Technical University of Denmark

Matthias Larsen, s103437
Joachim Jensen, s103430

UML

A Graphical User Interface (GUI) for weather radar and wind energy data visualization and analysis

02350 Windows Programming using C# and .Net, December 2012

DTU Informatics
Department of Informatics and Mathematical Modelling

Matthias Larsen, s103437
Joachim Jensen, s103430

UML

A Graphical User Interface (GUI) for weather radar and wind energy data visualization and analysis

02350 Windows Programming using C# and .Net, December 2012

UML,A Graphical User Interface (GUI) for weather radar and wind energy data visualization and analysis

This report was prepared by
Matthias Larsen, s103437
Joachim Jensen, s103430

Supervisors
Bjarne Poulsen

Release date:	December, 2012
Category:	1 (public)
Edition:	First

Department of Informatics and Mathematical Modelling
Technical University of Denmark
Asmussens Alle building 305
DK-2800 Kgs. Lyngby
Denmark

www.imm.dtu.dk
Tel: (+45) 45 25 33 51
Fax: (+45) 45 88 26 73
E-mail: reception@imm.dtu.dk

Abstract

Data for weather analysis and forecasts consists of enormous amounts of data.

Comparing these huge amounts of data quickly becomes difficult. Most people interpret images easier than numbers, hence the need for some sort of visualization for the data sets.

The data come from multiple sources (e.g. on site observations from measuring stations, meteorological forecasts from Numerical Weather Prediction models) and, consequently have very diverse formats (time series, georeferenced data, gridded data).

These diverse formats raise an important issue because no common or efficient platform for visualizing and analyzing these data exists except for two earlier attempts: one in MATLAB and one with Google Maps. These earlier application were not flexible enough to be used in a bigger context.

This paper shows the development and considerations throughout the project of an application that might serve as the beginning of a new common open source platform for analyzing, visualizing and comparing weather related data.

Contents

List of Figures

List of Tables

Listings

1 Introduction

The problem in this project is to analyse and develop an C# .NET application to manage UML class diagrams in a way so that it can be extended with other types of UML diagrams in the future.

This is interesting because different design patterns and methods can be used

2 Analysis

A lot of considerations have gone into the development process of the application. The following chapter explains the considerations with the most impact on the application.

TÆLLER 5

2.1 Choice of diagram

While state and sequence diagrams show how (part of) an application works in a detailed way, class diagrams give an overview over all or some classes in the application, what properties and methods they contain and how they are related to each other. In this way, class diagrams can be used to easily see how an application is modelled and thus how it can be altered or extended. Therefore, class diagrams are used more often than state and sequence diagrams when it comes to software architecture, and therefore an application to manage class diagrams has been chosen.

2.1.0.1 Solution strategy

Due to limited time, not all specifications for a UML class diagram can be implemented in the application, and thus we have made a prioritized specification list, an iteration plan, where the first element is of highest priority and will be designed and implemented first.

1. Different types of classes
2. Undo and redo functionality
3. CRUD for classes
4. Movable classes
5. Relations between classes
6. Save and load functionality
7. Automatic save functionality

One should think that *CRUD for classes* and *Movable classes* is more important than *Undo and redo functionality*, but as they are highly dependent on *Undo and redo functionality*, this has higher priority. E.g. if *CRUD for classes* was made before *Undo and redo functionality*, one could risk having to modify it all again in order to get undo and redo to work properly in the end.

2.1.1 Risks

Because of the solution strategy where a functionality has been given a priority corresponding to its importance, the risks of not implementing one of the specifications due to limited time have been minimized. If *Automatic save functionality* was left out, it would not cause the application to work improperly as it has no impact on the actual purpose - that is, creating an application that can manage UML class diagrams.
This is an advantage of using an iteration plan in the process of software development.

3

Design & Implementation

3.1 GUI

TÆLLER 35

The GUI consists of a menu, a two-column grid and a status bar. The features that can be interacted with in the GUI have been outlined below.

3.1.1 Editable classes

There are different gestures for creating, updating and deleting classes, each chosen to get the best user experience. In the left grid column, one can click on a button to create a specific type of class in the application. At the moment there are four types to choose among:

- Class
- Abstract Class
- Enum
- Struct

Clicking one of the buttons will also insert a box in the right grid column representing the class in the application. The color of the boxes are determined by the type of the classes.

Deleting a class is done by right clicking on the specific box representing that class.

If one wants to edit a class, this is done by double (left) clicking on the specific box. This will cause the box to switch to an editable field where one can type in the name, properties and methods for the class in accordance with the syntax of UML class diagrams. Clicking on the same box again or double clicking on another box will save the changes.

3.1.1.1 Regex

3.1.2 Movable classes

3.1.3 Undo and redo

3.1.4 Relations

3.1.5 Menu and shortcuts

3.1.6 Status bar

3.2 Save and load

3.3 Application layer

TÆLLER 20

dll different class types thread

3.4 Design patterns

2-3

TÆLLER 10

3.5 Tests

TÆLLER 10

3.6 Noget der ikke er udnervist i

TÆLLER 5

4 Conclusion

A weather visualization application that met the requirements was created successfully.

The implementation of different file types, data visualizing and database integration, to make it easier and to get higher performance, are completed with the help of tools and methods stated in this paper while keeping focus on extensibility, speed and user experience.

Due to lack of time, the support for NetCDF files were not implemented, but the flexibility of the software allows this to be done at a later date without breaking any existing functionality.

The final product ended up being a satisfactorily solution for a weather visualization application that can manage the enormous amounts of data in an intuitive way while being a viable candidate for a future common platform in the industry.

4.1 Future work

Besides fixing the currently known bugs stated in **??**, we have some ideas for the application, some of which have been described in 2 that did not make it to the existing application.

A

Use cases

We have two types of actors:

- User: Visitor.
- Admin: User with access to the administration panel.

A.1 Clicking on a radar at the map

Actor: User

Scenario:

- Checks if there is data for the date
- If images is not generated, it will generate them real time.
- Radar shows images over an interval.
- *One click for play/pause and double click for reset.*
- When there is no more data, the radar resets.

Alternative scenario:

1. No data is available for that moment and a message appears.

A.2 Clicking on a wind farm at the map

Actor: User

Scenario:

- Opens chart dialog.
- Chart data is retrieved in the background
- Data is visualized if available

A.3 Changing view in chart for wind farm

Actor: User

Scenario:

1. Changing to 2-week view.
    - Changing chart view interval to 2 weeks.
    - Load new data for the new interval.
2. Changing to weekly view.
    - Changing chart view interval to 1 week.
    - Load new data for the new interval.
3. Changing to daily view.
    - Changing chart view interval to 1 day.
    - Load new data for the new interval.

A.4 Changing data set in chart

Actor: User

Scenario:

- Selecting a new data set
- The data is retrieved from the server
- Visualize the data if available

Alternative scenario

- Unselecting a data set
- The data is removed from the chart

A.5 Using control buttons in chart for wind farm

Actor: User

Scenario:

1. Pressing fast backward.
    - Go two weeks back on 2-week view, one week back on weekly view and one day on daily view.

- Loads new data.

2. Pressing backward.
   - Go one week back on 2-week view, one day back on weekly view and one hour on daily view.
   - Loads new data.
3. Pressing play.
   - Change to daily view and go one hour forward every second.
   - Loads new data.
4. Pressing forward.
   - Go one week forward on 2-week view, one day forward on weekly view and one hour on daily view.
   - Loads new data.
5. Pressing fast forward.
   - Go two weeks forward on 2-week view, one week forward on weekly view and one day on daily view.
   - Loads new data.

A.6 Upload file

Actor: Admin

Scenario:

- Sign in with correct username and password.
- Add new file (CSV, WRK, ZIP) and enter correct timezone.
- If file is ZIP, it will decompress.
- File is uploaded.
- File is being parsed.

Alternative scenario:

1. Sign in with wrong username or password.
2. Adding file that is not correct format and get an error message.
3. Adding ZIP file, but files inside zip are not correct. They will be skipped.
4. File exists and will not be uploaded.

www.imm.dtu.dk

Department of Informatics and Mathematical Modelling
Technical University of Denmark
Asmussens Alle building 325
DK-2800 Kgs. Lyngby
Denmark
Tel: (+45) 45 25 33 51
Fax: (+45) 45 88 26 73
E-mail: reception@imm.dtu.dk