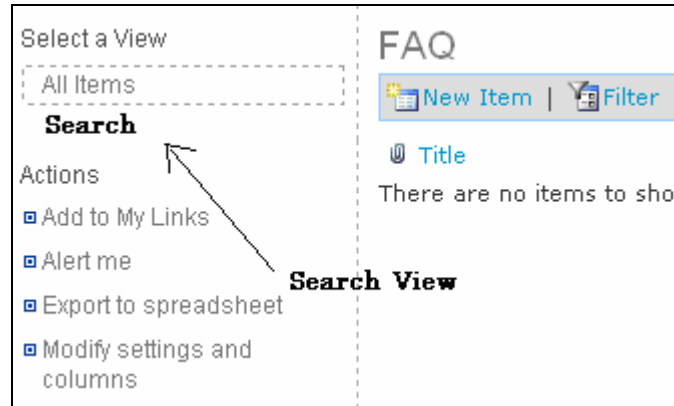


Figure 25 – Search view

The FAQ list must have a view called ‘Search’, since the web part will work on this view on the FAQ.

Figure 24 – Text box added to tool pane

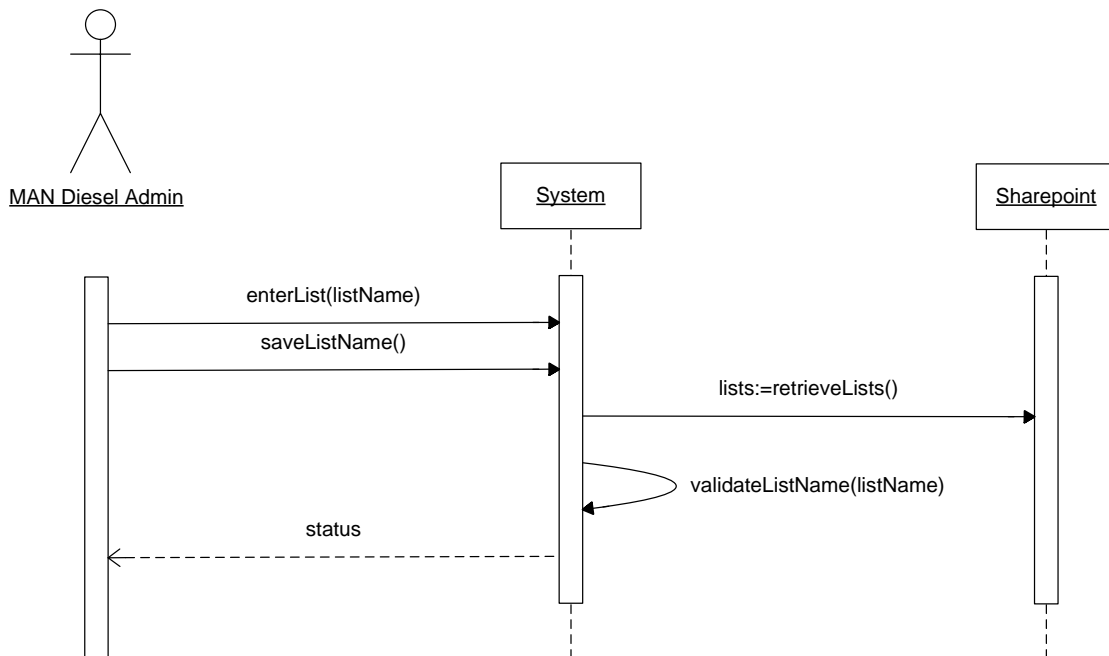


Figure 26 – System sequence diagram of Use Case 5

5.2.6 UC6 – Search in All Columns

Use Case ID: Domain	UC6 – Search in All Columns Search		
Created By:	PMM	Last Updated By:	PMM
Date Created:	9. February 2007	Date Last Updated:	17. February 2007

Actors:	MAN Diesel employee
Description:	Instead of searching in one column at the time, to limit the result which is brought back from a search, user should be able to search in all columns and get all the matches in the Sharepoint list.
Trigger:	User needs to search in all columns
Pre-conditions:	Web Part is visible
Post-conditions:	
Normal Flow:	<ol style="list-style-type: none"> 1. User enter a search input in the search textbox 2. User selects the 'All Column' index from the drop-down list. 3. User selects a search condition deciding which user input should be retrieve. 4. User hits the search button. 5. System split the user-input string, so it can be used to generate a valid query-string, if user-input contain multiple search inputs. 6. System generates a query-string 7. System update Sharepoint view query. 8. The Sharepoint list is filtered, and user gets search results.
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	Medium
Frequency of Use:	Daily
Business Rules:	
Special Requirements:	See section 3.3 Supplementary specification
Assumptions:	Search web part must be attached to a Sharepoint list.
Notes and Issues:	

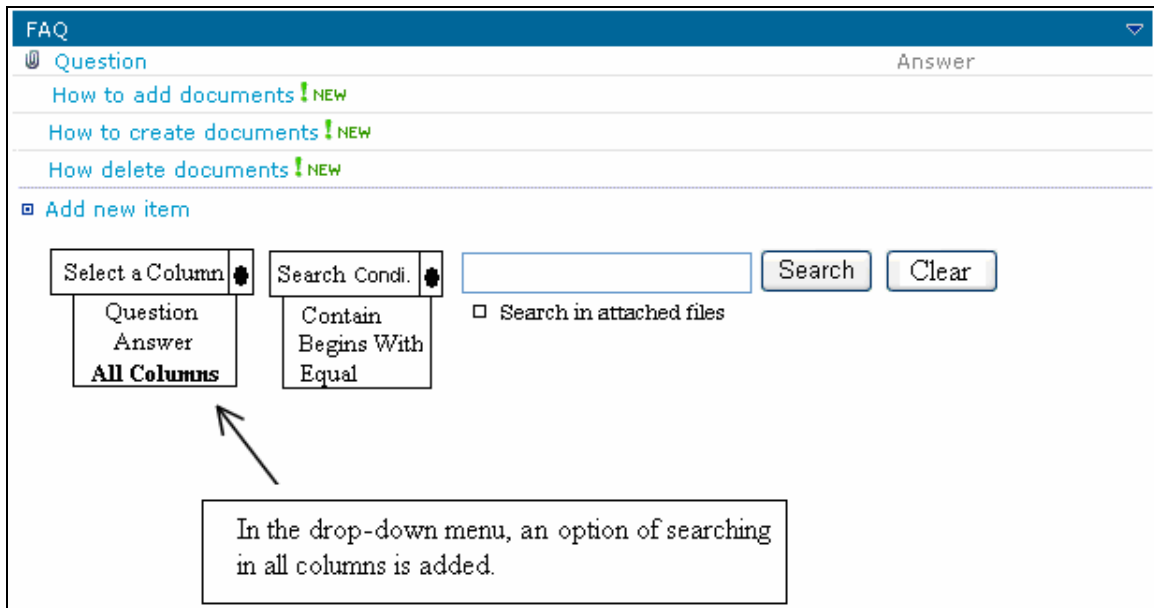


Figure 27 – How I imagine the interface when user needs to search in all columns

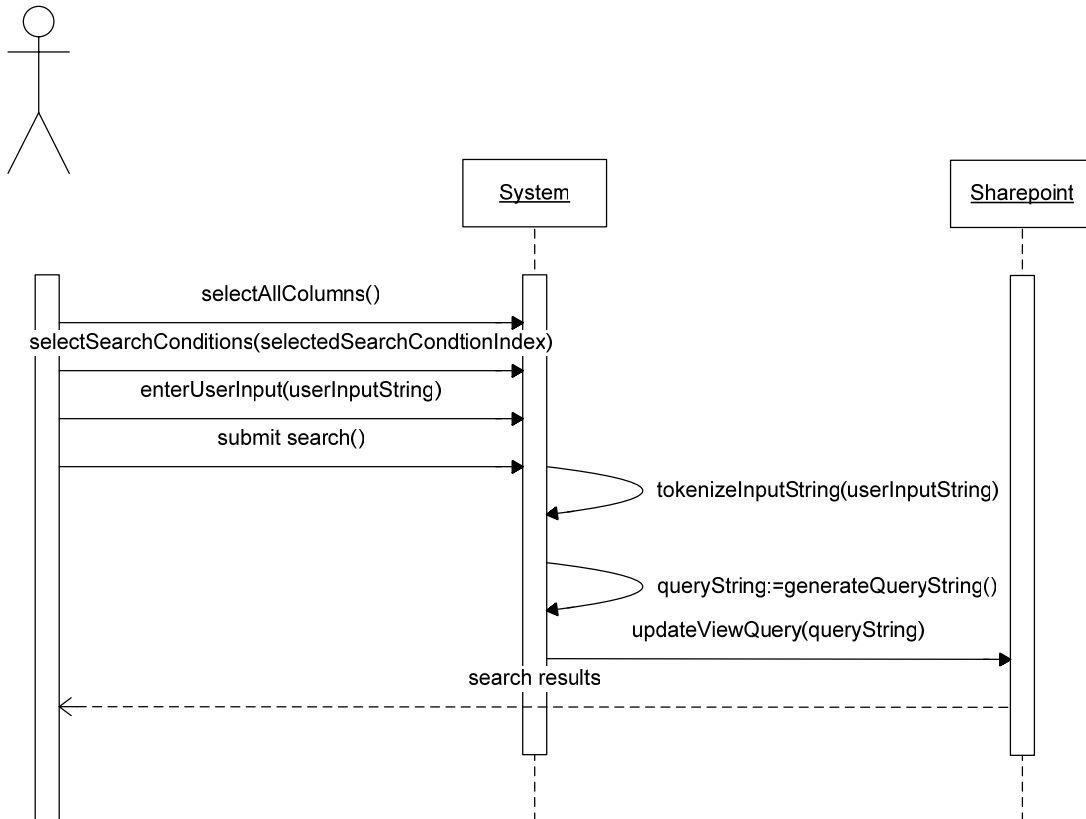


Figure 28 – System sequence diagram of Use Case 6

5.2.7 UC7 – Select Columns to Search in

Use Case ID: Domain	UC7 – Select Columns to Search in Search		
Created By:	PMM	Last Updated By:	PMM
Date Created:	22. February 2007	Date Last Updated:	27. February 2007

Actors:	MAN Diesel employee
Description:	User can select more than one column to search in without having to select the 'All Columns' in the drop-down list.
Trigger:	User needs to search in more than one column, and not all of them.
Pre-conditions:	Web Part is visible
Post-conditions:	
Normal Flow:	<ol style="list-style-type: none"> 1. User enter search input in the search text box 2. User selects the columns by checking the box in drop-down list. 3. User selects a search condition, deciding which user input should be retrieved. 4. User hits the search button 5. System split the user input string, so it can be used to generate a valid query-string, if user-input contain multiple search criteria. 6. System generates a query-string 7. System update Sharepoint view query. 8. The Sharepoint list is filtered, and user gets search results.
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	Low
Frequency of Use:	Rare
Business Rules:	
Special Requirements:	See section 3.3 Supplementary specification
Assumptions:	Search web part must be attached to a Sharepoint list.
Notes and Issues:	

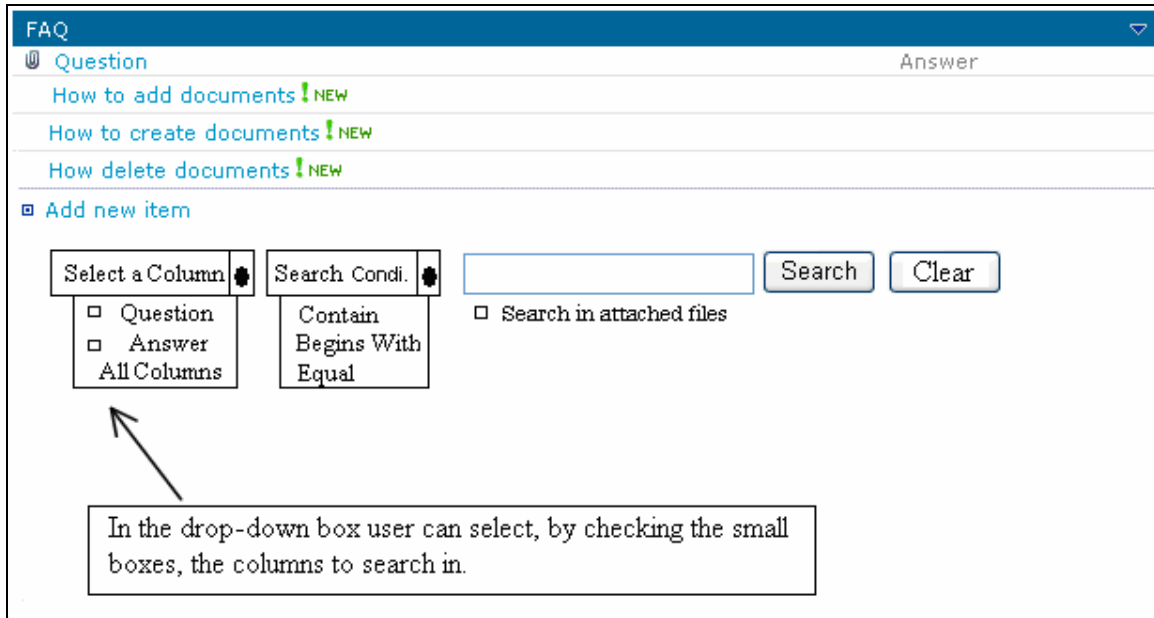


Figure 29 – How I imagine the interface when user needs to search in some columns

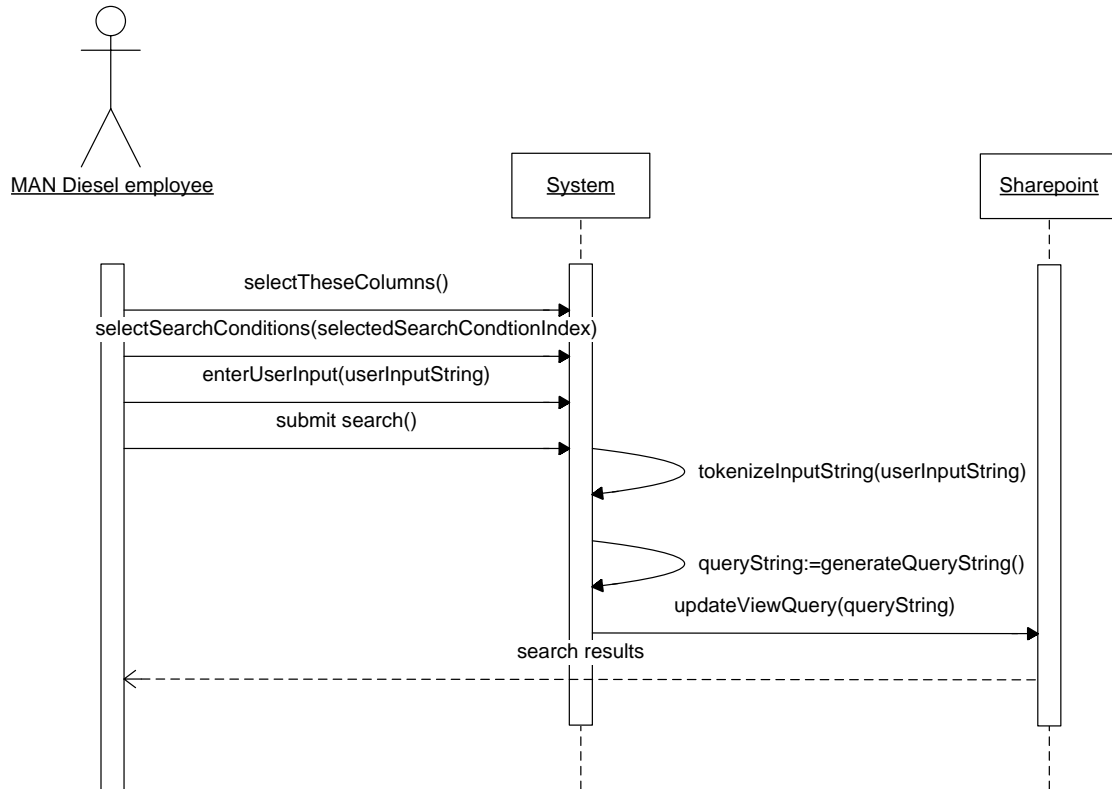


Figure 30 – System sequence diagram of Use Case 7

5.3 Conceptual-Model

5.3.1 Identification of conceptual classes and attributes

To get a better understanding of the system I have identified a number of conceptual classes and attributes by going through my Use Cases and perform linguistic analysis, which is identifying the nouns and noun phrases and consider them as candidate for conceptual classes and attributes. It has provided me with the following conceptual classes and attributes;

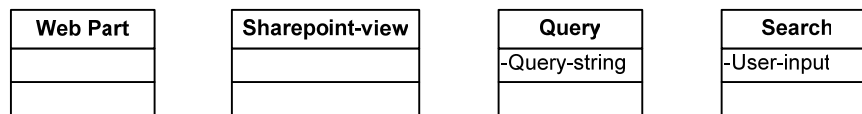


Figure 31 – Conceptual classes with attributes.

5.3.2 Association

To aid the understanding of the conceptual model or domain I establish associations between the conceptual classes. The associations are establish between the conceptual classes which have some relationship, e.g. the Search class can tell the Query class to generate a query-string and the web part access the Sharepoint-view. See figure 32.

5.3.3 Conceptual model diagram

From the identification of the conceptual classes, their attributes and how they are connected a domain model is created, which show a high level abstraction of the system.

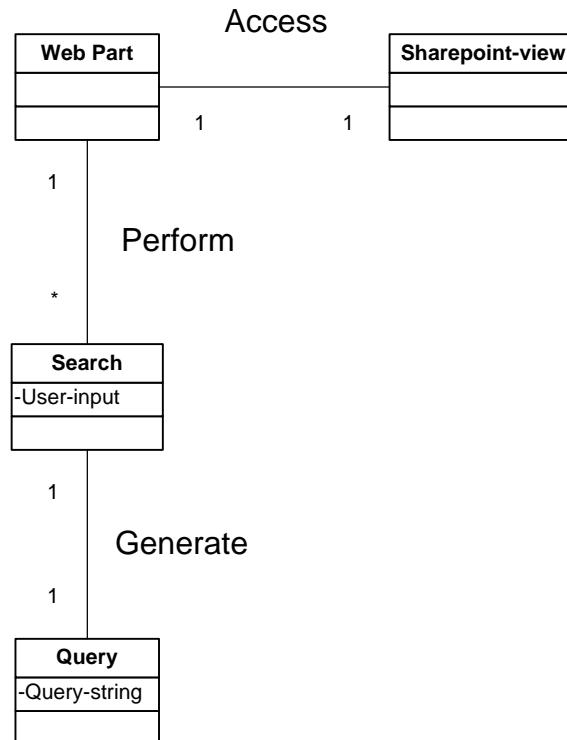


Figure 32 – Domain model

It is important to state that the conceptual classes is not software classes I merely perform this task to get a better understanding of the system I am trying to develop. However the conceptual classes can be of great help when I later begin to implement the system and the attributes can help give me an idea of what information these conceptual classes hold.

5.4 Summary

The purpose of this chapter was to look into possible approaches and try finding the right approach for creating the search web part.

Each of the Use Case found in the 'User Requirement Specification' phase was analyzed in detail and a conceptual model which describes the system at a high level abstraction was established.

CHAPTER 6

Design

6 Design

6.1 From Analysis to Design

Where I in the latter sections tried to understand the problem of this project and tried to find the right approach the design phase sets out to understand the approach that I decided to go a head with.

The process from a high level abstraction with conceptual classes to real software classes goes through what is called a Use Case realization. Use Case realization is created with help from sequence/interaction and class diagrams. The interaction diagrams show the flow in the software and the class diagram is used to show the content of a class and its connection to other classes.

By looking at the conceptual model diagram created in chapter 5 – Analysis I identified possible software classes. This however does not mean that all conceptual classes necessarily become software classes in the software program, and it doesn't mean that there can't be more software classes than there are conceptual classes. Which of the conceptual classes that becomes software classes is really first established during coding.

6.2 Patterns

For this project I have intentionally not used many patterns since it would only make the project more complicated and many of the patterns would simply be pointless to use.

6.2.1 Singleton

The singleton pattern is used in situations where only one instance of a class is needed. All object that need an instance of this class will use the same instance. To create a singleton pattern the class must keep an instance of itself as a private static member. Furthermore you need to add a constructor which is private, and the last step is to make a property which returns the instance. The following code show how it is implemented.

```
class Singleton
{
    private static Singleton instance = new Singleton();

    private Singleton() { }

    public static Singleton Instance
    {
        get { return instance; }
    }
}
```

6.3 Interaction Diagram

Earlier in the report I made some system sequence diagrams to show the interaction between user and system. Where system sequence diagrams showed the communication on a high level abstract the Interaction diagrams shows the interaction between software classes.

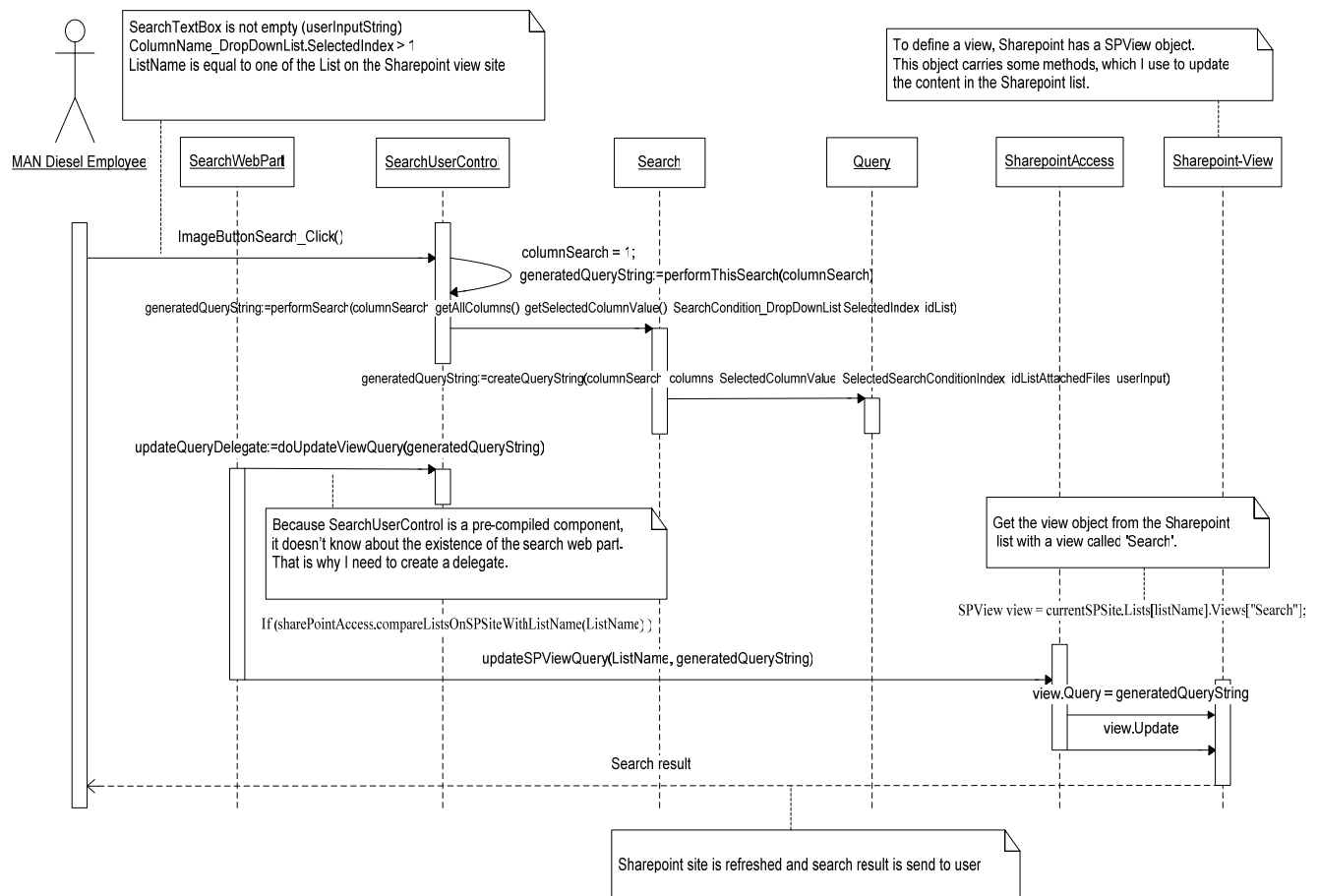


Figure 33 – Interaction diagram for Use Case 1

Figure 33 shows the flow in the software classes when the user hits the button. On the figure user has selected to search in one column before hitting the button. This information is passed to the function 'performThisSearch()', which takes a integer as parameter which tell the Query class which query-string is must create.

Now the SearchUserControl class uses one of the Search classes method 'performSearch()' to perform a search. What happens is that the 'SearchUserControl'

class passes all the data, from the different drop-down list, search textbox and information about which search type is requested, to the 'Search' class.

In the other Use Cases the Search class will now split the user-input string so it is readable. Since Use Case 1 is not the advanced search web part, which will take multiple user inputs, the tokenize method aren't seen used on this interaction diagram.

Now the Search class creates the query-string 'createQueryString()' by passing all the value received from the 'SearchUserControl' and the 'tokenized' user-input to the Query class.

The Query class generates a query-string which is returned to the 'SearchUserControl' class. To see how the query-string is generated look on the CD where I have placed the code.

With the used of delegates the generated query-string is send to the web part. The reason I need the delegates is because the 'SearchUserControl' is a precompiled component which doesn't know about the existent of the web part.

The 'SearchWebPart' class will through the SharepointAccess class access the SPview class on the Sharepoint site. The SharepointAccess class takes care of all communication with Sharepoint. When the SharepointAccess class have received the generated query-string the web part first checks whether admin have setup the web part correct. The web part must be attached to a list.

Now the SharepointAccess class call some of the SPView class methods and properties and sets the Query on a SPView object. Afterward the Query is updated so the changes take effect.

The rest of my interaction diagrams can be found in appendix B.

Use Case 7 has been dropped which why no interaction diagram for this Use Case has been made. I agreed with MD DEPT 9580 that this features is unnecessary to implement since the search in all column feature will work just fine in the situation where user needs to search in more than one column.

6.4 Class Diagram

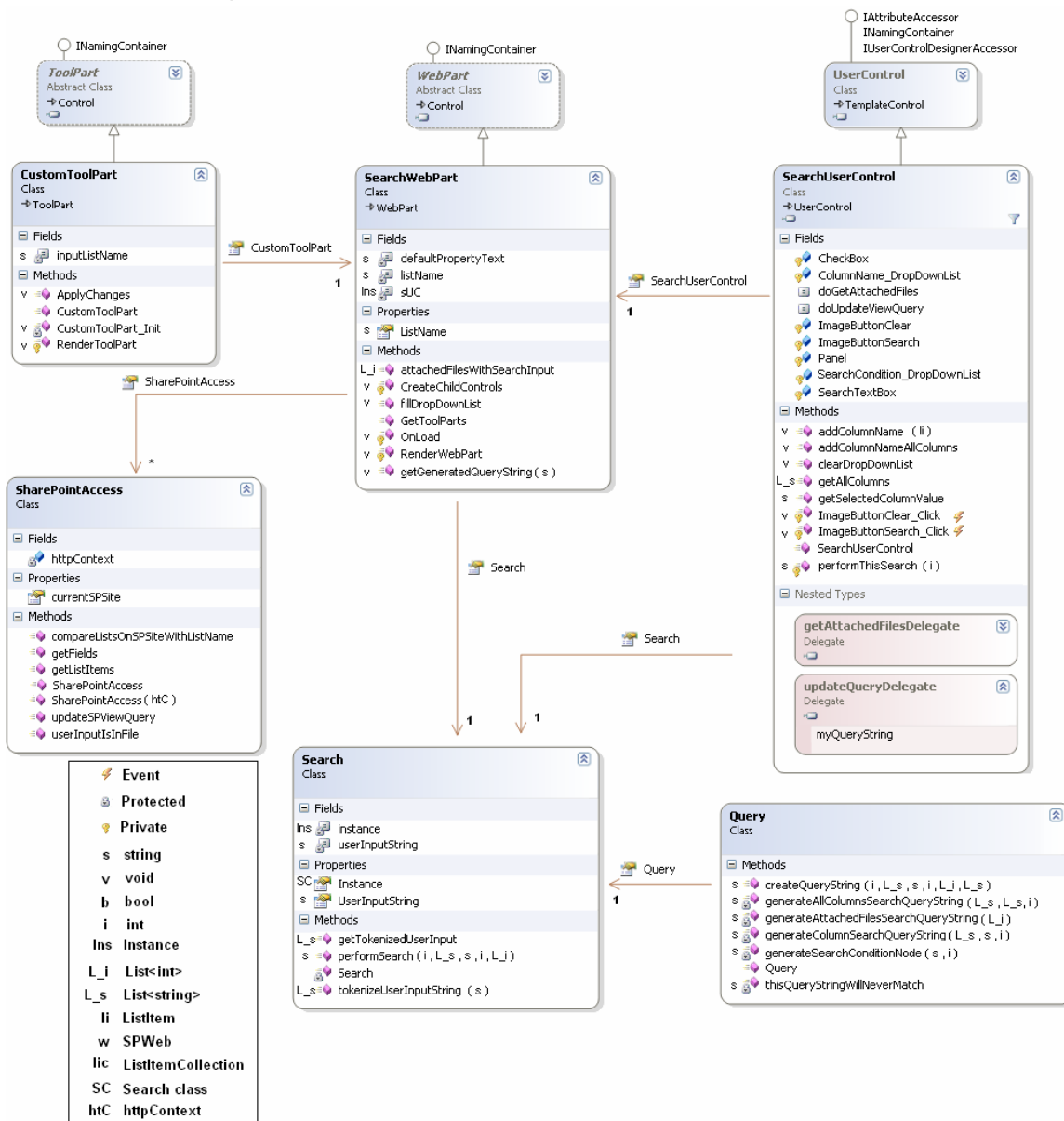


Figure 34 – Class diagram

The class diagram is constructed from the analysis and the conceptual model. After the implementation I have made reverse engineering on the finish class hierarchy and the above class diagram reflects the final result.

In the left side of the class diagram is symbol description which explain the attributes and methods, type, parameter value and access status.

The class diagram shows both the 'SearchWebPart' class and the 'SearchUserControl' class taking an instance of the 'Search' class, which is why I need to implement the singleton pattern to prevent the creation of multiple instance of the 'Search' class.

6.5 Logical Architecture

To organize the software classes I have decided to aim at dividing my classes into a three layered architecture. The reason I need three layers is because I want to separate user interface, business logic and data access from each other.

The layers should be organized so a data access layer and a business logic layer know nothing of the user interface layer, and the data access layer knows nothing of the business layer.

Although I aim a creating a layer architecture which follows the above rules it is related with some limits due to the way Sharepoint and web parts work. Often when creating web parts it is quite normal to have the data access layer in the web part which also is the user interface. However it is my goal to separate the user interface, business logic and data access from each other within the limits of Sharepoint.

The architecture I am going for in my project is a relaxed layered architecture[1, page 200], which basically means that higher layers can call upon several lower layers, unlike a strict layered architecture, where the top layer only calls upon the layer directly below it.

The top layer is my user interface or presentation layer where the web part and the web control class lies. Actually the web part and web control could be considered to be two layers since the web control doesn't know of about the existing of the web part. The user control is a pre-compiled component added to the web part.

The middle layer is my business layer, and the bottom layer is my data access layer.

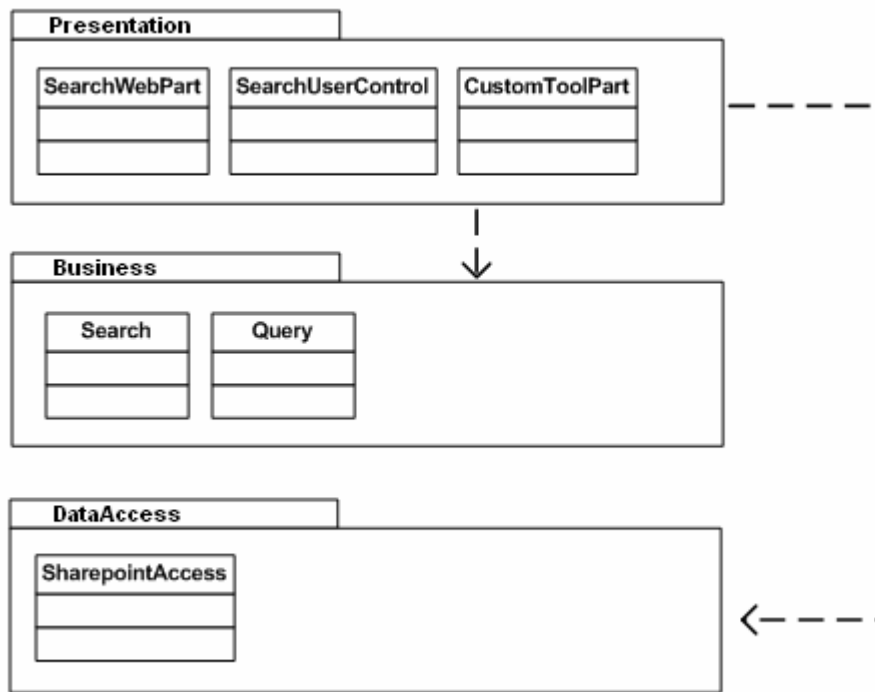


Figure 35 – Package diagram

6.6 Summary

In this chapter I made the transition from analysis to design, or from the conceptual model to a concrete software specification, with the use of interaction diagrams and class diagrams. Furthermore I establish a structure for the software classes and which design pattern to implement. Use Case 7 was dropped as we found it unnecessary and because Use Case 6 would cover it more or less.

CHAPTER 7

Implementation

7 Implementation

7.1 Implementing the Design

The following chapter is used to describe the implementation of the proposed design in chapter 6 - Design. I intend to give a description of each method, properties and attributes in the software classes in the class diagram.

7.1.1 SearchWebPart

The 'SearchWebPart' class is responsible for creating the search web part. The class inherits from the 'WebPart' class. The 'WebPart' contain a number of methods which control the behaviour the web part.

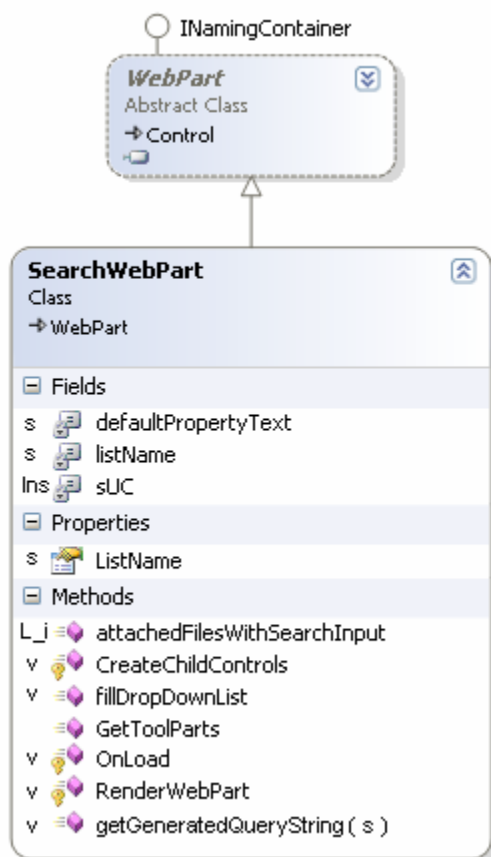


Table of attributes/fields:

Name	Type	Parameter	Description
defaultPropertyText	string		The value of the textbox in the tool pane
listName	string		The value of the list name given by user.

Table of properties:

Name	Type	Parameter	Description
ListName	string		Saves the name of the list entered by user.

Table of functions:

Name	Type	Parameter	Description
SearchWebPart			Constructor
CreateChildControls	void		This function is overridden to create my own composite control which is a group of controls, such as a button or a textbox, put together to create a user interface.
GetToolParts	ToolPart[]		This function is overridden to create my own tool pane. The function determines which tool parts are displayed in the tool pane and the order in which they are displayed. The tool pane is the box which emerges when a web part is in design mode.
RenderWebPart	void	HtmlTextWriter	This function is overridden, to get the function to display my composite control in the web part.
OnLoad	void	EventArgs	This function is overridden because the delegates need to be set immediately when the browser is loaded.
fillDropDownList	void		This function tells the user control to fill the drop-down list with the column names from the Sharepoint list.
attachedFilesWithSearchInput	List<int>		This function retrieves the item IDs from the Sharepoint list which has a match between the user-input and the content in the attached files.
getGeneratedQueryString	void	string	This function retrieves the generated query-string from user control.

7.1.2 CustomToolPart

The 'CustomToolPart' class add my toolpart(the textbox where user can add the name of list) to the tool pane which emerge when the web part is in design mode. The class inherits from the 'ToolPart' class.

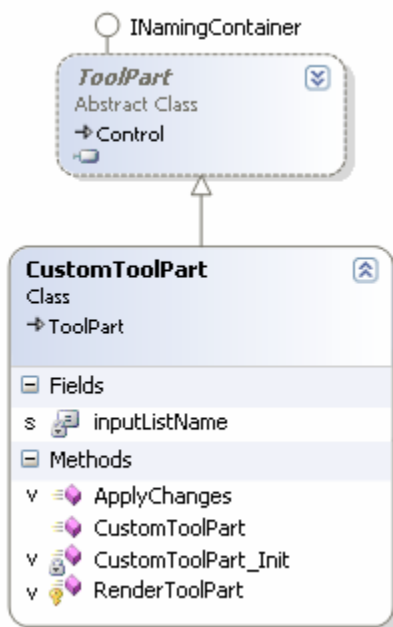


Table of attributes/fields:

Name	Type	Parameter	Description
inputListName	string		The attributes carries the value of the list name.

Table of functions:

Name	Type	Parameter	Description
CustomToolPart			Constructor, when called set the name of tool part and initialize the textbox control
customToolPart_Init	void	Object, EventArgs	Function sets the attribute inputListName equal to the value of the textbox by referring to textbox ID.
ApplyChanges	void		This function is overridden so when user enter a list name in the text box in the tool pane and press the apply button, the property in the SearchWebPart is set and saved. If the list exists drop-down list with column name is filled.
RenderToolPart	void	HtmlTextWriter	This function is overridden to get the function to display my tool part in the tool pane.

7.1.3 SearchUserController

The 'SearchUserController' class creates the controls in the web part. The class inherits from the 'UserController' class.

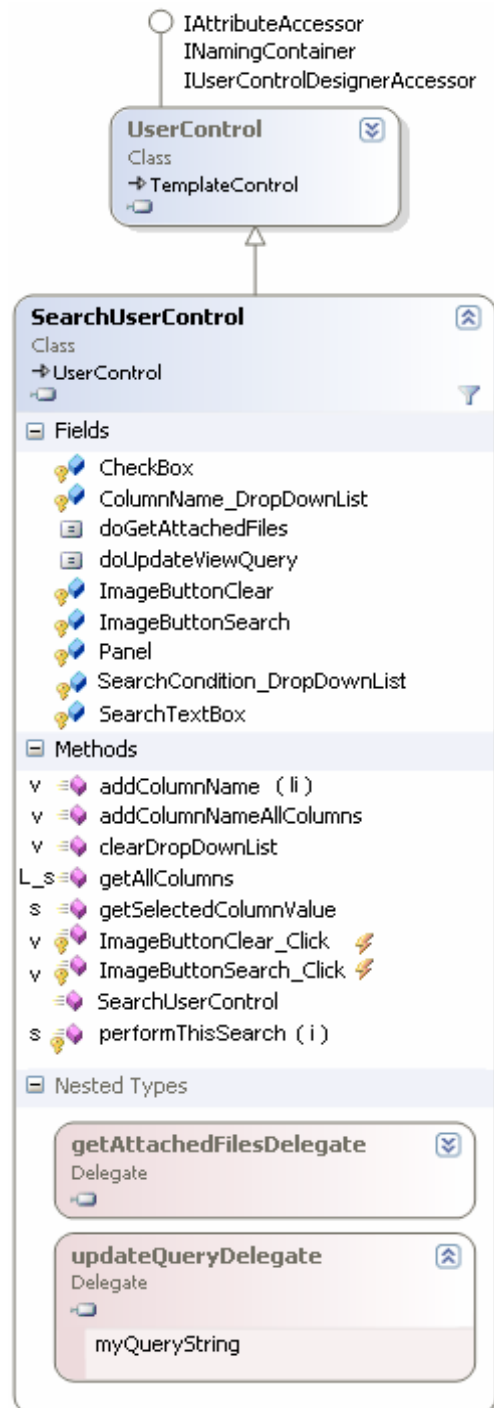


Table of attributes/fields:

Name	Type	Parameter	Description
updateQueryDelegate	void	string	Delegate which take the getGeneratedQueryString() method in the SearchWebPart class.
getAttachedFilesDelegate	List<int>		Delegate which take the attachedFilesWithSearchInput () method in the SearchWebPart class.
CheckBox			ID name of the check box
ColumnName_DropDownList			ID name of the drop-down list which contain all the column names in a Sharepoint list
SearchCondition_DropDownList			ID name of the drop-down list which contain search conditions. (Contain, Begin with)
ImageButtonClear			ID name of the button which clears the filter on the Sharepoint list
ImageButtonSearch			ID name of the button which launches a search
SearchTextBox			ID name of textbox where user enters the text to search for.
Panel			ID name of the panel that contain the other controls.

Table of functions:

Name	Type	Parameter	Description
SearchUserControl			Constructor
ImageButtonSearch_Click	void	Object, ImageClickEventArgs	When search button is click, this function launches the appropriate function based on the actions perform in the web part.
ImageButtonClear_Click	void	Object, ImageClickEventArgs	When the clear button is click, this function launches the methods that eventually update the 'Search' view query with an empty query-string that clears the filter on the Sharepoint list.
performThisSearch	string	int	This function passes the value of the decided search scenario to Search class, which perform the search and retrieves the generated query-string.
getAllColumns	List<string>		This function retrieves all the column names in the 'ColumnName_DropDownList' list, except the entries which doesn't exist in the Sharepoint list.

Name	Type	Parameter	Description
getSelectedColumnValue	string		This function retrieves the specific column which user wants to search in.
addColumnNames	void	ListItem	This function adds the name of the Sharepoint list columns to the 'ColumnName_DropDownList'.
addColumnNamesAllColumns	void		This function add the 'All Columns' index to the 'ColumnName_DropDownList'.
clearDropDownList	void		This function clears the 'ColumnName_DropDownList', except for the index '--Select Column --'.

7.1.4 Search

The 'Search' class performs the search and splits the user input string.

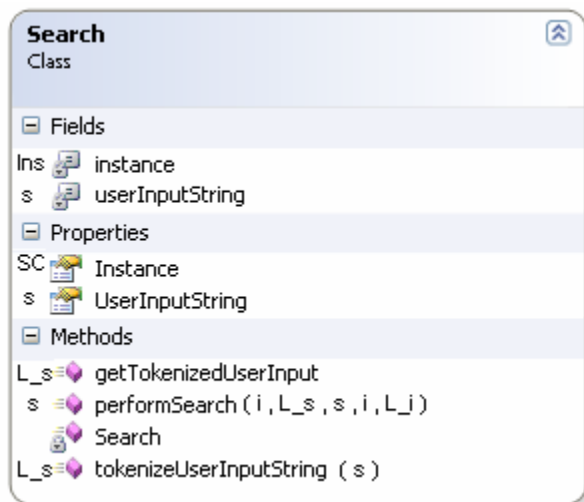


Table of attributes/fields:

Name	Type	Parameter	Description
instance	Search		Carries an instance of the 'Search' class.
userInputString	string		The attributes carries the un-tokenized user input string.

Table of properties:

Name	Type	Parameter	Description
UserInputString	string		Saves the input string entered by the user.
Instance	Search		Saves an instance of the Search class.

Table of functions:

Name	Type	Parameter	Description
Search			Constructor
tokenizeUserInputString	List<string>	string	This function split the user-input string, as many inputs is allow by using the '+' sign.
performSearch	string	int, List<string>, string, int, List<int>	This function performs the search by calling the function in the Query class, which generate a query-string. The function also passes the necessary data to the Query class, such as which query-string should be created, which index in drop-down lists have been selected, the tokenized user input and all column names.
getTokenizedUserInput	List<string>		This function retrieves the tokenized user input.

7.1.5 Query

The 'Query' class performs the building of query-strings.

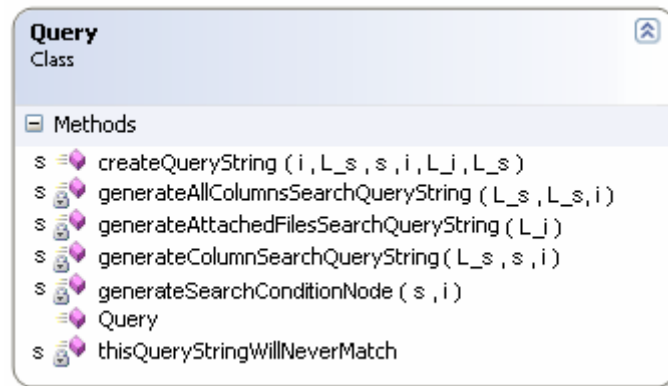


Table of functions:

Name	Type	Parameter	Description
Query			Constructor
createQueryString	string	int, List<string>, string, int, List<int>, List<string>	This function stands for generating the correct query-string.
generateColumnSearchQueryString	string	List<string>, string, int	This function generates the query-string, when user has selected to search in one column.
generateAllColumnsSearchQueryString	string	List<string>, List<string>, int	This function generates the query-string, when user has selected to search in all columns.
generateAttachedFilesSearchQueryString	string	List<int>	This function generates the query-string, when user has selected to search in attached files.
generateSearchConditionNode	string	string, int	This function generates the search condition node in the query-strings.
thisQueryStringWillNeverMatch	string		This function generates a query-string, which will 'never' match. If user search in attached files and no match occur it needs to clear the Sharepoint list for items.

7.1.6 SharepointAccess

The 'SharepointAccess' class controls all the access with Sharepoint.

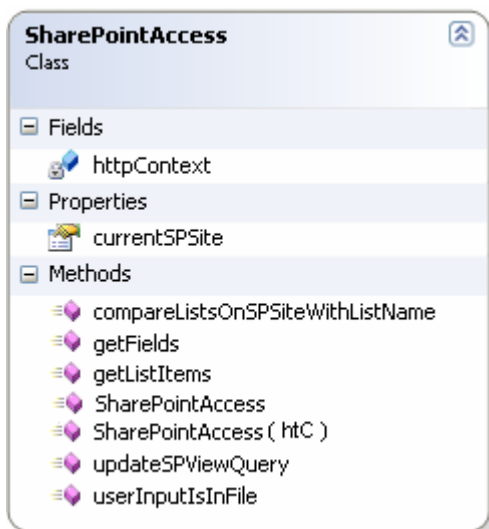


Table of attributes/fields:

Name	Type	Parameter	Description
httpContext	HttpContext		A Http context request

Table of properties:

Name	Type	Parameter	Description
currentSPSite	SPWeb		The current Sharepoint site

Table of functions:

Name	Type	Parameter	Description
SharepointAccess			Constructor
SharepointAccess		HttpContext	Constructor which take a HTTP request.
updateSPViewQuery	void	string, string	Updates the query on Search view.
compareListsOnSPSiteWithListName	bool	string	Compare the property ListName form SearchWebPart class, with the entire Sharepoint lists on site.
getFields	ListItem-Collection	string	Get the entire field/column names from the Sharepoint list.
getListItems	List<int>	string, List<string>	Get all the items from the Sharepoint list
userInputIsInFile	bool	string, List<string>	This function returns a true, if user input matches some of content in an attached file.

7.1.7 Implemented remarks

Of all the implementation I needed to make in this project I found the Query class to be the most interesting to implement because I had to use e.g. recursive calls.

In the beginning when the search web part could only search in one column and take one search input the methods that generated the query-string were fairly easy created.

```
<Where>
  <Contains>
    <FieldRef Name="ColumnName1" />
    <Value Type="String1" />
  </Contains>
</Where>
```

However as soon as I began implementing new functionalities on the core search web part and the query-string grew, I realised that the CAML query structure, made the query-string a bit more complicated to make. The query structure elements (<Or>, <Contains>) must contain two other elements.

Lets assume user have enters 2 input, the query-string would look as follow.

```
<Where>
  <Or>
    <Contains>
      <FieldRef Name="ColumnName1"/>
      <Value Type="String">String2</Value>
    </Contains>
    <Contains>
      <FieldRef Name=" ColumnName1"/>
      <Value Type="String">String1</Value>
    </Contains>
  </Or>
</Where>
```

A query-string with three inputs;

```
<Where>
  <Or>
    <Contains>
      <FieldRef Name="ColumnName1"/>
      <Value Type="String">String3</Value>
    </Contains>
    <Or>
      <Contains>
        <FieldRef Name="ColumnName1"/>
        <Value Type="String">String2</Value>
      </Contains>
      <Contains>
        <FieldRef Name=" ColumnName1"/>
        <Value Type="String">String1</Value>
      </Contains>
    </Or>
  </Or>
</Where>
```

A query-string with five inputs;

```

<Where>
  <Or>
    <Contains>
      <FieldRef Name="ColumnName1" />
      <Value Type="String">String5</Value>
    </Contains>
    <Or>
      <Contains>
        <FieldRef Name="ColumnName1" />
        <Value Type="String">String4</Value>
      </Contains>
      <Or>
        <Contains>
          <FieldRef Name="ColumnName1" />
          <Value Type="String">String3</Value>
        </Contains>
        <Or>
          <Contains>
            <FieldRef Name="ColumnName1" />
            <Value Type="String">String2</Value>
          </Contains>
          <Contains>
            <FieldRef Name="ColumnName1" />
            <Value Type="String">String1</Value>
          </Contains>
        </Or>
      </Or>
    </Or>
  </Or>
</Where>

```

After implementing the ability to have multiple user-input I had to implement the Search in “All Columns” Use Case which only made it more interesting since I had to make some adjustments to the methods so that they could handle multiple ColumnNames in the query-string. This is how it looks when user enters 2 inputs and is searching in all columns (2 columns) in the list.

```

<Where>
  <Or>
    <Or>
      <Contains>
        <FieldRef Name="ColumnName1" />
        <Value Type="String">String2</Value>
      </Contains>
      <Contains>
        <FieldRef Name="ColumnName1" />
        <Value Type="String">String1</Value>
      </Contains>
    </Or>
    <Or>
      <Contains>
        <FieldRef Name="ColumnName2" />
        <Value Type="String">String2</Value>
      </Contains>
      <Contains>
        <FieldRef Name="ColumnName2" />
        <Value Type="String">String1</Value>
      </Contains>
    </Or>
  </Or>
</Where>

```

When I implemented the “Search in Attached File” Use Case I couldn’t ask directly which columns items had a match between user-input and the content in their attached files, the same way I did when I search in the columns, so I had to make a work-around.

I solve the problem by first comparing the user input with all the attached files content. If there was a match, I retrieve the ID for the item. A list has an ID as it has a column name. With the ID’s I generated a query-string which I could filter the list with.

```
<Where>
  <Or>
    <Eq>
      <FieldRef Name="ID"/>
      <Value Type="String">1</Value>
    </Eq>
    <Eq>
      <FieldRef Name="ID"/>
      <Value Type="String">2</Value>
    </Eq>
  </Or>
</Where>
```

One of the requirements for the web part was that the results from the search was showed in the Sharepoint list and that the features of Sharepoint list could be used, such as sort, change the order of the columns and edit the column names. During the implementation I came across a smaller problem because the column name (display name) showed in browser was not the same as the name in database (internal name), so when I changed the column name I couldn’t search in the column. I solve the problem by simply showing the display names in the drop-down list and by using the internal name in query-strings.

The code for the web part can be found on the CD.

7.2 Summary

I used this chapter to describe all the software classes in detail what their methods, properties and attributes do, and I described how I solve some of the implementation.

CHAPTER 8

Test

8 Test

8.1 Purpose

Besides creating a web part which makes users capable of searching directly in a list, MAN Diesel A/S asked me to look into regression test tools in Visual Studio 2005 .NET Team Suite.

I intend to give a short description of each tool in VS05.NET TS and establish which of the tools can be used to uncover regression bugs and use it on my web part project.

As I mention in the start of my report regression testing is a type of testing method which tries to uncover regression bugs. Regression bugs can occur when you need to add new functionalities to a system. Take the situation where I had to add further functionalities to the core system web part. This situation will probably have the consequence that other functionalities in the code cease to work after the implementation.

In the section 3.3 Supplementary Specification in chapter 3 I stated that the performance of the search web part must have good response times since the main idea behind the web part is to save time. Therefore I am going to make a small test of the search performance it can be found in section 8.4.

In appendix C I have added a test which test for incorrect input from user on the web part. In chapter 9 – Deployment I have made a presentation of the web part which shows the cases when ‘correct’ input or actions have been entered/performed.

8.2 VS05.NET TS Test Tools

In VS05.NET TS you can perform 6 different tests.

- Load test
- Generic test
- Manual test
- Ordered test
- Unit test
- Web test

Load test is performed when you need to test the reliability of a system. Such a test is typical used for multi-user system where you need to get an idea of how much pressure a system can take when multiple users access the program’s services concurrently.

Generic test is performed when you need test an existing program or third party tool.

Manual test is performed when test couldn’t be automated.

Ordered test is performed when you want to run other test in a specific order.

Unit test is performed when you need to validate a part/unit of some source code is working properly.

Web test is performed to test the functionality of web applications and to test web application under load.

Of the test tools VS05.NET TS provides unit test is the only one which provides a way to deal with regression bugs. In unit testing you isolate each part of the program from each other and show that the individual parts are correct. This is why this test technique is particularly well to find regression bugs because it is easy to identify where the bug occurred and fix it.

I intend to use unit test on the classes which are difficult to implement and which surely will create regression bugs.

8.3 Unit Test

During the implementation of the Query class I realised that this class would be ideal to use unit test on, to validate that each method generated the right piece of the query-string. As I mention in 7.1.7 Implemented Remarks the methods in the Query class was challenging and therefore lead to quite some bugs during the implementation, due to the way CAML queries is structured.

I used a bottom-up approach during coding, where I wrote a part of the code and tested it, then I wrote some more and tested the sum of its parts.

I have made operation contracts for each of the function in the Query class. The contracts show input and expected output of the functions and help create the unit tests.

With the contracts written I have a sketch for the benefits and obligations of the functions in the Query class. Now take the generateColumnSearchQuerystring() method this function is expected to deliver a correct constructed CAML query which can search in one column. As input it takes a user input list which contains the tokenized user input string, the name of the column to search in and the search condition the user has selected from the drop-down list

Contract Query: generateSearchConditionNode

Operation:	generateSearchConditionNode(Search condition type: string, Selected search condition index, integer)
Cross Reference:	UC1 - Search in List, UC2 - Search Pattern & UC6 - Search in All Columns
Preconditions:	User selects a search condition.
Postconditions:	A start or end search condition node was created, which is to be attach to the query-string.

On figure 36 is an example of a unit test on the generateSearchConditionNode() method, which produced an error. This error tolled me that I had appended a wrong statement to the query-string. I expected it to append the name <Contains> when it really was

appending the value <Contain>. A small and stupid error but an error which would have made the search failed since the query-string must be entirely correct generated.

```

    /// <summary>
    /// A test for generateSearchConditionNode (string, int)
    /// </summary>
    [DeploymentItem("SearchBusinessLayer.dll")]
    [TestMethod()]
    public void generateSearchConditionNodeTest()
    {
        Query target = new Query();

        UnitTestProject.Project_SearchBusinessLayer_QueryAccessor accessor = new UnitTestProject.Project_SearchBusinessLayer_QueryAc

        string searchConditionStartNode = "SearchConditionStartNode";
        string searchConditionEndNode = "SearchConditionEndNode"; //Could also have been an empty string

        int selectedSearchConditionIndex0 = 0; //Contain

        string expectedContainStartNode = "<Contains>";
        string expectedContainEndNode = "</Contains>";
        string actualContainStartNode;
        string actualContainEndNode;

        actualContainStartNode = accessor.generateSearchConditionNode(searchConditionStartNode, selectedSearchConditionIndex0);
        Assert.AreEqual(expectedContainStartNode, actualContainStartNode, "Project.SearchBusinessLayer.Query.generateSearchCondition
            "expected value. <Contains>");

        actualContainEndNode = accessor.generateSearchConditionNode(searchConditionEndNode, selectedSearchConditionIndex0);
        Assert.AreEqual(expectedContainEndNode, actualContainEndNode, "Project.SearchBusinessLayer.Query.generateSearchConditionNode
            "expected value. </Contains>");

    }
}

```

Test Results

Test run failed. Results: 0/1 passed; Item(s) checked: 1 [Rerun original tests](#) [Debug original tests](#)

Result	Test Name	Project	Error Message
Failed	generateSearchConditionNode1UnitTestProject		Assert.AreEqual failed. Expected:<<Contains>>, Actual:<<Contain>>. Project.SearchBusinessLayer.Query.generateSearchConditionNode did not return the expected value. <Cor

Figure 36 – Unit test on the generateSearchConditionNode() function.

I have added the rest of the contracts in appendix D. The unit tests lies together with the rest of my code on the CD.

8.4 Search Performance

8.4.1 Purpose

To test the performance of the search web part I have used VS05.NET TS web-test tool which allow me check how along an action takes to perform.

For this performance-test I have chosen to create a test-list where I manual have entered 60 items in the list and afterward attached a file on each item.

The web test tool allow me to record an action perform in the browser, e.g. searching in one column. The recorded action is then tested and the time it takes to perform the action is displayed.

I intend to test the response time when searching in one column, all columns and attached files, and I am going to get the response time for respectively 10, 30 and 60 items each item having a file attached. Furthermore I am going to test for 1 or 4 user input in the search textbox.

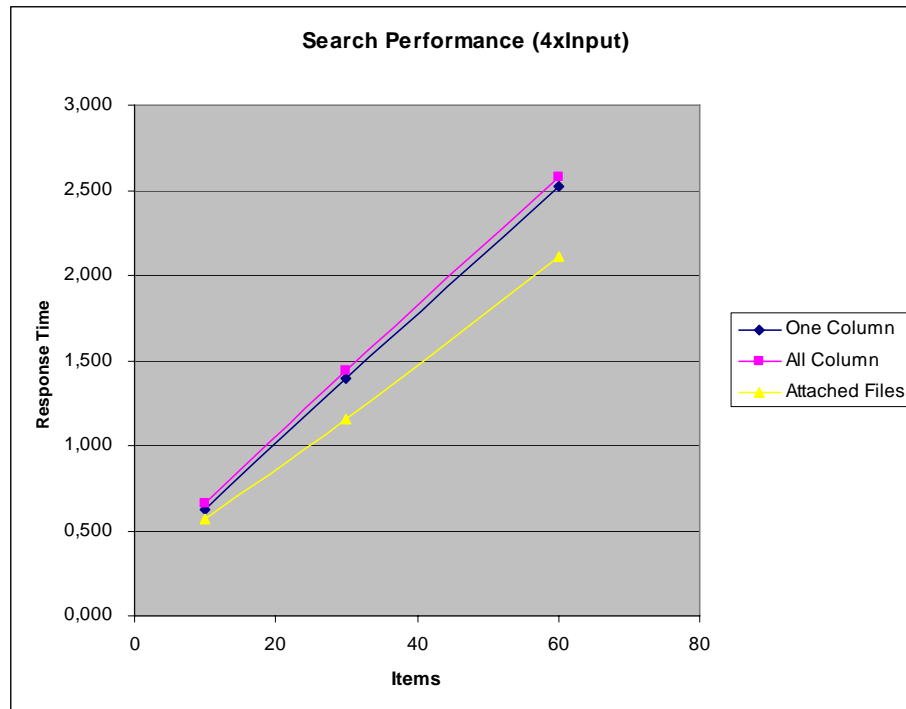
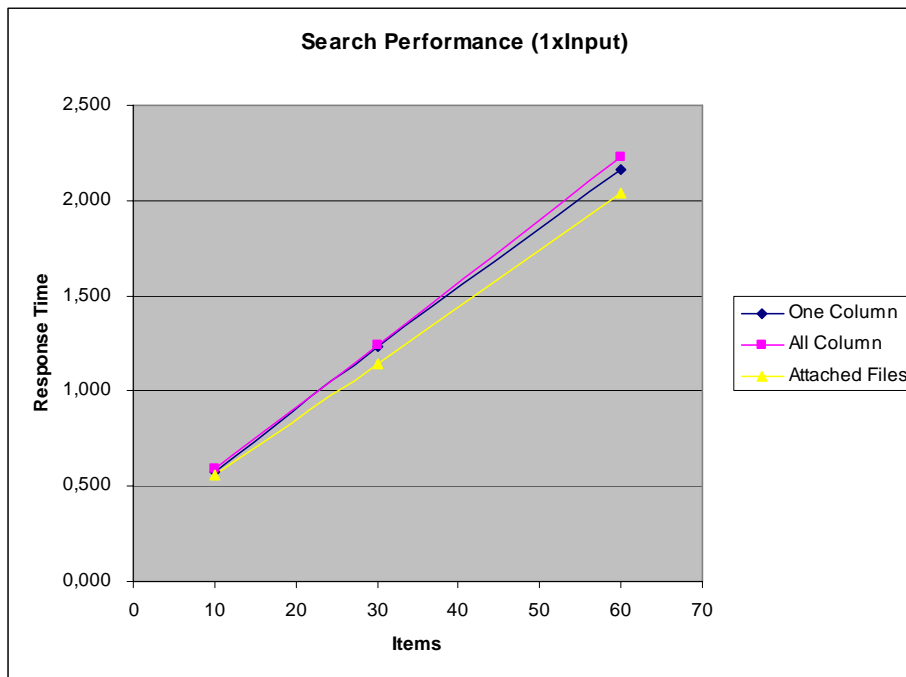
I intend to create a graph for one user-input and another graph with four user-input.

The response time for the selected search scenarios should be around 1-2 seconds for 60 items otherwise it will take to long time. The graph with 4 inputs should show a minor increase in response time compared to the graph with one input.

I expect the search in all columns compared to the search in one column to take longer time, due to the fact that more data will be process because the query-string is bigger.

8.4.2 Results

From the results generated in the web test I have made the following graphs and tables:



One Column					One Column (4xInput)				
Test 1	Test 2	Test 3	Average (sec)	Items	Test 1	Test 2	Test 3	Average (sec)	Items
0,55	0,62	0,56	0,577	10	0,66	0,6	0,6	0,620	10
1,25	1,22	1,24	1,237	30	1,37	1,43	1,39	1,397	30
2,12	2,17	2,21	2,167	60	2,44	2,57	2,57	2,527	60
All Column (3 columns)					All Column (4xInput) (3 columns)				
Test 1	Test 2	Test 3	Average (sec)	Items	Test 1	Test 2	Test 3	Average (sec)	Items
0,61	0,6	0,56	0,590	10	0,69	0,65	0,65	0,663	10
1,23	1,27	1,23	1,243	30	1,5	1,47	1,35	1,440	30
2,22	2,23	2,24	2,230	60	2,63	2,52	2,58	2,577	60
Attached Files					Attached Files (4xInput)				
Test 1	Test 2	Test 3	Average (sec)	Items	Test 1	Test 2	Test 3	Average (sec)	Items
0,59	0,52	0,56	0,557	10	0,53	0,58	0,59	0,567	10
1,21	1,11	1,12	1,147	30	1,25	1,15	1,06	1,153	30
2,04	2,12	1,96	2,040	60	2,09	2,09	2,16	2,113	60

As the tables show I have generated 3 response times for each test scenario and the average response times has been plotted on the graph with the number of items. The results from the plotting show two graphs with a linear tendency.

The graph with 4 input show a minor increase in response time compared to the graph with one 1 input and the search in all columns took a little longer to perform compared to search in one column, which was as I expected. I have deliberately not tried to compare the attached files search with the other searches since it was implemented in a different way as I described in section 7.1.7 Implemented Remarks.

The graph and tables further show that a search with 60 items takes around 2 seconds, the highest search times being 2,63 seconds in all columns test 1 which perhaps is little bit to much.

Overall the search time is acceptable within this scale of 60 items, but on the same time I must admit there is room for improvements.

8.5 Summary

I used this chapter to look-into VS05.NET TS to find a regression test tool. VS05.NET TS contain the regression test tool unit test which I have used on the Query class in my web part project. To create the unit test I used the UP artifact Contract to show the input and output of the methods I tested. Furthermore I have tested the response time on the web part.

I have added the contract in appendix D and the unit test can be found on the CD together with the rest of the code.

CHAPTER 9

Deployment

9 Deployment

9.1 Adding the Search Web Part to Sharepoint

To add the search web part to the Sharepoint environment there is number of things one must do.

The first thing is to add the web parts assemblies to the Sharepoint BIN directory.

The next thing is to create a web part file (.dwp) which contain the following content.

```
<?xml version="1.0" encoding="utf-8"?>
<WebPart xmlns="http://schemas.microsoft.com/WebPart/v2" >
  <Title>Custom SPList Search</Title>
  <Description>Search for items in custom list</Description>
  <Assembly>SearchWebPart</Assembly>
  <TypeName>MD.GIE1.WebParts.SPListSearchPresentationLayer.SearchWebPart
</TypeName>
</WebPart>
```

The web part must be placed in the Sharepoint directory WPCATALOG and the pictures for the buttons in WPRESOURCES.

The last thing before the web part is added is to tell Sharepoint that the web part is safe. An entry in the in the Safe Control list in the WEB.CONFIG file must be added.

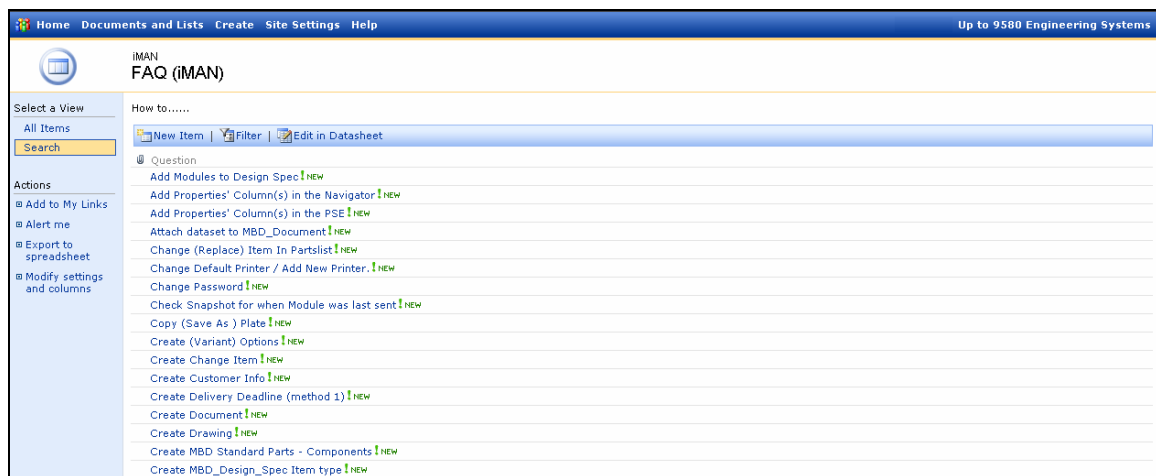
```
<SafeControl Assembly="SPListSearchWebPart"
Namespace="MD.GIE1.WebParts.SPListSearchPresentationLayer" TypeName="*"
Safe="True" />
```

Now the web part will appear in the Virtual Server Gallery.

9.2 Presentation of the Search Web Part

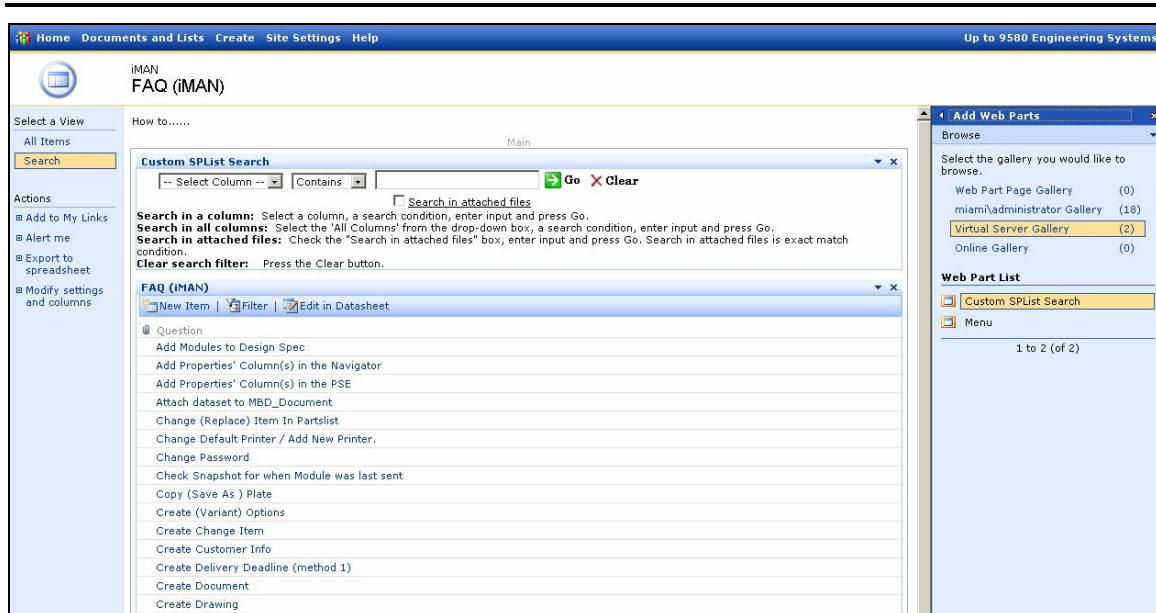
The web part now lies in the Virtual Server Gallery. To apply the web part there are some basic rules which must be met.

Remember the web part must have a view called 'Search' and the web part must be attached on the list's Search view page. Furthermore the appropriate user rights to filter a list must be given to the user.



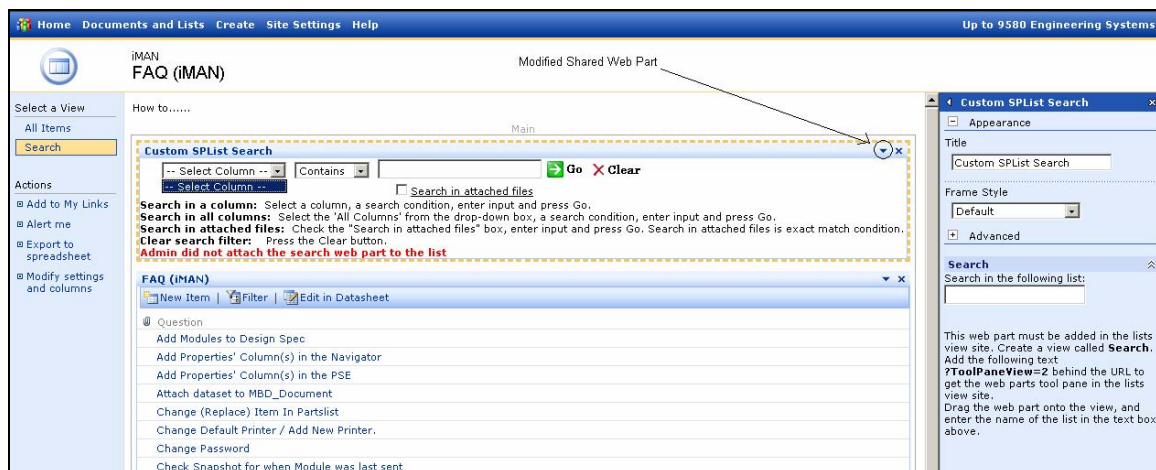
To attach the web part to the list's Search view page I append the following text '*?ToolPartView=2*' to the end of the URL.

```
http://<servername>/sites/<sitename>/Lists/<List Name>/Search.aspx?ToolPaneView=2
```

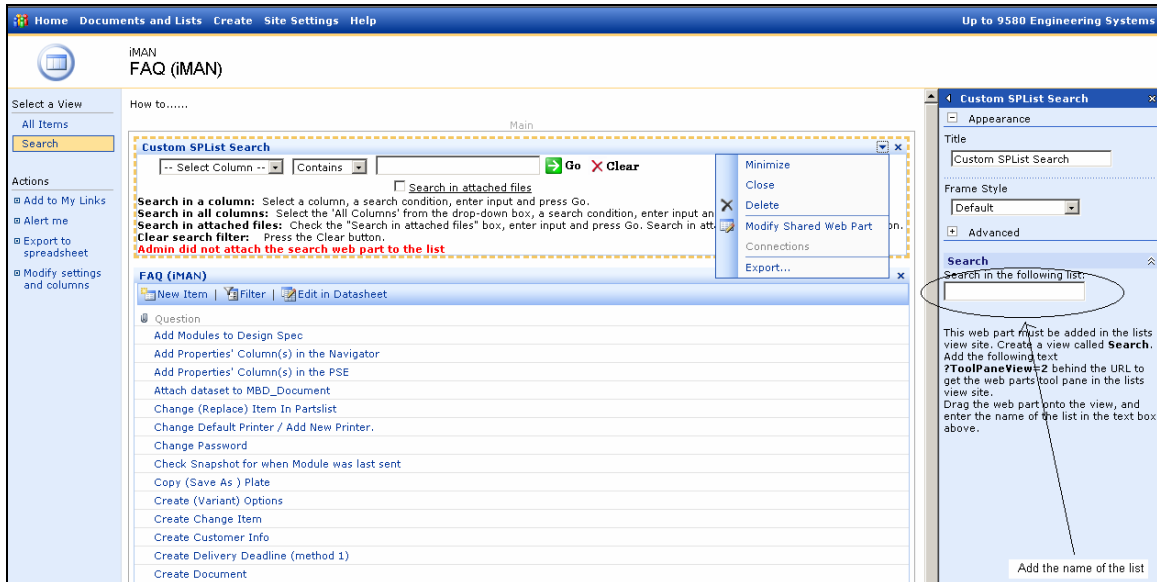
This will show the web part tool pane from where web parts can be added to the page.

The web part lies in the Virtual Server Gallery. I now drag the web part called 'Custom SPLIST Search' from the Web Part List onto the page.

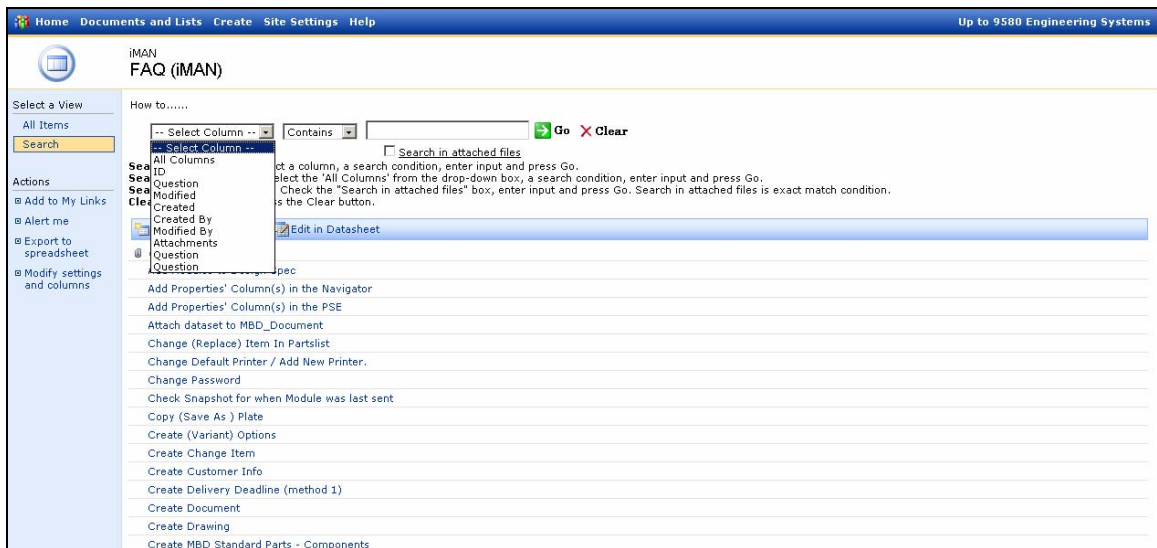


When dragged onto the page a message appears (in red) which tell admin or user that the search web part must be 'attach' to the list, in this case the FAQ (iMAN) list.

To 'attach' the list I must modify the web part. The web part is modified by selecting the 'Modified Shared Web Part' in the search bar. See picture above.

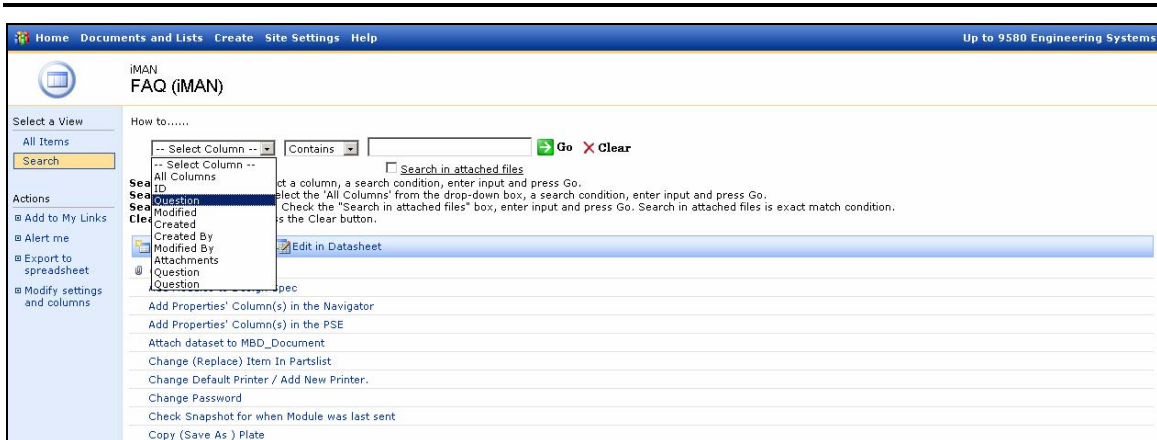


This will show the search web part tool pane. In the text box I need to add the name of the list and apply before the search web part is 'attach' to the FAQ (iMAN) list.



Now the search web part has been 'attached' which can be seen by the fact that the 'column' drop-down list has been filled.

To search in one column I select the Question column from the drop-down list and select a search condition (Contain or Start with). In this case I type the following input ("part") in the textbox and hit the 'Go' button.



This request resulted in all the items which contained the input “part” in the Sharepoint list. To clear the list and see all items again I hit the clear button.

To search in more than one column I select the ‘All Columns’ and a search condition from the drop-down lists. Of course this is not possible in the ‘FAQ (iMAN)’ list since it only contains one column. So instead I perform the search on a task list.

The screenshot shows the iMAN Tasks page with a search for 'progress'. The search results table is as follows:

Title	Status	Priority	Assigned To	Due Date	% Complete
Log Engine info i OPERA	In Progress	(1) High	Jens Chemnitz	5/25/2007 12:00 AM	60%
Design af løsning til Hotspot	Not Started	(2) Normal	Hans-Henrik Rindum	6/22/2007 12:00 AM	0%
Analyse af MBD Add. Insert of item i PSE-BOM	Completed	(3) Low	Jens Chemnitz	3/12/2007 12:00 AM	100%
Replace 'INCOMPLETE_MODULES'	In Progress	(2) Normal	Harbo Poulsen	4/26/2007 12:00 AM	40%
Release step af dataset i E&S projektet	In Progress	(2) Normal	Hans O. Mortensen	3/30/2007 12:00 AM	50%
ERL	In Progress	(3) Low	Bo G. Nielsen	3/30/2007 12:00 AM	70%

I type the input (“progress”) and hit the ‘Go’ button, and get a number of result in the ‘Status’ column and one result in the ‘Title’ column.

The screenshot shows the iMAN Tasks page with search results for 'CADAM'. The search results table is as follows:

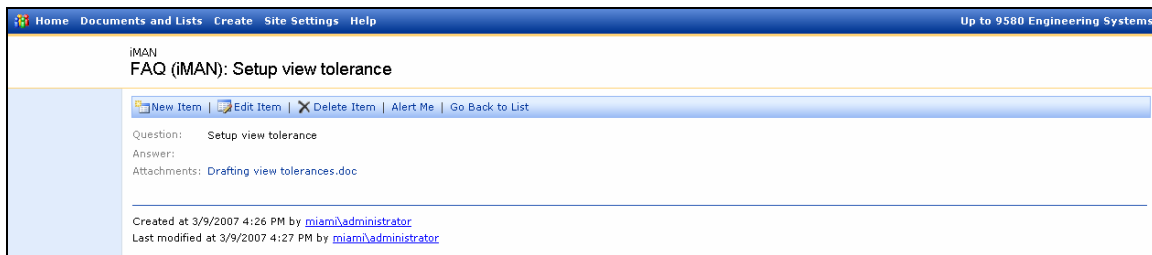
Title	Status	Priority	Assigned To	Due Date	% Complete
Data fra iMAN til Skibs- og Engine info i OPERA	In Progress	(1) High	Jens Chemnitz	5/25/2007 12:00 AM	60%
Replace 'INCOMPLETE_MODULES'	In Progress	(2) Normal	Harbo Poulsen	4/26/2007 12:00 AM	40%
Release step af dataset i E&S projektet	In Progress	(2) Normal	Hans O. Mortensen	3/30/2007 12:00 AM	50%
ERL	In Progress	(3) Low	Bo G. Nielsen	3/30/2007 12:00 AM	70%
Kommunikationen med TCS	In Progress	(2) Normal	Jens Chemnitz		80%
Udtræk fra SAP	In Progress	(1) High	Lars Nikolajsen	4/18/2007 12:00 AM	70%
ASPI load	In Progress	(2) Normal	Frands H. Bromose	4/28/2007 12:00 AM	60%
Omlægning af iMAN-SAP interface	In Progress	(1) High	Gert M. Morton	6/14/2007 12:00 AM	30%
Progress Module	Not Started	(2) Normal	Frands H. Bromose	5/31/2007 12:00 AM	0%

The last search options are to search in the items attached files. To do this I check the small box (Search in attached files) below the text box and enter the input (“CADAM”).

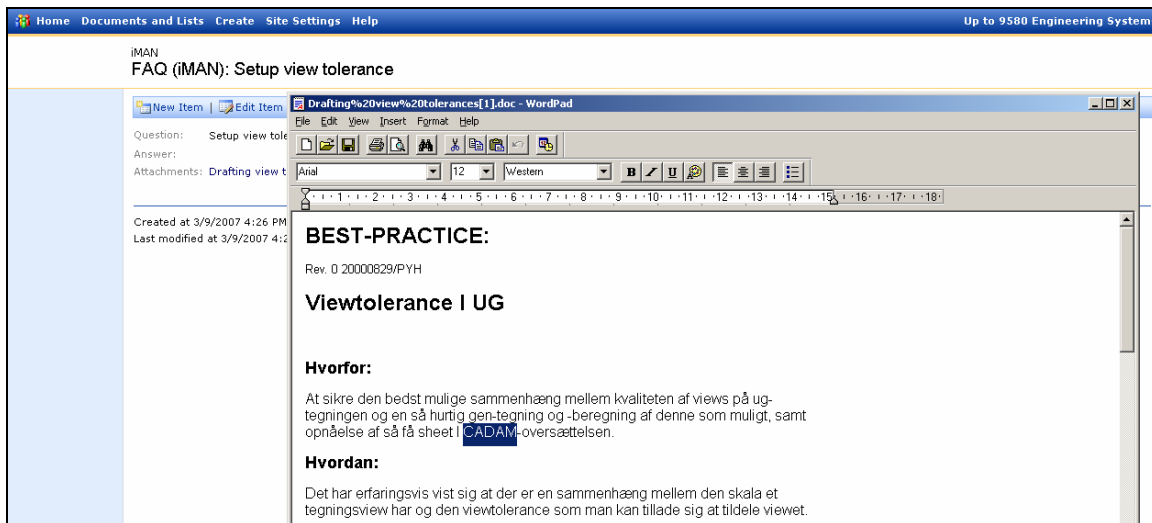
The screenshot shows the iMAN FAQ (iMAN) page with search results for 'CADAM'. The search results table is as follows:

Title	Status	Priority	Assigned To	Due Date	% Complete
Question					
Setup view tolerance					

This gave me the following result (see picture above) an item which contained a document called - Drafting view tolerances.doc (see picture below).



From the picture below, it is clear that the input (“CADAM”) I gave the search web part matches the content of the document.



CHAPTER 10

Conclusion

10 Conclusion

10.1 Purpose

The purpose of this project was to create a web part for MD DEPT 9580 which was able to search directly in a Sharepoint list, which took care of the problems that occur when a list reaches an unmanageable size.

Besides developing a web part MD DEPT 9580 asked me to look into the regression test techniques in Visual Studio 2005 .NET Team Suite to see which tools the test environment provides to find regression bugs.

10.2 Summing up

For this project I worked according to the UP process which is an iterative and evolutionary process where functionalities gradually are implemented into the system.

Together with DEPT 9580's requirements for the search web part was gathered and rated according to their importance to the project.

A detail analysis of each requirement was performed with the help of UP artifacts such as Use Cases text and system sequence diagrams, and a conceptual model was established to get a high level abstract of the system

With class diagram and interaction diagrams I made the transition from analysis to design and created a more concrete software specification of the system.

System was implemented and software classes were described in detail. Use Case 7 was not implemented as MD DEPT9580 and I found this feature unnecessary due to the fact that Use Case 6 covered it more or less.

The different test tools in VS05.NET TS were analyzed to find a regression test tool, which could be used on the web part. The unit test tools in VS05.NET TS was used on the web part to find regression bugs and the unit test was created with help from the UP artifact Contracts. The web test tool in VS05.NET TS was used to test the performance of the web part.

10.3 Evaluation

In my opinion the search web part delivers a fine and concrete solution to a significant problem in Sharepoint. A list, which reaches a certain size, will contain so many items that users eventually will spend a long time on finding a specific item. This is not very user-friendly and then you would think that the provided search functionality in Sharepoint would help, this is not the case, the search functionality in Sharepoint is a pain and simply not good enough. Often when using the Sharepoint search you will experience that you receive a lot of 'garbage' results which you then need to filter to actually find some results which can be used.

With the search web part the "long" time previously spent on finding a specific item in a list is over. My web part is attached to a list view which then allows users to search in an exact list either by searching in one column, all columns or in the attached files of the list.

Concerning the supplementary requirements I have added explaining text in the web part that should increase the usability, which already is fairly easy to master without much knowledge. The web part works on the custom made lists and it even works on the pre-defined list in Sharepoint.

In chapter 9 – Test, I made a performance test of the web part and it showed some acceptable response times in the given test scenario, although it also illustrated that there is room for improvement within this area.

To make the web part work on other lists the only thing admin must do is to change the list name in the tool pane on the web part and insure the list to search in is placed on the Sharepoint site. I have made the button and colour of the web part fairly anonym so the web part seems like it is a part of the list.

The unit test tool in VS05.NET TS which can be used to find regression bugs was very helpful when I implemented the Query class. It was very important that the query-strings generated in Query class were created correct since it would otherwise have made the Sharepoint list-view fail. Together with the debug tools I could tell how the actual query-string generated in Query class looked like.

During the project I work according to the way UP proscribes which divide the development process into a number of iterations. In 3.4 - Iteration Plan the number of iteration it took me to implement the web part can be seen.

Although I aimed at organizing my software classes into a three layered architecture where all business logic was in one layer and all user interface related functionalities in another layer etc., it is clear from the code on the CD that it isn't a completely separation.

10.4 Perspective

The perspective of the web part is, it in the moment is deployed on the DEPT 9580 portal web site and is going to be until MD decides to upgrade to Sharepoint 2007. When this occur the web part might need some adjustment to work properly on Sharepoint 2007.

10.5 Future Improvements

Of the future improvements I have in mind a total separation of the code in a user interface layer, a business layer and a data access layer is something which I rate high.

Beside this I believe that the following improvements should be added in the future;

- **Sharepoint 2007** – When MD in a likely near future goes from Sharepoint 2003 to 2007 I imaging that it might be necessary to make some adjustment to the web part.
- **Search performance** – To improve the search web part their might be seconds to gain by employing asynchronous techniques in the web part.
- **Search result priority** – The results received from a search should be sorted so the most viewed result appears in the top of the list.
- **User right independent** – Due to the way Sharepoint works you need certain rights to filter a list. The web part shouldn't depend on which user-rights an employee has. When user hits the button he must be given the appropriate rights in that moment.
- **View independent** – The web part shouldn't be restricted to only work on one of the lists view pages, or on a view called 'Search'. In the tool pane where the list name is set it should be possible to set name of the view.

Literature list

11 Literature list

- (1) Applying UML and Patterns
An introduction to Object-Oriented Analysis and
Design and Iterative Development
Third Edition
Craig Larman
- (2) Unified Process
http://en.wikipedia.org/wiki/Unified_Process
(January 2007)
- (3) Using RUP/UP: 10 Easy Steps
A Practical Guide
Software Development Best Practics
<http://www.x-tier.com/public/RUPUPIn10EasySteps.doc>
(January 2007)
- (4) What, no supplementary specification
FURPS+
<http://www-128.ibm.com/developerworks/rational/library/3975.html>
(January 2007)
- (5) Using SharePoint's SPView Class and CAML as a Query Language
http://www.devx.com/dotnet/Article/31762?trk=DXRSS_LATEST
(January 2007)
- (6) Collaborative Application Markup Language
http://en.wikipedia.org/wiki/Collaborative_Application_Markup_Language
(January 2007)

Appendix

Appendix A

Guidance for Use Case Template

Document each use case using the template shown in the Appendix. This section provides a description of each section in the use case template.

Use Case Identification

Use Case ID

Give each use case a unique integer sequence number identifier. Alternatively, use a hierarchical form: X.Y. Related use cases can be grouped in the hierarchy.

Use Case Name

State a concise, results-oriented name for the use case. These reflect the tasks the user needs to be able to accomplish using the system. Include an action verb and a noun. Some examples:

- View part number information.
- Manually mark hypertext source and establish link to target.
- Place an order for a CD with the updated software version.

Use Case History

Created By

Supply the name of the person who initially documented this use case.

Date Created

Enter the date on which the use case was initially documented.

Last Updated By

Supply the name of the person who performed the most recent update to the use case description.

Date Last Updated

Enter the date on which the use case was most recently updated.

Use Case Definition

Actors

An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks. Different actors often correspond to different user classes, or roles, identified from the customer community that will use the product. Name the actor that will be initiating this use case and any other actors who will participate in completing the use case.

Trigger

Identify the event that initiates the use case. This could be an external business event or system event that causes the use case to begin, or it could be the first step in the normal flow.

Description

Provide a brief description of the reason for and outcome of this use case, or a high-level description of the sequence of actions and the outcome of executing the use case.

Preconditions

List any activities that must take place, or any conditions that must be true, before the use case can be started. Number each precondition. Examples:

1. User's identity has been authenticated.
2. User's computer has sufficient free memory available to launch task.

Postconditions

Describe the state of the system at the conclusion of the use case execution. Number each postcondition. Examples:

1. Document contains only valid SGML tags.
2. Price of item in database has been updated with new value.

Normal Flow

Provide a detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions. This dialog sequence will ultimately lead to accomplishing the goal stated in the use case name and description. This description may be written as an answer to the hypothetical question, "How do I <accomplish the task stated in the use case name>?" This is best done as a numbered list of actions performed by the actor, alternating with responses provided by the system. The normal flow is numbered "X.0", where "X" is the Use Case ID.

Alternative Flows

Document other, legitimate usage scenarios that can take place within this use case separately in this section. State the alternative flow, and describe any differences in the sequence of steps that take place. Number each alternative flow in the form “X.Y”, where “X” is the Use Case ID and Y is a sequence number for the alternative flow. For example, “5.3” would indicate the third alternative flow for use case number 5.

Exceptions

Describe any anticipated error conditions that could occur during execution of the use case, and define how the system is to respond to those conditions. Also, describe how the system is to respond if the use case execution fails for some unanticipated reason. If the use case results in a durable state change in a database or the outside world, state whether the change is rolled back, completed correctly, partially completed with a known state, or left in an undetermined state as a result of the exception. Number each alternative flow in the form “X.Y.E.Z”, where “X” is the Use Case ID, Y indicates the normal (0) or alternative (>0) flow during which this exception could take place, “E” indicates an exception, and “Z” is a sequence number for the exceptions. For example “5.0.E.2” would indicate the second exception for the normal flow for use case number 5.

Includes

List any other use cases that are included (“called”) by this use case. Common functionality that appears in multiple use cases can be split out into a separate use case that is included by the ones that need that common functionality.

Priority

Indicate the relative priority of implementing the functionality required to allow this use case to be executed. The priority scheme used must be the same as that used in the software requirements specification.

Frequency of Use

Estimate the number of times this use case will be performed by the actors per some appropriate unit of time.

Business Rules

List any business rules that influence this use case.

Special Requirements

Identify any additional requirements, such as nonfunctional requirements, for the use case that may need to be addressed during design or

implementation. These may include performance requirements or other quality attributes.

Assumptions

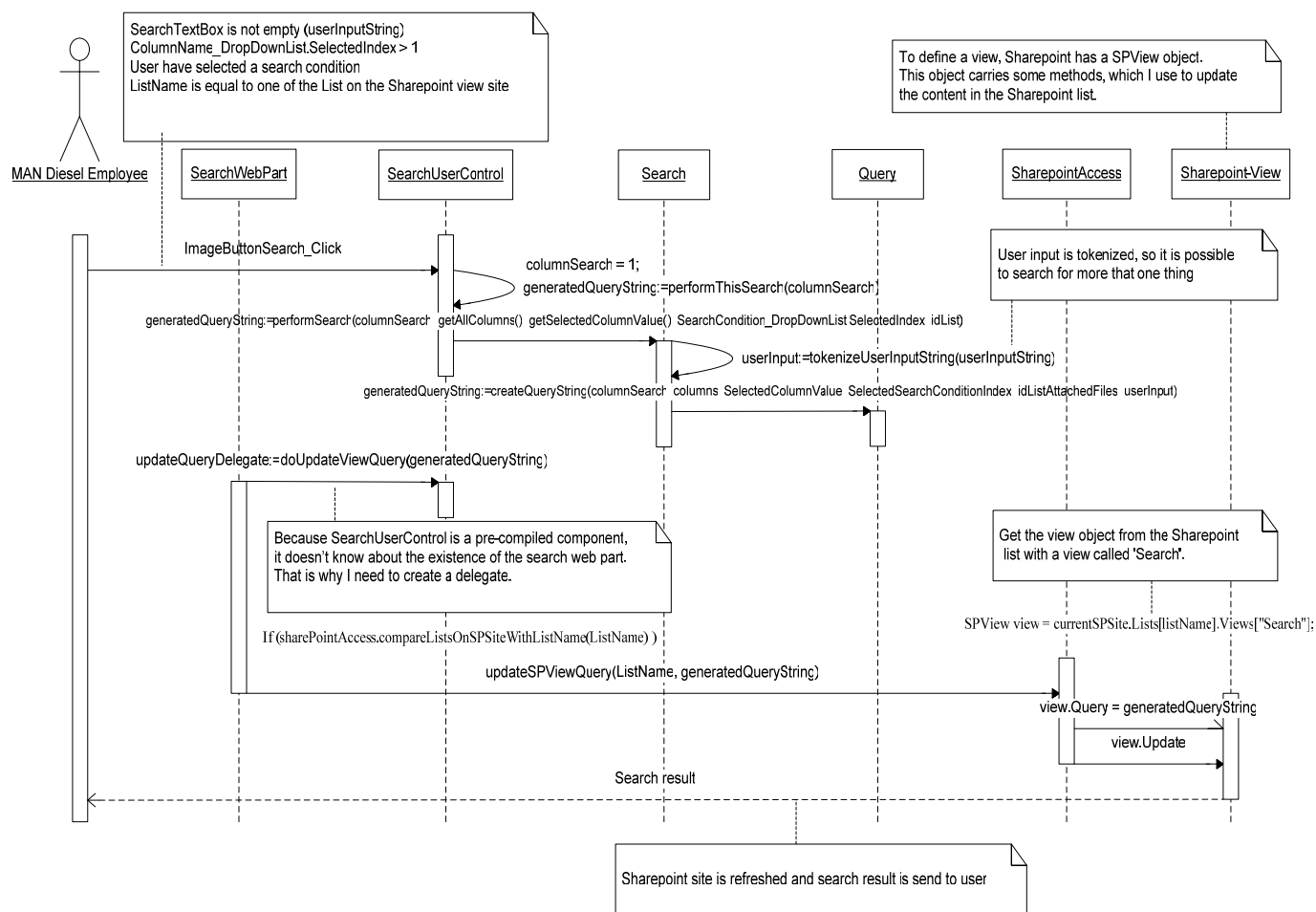
List any assumptions that were made in the analysis that led to accepting this use case into the product description and writing the use case description.

Notes and Issues

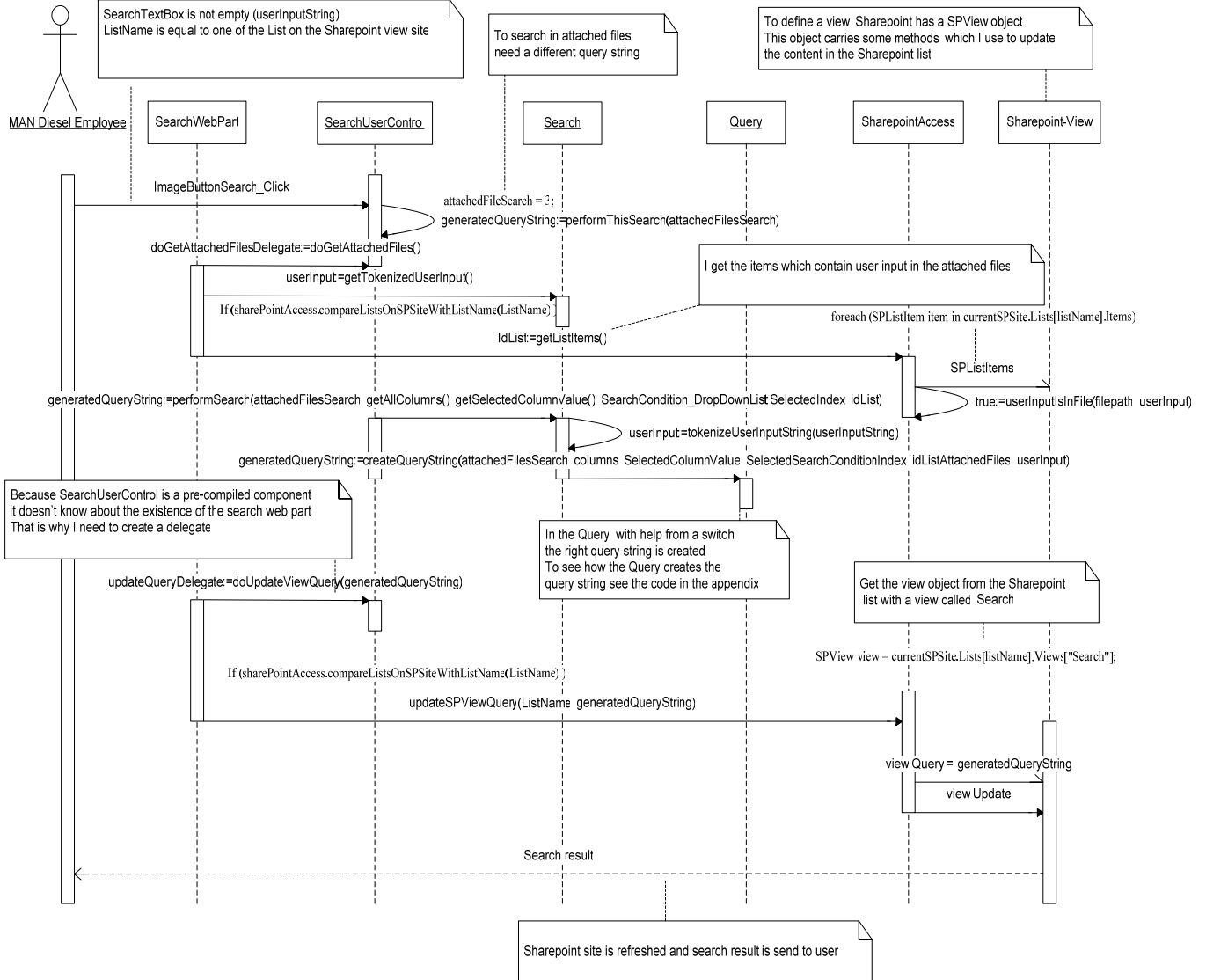
List any additional comments about this use case or any remaining open issues or TBDs (To Be Determineds) that must be resolved. Identify who will resolve each issue, the due date, and what the resolution ultimately is.

Appendix B

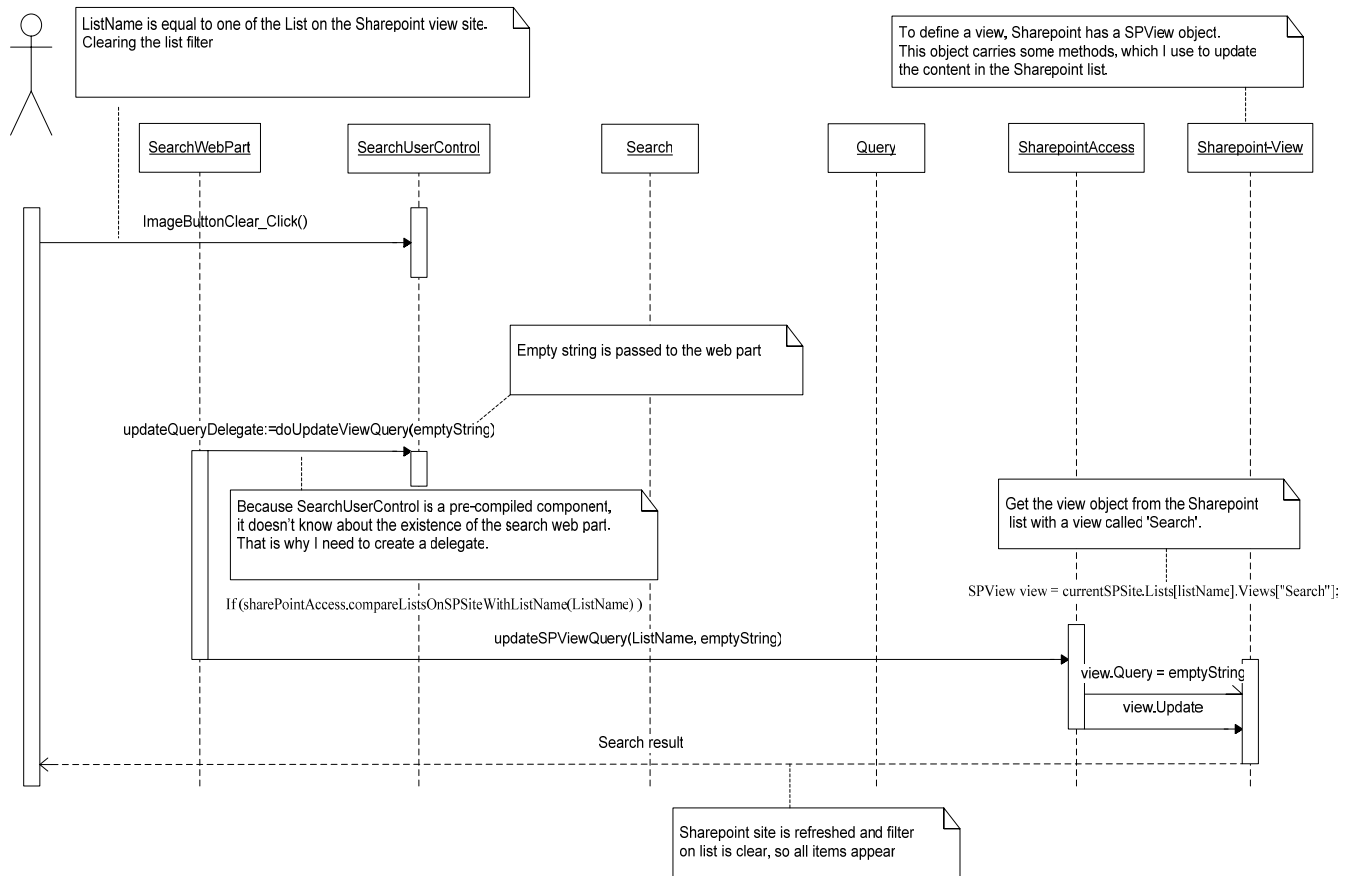
Interaction diagram Use Case 2



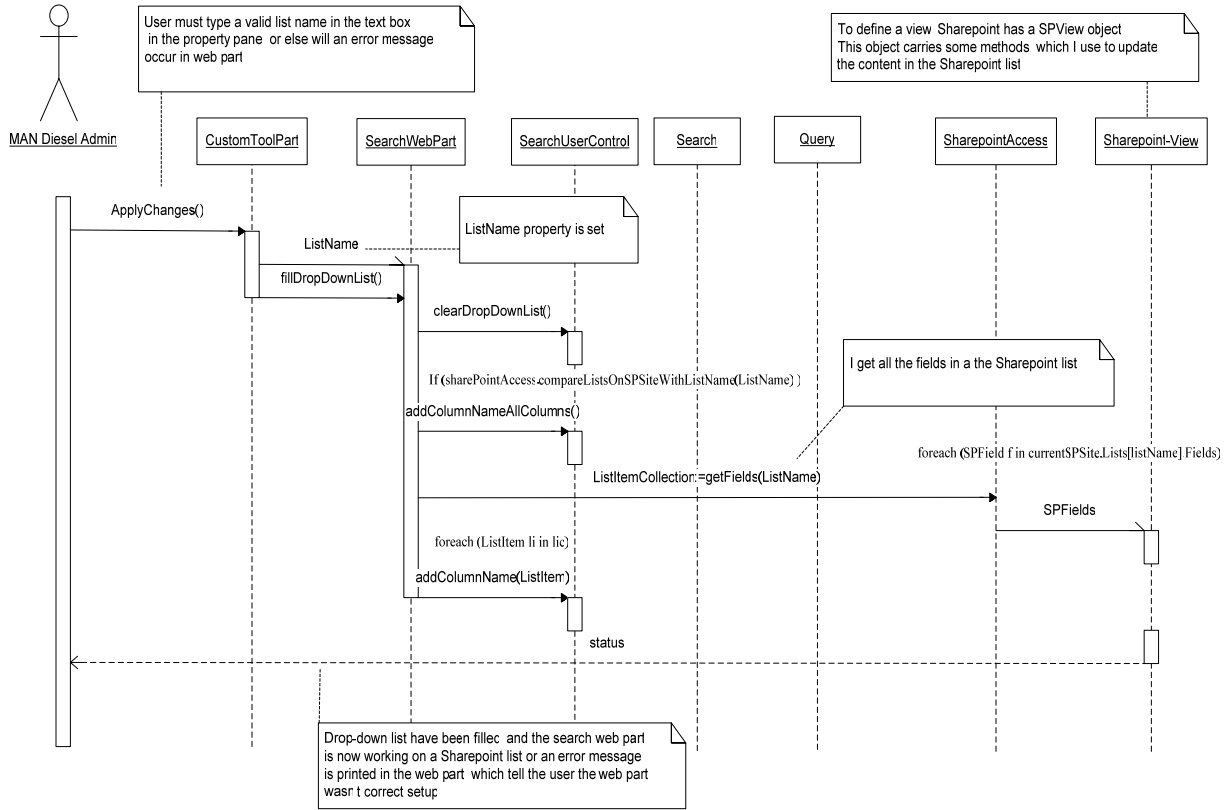
Interaction diagram Use Case 3



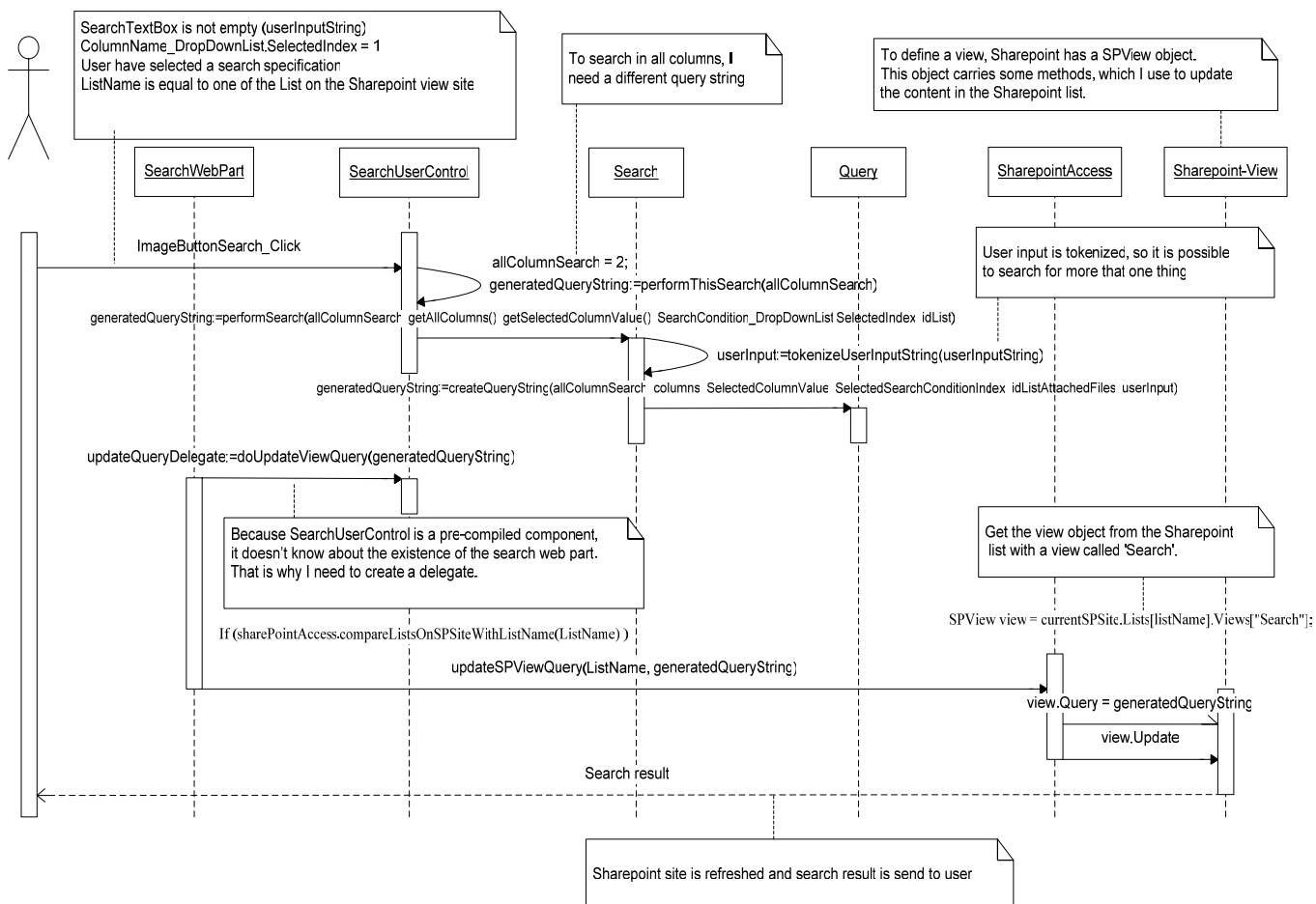
Interaction diagram Use Case 4



Interaction diagram Use Case 5



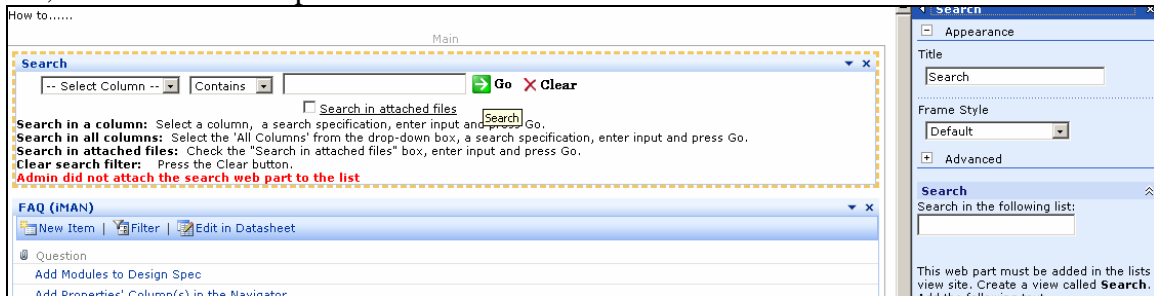
Interaction diagram Use Case 6



Appendix C

Test

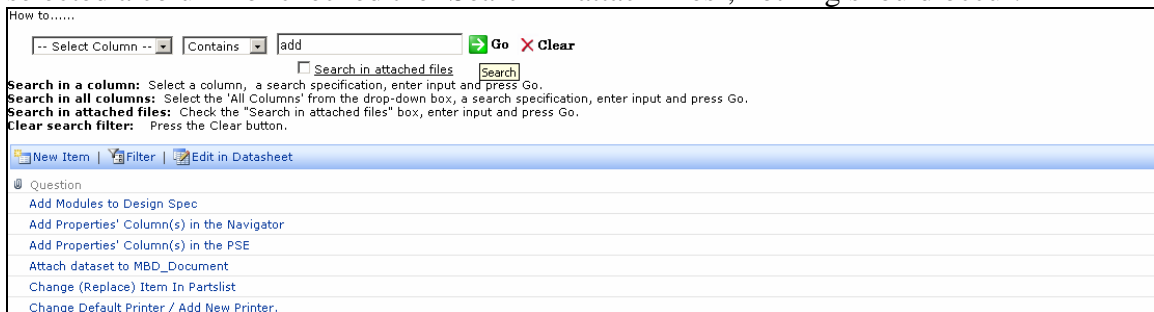
During the phase where user has drag the web part onto the site, and hasn't type anything in tool pane, and tries to hit the go and clear button nothing is supposed to happen in the list, as the search web part isn't attach to the list.



As expected nothing occurs in list below. If user enters a list name which doesn't exist the same will occur. A message is printed in the web part which tells the user that the web part hasn't been attached to the list.



When the web part has been attach correct, and if user tries to enters some text, but hasn't selected a column or checked the 'Search in attach files', nothing should occur.



As nothing occurs in list, since a column hasn't been selected.

Appendix D

Contracts

Contract Query: createQueryString

Operation:	generateColumnSearchQueryString(Selected query-string : integer, Columns : List<string>, Selected column value : string, Selected search condition index : integer, ID list : List<int>, User input list : List<string>)
Cross Reference:	UC1 - Search in List, UC2 - Search Pattern, UC3 - Search in Attached Files, UC6 - Search in All Columns.
Preconditions:	User needs to search.
Postconditions:	A query-string which follows the CAML query structure is created to search.

Contract Query: generateColumnSearchQueryString

Operation:	generateColumnSearchQueryString(User input list : List<string>, Column name : string, Selected search condition index : integer)
Cross Reference:	UC1 - Search in List, UC2 – Search Pattern.
Preconditions:	User needs to search in one column.
Postconditions:	A query-string which follows the CAML query structure was created to search in one column.

Contract Query: generateAttachedFilesSearchQueryString

Operation:	generateAttachedFilesSearchQueryString(ID list : List<int>)
Cross Reference:	UC3 - Search in Attached Files.
Preconditions:	User needs to search in the attached files.
Postconditions:	A query-string which follows the CAML query structure was created to search in the attached files.

Contract Query: thisQueryStringWillNeverMatch

Operation:	thisQueryStringWillNeverMatch()
Cross Reference:	UC3 - Search in Attached Files.
Preconditions:	User enters some input which doesn't match the content in the attached files.
Postconditions:	An empty query-string was created.

Contact Query: generateAllColumnsSearchQueryString

Operation:	generateAllColumnsSearchQueryString (User input list: List<string>, Columns : List<string>, Selected search condition index : integer)
Cross Reference:	UC2 – Search Pattern, UC6 - Search in All Columns.
Preconditions:	User needs to search in all columns.
Postconditions:	A query-string which follows the CAML query structure was created to search in all columns.