Durham University
School of Engineering and Computing Sciences

# Design Document 2009/2010 Windfarm Planner

airnegy

**Development By:**
**SEG 2**

## Document Information

| | | | |
|---|---|---|---|
| **Project Name:** | Wind Farm Planner | | |
| **Prepared By:** | Airnegy | **Preparation Date:** | 28/04/2010 |
| **Email / Phone:** | cs-seg2@durham.ac.uk | | |
| **Document Version No:** | 3.6 | **Document Version Date : 28/04/2010** | |

## Version History

| **Ver. No.** | **Ver. Date** | **Revised By** | **Description** |
|---|---|---|---|
| 1.0 | 08/02/2010 | Eloise Stancombe | Create Document |
| 1.1 | 10/02/2010 | Eloise Stancombe | Draft Sections 1.1, 1.2 |
| 1.2 | 12/03/2010 | Seg 2 | Draft Section 1, 2 |
| 1.3 | 18/03/2010 | Seg 2 | Draft Sections 2, 3 |
| 2.0 | 06/04/2010 | Prathna Singh | First Draft Complete |
| 2.1 | 10/04/2010 | Seg 2 | Final Draft Sections 1 |
| 2.2 | 12/04/2010 | Seg 2 | Final Draft Sections 2, 3 |
| 2.3 | 16/04/2010 | Eloise Stancombe | Document Formatting |
| 3.0 | 21/04/2010 | Prathna Singh | Final Draft |
| 3.1 | 23/04/2010 | Eloise Stancombe | Partial Edit |
| 3.2 | 24/04/2010 | Seg 2 | Updated Sections 1.4, 1.5, 3.1 |
| 3.3 | 25/04/2010 | Prathna Singh | Partial Edit, Updated Sections 1.4, 2.2 |
| 3.4 | 26/04/2010 | Seg 2 | Updated Section 1, 2 |
| 3.5 | 27/04/2010 | Prathna Singh | Updated Section 3 |
| 3.6 | 28/04/2010 | Seg 2 | Final |

## Table of Contents

# ***Introduction***

## 1.1 Purpose

The design document will outline all parts of the software and their functionality. This will enable us to take a high level view of the system whilst giving a complete description of the software. We will use an object orientated design approach to the project. Object orientated design is the process of planning a system of interacting objects for the purpose of solving a software problem.

The purpose of the Airnegy Wind Farm planner is to provide proof-of-concept software that will be used by England's Regional Development Agencies (RDAs); a group of 9 organizations that where launched in 1999 in order to improve economic development, social and physical regeneration. As such each RDA has been tasked with contributing to the UK's targets on renewable energy. In support of this, each RDA is examining the potential for their region to contribute wind-powered energy to the national grid infrastructure.

Proof-of-concept software aims to aid the RDA for the North East in planning for wind farms. This will be done by providing software that will allow an RDA to associate wind speed information with electricity infrastructure information. Thus enabling the user to add, view and edit information such as name, location (Latitude/Longitude), date online, number of turbines, turbine model name and power output (MW) about existing or planned wind farms. Also the user should be able to add, view and edit information based on the Long Term Development Statements that Network Distribution Operators (NDOs), information such as name, location (Latitude/Longitude), size in kilovolts (33kV, 66kV, 132kV), maximum capacity (megawatts) and current capacity (megawatts) concerning electrical substations. All information will be queried by the user.

**System Benefits**

This system will be of use to the RDA because it will enable them to quickly identify ideal sites for wind farms, based on a number of criteria. The software will be able to calculate the cost implications involved with setting up the wind farm as well as expected yearly yield thus enabling the RDA to make sure the wind farm is cost effective. This will be dealt with in the financial report of the system, with the user able to choose which wind farms, substations and wind turbines to use to calculate the report. Another benefit of the system is that the RDA will be able to map where all wind farms and substations are in the UK and store their details on the system. There will also be the benefit of being able to minimise the energy deficit of the substations, as this can be used as a search criteria. This will mean that the substations' energy output is maximised. The main benefit of the system is that of being able to highlight a windy area around a certain point and then within that area being able to highlight the best points to place a farm. The user will be able to specify the size of the area and height they wish to search at, as well as the point (coordinate), which can either be one of their own choosing or the co-ordinate of a substation, from which to search around.

## Objectives

The objective of the system was to create a piece of software that enables the RDA to be able to optimise the positioning of wind farms in the UK. The main objective is to closely relate the user requirements to the final piece of software to produce s product that fulfils the needs of the RDA to aid them in the workplace and achieving their governmental targets.

## Goals

The goal of the system was to make the process of deciding new wind farm sites more efficient for the RDA, based on a number of pre-determined factors. There was also the want of then using this information for a more thorough analysis of the site to have a better idea about suitability especially the financial aspects. The system needs to be able to deal with large quantities of data provided by national institutions and identify specific regions where it would be ideal to create a new wind farm or add more turbines to an existing wind farm, to supply energy to underactive substations. The system should let the user traverse through the data via a GUI and queries. It will provide a graphical representation for the user, which will detail the positions of the substations and other data. Another goal is enabling the user to be able to specify preferences. The user guide will also be implemented and will be part of the system, so that it is easy to access and as it will be electronic it can be expanded with the system to ensure a continuing fully comprehensive guide to the system. Being able to add, edit, view and delete all information held on wind farms, substations, turbine models and turbine models to wind farms is an essential goal of the system. The financial report was a key part of the requirements as it highlights whether or not the proposed wind farm and site is a financially viable option based on different factors like cost of connection to a substation.

## 1.2 Scope

The system will:

- Add/edit/delete wind farms
- Add/edit/delete wind turbines
- Add/edit/delete substations
- Import data e.g. wind speed from a file
- Store all the imported Wind Speed data
- Be able to change the settings of outputs like km/miles
- Create a financial report
- Create a database storing all information held on wind farms
- Be able to restore and back-up all data held in the database
- Have a visual representation of the wind farm/substation/wind speed data on a map.
- User can query the wind speed data to determine wind speeds at different heights for a location.
- Find potential Wind farms by substation
- Model potential Wind Farm revenue yield
- The database will have forms of validation and verification for all data entered into the system
- User Guide will be created
- Tool tips will also be integrated into the system

## 1.3 Changes to Requirements

Here are some of the changes that we made to the requirements. The reason for us making these changes was to improve the system from the perspective of a Software Engineering team. These changes have been brought about to enhance the functionality of the program.

| Requirement ID | Reason and Consequence | Original | Change |
|---|---|---|---|
| 6.1.1.6.4 In Requirement Document, Functional | We changed this because we felt that a circular area was more accurate, therefore the calculation of finding the specified area had to change. | If the supplied area is >0 then the system should return wind speeds for a square area around the supplied location. The square will have side length of twice the distance “d”. The area will be $(2d)^2$. | We will find the wind speed for a circular area around the supplied location |
| 6.3.3 Distance between 2 coordinates | We changed the formula because the new formula was more optimal, it took fewer lines of code and less execution time. | Distance (Km) = cos(((sinLat1 * sinLat2) + (cosLat1 * cosLat2)) * cos(long2 – long1)) * 6371 | The formula has been changed to Pythagoras theory. |

**Additional Requirements Section**

| ID | Requirement | Description |
|---|---|---|
| | Setting option for Coordinates | Let the user be able to choose if they want to see and type in coordinates in Longitude/ Latitude or OSGB36 |
| | Setting Option for distance | Let the user be able to choose if they want to see and type in the metric or imperial system. |
| | Local Database | Be able to store database locally on the user’s personal computer |
| | Restore Function | Since there is backup function, there should also be a restore function |

For the rest of the document, we have abided in by the requirements given to us as much as possible.

## 1.4 Definitions

These definitions are for the design document, some definitions that are not stated here will be in the design document:

Object orientated design - The process of planning a system of interacting objects for the purpose of solving a software problem.

OSGB36 – The British National Grid referencing system used in Great Britain.

Lat/Long – A coordinate system used all over the world measured in degrees minutes and seconds.

Database – Use tables as a model for storing, managing and retrieving information.

Regional Development Agency (RDA) - Spread economic prosperity and opportunity

IEEE standards - The engineering process by creating, developing, integrating, sharing, and applying knowledge about electro- and information technologies and sciences.

System Architecture - Set of conventions, rules, and standards employed in designing the system's various components.

Unified Modelling Language (UML) - standardized general-purpose modelling language in the field of software engineering. UML includes a set of graphical notation techniques to create visual models of software-intensive systems.

## 1.5 References

1) Software Design Document, www.wikipedia.org,
http://en.wikipedia.org/wiki/Software_design_document [30/3/2010]

2) Object Orientated Design, www.wikipedia.org,
http://en.wikipedia.org/wiki/Object-oriented_design [30/3/2010]

3) RDA's, www.englandsrdas.com,
http://www.englandsrdas.com/about-the-regions/north-east [1/4/2010]

4) IEEE, http://standards.ieee.org,
http://standards.ieee.org/sa/sa-view.html [27/4/2010]

5) System Architecture, www.businessdictionary.com,
http://www.businessdictionary.com/definition/system-architecture.html [27/4/2010]

6) UML, www.wikipedia.org,
http://en.wikipedia.org/wiki/Unified_Modeling_Language [27/4/2010]

7)

## 1.6 Overview

The rest of the document will portray a detail description of how the core components of our system “Airnegy Wind Farm Planner” are designed. This helps give a better understanding on how to develop software that meets the requirements of our end user the RDA.

This document is technical due to the main audience will be a programming team and will need to have a good understanding of computer technical terminology. There is however an Acronym and definition section if the reader does not know specific terminology. The reason for this being the audience of this document is for the programming team to have a good understanding to carry out producing the software in the way it is intended and to not stray away from this specific guideline. All information is provided to do so.

This document is structured with all major components expected by the IEEE standards. There will also be diagrammatic explanations to show our understanding of the professional design principles to clearly explain our thought process. Some information is left out or not touched upon greatly if and only if it is very intuitive thinking that the programmers and designers will understand.

Diagrammatic explanations are used in the remainder of the document to present information more easily as well as show our understanding of professional design principles and to explain our thought process.

The first part of the document is showing the preliminary design of the system architecture and how the system has been decomposed. Once this is explained we will delve deeper into each component and see what they consist of and show their connection to other components if there are any. We also have appendices to help the reader understand concepts more if there is confusion.

The next section will then contain how the user should perceive the software to look visually with an explanation of why the designs should be used.

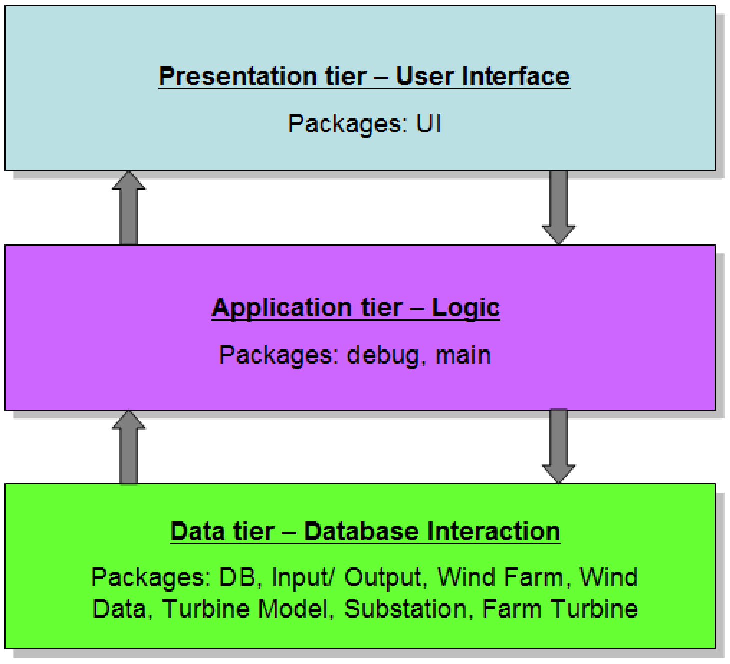
Finally there will be a requirements matrix to show that all requirements have instructions on how to be implemented and in which section of the document.

# ***Decomposition Descriptions***

## 2.1 Architecture

**Architecture Form Used**

We have decided to use the Three Tier Architecture. This means that we have broken the system up into 3 fundamental sections. Each section can travel up or down one layer but never skip in between.


Presentation tier – User Interface
Packages: UI
Application tier – Logic
Packages: debug, main
Data tier – Database Interaction
Packages: DB, Input/ Output, Wind Farm, Wind Data, Turbine Model, Substation, Farm Turbine

Advantages:

1. Due to the system being split into layers, this means that if anything has to be changed within the layer, it won't affect other layers in the model.
2. The user won't be able to access the data tier directly, so it is more transparent and they do not have unauthorised access to sections of the database.

Disadvantages:

1. The communication points have doubles since the user cannot access the data tier directly.
2. When processing the wind data, to put it into the database, it will take a long time to process, like 6 hours. This is only done initially, or when the data has to be changed when it has been updated by the RDA.

## Architecture Alternatives

- Model View Controller – The only difference in this architecture is that the user would be allowed to contact the Data Tier directly. Reason Why we don't want to use this….
- Transaction Processing System - This system is based on the user querying the database for specific information to either manipulate or use the information for other queries. The only thing is that if a process does not complete itself, our system does not have to rollback. This is because multiple processes does not happen within our system.

*How and why was the system decomposed?*

The system will be decomposed in an Object Oriented way, which means we will break the system down into objects which are related to the solution to the problem.

The java classes will be separated into packages based on the activities that they will perform. Data types (such as windfarms, substations, farmturbines, turbinemodels and winddata) and their respective controller classes will be separated into their own packages, each containing one data type and controller. User interface classes will be placed in the 'ui' package. Database classes will be put into the 'db' package. The packages will be grouped into our three layer architecture. The contents of the logic layer, such as calculations, conversions, settings and the main thread class will be put into the 'main' package. We will then design the system in an object-oriented manner, with all data types having get and set methods. Data type controller objects will keep track of their respective data types.

The system will be decomposed because not only will it be easier for the design team to work on different sections without disrupting fellow team members work, but will make the system easier to understand.

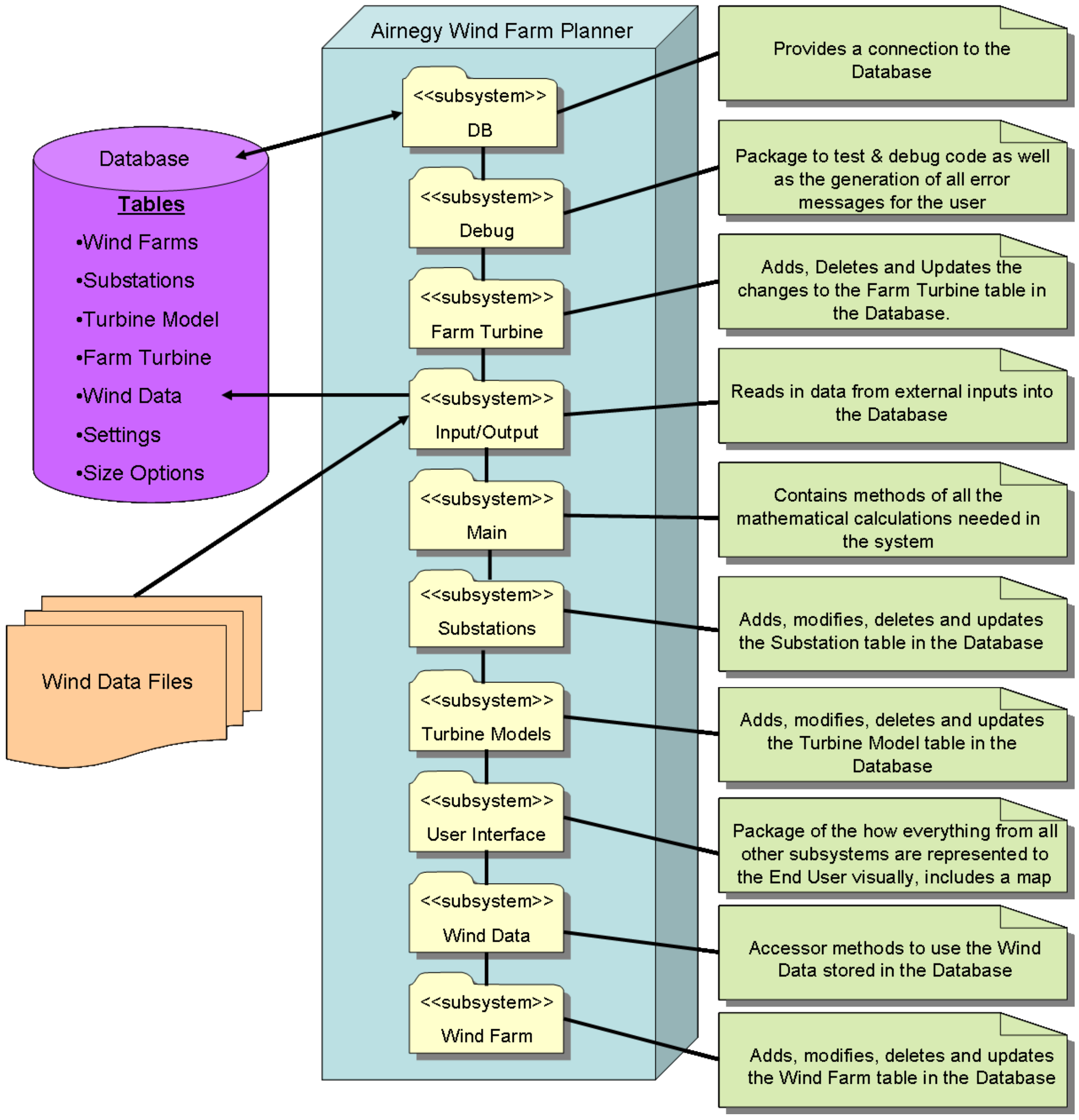
The Main components are:

- Database, this is where all the data is to be stored that they user wishes to input, it is also used to record the setting that the user wants
- Files – this the wind data files that the user needs to import into the database through the software
- Software – this is where the logic and user interface is to be implemented and broken down into further components of the packages used within the software.

*How do the individual parts work together?*

This will be explained in the interface specification in more detail than the UML diagram provided below to show the subsystems and what they may interact with.

## 2.2 High Level Overview


Airnegy Wind Farm Planner
Database
Tables
•Wind Farms
•Substations
•Turbine Model
•Farm Turbine
•Wind Data
•Settings
•Size Options
Wind Data Files
<<subsystem>> DB
<<subsystem>> Debug
<<subsystem>> Farm Turbine
<<subsystem>> Input/Output
<<subsystem>> Main
<<subsystem>> Substations
<<subsystem>> Turbine Models
<<subsystem>> User Interface
<<subsystem>> Wind Data
<<subsystem>> Wind Farm
Provides a connection to the Database
Package to test & debug code as well as the generation of all error messages for the user
Adds, Deletes and Updates the changes to the Farm Turbine table in the Database.
Reads in data from external inputs into the Database
Contains methods of all the mathematical calculations needed in the system
Adds, modifies, deletes and updates the Substation table in the Database
Adds, modifies, deletes and updates the Turbine Model table in the Database
Package of the how everything from all other subsystems are represented to the End User visually, includes a map
Accessor methods to use the Wind Data stored in the Database
Adds, modifies, deletes and updates the Wind Farm table in the Database

## 2.3 Responsibilities of Core Components with Software

**Interface Specification**

Due to the copious amounts of methods needed and the repetition in methods for different tables in the database, we have given a brief overview of what methods should be implemented. Also methods that don't seem intuitive an explanation is given to why it is needed. Also any mathematical formula needed will be provided unless stated otherwise.

**Package: Database**

Class: DatabaseConnector.java
Purpose: Initiates and maintains access to the database
Programming notes: Programmer needs basic SQL knowledge.

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| connection | Establishes a connection to the database using SQL Lite | Server Address, Driver Name | - | - |
| Get…. | Retrieves data from the database from the specified table | - | ArrayList of data from specified table | - |
| Update…. | Updates data in the database dependent on what the user wants to change | - | Tells the user that the database has been updated | - |
| Delete… | Deletes data the user does not require any more | Wind farm ID | Tells the user that the database has been updated | - |
| Terminate | Terminate Connection to the database | - | - | - |

## Package: Debug

Class: Assert.java
Purpose: Used for testing & debugging code
Methods:

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Debug | Modified version of the assert statement that provides extra information to the debugger | Boolean debug | TRUE/ FALSE | - |

Class: ErrorDialog.java
Purpose: Notifies the user of any errors or issues
Methods:

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Notfiy | Notify users of any information or errors during program execution | Type of notification( error / information), String Message | Dialog Box | Should be used every time when notifying the user |

## Package: farmturbine

Class: FarmTurbine.java
Purpose: Validates data before adding data to the farmTurbine table in the database
Methods:

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| setTableID | makes sure the ID is not already in the table and then sets the ID | String windFarmID | If the ID already exists, tell the user. | windFarmID > 0 &&windFarmID < 50000 |
| setNumberOfTurbines | Check value is bigger than 0 and less than 1000 | Int num | - | - |

Class: FarmTurbineController.java
Purpose: Maintains a list of all farm-turbine relationships that exist and provides accessor methods to them
Methods:

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| FarmTurbine Controller | Gets all already existing turbine models from the database and add each one to an arraylist | - | An Arraylist of all existing turbine models | - |
| addFarmTurbine | connects to the database and adds information to the table | Farm turbine object | Tells the user that the data has been/ not been added to the database | - |
| removeFarmTurbine | connects to the database and removes the specified row of data out of the table | Farm turbine object | Tells the user that the data has been/ not been removed to the database | - |
| getFarmTurbines | | windFarmID, turbineModelID, numberOfTurbines | | |
| updateTheDatabase | | OldFarmTurbine, newFarmTurbine | | |

**Package: Input/ Output**

Class: WindSpeedReader
Purpose: Reads in text files according to the wind speed and adds the data to the database.
Methods:

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| windSpeedReader | Reads in 3 files dependent on wind speeds, all files have the same layout, so need to allocate the data into the Wind Data Table in the Database. | 3 files, one for speed 10, 25 and 45 | - | - |

**Package: Main**

Class: Calculations
Purpose: All mathematical calculations needed to meet requirements
Methods:

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| circleAroundPoint | See appendix A.1 | Coordinates, radius | | |
| areaAroundPoint | See appendix A.2 | OSGB coordinates | | |
| distanceBetween2Points | See appendix A.3 | Coordinates | Double Distance | |
| wholesale | See appendix A.4 | Yield | String Wholesale | * In Kilo Watts |
| Renewable Energy Certificate (REC) | **See appendix A.5** | Yield | String REC | * In Kilo Watts |
| Revenue | See appendix A.6 | REC, Wholesale | String Revenue | |

Class:  LatLongToOSGB
Purpose: Convert latitude longitude values into OSGB 36 values.
There is a need for the changing between values dependent on what the user uses as input preference.

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Read | Reads in the latitude longitude values in as a user would input it in, then converts into decimal longitude latitude. | String Latitude, String Longitude | Should be in the form decimal latitude longitude values. | form e.g. 23 45 68N<br><br>lon = lonDeg + (lonMin/60) + (lonSec/3600) |
| Convert | See appendix A.7 | Double latitude decimal, double longitude decimal | Double northing, Double easting | |

Class: OSGBToLatLong.java
Purpose: Converts OSGB 36 northing, easting values to Longitude, latitude values. There is a need for the changing between values dependent on what the user uses as input preference. Decimal Format is needed for the Programmer, and the user format for the user.

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Convert | See appendix A.8 | Double northing, double easting | Double longitude decimal, Double latitude decimal | *accepts only northing easting WITHOUT the alphabetic values. |
| Decimal to Formatted Longitude latitude to User format | See appendix A.9 | Double Longitude, Double latitude | String Longitude, String Latitude | |

Class: OSGB
Purpose: The format of the wind data files are not in a standard format, so as a team we have decided it would be better if we converted it into OSGB and stored it in the Database like that. The data given in the wind data files are in the form (600, 1299), we then would like to convert it into OSGB 36 which looks like this NN 166 712. NEED TO ATTACH APPENDICE, DAN

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Wind data coordinates to OSGB36 | | String coordinate, e.g. 600, 1299 | String coordinate, e.g. NN 166 712 | *only round value in the last stage |
| OSGB 36 to wind data coordinates | | String coordinate, e.g. NN 166 712 | String coordinate, e.g. 600, 1299 | *Only round in the last stage |

Class: Settings.java
Purpose: This is to set the default settings that the user wishes to use and how they would like to view information. This is for if they wish to see and use kilometres or miles and latitude/longitude or OSGB36. Also users should be able to change the value of the Energy Certificates and Wholesale prices. These Settings are stored in the database in the Settings Table.

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Settings() | Makes a connection to the Database | - | - | - |
| Update Energy Certificates | Takes the value that the user inputted and changes the current value in the database to the one the user inputted | double energyCertificates | Tells the user that they have successfully/ unsuccessfully updated the setting. | - |
| Update Wholesale Prices | Takes the value that the user inputted and changes the current value in the database to the one the user inputted | double wholeSalePrice | Tells the user that they have successfully/ unsuccessfully updated the setting. | - |
| Update Number system Preference | If true then using the imperial system else metric system. | Boolean imperial | Tells the user that they have successfully/ unsuccessfully updated the setting. | - |
| Update Coordinate Preference | If true uses the longitude latitude system, else OSGB36 | Boolean latlong | Tells the user that they have successfully/ unsuccessfully updated the setting. | - |

**Package: Substations**

Class: substation.java

Purpose: Validates data before being entering it into the Substation table in the database

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Set Name | Value must be 20 characters or less | String Name | If Value is too long, tell the user. | - |
| Set size | Check value is 33 or 66 or 132, the sizes of substations. | int size | If not one of the specified values "Value entered for substation size is not equal to 33 or 66 or 132" | - |
| Set Current Capacity | check value is within sensible parameters | Int currentCapacity | If value is not within range, tell user. | * has to be greater than zero and smaller than Max capacity |
| Calculate Current capacity | See appendix A.10 | turbinemodels. TurbineModel a, double percentage, double speed | - | - |
| Set Max Capacity | check value is within sensible parameters | Int maxCapacity | If value is not within range, tell user. | * greater than zero. Also entered in Kilowatts |
| Calculate Power Deficit | See appendix A.11 | Int mazCapacity, currentCapacity | - | |
| Calculate Turbine Deficit | See appendix A.12 | TurbineModel. Model | Int numberOfTurbines | |
| Substation() | Adds data to the database | int substationID, String location, String name, int size, int currentCapacity, int maxCapacity | Tells the user if the data was successfully/ unsuccessfully added to the database | |

Class: SubstationController.java
Purpose: Keeps track of all the substations that exist in the system and provides accessor methods to them.

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Substation Controller | Gets all already existing substations from the database and adds each one to an the arraylist | - | - | - |
| Add Substation | Adds the data to the Substation table in the database and user the validators in the substation class. | Substation object | Tells the user if the data has been successfully been added or not | - |
| Remove Substation | connects to the database and removes the specified row of data out of the table | Substation Object | Tells the user if the data has been successfully been added or not | - |
| Get Substation | Returns all the substations stored in the database | - | All substations stored in the database | - |
| Update Substation | Takes the new substation object and replaces it with the old object | Substation object | Tells the user if the data has been successfully been added or not | - |

**Package: Turbine Models**

Class: TurbineModel.java

Purpose: Validates data that needs to be added into the Turbine Model table in the database.

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Get output curve | Need to get 5 values to store in the table. check value is bigger than 0 and less than 100 | Float startUpSpeed, float optSpeed, int quarter1Output, Int quarter2Output, Int quarter3Output | - | * these 5 values will be then make us able to calculate the energy a turbine can generate. |
| Set Turbine Model ID | Gives the turbine Model a unique ID | String ID | - | * Need to check if the ID doesn't exist in the table |
| Set Manufacturer | Makes sure the manufacturer name is less than 20 characters long | String Manufacturer | Tells the user if the Name is too long. | - |
| Set Height | Makes sure the height of the model is sensible | Int Height | If the value is too big, tell the user. | |
| Set Turbine Model Name | Makes sure the name is less than 20 characters long | String Name | If the value is too big, tell the user | |
| Calculate The power output speed | See Appendix A.13 | Float wind speed | Gives the user the output speed | |

**Package: Wind Farm**

Class: Windfarm.java

Purpose: Provides a set of validation for the wind farm table in the Database.

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Set location | Make sure the location is 20 characters or less | String Location | If the location is longer than that, tell the user. | |
| Set Wind farm name | Make sure the name is 20 characters or less | String Name | If the name is longer, tell the user. | |
| Set calculation to substation | See appendix A.14 | String Longitude, String Latitude, int Cost | String Calculation | |
| Calculate Energy Yield | See appendix A.15 | Turbine Model Object, double percentage, double speed | String Energy Yield | |
| Calculate wholesale return | See appendix A.16 | double percentage, double speed | String Wholesale | |
| Calculate Revenue | See appendix A.17 | double percentage, double speed | String Revenue | |
| Get total number of turbines | Presents how many turbines | - | Int Turbines | |
| Get turbine model names | Returns all the models names | - | String Arraylist | |

Class: WindFarmController.java
Purpose: Keeps track of all the wind farms in the database and provides accessor methods.

| Method | Description | Input | Output | Special Guidelines |
|---|---|---|---|---|
| Wind farm controller | Provides a connection to the database | - | - | - |
| Add Wind farm | Adds a wind farm object to the wind farm table in the database | Wind farm object | Tells the user if data has been successfully/ unsuccessfully been added to the database | *Must check if it already exists in the database table |
| Remove Wind Farm | Removes a wind farm object from the wind farm table in the database | Wind Farm object | Tells the user if data has been successfully/ unsuccessfully been removed to the database | - |
| Get Wind farms | Gets all the wind farms in the table and puts them into an arraylist | - | Arraylist of wind farms | - |
| Update Wind farm | Gets the new wind farm object and replaces the old one with it | Wind farm object | Tells the user if data has been successfully/ unsuccessfully updated the database | - |

Package: User Interface
Class: Deftero.java
Purpose: This is the main frame for which all the methods explained above is presented to the user. I do not think that it is suitable to go through what methods should be used. In the GUI design section we will discuss what the user interface will look like.

Class: Map.java
Purpose: Once again it is not suitable to list methods, due to using all the above methods. In the GUI design section there will be mock ups of what the Interface for the map will look like.

## 2.4 Intermodule dependencies

Insert Class diagram

## 2.5 Interpocess Dependencies

Insert Sequence diagrams
Insert Activity Diagrams

## 2.6 Requirements Traceability Matrix

| Unique UAT ID | UAT Description | Design Reference |
|---|---|---|
| 2.1 El-FR1 | Add Electrical Substation | |
| 2.2 El-FR2 | Delete Electrical Substation | |
| 3.1 T-FR1 | Add Turbine Model | |
| 3.2 T-FR2 | Turbine Power Output Curve | |
| 3.3 T-FR3 | Delete Turbine Model | |
| 4.1 WD-FR1 | Import Wind Speed Data | |
| 4.2 WD-FR2 | Query Wind Speed Data | |
| 5.1 WEl-FR1 | Add Wind Farm | |
| 5.2 WI-FR2 | Delete Wind Farm | |
| 6.1 MWF-FR1 | Find Potential Wind Farm Site By Substation | |
| 6.2 MWF-FR2 | Model Potential Wind Farm | |
| 6.3 MWF-FR2 | Model Potential Wind Farm Revenue Yield | |