# Table of Contents

# Project Establishment

## *Background and Purpose*

Background for this project is to develop a stock-screener, which is like a search engine system for an on-line shares community website: www.freyr.dk

This final project is implementing after a thorough study and understanding the concepts of Project management, System Development Methods, Database, Programming and Web services which I learned in earlier semesters. By doing this project, I hope to get better practical knowledge and get real world job experience.

This Project has to cover:

1). Create a "crawler" to extract information from the websites and save it into the database regularly.

2). Create a web front-end for the database, where users can search for the shares (the screener).

## *Goals*

I am defining here **SMART** goals which I wish to reach in this 10 weeks project.

1). I have to complete the project establishment, requirement specifications together with customer within 2 weeks and selecting process model.

2). Every 2nd.week I will track differences between estimated time and real time phase to control delays, by updating the time plan.

3). I have to be able to complete three XP iterations before week 43.

4). Report and product has to be submit to the school before deadline that is 31.Oct.2008 at 12:00pm.

## *Requirement Specifications*

I made few user stories together with customer and wrote down here as customer accepted functional requirements for system. These requirements could be changed or canceled or added new requirements by customer further. Working papers for user stories are found in the appendix.

| Uno. | Requirement | Description | Estimate time |
|---|---|---|---|
| 1 | Admin Interface to manage criteria | Criteria is defined by the four fields:<br>Name, URL, XML path, Convert Expression. | 1 day |
| 2 | Import information from Reuters | Shares information has to extracted from Reuters website and added to database, including fields PE/Ratio, shares_out and Div&Yield. | 2 days |
| 3 | Distribution graph for each criteria | Distribution graph helps to find good values for criteria, So Freyr prefer to have a graph for it. | 2 days |
| 4 | Import should run daily for all symbols | There is a list of all symbols in a Frey's database . All those symbols have to be updated everyday. | 3days |
| 5 | API : Get information for symbol. | There has to be an API , where it is possible to get all known data has to able to search. | 1days |
| 6 | Show a Search Form | When user enter into application, it has to display a search form which has fields such as Criteria, Market Cap, PE/Ratio, Dividend Yield, 52w Price change etc., | 2days |
| 7 | Show Search Results | When user enters Criteria and clicks on Search button then related search form result has to be shown.<br>Company name and symbol has to link to details page when user clicks on them. | 3 days |
| 8 | History of Extracted values | Extracted values should be saved also for history purpose | 2 days |
| | | | |

## Risk Analysis

By Risk analysis, we improve the accuracy of traditional scheduling, cost estimation and systems engineering skills. Risk is a combination of the severity and probability of the hazard event. This means that by analyzing and categorizing the risks, I get a better understanding for the risks that can appear, and by the better understanding I have more chance to avoid them.

In the table below, are the risks I identified, their probability, the effect on my project, and how accepted this risk is. Together with this are some solutions on how I can avoid the risks.

| Risk | Probability | Effect | Acceptability | Risk Solutions |
|---|---|---|---|---|
| Requirements are wrong. | Low | Medium | Acceptable | Find the missing requirements in next iterations. |
| System is not user friendly. | Low | High | Intolerable | Test together with customer to improve. |
| Software is not good enough for safety. | Low | High | Intolerable | Improve safety requirements in further iterations. |
| Code and report is lost. | Low | High | Intolerable | Take regular backup to Frey's server. |
| Not well Documented | Medium | High | Intolerable | Make use of weekly meetings with supervisor to get input on what to improve. |
| Delay of project report submitting. | High | High | Acceptable | I fail the semester, But, I guess there are 2 more chances to finish. Freyr can use the well-developed parts for further development. |
| Working with new technologies. Not sure that technologies can solve the problem. | High | High | Acceptable | Work with Freyr to gain better knowledge, document if something can not be solved. |

By those risks and their solutions I found are suitable to use XP process model to minimize the risks and saves the cost of project.

## *Strategy Analysis*

Strategy analysis is a way to find strategic elements to include in the project. By looking at the conditions of the project, and the strengths and weaknesses of those, it is possible to find their strategic solutions. Like with risk analysis, it is a way to get a better understanding of the project, to anticipate future problems with more ease. Below are the strategic conditions I have found for the project:

| Conditions | Strengths | Weaknesses | Strategic element |
|---|---|---|---|
| Technical Issues | Good Knowledge on System development methods, Programming and Sql server.<br>Some Knowledge on Python, framework and My Sql. | Don't have experience with Django at all. | Self study and finding information on Internet.<br><br>Help from Freyr developers about introduction. |
| Developer(s) | Only my self.<br><br>Getting help with Pair programming from my husband on Python and Django. | Only myself working on project report, analysis & design phases and missing team work. | Confirm project steps with supervisor.<br><br>Pair Programming and Short meetings with customer.<br><br>Testing and review each iteration to improve quality. |
| Result | Clear defined goals. | Difficult to estimate , due to experimenting on new subjects and complexity. | Use XP with many short iterations. |
| Users | Small time and private investors after release the product. | No real users at developing phase. | Work more close with Freyr.<br>Show prototype to customer to try. |
| Environment | No technical problems, No disturbance, good cooperation from Freyr. peaceful environment at home. | Faraway from home to school to seek help from teachers. | Internet available at home, So I can discuss about project with Freyr on line. |

**Strategy:** Following these strategic elements, I am going to use agile method with 3 iterations and prototypes by using XP process model.

## *Selecting process model*

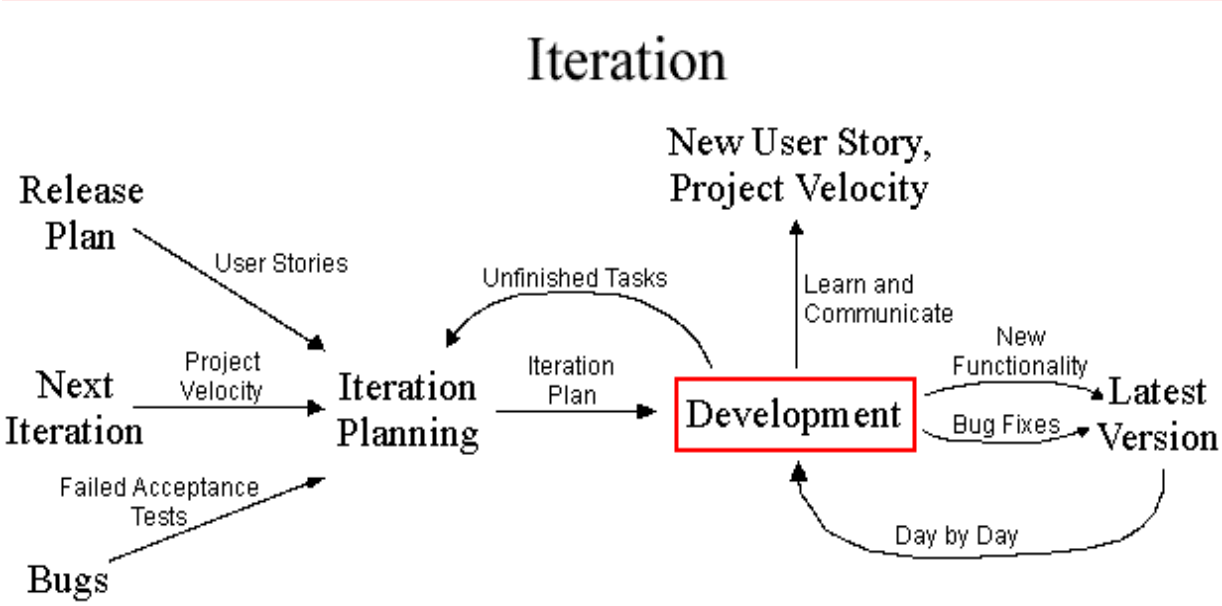
By the risk and strategy analysis, I have selected to work with the extreme programming (XP) process model.

XP is one of the agile process models. To be agile means to move quickly, and light weight - this sounds like a good features that I can use in this project.

- Waterfall. I used this for the first 2 semesters and also I used Iterative process model in 3rd and 4th semester, So I am curious to try with XP in practical and get better knowledge, what I learned theoretical in system development methods.
- There is more uncertainty in the project, and agile is a way to lower that.
- XP allows to do continue releases, So customer can start use them before completion of project.
- Since I have very little knowledge on Python and Django, it is good opportunity to gain knowledge by working on pair programming together with Frey's developer.
- By using XP, I can reduce the cost of changes in project.

For these reasons, I would like to select an agile process model, since it allows me to produce more than one iteration, and to put more focus on what is the important parts to get done first, and this is then XP.

Below is this diagram showing the phases of XP process model life cycle.



## *Time planning*

As a final point on my project establishment, an initial time-plan is good. Since I am only one person on the project, and since the estimates of what must be done this early in the project is very uncertain, the time-plan will only be rough estimates of the project milestones. The milestones are defined by the artifacts they produce, so it is clear when a milestone has been reached.

| Title | Description | Artifacts | Time estimate | Week |
|---|---|---|---|---|
| Project establishment | Discussion with Freyr about the project requirements, write about process model selection, create initial time plan. | Project contract and initial time plan. | 1 week | 34 |
| Release planning | Find user stories for the project, give user stories to customer to select for first iteration. | List of minimum 10 user stories. | 1 week | 35 |
| Introduction to Freyr | Get introduction to the Freyr setup, look at Python, Django, and MySQL. | Document describing how is C# vs Python, ASP.NET vs Django, and MySQL vs MSSQL. | 1 week | 36 |
| Iteration planning #1 | Agree on user stories to implement in first iteration. | List of selected user stories. | 1/2 day | 36 |
| Iteration development #1 | Work on the first iteration. (My guess is database related tasks) | Detailed design, tests, code, and test report for selected user stories. | 2 weeks -1 day | 37+38 |
| Iteration planning #2 | Review the produced artifacts. Agree on user stories to implement in second iteration. | Review document. List of selected user stories, and/or error reports. | 1 day | 38 |
| Iteration development #2 | Work on the second iteration. (My guess is crawler related tasks) | Detailed design, tests, code, and test reports for selected user stories. | 2 week -1 day | 39+40 |
| Iteration planning #3 | Review the produced artifacts, Agree on user stories to implement in third iteration. | Review document. List of selected user stories, and/or error reports. | 1 day | 40 |
| Iteration development #3 | Work on third iteration. (My guess is web GUI) | Detailed design, tests, code, and test reports for selected user stories. | 1 week -1 day | 41 |
| Review of iteration #3 | Review of the produced artifacts. Evaluate cooperation with Freyr. | Review document. | 1 day | 41 |
| Report writing. | Finish the report. | Final report. | 1 week | 43 |

I notice that week 42 is autumn vacation. I am not certain if the "release planning" is optimistic for 1 week, but I hope to get help from Freyr on the analysis and understanding of the problem, so I can finish in 1 week. If it takes one week more, I can use later "iteration development #2" for 1 week only.

### *Cost Estimation*

I am using **Agile COCOMO II** tool to estimate Cost and effort for this project. It is a simple method of software cost estimation by analogy to a previous project.
Purpose of using Agile COCOMO II tool is, the ready ability to move quickly in giving estimates with ease. It works by using the lessons learned in previous projects, and by comparing cost/effort "drivers" with what I know about the current project.

I used the cost drivers for which I had knowledge enough to estimate, compared with my previous project SIM-ATC. The estimated effort is 3 months, which is roughly the time of this project.

The estimation came up with 2.81 man months for the full project, which I hope is realistic. The full details for the cost estimation output can be found in the appendix.

## Selecting programming language and tools

In my previous projects I have used C#, .Net, and ASP.NET for implementation, and other Microsoft tools such as MSSQL server and Visual Studio .NET. I have some kind of familiarity with these tools, but, it is not a requirement that I must use those for this project.

Since I chose XP for process model, I think it is a good idea to select tools and programming language which are a good match for this. At the same time, another purpose here is that I think it will be a good idea to look at what Freyr.dk is already using, so it will be easier to fulfill the strategic element of communicating with Freyr – it is more easy if we use the same tools. As a third point, using new tools is a way to show that I am capable of applying learned theory regardless of the tools.

I have selected to use Python programming language, Django application framework, and MySQL for database. My selection is argued as follows:

- Django framework is presented as a ".. high-level Python web framework that encourages rapid development and clean, pragmatic design..", which is good fit for an agile process.
- Python, Django, and MySQL is already in use by Freyr.
- All are available both for Windows platform which I am familiar with, and for Freyr.dk's selected platform.
- They are new technologies, but not radical new, so they should be able to learn in the project time frame.

## Comparing with known technology

In order to assess these new tools, I will create a small comparison. This comparison is mostly for my own use to see the differences, but also to document what I have learned during the project.

### *Python and C#*

Python and C# are both object oriented programming languages, and they are both procedural, but from that they are also some different. I will try to give a short overview of how is Python different

from C# in relation to the features I am using in my project.

I think it is easy to get started with Python. The basic structure is almost like C#, but just little different. Instead of curly-brackets, Python is using indenting to decide the scopes in the code. Also I think it is more object oriented, everything in Python is objects, also methods and documentation in the code. It has types, but it is more dynamic, so I do not need to bother about declaring types on variables.

I show below here a small example program I made while starting with Python, and a C# program with same features, to give idea of basic Python structure:

C# program to calculate fibonacci numbers normal and recursive.

```
using System;

namespace simple {

   public class Fib1 {
         // iterative
         public int Calc(int HowMany) {
            int a = 1, b = 1;
            for (int i = 1; i < HowMany; i++) {
                  int tmp = a;
                  a = b;
                  b = tmp + b;
            }
            return a;
         }
   }

   public class Simple {
         // recursive fibonacci
         public static int Fib2(int HowMany) {
            return Fib2(HowMany, 1);
         }
         public static int Fib2(int HowMany, int a) {
            if (HowMany > 1) {
                  return a + Fib2(HowMany - 1, a + 1);
            }
            else {
                  return a;
            }
         }
         public static void Main(string[] args) {
            Console.WriteLine("by class: " + new Fib1().Calc(10));
            Console.WriteLine("by recursive: " + Fib2(10));
         }
   }
}
```

There is not any news here, I created a static method and an object method, and both are working to calculate Fibonacci, one does it iterative, and another recursive.

Python program doing the same

```
# class to calculate fibonacci
class fib1:
   # method to calculate
   def calc(self, how_many):
      a, b = 1, 1
      for i in range(1, how_many):
         a, b = b, a + b
      return a

# method to calc fibonacci recursive
def fib2(how_many, a=1):
   if how_many > 1:
      return a + fib2(how_many-1, a + 1)
   else:
      return a

print "by class: ", fib1().calc(10)
print "by recursive: ", fib2(how_many=10)
```

We can see here the differences now:

- The program is much smaller, exactly half size if we count the “using” statement in C#. I think smaller program means more easy to read and understand.
- There is less unnecessary syntax. The C# example is full of () and {}s. This also makes it easier to understand.
- Indentation in Python takes some time to get used to. It is annoying that I have to be so precise.
- Python is using files for modules, it is the same as “name space” in C#, but it seems more simple that one file is one name space.
- There are no type declarations for variables in the Python. This makes it easier to read, but also harder to know what to pass if in doubt.
- “if”, “for”, and structures in general are similar to C# and Python, so it is easy to understand concepts of both.

### Static or dynamic language

It is not necessary to compile Python programs, the Python virtual machine (PVM) will do that automatically when program is loading. C# is also running in a virtual machine Common Language Runtime (CLR), so in that way they are similar.

Static language is to my understanding that the program can be evaluated at compile time, and not while it is running. The opposite is then dynamic, it evaluates when it is running, meaning some errors you can only see while program runs, also simple errors like for example typos of method names, those are shown during compile in C#.

But, dynamic language like Python also offer me some functionality that I can not get from C#, which I used during my Extractor module. It is using eval() function, which evaluates a piece of code at runtime, that way I can put python code into the database, load it, and have it evaluated when I need it.

### IDE support

For my project, I used the Python IDLE.IDE . It is a very simple IDE, almost like a text editor, that can do syntax coloring, and helping with the indentation.

Compared to Visual Studio, this is very basic. Visual Studio has many features besides writing code, such as managing all files of a project, browsing structures such as name spaces, classes, and methods, and also it has a whole set of features for the GUI construction.

## *Django and ASP.NET*

I chose to compare Django with ASP.NET, since both are web application frameworks. There are some parts of Django which are not part of ASP.NET but other .NET frameworks – in those places I will mention that.

Django is a web application framework, which is using Python as the programming language, just like ASP.NET is a web application framework, which is using C# as its programming language. It is based on the MVC pattern, but they use some different words than MVC. I will note in this chapter what I see as the proper MVC components. It is hard to say if ASP.NET is a MVC framework or not – you can chose to say it is, where the Code-behind is the controller, the ASP page is the view, and the model is your database, or, you can say that it is not, because there is no clear distinction between controller and view, and there is no model.

## Models

One part of Django is an object relational mapper (ORM), that means some framework for mapping objects to and from a relational database. Django supports more databases, and it is very easy to configure which database you want to use, by the application settings file. For development it is possible to use a file-based database, SQLite, which is like Microsoft Access, and does not require any server installed.

The models is where to program the business objects for use by the application. Each class has defined attributes which specify how to map to database, and it must inherit from Django class models.Model. In my example below I am showing how one model can look like:

Models.py file from search application

```
from django.db import models

# The business object Stock Property
class StockProperty(models.Model):

        # Define the attributes and their mapping to database
        name = models.CharField(max_length=100)
        url = models.URLField()
        xml_path = models.CharField(max_length=250)
        convert_expression = models.CharField(max_length=250, null=True, blank=True)
        frontpage = models.BinaryField()

        # string convert method
        def __str__(self):
                return "%s url=%s, xml_path=%s, convert_expression=%s" % \
                        (self.name, self.url, self.xml_path, self.convert_expression)
```

By this model, I can now work with it in the rest of the application through the API that is provided for all models. Below I show some examples of how I am working with this model.

Views example working with models API

```
# Views for search application
from django.shortcuts import render_to_response

def index(request):
        """ Index shows the a default criteria """
        # find all StockProperty which have frontpage=True
        list = StockProperty.objects.filter(frontpage=True)
        return render_to_response("index.html", {"properties": list})

def all(request):
        """ Show all criteria """
        list = StockProperty.objects.all()
        return render_to_response("index.html", {"properties": list})

def change_name(request, the_id, new_name):
        """ Change the name of one criteria """
        # load from database
        p = StockProperty.objects.get(id=the_id)
        # modify name
        p.name = new_name
        # save to database again
        p.save()

def create(request, the_name, the_url, the_path):
        """ Create a new criteria """
        p = StockProperty(name=the_name, url=the_url, xml_path=the_path)
        p.save()
```

So, it is very simple to work with the business objects in Django, it will handle all the SQL for me.

By using a database client, I can go to the database and verify that all is done correctly.

ASP.NET does not have any ORM features like this that I know of. They have ADO.NET, but to my understanding, that works in a different way, with “generic” containers (like GridView), that maps to the database, and you have to create the SQL for updates, inserts and deletes. I think working with Django is more easy here, because it is easier to relate to the business objects that you write yourself.

## Dynamic Admin Interface

One other thing I used in Django was its Dynamic Admin Interface (DAI). The DAI is a full web application that Django creates by looking at the models I have defined. It creates a GUI where I can perform CRUD operations on all the models. This is very good, for example for configuring the application.

***SCREENSHOT HERE***
In the screen-shot we see the admin interface where I am editing one of the models I created for the project.

It is possible to configure how the models are presented in the admin interface, for example I have an admin configuration like the following, where I define which fields of StockProperty that I want to show in list.

Admin.py showing admin interface customization

```
from models import StockProperty
from django.contrib import admin

class StockPropertyAdmin(admin.ModelAdmin):
    list_display = ['name', 'url', 'xml_path', 'convert_expression']

admin.site.register(StockProperty, StockPropertyAdmin)
```

I think it is very simple to work with the models using the DAI. I do not need to work with SQL at all to create configuration in the database, I can just use the DAI GUI, which is much easier than SQL :-).

This is something that ASP.NET does not have at all, there are maybe some tools you can buy to get this, but it is outside the scope of my project to look at that.

## Design URLs

In ASP.NET, one .aspx file is the same as one URL.

With Django I must define all URLs, and what views should execute when an URL is requested by the client's browser. Below I show one example of how I define the URLs for one of my applications in the project:

Urls.py defining urls to views map.

```
from django.conf.urls.defaults import *
from search.views import do_query, show_criteria

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    (r'^search/', do_query),
    (r'^show/(?P<criteria>\d+)', show_criteria),
```

```
    (r'^admin/(.*)', admin.site.root),
)
```

I define the search/ url to go to a query view, and then a show/ID to send the ID to a show_criteria view. It is not so easy to work with those parameters, so I do not use that a lot. It is using regular expressions, which are quite hard to understand.

It is hard to say what is better, but I think the Django model offers more flexibility, while the ASP.NET way is just simple. I like with Django that it is possible to get parameters from the URL directly.

## Writing views

Some example of views here, I already put one above.....

## Design templates

Some example of ASP.NET .aspx file here, and compare with a Django template.....

## Custom admin commands

Django-admin.py is a tool to administer a django application, for example, to create a project, to start a development server, etc.  It can also be extended with your own admin commands. I am thinking to use this for solving periodical jobs (eg. It runs from commandline, but inside the django framework).

I should write something here about that when I get more knowledge.....

## IDE support

Tool support is better for ASP.NET, since it is supported by the Visual Studio .NET, but, for Django, I am now using Eclipse with Python extensions, and by that I get the HTML support of Eclipse which is better than the HTML support in Visual Studio .NET, and the Python support, which is as good as it gets for a language like Python (see more where I compare Python and C#).

# *MSSQL and MySQL*

Very short resume on what this chapter can be about.

How is installation of MySQL, is it easy to get started with it? Is it easier/harder than MSSQL?

About InnoDB: MySQL has many storage engines (ways of storing data on disk). InnoDB is the engine which provides transactions. Is this good? Must I use this engine? Does MSSQL have transactions always? (maybe this is not needed to write about?)

What are the tools for working with MySQL? And for MSSQL? Which is easier to manage?

How is the SQL (queries, inserts, updates, create, drop) that works on MSSQL and MySQL, are they very different, or the same?

What about Transact-SQL (Stored procedures and functions)? Does MySQL support those? Do I need that for my project?

Is there any real difference if I choose MySQL or MSSQL for my project? Why is MySQL good for my project?

## Iteration Planning

Iteration planning is much like the traditional analysis part in XP. First the customer and I select which user stories that are going to be implemented for the coming iteration, based on the calculations on project velocity from the previous iteration. For the first iteration, we just used my estimates as truth, then later we can modify depending on how the first iteration went. This is very smart with XP I think.

For each of the selected user stories, it must be analyzed, and from the user stories, tasks are created. One task is equivalent of a more detailed description of one part of the solution to the user story. These tasks contains more knowledge now when we have analyzed the problem, so they also contain a more precise estimate for the tasks.

Below I have listed the tasks that were created during the project for each iteration, together with estimates and follow-up for their real time spent, this allows me to do the project velocity measurement.

| INo | TNo | UNo | Task name and Description | Time estimate | Actual time |
|---|---|---|---|---|---|
| 1 | 1 | 1 | **Setup Django Project**:<br>A Django project and Django application has to be created. | 4 hours | 3 hours |
| 1 | 2 | 1 | **Create a stock property model**<br>Create stock properties with fields such as Name,URL, XML path, convert expression and also has to create the table in database. | 3 hours | 4 hours |
| 1 | 3 | 2 | **Create the stock property for three fields**<br>Create the stock property with for the fields such as PE/Ratio, Shares out, Div_Yield. | 2 days | 1 day |
| 1 | 4 | 2 | **Create a model to hold values**<br>A model has to be created, which can hold information on the values extracted, and the symbols which they belong to. | 3 hours | 3 hours |
| 1 | 5 | 2 | **Extract values and Save in database**<br>Save the values into database after download the URL, and execute XML path and converter expression. | 2 days | 2 days |
| 1 | 6 | 3 | **Calculate X,Y coordinates for Distribution graph**<br>Group the companies by values from database. So values can fit to the graph. | 1 day | 1 day |
| 1 | 7 | 3 | **Create image by the graph data**<br>The calculated graph data has to be plotted in an image, the image has to show the data, with X axis legend. | 1 day | 1 day |
| 1 | 8 | 3 | **Create the URL to show the graph**<br>Create a view that takes a stock property ID and returns the graph image for it and then create a URL mapping for a view. | 1 day | 3 hours |

Ino is the iteration number, Tno is the task number, and Uno is the user story number.

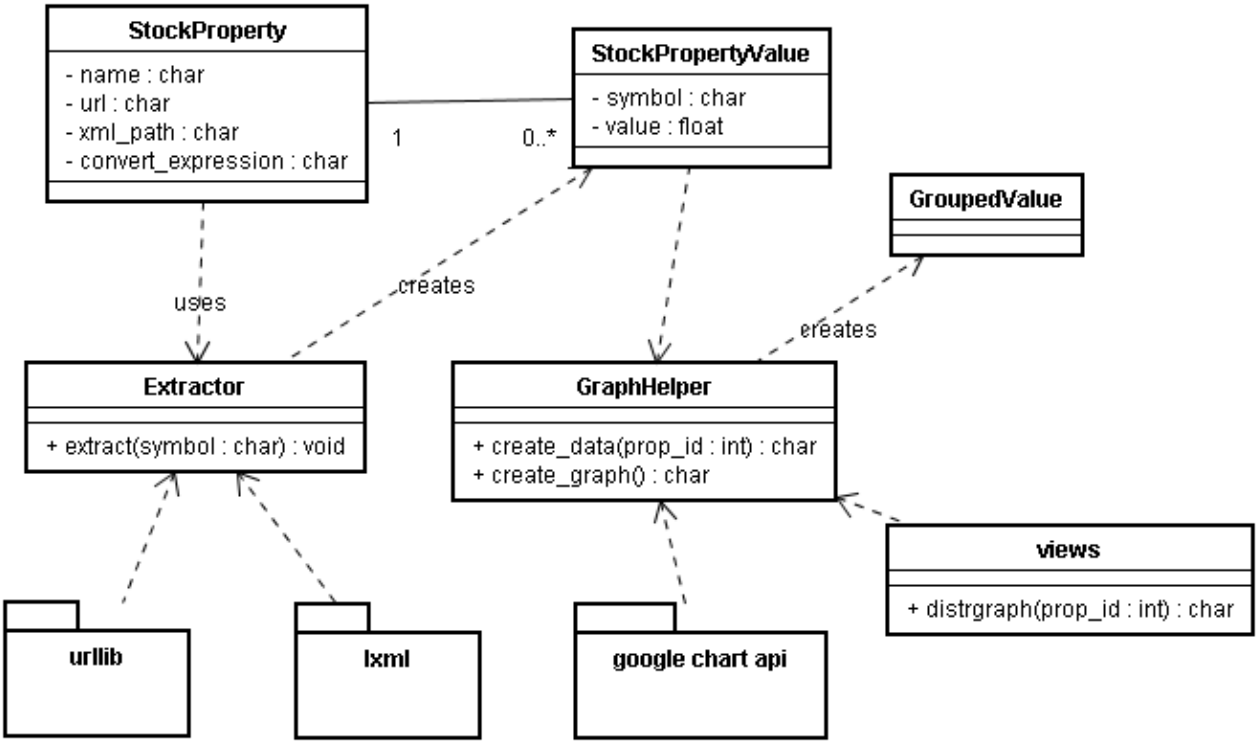
In the appendix, the working papers for the tasks have been attached.

# Design

*** This should eventually show some design of the final project ***

What I at least want to show here:

- Final business object model
- Final domain object model
- Sequence diagram for how data comes through the system
- Use-case diagram




This is the class diagram after iteration 1.

# Testing

For testing, there are two parts that we have learned about in the classes. One is the user acceptance test, which is performed by customer, as a black box test, and then there is the module tests, which are performed by developers during development, usually as white box testing.

By using XP, I am getting good experience with using both. In iteration planning, each user story will be as one user acceptance test, and each task created from the user story will be one module test.

## *Module testing*

I have tried to follow XP by creating tests before I implement the code. While it is not possible for all the tasks (some tasks that are not product related, it is hard to create test), then most tasks it is possible to create a test, and then implement the feature afterwards.

I think this approach is very good, by creating the test, I am forced to think about how the functionality must be, and same time, the test functions as a kind of documentation for how my program works.

Django has a framework, where I create tests inside each application, and then execute one script, that will run all the tests. Below I show the example of one such test, which is the test for task no. 4:

Test for task 4, test existence and CRUD operations on a model.

```
# Test for Task number 4.
class StockPropertyValueTest(unittest.TestCase):

    def testInstantiation(self):
        from models import StockPropertyValue
        o=StockPropertyValue()

    def testCrud(self):
        from models import StockPropertyValue, StockProperty

        # check that database is empty
        ol = StockPropertyValue.objects.filter(symbol="tdc.co")
        self.assertEquals(0, len(ol))

              # create stock property
        s = StockProperty(name="test", url="http://google.com",xml_path="test path",\
                    convert_expression="convert")
        s.save()
       # create one object and save to datbase
        o = StockPropertyValue(symbol="tdc.co", value=6, stock_property=s)
        o.save()

        # show the id just for ourself
        print o.id

        # read from database
        o2 = StockPropertyValue.objects.get(id=o.id)
        self.assertEquals(6, o2.value)
        self.assertEquals("tdc.co", o2.symbol)

        # update object and save to datbase
        o2.symbol = "test2"
        o2.value = 9
        o2.save()
        # compare objects
        o3 = StockPropertyValue.objects.get(id=o2.id)
        self.assertEquals(o.id, o3.id)
        self.assertEquals(9, o3.value)
        self.assertEquals("test2",o3.symbol)
```

```
    # delete
    o3.delete()

        # verify its gone
    ol = StockPropertyValue.objects.filter(symbol="test2")
    self.assertEquals(0, len(ol))
```

After each task, I run the tests, so I can see if anything has become broken in other parts of the application. Below is output of when I run the tests:

** SCREENSHOT HERE **

## *User acceptance testing*

Here I have transformed each of the user stories into a test that can be performed by customer when I deliver the functionality. In my table I have shown the created tests, and same time, I have used the table for collecting the information about when user accepted the test, and comments if it was rejected or accepted.

| Uno | Test description | Notes | Delivered | Accepted |
|---|---|---|---|---|
| 1 | Install the software according to installation document.<br><br>Open url in browser: http://yani.dk:7400/admin/search/stockproperty/<br><br>(Replace yani.dk with Freyr's own IP)<br><br>Verify all four required fields are available. | | 19-9-2008 | |
| 2 | Install the software according to installation document.<br><br>Open url in browser: http://yani.dk:7400/admin/search/stockproperty/<br><br>(Replace yani.dk with Freyr's own IP)<br><br>Verify P/E Ratio, Shares Out , Div and Yield are available and correct. | | 19-9-2008 | |
| 3<br><br>1. | Install the software according to installation document.<br><br>Open url in browser: http://yani.dk:7400/distrgraph/4/ - graph with data, verify the graph is there and check if graph is suitable for test data. | | 19-9-2008 | |

| | | | | |
|---|---|---|---|---|
| | http://yani.dk:7400/distrgraph/99/ - graph without data, verify the graph shows “No data available”.<br><br>(Replace yani.dk with Freyr's own IP) | | | |