Customer driven project, TDT4290

Automatic Import of Completed Ads

Adresseavisen AS

Audun Skjervold
Erlend Løkken Sigholt
Hong-Dang Lam
Truls Hamborg

November 6, 2013

Contents

1 Abstract

This paper describes the process of developing a framework for Automatic Importing of Completed Ads, on request from Adresseavisen AS, which is a major regional newspaper. It will describe the project, from the preliminary study phase and project management/organization, to implementation and completion.

The goal is to create a framework which will facilitate automatic importing of completed real estate ads in the form of pdf-files and associated data, with possibility to expand to other types of ads. The accompanying data will be placed into the internal database of Adresseavisen and their internal order system for ads, and the pdf is to be saved in their archives.

Adresseavisen, as a customer, is interested in the framework in itself, with accompanying developer documentation for possible further development. In addition to this, we will produce full documentation of the process as a deliverable in the course which this project is for.

2 Preface

This document was written for the project-based course *TDT4290 Customer Driven Project*, at the *Norwegian University of Technology and Science, NTNU* during the fall semester of 2013.

The project team consisted of four students from the Department of *Computer and Information Science* at *NTNU*. The project was *Automated Importing of Completed Ads*, on behalf of *Adresseavisen AS*.

The team would like to thank Asle Dragsteen Liebech<insert compliments/thanks> and our supervisor Zhu Meng for spectacular supervising.

3 Introduction

3.1 Project name - Automatic Import of Completed Ads

The project is named *Automated Importing of Completed Ads* by the customer; *Adressavisen AS.* The team is to create a framework for automatic import of real estate ads into Adresseavisens internal database and order system.

3.2 Customer - Adresseavisen AS

Adresseavisen is a regional newspaper in Trondheim, Norway. It publishes it's newspaper in Trøndelag and Nordmøre on a daily basis except for Sundays. It is an independent, conservative newspaper with a daily circulation of approximately 85000 NOK. Stocks in Adresseavisen are traded on the Oslo Stock Exchange.
"In addition to the main newspaper, Adresseavisen owns several smaller local newspapers in the Trøndelag region. They also own and operate a local radio station, Radio-Adressa, and a local TV station, TV-Adressa (prior to 30 January 2006: TVTrøndelag). They also have a stake in the national radio channel Kanal 24. In addition, the newspaper owns the local newspapers Fosna-Folket, Hitra-Frøya, Levanger-Avisa, Sør-Trøndelag, Trønderbladet and Verdalingen." [3]

3.3 Project Purpose

The purpose of this project is to create a framework for automated import of complete ads in the form of pdf-files. The pdf-files will be accompanied by data, which shall be added to the database, and used to create an order in Adresseavisens internal order system for ads. This framework shall also be able to provide certain information that is to be given to customers when they wish to have an ad featured in Adresseavisen, such as available dates and ad-packages.

3.4 Project background

Adresseavisen currently uses a system where ads to be featured on website, or published in newspapers are created in the system, with the use of one of very many templates. This is very complex and an increasing amount of customers uses their own fully completed ads, in the form of pdf-files. Adresseavisen now has the need for a system which can automate and simplify the process of receiving these ads into the system, with the purpose of eventually replacing the old system.

3.5 Problem Description

Webassistenten will take a completed ad in the form of a pdf file and save it to the appropiate folder, these folder will have the correct ID of the customer so adressa can easily identify which pdf belongs to which customer. Webassistenten will require certain data along with the pdf, ie. date, module etc. These data will be saved into a database internally in their system.

3.6 Existing Technology

<Already existing technology that the customer could have used rather than employing us to develop new software.>

3.7 Stakeholders

Customer:
Adresseavisen AS

Customer representatives:
Asle Dragsten Liebech - asle.dragsten.liebech@adresseavisen.no
Jostein Danielsen - josetein.danielsen@adresseavisen.no

Group Members:
Audun Skjervold - <email>
Erlend Løkken Sigholt - <email>
Truls Hamborg - <email>
Hong-Dang Lam - <email>

Group Supervisor:
Meng Zhu - zhumeng@idi.ntnu.no

3.8 Measure of Project Effects

The completed ads produced through our product will not be distinguishable from completed ads produced without it. For this reason, we will not see any effects of this project.

We do however hope that Adressa and their customers will see effects in the form of time and money saved. We also believe that the real estate agents to a larger extent will find the submitted ads being accepted in the format they want them to be.

3.9 Duration

The introduction to the course started on Wednesday 21.08.2013 when we all met in S6 for information and were introduced to the customer and the project. The final delivery is due on 22.11.2013.

4 Project Management

4.1 Terms and Resources

4.2 Project Organization

4.3 Roles

Role	Name	Responsibility
Scrum Master	Truls Hamborg	Scrum master/process manager leads the stand-up meeting. Makes sure the team follows the agreed-upon process.
Documentation Manager	Truls Hamborg	Ensures documentation coherency.
Product Owner (scrum role)	Adressa	The customer, owns the product
Lead coder	Erlend Sigholt	Oversees coding conventions and makes sure we follow the "clean coding" principles.
Secretary	Erlend Sigholt	Takes notes during meeting(s)
Implementation manager	Audun Skjervold	Takes care of learning the framework and implementation of the code.
Customer and supervisor contact	Hong-Dang Lam	Contact-person for customer and supervisor
System Architect	Hong-Dang Lam	Create and maintain the architecture of the system

Test Manager: Everyone is responsible for testing their own code.

4.7 Issues

The project didn't go as smoothly as we had hoped. The group agreed on pretty much everything, but the issues we had was mostly related to the software and tools not cooperating.

4.7.1 C# Compiler

C# compiler was not found due to uninstallìng of RC, this lead to the "add new" window not opening, and therefore not possible to add the entity framework. This command in cmd fixed it:

```
gacutil /u Microsoft.VisualStudio.CSharp.Services.Language.Interop
```

4.7.2 GitHub DDoS

On October 14, the day of the midterm delivery, GitHub experienced a DDoS attack. This resulted in some of the files we were working on for the report becoming locked mid-commit and mid-pull, and preventing us from performing a new git pull. This also prevented some of us from compiling, due to a previous commit containing errors having been pulled.

4.7.3 Windows on Mac

Mac is not Windows.

4.7.4 SDM

SDM

4.7.5 MSSQL

DB

4.8 Internal Risks

4.8.1 Low experience with the development process

What:	The group doesn't have much experience with lengthy projects
Probability:	High
Impact:	Low-High
Action:	Reduce by regularly reviewing progress and making use of supervisor meetings etc.

4.8.2 Unfamiliarity with technology

What:	Some members of the group have no experience with the programming language in use, and only one has used the full extent of the relevant framework. Will require extra time for learning.
Probability:	Moderate
Impact:	Low-Moderate
Action:	Reduce by group members with experience coaching others in the relevant technology and practices. Set time aside for learning.

4.8.3 Unfamiliarity with tools

What:	Some members of the group have no experience with the tools we're using, it's complicated to install due to the different versions of the tools & there's so many tools needed to start.
Probability:	Moderate
Impact:	Low-Moderate
Action:	Reduce by group members with experience coaching others in the relevant technology and practices. Set time aside for learning. Debugging via google.

4.8.4 Illness

What:	As winter approaches the probability for illness in the group increases. Being a small group, this might be critical.
Probability:	Moderate
Impact:	Low-High
Action:	Do not tailgate deadlines.Work steadily, and have margins and practices. Expect some members to not produce at 100% every week

4.8.5 Other engagements

What:	Group members have extracurricular activities that require time certain dates during the semester. This might cause reduced work-output and/or absence.
Probability:	Moderate
Impact:	Low
Action:	Plan for it well in advance. It seems however that the time required is fairly concentrated and/or pre-planned, so it should be easy to plan around.

4.8.6 Other subjects

What:	Customer driven projects isn't the only subject we're assigned to this semester, the other subjects might have projects too so we might not be able to allocate enough time for the project.
Probability:	High
Impact:	Moderate
Action:	Try to plan for it in advance. All the courses have clear deadlines and we should be able to plan ahead.

4.8.7 Underestimation of implementation

What:	We have estimated that the implementation scope is not a significant majority of the project, and possibly even smaller than the process and documentation parts. If we somehow have misjudged this, we will face significant delays/increased workload when the plan will have to be adjusted.
Probability:	Low
Impact:	High
Action:	Ensure to plan for more time for the project than we expect to need. Do not rush the pre-study, and familiarize ourselves sufficiently with all aspects. Ensure we have time to work overtime if necessary during the implementation period/sprints.

4.9 External Risks

4.9.1 Deaths

What:	There's always a chance of some family members dying.
Probability:	Low
Impact:	High
Action:	Be understandful, try to reschedule working hours

4.9.2 Customer

What:	Customer might be on vacation, in a meeting or not able to respond instantly. Database setup
Probability:	Medium
Impact:	Low
Action:	Wait and call again or send and email or short message service Set up our own local database

4.10 Procedures for Quality Assurance

5 Preliminary Studies

This section contains the information we found in our preliminary studies, and the choices we made from the information we gathered. Covered here are process-related topics such as development methodology, pros and cons of these, and which we chose, as well as technology-related topics such as frameworks and standards.

5.1 Development Methodology

The course compendium proposes two types of development methodologies: the sequential method *Waterfall*, and the agile method *Scrum*. This subsection supplies a brief introduction to these two approaches, followed by our argumentation for and against the two approaches in the case of our particular project. This subsequently followed by a conclusion as to which approach(es) we chose for our project.

5.1.1 Waterfall

The Waterfall development method is a sequential design process. It is divided into clearly defined, mostly separated phases, although there often is some overlap between them. The first phases focus on gathering requirements and writing initial documentation like design/architecture. Later phases move on to actual implementation, then testing, followed by final report. Maintenance after a "completed" project might also in some cases be part of the process. <add figure depicting process>

5.1.2 Scrum

The scrum development method is an agile approach to development. It is an iterative process consisting of several cycles, each containing most of the phases of a sequential method, and resulting in a functioning prototype. <add figure depicting process>

5.1.3 Our choice

Each of the proposed methods have their own advantages and disadvantages, that make each of the methods respectively a better or worse fit for our project than the other.

While the different methods have various advantageous features, not all of these are applicable to our particular project, and others are negligible.

Below we have highlighted the most important advantages and disadvantages of each method, while discussing whether or not the particular point is relevant to our project, leading to a conclusion and our choice of method.

Waterfall

The following points are advantages of using the Waterfall method for projects, both in general and for ours specifically.

- Waterfall is suitable for small projects because they are manageable to fully plan. This fits our project description.

- It is easier to get every involved party on the same page with a thorough plan, such as provided through the Waterfall method's early phases. This is beneficial to any project, but even more so in one such as ours, where the team consists of students who may have other projects in other courses running in parallel with this.

These next points are generally advantages of the Waterfall method, but are mostly not applicable for our project for various reasons.

- Waterfall is a good method if you know everything about the project beforehand, or are able to acquire the required information and the full project specification before the implementation phase. Due to the course schedule and the relatively limited time frame of our project, we felt we needed to start implementation earlier than a long planning and requirements phase would allow.

- Sequential methods require little underway feedback. This give them the edge over agile methods when access to the customer is restricted but specifications are expected to remain the same. While the specifications for our project were expected to remain unchanged, our goal was to include the customer in the process.

- The method supplies strong documentation as the first phases are focused entirely on creating these documents. However, while the course relies heavily on documentation, the customer had no use for most of this documentation, making this point less important for our project.

Lastly we have the biggest disadvantage of following a sequential method, which applies to any project doing so.

- Sequential methods don't handle change to the requirements particularly well, making this approach risky in the case of the customer wishing to modify the requirements during the implementation phase.

Scrum

Advantages of agile methods such as scrum include the following:

- Agile methods support rapid production of prototypes to present to the customer. This allows for easier correction of misunderstandings, because they become apparent earlier in the process through the functioning prototypes.

- The method utilizes stand-up meetings, a scrum master, and optionally (and preferably) a kanban/scrum board. These things do carry overhead, but provide both the team and customer with frequent feedback, making the benefit much greater than the disadvantage of the overhead.

- The approach is highly supported by online tools - e.g. Trello for the kanban board - that let both parties (the team and the customer) stay up-to-date on the planning and prioritization of tasks during implementation.

While the next point generally is an advantage of agile development over a sequential process, it does contain some risks for a small project such as ours, as explained below.

- Agile methods handle change to requirements very well, due to the high level of underway involvement of the customer. The risk of weighting this point when deciding on an approach is the possibility that there simply will not be many changes, making this point irrelevant. Because we expected there to be necessary changes to the requirements and specification during the implementation, we decided this was important for our project.

There is one major disadvantage - or rather, risk - of choosing an agile approach.

- Agile methods rely on team members - or at least the project manager - having experience with the full development process, assuming the entire project is to be completed in agile fashion.

There is another important point when using an agile method, which can be the deciding factor for some projects.

- Iterative approaches are heavily reliant on easy access to the customer for continuous feedback and extraction of requirements. This is okay for our project, as our customer is readily available through both e-mail and phone.

From our analysis of the two methods, we concluded that neither were a perfect fit for every part of our project. We felt that our inexperience with projects such as this one made it too difficult to complete the entire project through an agile process, but we felt too unsure about the scope of the project to plan everything ahead and do a straight sequential process. We did, however, wish to plan an outline for the project, create a general architectural overview and gather the most important requirements before we started implementation.

This led us to a decision of employing the waterfall method, or at least something similar, for the complete process, with a relatively long period of planning before starting implementation.

The customer suggested using an agile approach for implementation because it is the same as they use. Using the same method as the customer might be beneficial to the project, as it improves communication and workflow. We decided to complete the implementation phase as an agile process, divided into two-week sprints, each consisting of a sprint planning meeting, then several days of implementation, with semi-daily stand-up meetings, and ending in a sprint review meeting and a functioning prototype.

5.2 Frameworks

5.2.1 Software Development Model

5.2.2 Programming languages, Communication Protocols and File Formats

5.2.3 Web Api

5.2.4 Server

5.2.5 Database

We had a lot of problems with the database, the customer said he'd put up a database server for us to use. However this turned out to be difficult to achieve due to the security/confidentiality issues on their server. We had to try to set up the database server ourself on our local machine, this proved to be rather difficult due to microsoft tools not giving enough information, and thereby not installing the necessary tools. The customer uses a MSSQL server and they provided an 11GiB .bak file which we could use to restore the database.

5.3 Technology

This subsection covers the technology we utilized during our project. It covers the various operating systems our group members have been working on, editors and IDEs for documents and code, as well as frameworks and platforms used in development. Additionally it covers tools for version control and file sharing between the group members, as well as communication platforms.

5.3.1 Windows 7, 8

Microsoft Windows 7 and - 8 are operating systems by the Microsoft Corporation. They logically provide good support for .NET developments, seeing as .NET targets the Windows platform and is made by Microsoft. Visual Studio is made for Windows, and was our main IDE, so all group members had access to PCs with Windows installed.

5.3.2 Ubuntu Linux

This OS is perhaps the most widely used distribution of Linux, developed by Canonical Ltd. It provides good support for many development tools, except of course Windows development. However we did find support for using it for some Windows development. This operating system was used by one group member on a laptop, when working on-site at NTNU. For coding, Mono with MonoDevelop was used, while other tasks were mostly unaffected. The operating system provides good support for other parts of the process, such as Git and LaTeX.

5.3.3 Mac OSX

5.3.4 Visual Studio 2012

Visual Studio is Microsofts IDE for development for their platforms. This is the main IDE we developed the framework on, seeing as it has very good integration with C# and .NET platforms, which we were required to use. We decided to use the ultimate version because this version provides everything we might need.

5.3.5 .NET and Mono

We were required to use ASP .NET MVC for our framework. ASP .NET MVC is a framework for web applications which enables the use of the Model View Controller (MVC) pattern. It is part of Microsofts .NET Framework suite, which is the preferred way of interactiong with Windows systems and OSes.

Mono is the open source-, cross-platform version of the .NET suite, which we used when not developing on Windows machines. It is available both for Windows, OS X, most Linux Distributions, Android, and various other operating systems.

5.3.6 MonoDevelop

This is an open source IDE for development with Mono, available for OS X and most Linux distributions. This was the IDE used when not developing on Windows machines.

5.3.7 Doxygen

Doxygen is a documentation generator that generates software reference documentation directly from source file comments and tags. It supports multiple programming languages, and outputs documentation in several formats, including, but not limited to: HTML, LaTeX,and man pages.

We used Doxygen in our project to generate the API documentation requested by the customer. It was chosen due to easy setup and configuration, support for C#s XML-comments and tags, as well as multi-platform support (Windows, Linux and Mac OSX).

5.3.8 LaTeX

We quickly chose LaTeX for our typesetting, due to it being the de-facto standard for academic typesetting, with good support for both code snippets, tables, references and bibliography.

Most of our group also had at least some experience using it, and some were quite experienced, which made the choice easier.

5.3.9 Git

For our version control and source repository, we chose Git. This because we had most experience with it, and found it easy to set up via GitHub (where we all had accounts already). It also has the advantage of being distributed, so we could avoid a single point of failure, and having a staging area where one can selectively commit files according to whether they're ready or not, instead of risking accidental changes which might break something.

Both the source code and the entirety of the report source files were stored on GitHub, since both would be catastrophical to lose, and were quite important to have under version control in case we needed to track problematic changes.

5.3.10 Trello

To support our agile process and sprints, we used Trello for planning and control of workflow. It is an online Kanban Board tool, where we can create work packages and issues, while tracking who does what, and tracking backlog, finished modules, and work in progress.

5.3.11 Web API

ASP.NET Web API is a framework for building web APIs on top of the .NET Framework, it lets you create calls from the browser to the methods in the

5.3.12 Dropbox

Dropbox is a syncing service that let you choose a local folder on your machine that will be synced to the cloud. Dropbox lets you share folders and files inside a shared Dropbox folder.

We used Dropbox for sharing and synchronizing internal documents that usually were only useful for a limited time, but might be referenced later.

5.3.13 Google Drive

Google Drive is Google's productivity suite/office pack. The difference between Drive and other office solutions (like Microsoft Office, OpenOffice/LibreOffice) is that Drive exists in the cloud and lets the user simultaneously work on a document.

Google Drive was used for simultaneous collaboration on documents, where the content was up for discussion, or it was advantageous to see what the others where writing, as well as for dynamic internal documents like work logs.

5.3.14 Microsoft SQL server

The customer uses Microsoft SQL server as their database, and we received a backup dump of their database.

5.3.15 Entity Framework

The customer's preference was that we utilized Microsoft's entity framework to connect to the database. The entity framework is an object-relational mapper that enables .NET developers to access the database without writing the typical data-access code that developers typically need to write. EF lets the developer work with domain specific object and properties without worrying about the underlying database table.

5.4 Extra Tools

5.4.1 LaTeX Editors

We have used several editors for our LaTeX documents. Some members have used TeXstudio, others have used TeXworks, while others again have used Gummi.

TeXworks is a simple, lightweight working environment for LaTeX documents, and is modeled on TeXShop. It provides several compilers, and a raw text editor, as well as a pdf viewer, but little else.

TeXstudio, a fork of Texmaker, is better described as an integrated development environment (IDE). Relative to TeXworks, it better supports simultaneous work on several LaTeX documents. It also supplies multiple compilers, a pdf viewer and a text editor, but it includes some nice-to-have features that are not available in TeXworks. Some examples of these features are:

- Easy navigation between included files
- Compilation of main document directly from included files
- View included images through hover

Gummi is a free, open-source, lightweight working environment for LaTeX which is available in the repositories of most widely used Linux distributions, and for Windows as well. It provides live preview of the document without manual compilation, support for multiple compilers, bibliography management, and SyncTeX support. It also provides easy insertion of tables and images.

5.4.2 LaTeX Compilers

We have also used several LaTeX compilers, most notably pdfLaTeX and BibTeX. We have used BibTeX specifically to handle citations and references, and pdfLaTeX for general compilation of our documents.

5.4.3 Git Tools

We have used various tools for Git, such as Git Bash and GitHub For Windows, as well as Git for Linux. GitHub For Windows is a graphical user interface for Git, and provided good visualization of branching, while Git Bash and Git for Linux were used for straight-forward commit, pull and push.

5.4.4 Raw Text Editors

A variety of raw text editors were used, mainly for meeting notes and similar small and temporary documents. Sublime Text was used by most of the group, allowing us to read and edit text files with no file ending. Some also chose to use gedit.

5.4.5 Communication

Various software was used for communication when the group members were at separate locations. Foremost of these were Facebook Messenger and e-mail. We also used IRC, to which some connected through means such as PuTTY, to connect through ssh to a server running irssi in screen, while others ran mIRC locally or connected through ssh in another terminal (such as the native terminal of their OS).

5.5 Templates

We have created the following templates for documents used in the process:

- Foo
- Bar

We have established several standards for the project, as seen in the rest of this section.

5.6 Documents

For internal documents we have established the naming standard:
MM_DD_<Description>_<Version if applicable>
This is to ensure documents are properly sorted, and that they are easily identifiable.

5.7 Coding

We will be using C# as a programming language, and will consequently be following the C# coding standards, as outlined by Microsoft <cite http://msdn.microsoft.com/en-us/library/vstudio/ff926074.aspx here>.

The guidelines are summarized in the following section.

5.7.1 Documentation

All public classes, methods, and preferably properties/fields shall be documented with comments which will enable generation of documentation. Example:

```
/// <summary>
/// This is a summary of what the class contains and
   its intended function
/// </summary>
/// <author>Author Name</author>
public class ExampleClass
{
        /// <summary>
        /// This summary tells what the method does,
           any side-effects, and how/why to use it.
```

```
        /// It should NOT say how the method does what
           it does, unless this is absolutely
           neccessary.
        /// </summary>
        /// <param name="intName">int</praram>
        /// <param name="stringName">String</praram>
        /// <returns>String</returns>
        /// <author>Author Name</author>
        public abstract String ExampleMethod(int
           intName, String stringName);
}
```

5.7.2 Naming and variables

Use CamelCase for classes, method names and properties. Example:

```
public class ExampleClass
{
        public abstract void ExampleMethod(int intName,
           String stringName);

        private int ExampleProperty { get; set; }
}
```

Variables shall be named after the lowerUpper scheme, where the first word is in lowercase, and any others starts with an uppercase letter. Example:

```
int exampleVariable = 1;
int stringExample = "This is an example";
```

5.7.3 Comments and layout

Blocks shall start and end with curly brackets on their own line.

Comments shall have a space between the double slashes and the actual comment. Continuation lines shall be indented. All comments shall start with a capital letter, and end with a period.

There shall be only one statement per line. The same goes for declarations. Parantheses shall be used to separate clauses in expressions, to ease understanding.

```
// This is a single line comment
void Foo()
{
```

```
        // The following is correct:
        int x;
        int y;

        // The following is incorrect:
        int x,y;

        // This is a multi line comment, with more text
           this is line two of a multi line comment


        if(true)
        {
                StatementOne();
                StatementTwo();

                if ((var1 && var2) || (var3 && var4))
                {
                        Bar();
                }
        }
}
```

5.7.4 Variables, types, and declaration

Implicitly typed local variables can be used when the right hand side clearly indicates type, or it's not important.

Use in-line instantiation with constructors when possible, instead of instantiation and assignment.

Short strings shall be appended with the use of the + operator. Longer ones in loops shall use StringBuilder.

Example:

```
// Apparent use of string. Use of var ok:
var name = "SampleString";

// Type inconsequential:
foreach(var v in collection)
{
        //Type-independent method:
```

```
        handleVar(v);
}

// Array instantiation with constructor:
int[] numbers = { 1, 2, 3, 4 };

//Use of var requires explicit instantiation
var numbers2 = new int[] { 1, 2, 3, 4 };

//Avoid this if you could have used the above:
int[] numbers3 = new int[4];
numbers3[0] = 1;
numbers3[1] = 2;
// Etc.

//Short string example
string simpleString = "This is our " + var1 +
   "test-string." + var2 + "something."

//String builder example
string longString = "LongLongLong";
var longBuilder = new StringBuilder();
for(int i = 0; i < 1000; i++)
{
        longBuilder.Append(longString);
}
```

5.7.5 Try-catch, exceptions and using

Exception handling shall be done by try-catch statements. Code shall not unexpectedly throw exceptions; only when something unrecoverable has happened.

In the case of a try-finally statement, a using statement shall be used instead, if the only function of the finally-block is disposing/closing of the used object.

```
Socket socket = new Socket();
try
{
        socket.SomeMethod();
}
finally
```

```
{
        socket.Close();
}
// Can be replaced by:
using (Socket socket = new Socket();)
{
        socket.SomeMethod();
}
```

5.8 Static Members

Static members shall always be called by class name, and never accessed in a derived class when defined in a base class.

5.8.1 Clean Coding

We have also endeavoured to follow the ten Clean Coding principles, as outlined by one extra pixel's post. [2] The ten principles are as following:

1. Revise your logic before coding
2. Clearly expose the structure of the page
3. Use the correct indentation
4. Write explanatory comments
5. Avoid abusing comments
6. Avoid extremely large functions
7. Use naming standards of functions and variables
8. Treat changes with caution
9. Avoid indiscriminate mixing of coding languages
10. Summarize your imports

5.9 APIs

5.9.1 ASP .NET Web API

One of the agreed upon requirements for the project was that we conform to ASP .NET Web API. This to make it easier to interact with our framework from other systems (both existing and future ones). An introduction to using this API can be found at http://www.asp.net/web-api or http://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx.

5.10 Summary

6 Requirements

Key

A = Availability
I = Interoperability
M = Modifiability
T = Technology requirement
F = Other functional requirement

Functional Requirements

ID	Requirements	Priority	Complexity
F1	Receive PDF-files and store them in an appropriate folder	H	L
F2	Insert the data accompanying a submitted ad into the database	H	M
I1	Create an order in the internal order system for a submitted ad	H	M

Functional Requirements

ID	Requirements	Priority	Complexity
M1	The software must support easy addition of other types of ads	H	L
S1	The product should provide a high degree of stability, so the customer can meet their availability demands when our product is integrated into their systems	M	M
T1	The software must be developed on the .NET platform	H	L

6.1 User Stories

We created the following user stories from the customer-provided project description, and meetings with the customer.

Actor	C1.Customer
Description	Customer submits a completed ad
Example	Real estate agent sends a completed ad in pdf-format to the system, with accompanying data. The data is automatically put in the correct databases/tables

Actor	C2.Customer
Description	Customer reviews and selects product options
Example	A Real estate agent wants to insert an ad in the system. The system displays available products, and when one is selected, will list the next five available booking dates for this product, and its options.

Actor	D1.Developer
Description	Decides to develop a plugin for the system
Example	Starts IDE of choice, and develops a plugin/extension, using the interfaces and polymorphism provided.

Actor	D2.Developer
Description	Developer wishes to use framework in/with other application
Example	The developer, being already familiar with Microsoft Web API, quickly integrates the Web API compliant framework with his intended target.

Actor	D3.Developer
Description	Wishes to quickly understand system components for extension
Example	Reads attached developer documentation in IDE, or as attached html/xml. Quickly sees what each individual method does, and can easily extend the system.

Actor	C3.Customer
Description	Customer tries to submit an ad with insufficient information
Example	The customet tries to add a pdf-file, but doesn't provide the necessary data; I.e. didn't specify a price or place. The system rejects the ad. The database remains coherent.

6.2 Non-functional Requirements

From these user stories, the following Non-Functional Requirements can be extracted.

- Compliance (with Microsoft Web API)
- Extensibility (through polymorphism)
- Documentation (for Developers)

7 Architecture

7.1 Stakeholders

7.1.1 Customer

Our goal with this course is to make a product that work as the customer intended but also performs satisfactorily. The code we're writing needs to be written following the clean coding standard and use interfaces/polymorphy so their developers can further develop this solution.

7.1.2 Course Evaluator

The course evaluator needs to be satisfied if we want a good grade, that means that we need to satisfy all the other stakeholders and deliver a well documented report of everything.

7.1.3 Supervisor

The supervisor said in the first meeting that he would fight for our grade given that he was satisfied. This means that we should complete the project deliver a well documented report.
We should also take advices from him, especially when there's issues among us in which we need to tell the supervisor so he can help solve the issue.

7.1.4 Implementers

The architecture should be easy to implement and make sense to the coders.

7.2 Quality Attributes

The customer was very specific when it came to what they wanted.

7.2.1 Modifiability

The solution we're making will be used for other ads than real estate ads, therefore we need to make it modifiable so other developers later on can further develop using our solution as a base.

7.2.2 Performance

We want the system to perform with a satisfactory performance.

7.2.3 Availability

The system should be available for the users when they need it.

7.2.4 Interoperability

Needs to inter-operate with already-existing order system, this will be done by using the technology the customer tells us to use; Web-api, mssql etc...

7.2.5 Readability

The customer wants us to write readable code. We are supposed to do a polymorphy/interface type of programming so it's easier for other developers to extend via plugins etc. therefore it's important that the code is readable so other's can easily understand what's going on without much hazzle.

7.3 Views

7.3.1 Process view

There is no need for us to supply a process view, because we do not have access to their server. We are only supposed to write the code for their system, without taking into account how the processes interoperate.

7.3.2 Logical view

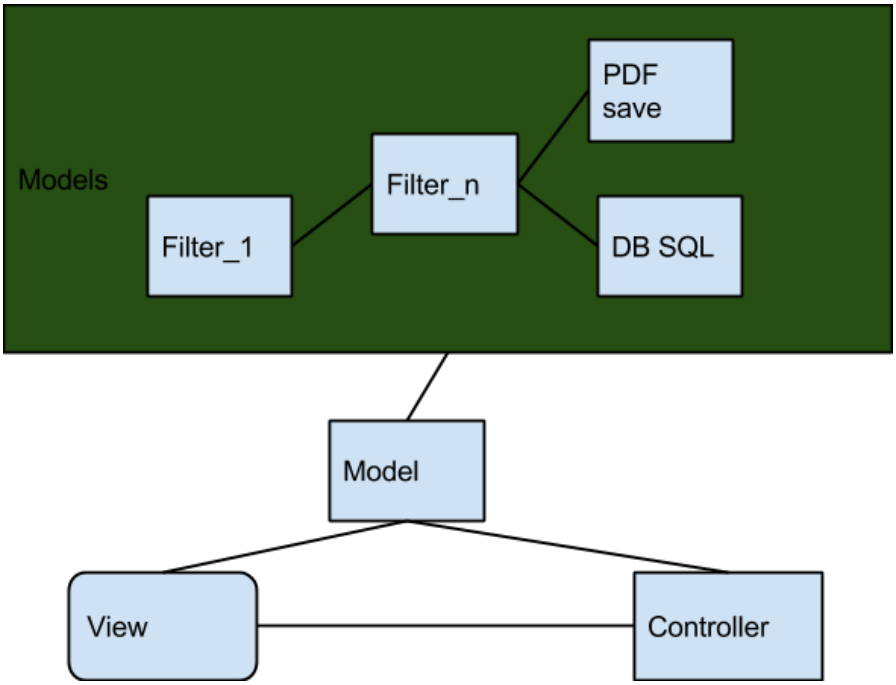

Figure 1: Logical view

7.3.3 Scenario view

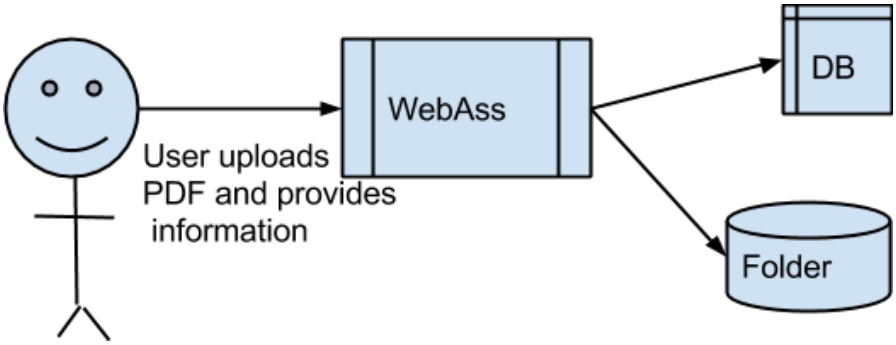

Figure 2: Scenario view

7.3.4 Physical view

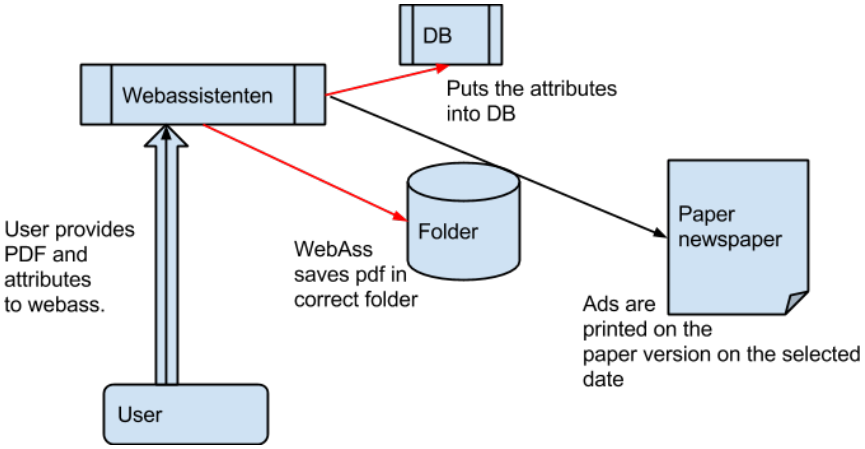

Figure 3: Physical view, we implement the red arrows

7.4 Class diagram

From these views, we made this class diagram.

7.5 Patterns

MVC due to the technology and pipe & filter to filter the data and due to the modifiability requirement.

7.6 Tactics

7.6.1 Modifiability

- Increase semantic cohesion
- Decrease coupling
- Split modules

7.6.2 Performance

- Write optimal code

7.6.3 Availability

- Our code should not crash the customer's system, but it's their responsibility that the system is available.

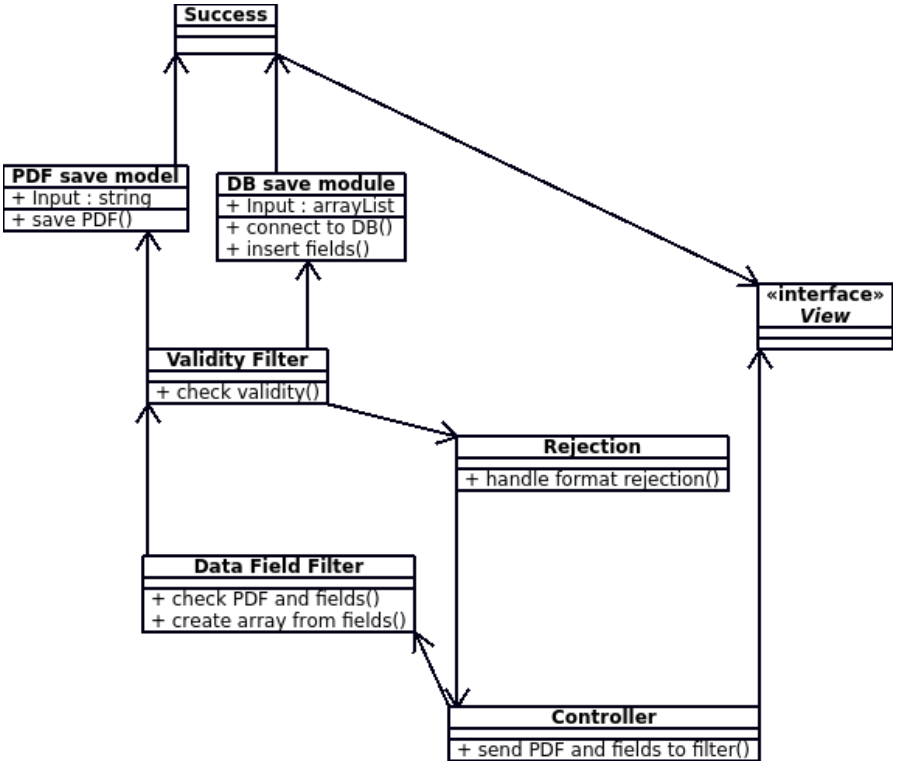

Figure 4: Digital class diagram

7.6.4 Interoperability

– The technology and tools we're using should be sufficient to ensure interoperability

7.6.5 Readability

– We will follow the clean coding principle and use camelCase coding. Refer to the Templates and Standards section.

8 Sprint 1

8.1 Plan

8.2 Backlog

8.3 Design and Implementation

8.4 Testing and Results

8.5 Retrospective

9 Sprint 2

9.1 Plan

9.2 Backlog

9.3 Design and Implementation

9.4 Testing and Results

9.5 Retrospective

10 Sprint 3

10.1 Plan

10.2 Backlog

10.3 Design and Implementation

10.4 Testing and Results

10.5 Retrospective

11 Testing

This chapter will describe the methods used for testing the requirements, the reasoning behind their choice, and the results of the testing.

11.1 Testplan

<Unsuitability of TDD or Unit Testing> <About testing of functionality>

11.2 Test Results

<Whatever the results of the testing was, and thoughts about it>

12 Evaluation

13 Conclusion and future work

References

[1] Yusuf Arslan. Is agile and scrum really better than waterfall? http://yusufarslan.net/agile-and-scrum-really-better-waterfall. Read on 26 August, 2013.

[2] Pamela Rodríguez Domínguez. 10 principles for keeping your programming code clean. http://www.onextrapixel.com/2011/01/20/10-principles-for-keeping-your-programming-code-clean/. Read on 2 September, 2013.

[3] Wikipedia. Adresseavisen. http://en.wikipedia.org/wiki/Adresseavisen. Accessed 7 October, 2013.

A Database setup

The database dump we got from the customer was a 10.8GiB .bak file which contained the existing internal database of the adressa system, which is a MSSQL database. We thus had to install MSSQL locally on our own computers to be able to restore this backup file before we could integrate the database into our project via the entity framework. To manage and restore the database, we had to use SQL Management Studio. Found here: http://www.microsoft.com/en-us/download/details.aspx?id=8961

This package let us install the 2012 version of the management studio, or choose to update from an existing 2008 version of MSSQL Management Studio which it claimed was already installed. However, this package did not install an instance of the SQL server.

It was therefore impossible to connect to a MSSQL server instance because none existed, which made it impossible to restore the .bak file (database backup dump), because there were no SQL server instances to restore to. We tried to install "SQL server with tools express" which did in fact install a SQL server instance and we were allowed to connect to it via the management studio, and we were able to click "restore database". We then navigated to the appropiate folder and chose the .bak file to restore. When we clicked "OK", it started to restore the database, but after a few minutes we got an error message saying that the database was too big to be restored, the express version can only restore a database up to 10.2GiB while the .bak file we got was 10.8GiB. The last tool we installed was Microsoft SQL Server 2012 Developer 32/64-bit, it had a MSSQL server instance. We could now restore the .bak file! http://www.katieandemil.com/sql-server-2012-restore-database-backup-file BAM! By using the entity framework we could add a ADO.NET Entity data model of the database. (http://www.entityframeworktutorial.net/Entit framework5-introduction.aspx)

B Evaluation