Projektbericht

Marc Fuchs
Matrikel-Nr.: 16452
Student der Medieninformatik (Bachelor)
an der
Hochschule der Medien Stuttgart
angemeldet als
Software-Projekt (EDV-Nr. 20556)

Kurzbeschreibung

Das Projekt WindGUI – Visualisierung von Windenergiedaten - entstand in Zusammenarbeit mit Prof. Dr.-Ing. Johannes Maucher als Betreuer seitens der Hochschule und Dr. Martin Felder vom Zentrum für Sonnenenergie- und Wasserstoff-Forschung (Standort Stuttgart-Vaihingen).
Ziel war die Entwicklung einer Datenbank mit angegliederter graphischer Benutzeroberfläche (GUI), um

1. Daten von verschiedenen Windpark- und Windkraftanlagenbetreibern zu erfassen und in einer Datenbank persistent zu halten. Diese Daten haben sowohl statischen (Standort etc.) als auch dynamischen (Windgeschwindigkeit) Charakter.
2. die erfassten Daten graphisch in Form von Histogrammen und Polarplots darstellen zu können

Die Umsetzung des Projektes erfolgte in der Skriptsprache Python (Version 2.5); als GUI-Toolkit kam PyQT und als Datenbank MySQL zum Einsatz.

Das Projekt ist ein „Ein-Mann-Projekt“, d.h. die Programmierung erfolgte ohne Hilfe Dritter, wobei natürlich Input von seitens meiner Betreuer sehr wohl mit in die Programmentwicklung einfloss, die praktische Umsetzung erfolgte aber durch meine Person.

Desweiteren versichere ich hiermit, dass ich nur die in diesem Dokument aufgeführten Quellen, Bücher und Programme und keine weiteren Hilfsmittel benutzt habe.

Ludwigsburg, den 23.Juli 2009,

_____

Gliederung des Projektberichtes

1. Der Weg zum fertigen Programm
    1.1 Auswahl der Tools und Entwicklungsumgebungen
    1.2 eingesetzte Bücher
    1.3 Programmierparadigmen und Design Patterns
    1.4 Probleme während der Umsetzung sowie deren Lösung
    1.5 zeitlicher Ablauf des Projektes

2. Die Installation von WindGUI
    2.1 Grundsätzliche Funktionsweise
    2.2 Systemvoraussetzungen
    2.3 Tutorial und Anleitung zur Einführung in WindGUI

3. Abschließende Beurteilung der gewonnenen Erfahrung
    3.1 Betrachtungen über Python
    3.2 Persönlicher Lernerfolg durch das Projekt
    3.3 Fazit des Projekts und mögliche Weiterentwicklungen

1. Der Weg zum fertigen Programm

1.1 Auswahl der Tools und der Entwicklungsumgebung

Zuerst möchte ich an dieser Stelle betonen, dass mir bezüglich der Umsetzung sowohl von meinem betreuenden Professor Dr. Johannes Maucher als auch von Dr. Martin Felder stets freie Hand bei der Entwicklung gelassen wurde, was ich sehr geschätzt habe, da ich so sehr gut verschiedene Strategien und Design-Patterns ausgiebig erproben und auch einsetzen konnte. Die Umsetzung in Python hat mir gezeigt, dass in dieser Sprache sehr schnell viele Möglichkeiten ausprobiert und miteinander verglichen werden können und Python somit nicht umsonst als Paradebeispiel für eine Sprache steht, die für Rapid Prototyping geradezu prädestiniert ist.

PyDev
Am Anfang des Projektes stand noch völlig offen, welche Entwicklungsumgebung (IDE) eingesetzt wird, da Python „von Haus aus“ zwar eine IDE namens IDLE mitliefert, diese aber, um es dezent auszudrücken, doch etwas „befremdliche“ Bedienung ermöglicht. Aber dieser Aspekt liegt immer im Auge des Betrachters, somit maße ich mir keine Aburteilung von IDLE an.
Aber da ich im Laufe meines Studiums hauptsächlich mit Java in Berührung gekommen war und dabei die IDE Eclipse eine zentrale Rolle spielte, lag die Entscheidung nahe, das Eclipse-Plugin PyDev zur Erstellung des Quellcodes zu benutzen. PyDev bot sich auch insofern an, als der Umstieg als solcher eigentlich gar keiner war, da ja immer noch Eclipse das Fundament bildete und ich in meinem bisherigen Studium genügend positive Erfahrungen mit Eclipse sammeln konnte.

Python(x,y)
Als wirklichen Meilenstein im Einsatz von Python kann man die wissenschaftliche Entwicklungssoftware Python(x,y) bezeichnen, da diese schon eine Vielzahl an Bibliotheken und Modulen speziell zu Python mitliefert und gerade für den wissenschaftlich-mathematischen Einsatz z.B. durch matplotlib, numpy und scipy schon bei der Grundinstallation sehr mächtige und universell einsetzbare Bibliotheken zur Verfügung stellt.
Gerade matplotlib bietet mannigfaltige Möglichkeiten zur Visualisierung von Daten mit einer quasi selbst erklärenden Syntax. Als Matlab-User wird man mit matplotlib sehr schnell vertraut, da viele Funktionen ähnlich sind und man sehr schnell zu ansprechenden Resultaten kommt. Ich war insbesondere bei der Erstellung von Histogrammen und Polarkoordinatenplots innerhalb

kürzester Zeit in der Lage, optisch ansprechende Ergebnisplots zu erstellen.

MySQL
Die Entscheidung für MySQL als Datenbank fiel mir leicht, da ich schon während meines Praxissemesters bei T-Mobile sehr viele Erfahrungen damit sammeln konnte und MySQL einige interessante Funktionen bietet, die einen enormen Geschwindigkeitsvorteil bei bestimmten Operationen erlauben sowie die Möglichkeit bieten, verschiedene Datenbank-Engines wie MyISAM und InnoDB einzusetzen.
Darüber hinaus war es auch Bedingung des Projekts, kostenlose Software einzusetzen. Hierbei zeigte sich, dass die eingesetzten Tools ihren teilweise sehr teuren Pendants (ich wage an dieser Stelle Oracle als Beispiel zu erwähnen) in vielen Punkten in nichts nachstehen.
Nichtsdestotrotz hat auch MySQL seine Schwächen, da beispielsweise das Erstellen von Datenbank-Triggern noch in den Kinderschuhen steckt, hier hat Oracle (noch) „die Nase vorn“. Aber MySQL schließt auch in diesen Disziplinen zu seinen kostspieligen Mitbewerbern auf.

phpMyAdmin
Zur Administration der Datenbanken während des Entwicklungsprozesses wurde phpMyAdmin eingesetzt, ein in PHP geschriebenes Tool zur Verwaltung, Erstellung und Vergleich von Datenbanken sowie Operationen auf den Tabellen. Darüberhinaus erlaubt es, die Geschwindigkeit von Datenbank-Operationen zu ermitteln und damit z.B. bestimmte SELECT-Statements hinsichtlich ihrer Geschwindigkeit zu optimieren. Zudem erlaubt phpMyAdmin Tabellen anzulegen oder abzufragen und dabei den dazu benutzten SQL-Code anzeigen zu lassen, sodass dieser per Copy&Paste in die entsprechenden Python-Module übernommen werden kann.

PyQT
Die Entscheidung für ein GUI-Toolkit zur Erstellung von graphischen Oberflächen war schon etwas kniffliger, da zwar Python „von Haus aus“ eine GUI mitliefert (Tkinter), diese aber in den Möglichkeiten doch sehr eingeschränkt ist.
Schnell war klar, dass hier eine andere Möglichkeit gefunden werden musste, denn gewisse Grundzüge der GUI sollten doch ansprechend darzustellen sein und vertraut aussehen, was bei Tkinter definitiv nicht der Fall ist.
Die Entscheidung engte sich daher auf die beiden häufig eingesetzten GUI-Toolkits wxPython und PyQT ein.
Letztlich entschied ich mich für PyQT, ein Wrapper für das in C++ umgesetzte Toolkit Qt, da dieses sehr ansprechende Oberflächen erlaubt. Als Beispiele für Programme mit Qt sind beispielsweise Google Earth, Skype und Mathematica zu nennen.

Folglich entschied ich mich, hier also PyQT zu benutzen. Ein Schritt, den ich nicht bereute, da PyQT mit relativ wenig Befehlen durch eine ausgeklügelte Klassenhierarchie und Layoutmanager sehr komplexe und zugleich logisch aufgebaute Oberflächen ermöglicht.

1.2 Eingesetzte Bücher und sonstige Hilfen

Da ich zu Beginn des Projektes absoluter Neuling in Bereich Python war, war der erste Schritt natürlich die Aneignung der grundsätzlichen Syntax sowie der Datenstrukturen und spezifischen Eigenheiten dieser Sprache. Dafür hatte ich zwei hervorragende Bücher zur Hand:

-*Python – das umfassende Handbuch*
von Johannes Ernesti und Peter Kaiser (Galileo Computing)

-*Python*
von Farid Hajji (Addison-Wesley)

Mittels dieser beiden Bücher konnte ich mir relativ schnell die Grundzüge und Eigenschaften von Python aneignen (insbesondere die Python-spzifischen Datenstrukturen wie etwa Dictionaries) und darüber hinaus gut als Nachschlagewerk gebrauchen; insbesondere das Buch von Farid Hajji bietet eine hervorragende Python-Referenz.
Der zweite Schritt war dann, mir gezielt Wissen über die Erstellung von graphischen Oberflächen anzueignen. Da ich mich bewusst für PyQT entschieden habe, erwarb ich das Buch

- *Rapid GUI Programming with Python and Qt – The Definitive Guide to Python Programming* von Mark Summerfield (Prentice Hall).

Meiner Meinung nach ist dieses Buch mithin eines der besten Programmierbücher, welche ich bis dato gelesen habe. Es wird nicht einmal besondere Erfahrung mit Python zum Einstieg in die Oberflächenentwicklung mit PyQT vorausgesetzt und bietet in den ersten Kapiteln sogar eine Art „Crashkurs“ für absolute Python-Neulinge.
Dieses Phänomen ist mir auch schon bei anderen englischsprachigen Büchern aufgefallen, die von einer relativ „populärwissenschaftlichen“ Ebene zu Beginn des Buches sukzessive immer mehr ins Detail gehen, gegen Ende auch die komplexe Materie erfassen und dabei der Roten Faden trotzdem immer klar erkennbar lassen.

1.3 Programmierparadigmen und Design Patterns

Eine gute und saubere Strukturierung eines Programms ist Grundvoraussetzung für sauberes und vor allem nachvollziehbares Software-Design.
Unter dieser Prämisse begann ich mit der Entwicklung. Viele Dinge waren von Anfang an klar: Als wichtiges Entwurfsmuster ist der Model-View-Ansatz zu nennen; also die Trennung von Programmlogik und -oberfläche.

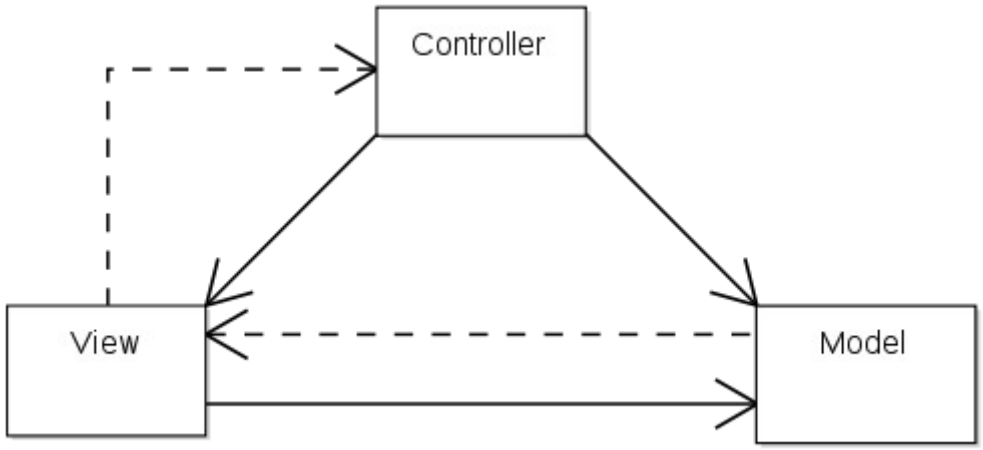

Abbildung 1: Model-View-Controller Pattern (herkömmliche Form)

Dabei hielt ich mich jedoch bewusst nicht sehr streng an die spezielle Trennung von Bedienung und Darstellung (also View und Controller), da diese nämlich insbesondere bei PyQT soweit ineinander verzahnt ist, dass dies gar nicht möglich ist.
Es gibt zwar bei PyQT die Möglichkeit, durch den Einsatz der sogenannten Convenience Item Widgets vorgefertigte Entwurfsmuster zu verwenden, die auf einer Art Model-View-Delegate-Ansatz beruhen, ich bin jedoch von diesem Ansatz aus 2 Gründen bewusst abgewichen:

1. Die Convenience Item Widgets sind gut bei kleineren Anwendungen, aber bei großen leidet die Flexibilität, weil eben nur bestimmte Darstellungsweisen umgesetzt werden können
2. Jeder, der dieses Projekt weiter entwickelt, muss sich erst mühsam in diese Convenience Item Widgets einarbeiten, um überhaupt zu verstehen, etwa hinsichtlich der Frage, warum z.B. QVariant-Objekte zur Parameterübergabe zwischen Model und View verwendet werden

Stattdessen wählte ich einen eigenen Ansatz, der weitestgehend selbsterklärend ist. D.h. Die Parameter wurden quasi „nativ“ übergeben und, wenn notwendig auf einen anderen Datentyp „gecastet“. Dies war z.B. notwendig bei Datenbank-Queries. Da hier die MySQLdb-Bibilothek eingesetzt wurde, liefert diese immer *Tuple*-Instanzen zurück. Damit die Inhalte dieser Tupel aber richtig angezeigt werden, müssen die Tupel-Werte

in Strings oder Integer-Werte umgewandelt werden.

Hierzu ein Beispiel:

```
def getSelectedCols(self):
        colList = []
        """
        returns a list of the names of all SELECTED columns
        """

        sql = "SELECT * FROM `wka`.`showedparameters`"
        self.curs.execute(sql)

        for row in self.curs:
            colList.append("%s" % row)

        return colList
```

Zuerst eine kleine Anmerkung:
Der Text zwischen den 3 Anführungszeichen ist ein sog. Docstring und kann als Kommentar sowie zur Beschreibung der Funktionalität einer Methode verwendet werden. Hiervon machte ich auch ausgiebig Gebrauch, d.h. es finden sich im Quelltext nur wenige „herkömmliche“ Kommentare (die ja mit dem #-Zeichen definiert werden), vielmehr beschrieb ich die Funktionen mit eben diesen Docstrings.
Docstrings können dann in späteren Schritten auch verwendet werden zum Erstellen einer Online-Dokumentation.
Diese Funktion gibt Spalten einer Tabelle als Liste zurück. Da die Datenbankabfrage über MySQLdb Tuple-Datentypen liefert, iterierte ich nun über den ResultSet des Abfrageergebnisses, „castete“ den jeweiligen Wert in einen String und hängte diesen an die Liste an.

Ebenfalls wichtig war mir die Kapselung und Trennung der Anwendungslogik in eigenen Klassen und Bibliotheken. Ein Grundsatz, den ich verfolgte, war, dass die Programmlogik auch eigenständig ohne GUI nutzbar sein soll.
Das eröffnet z.B. auch später die Möglichkeit, das Programm komplett ohne GUI nur von der Python-Kommandozeile aus zu bedienen, was bei Batchprozessen, wenn viele csv-Dateien als Input vorliegen, durchaus Sinn macht.

1.4 Probleme während der Umsetzung sowie deren Lösung

Im Grunde lief das Ganze ohne größere Probleme ab, die meisten Fragestellungen und Probleme waren kleinerer Natur, wie etwa falsche

Typkonvertierungen oder falsch gesetzte Slots und Signale in PyQT, und konnten durch konsequentes Debugging einfach verbessert werden.

Ein wenig knifflig war auch die Erstellung der Mapping-Dialoge, da diese nach meiner Vorstellung genau so viele Auswahlboxen haben mussten wie die zugrunde liegende csv-Datei Spalten hat. Dieses Problem habe ich nach einiger Überlegung mit Listen gelöst, deren Länge durch die Anzahl der Spalten in der csv-Datei bestimmt wird und diese Länge dann als Grundlage für die Anzahl der Auswahlboxen genommen wird.

Ein weiteres Problem war die relativ langsame Einlesegeschwindigkeit in die Datenbank, wenn dies über die csv-Bibliothek erfolgt. Bei kleineren Datenmengen macht dies noch nicht viel aus, bei csv-Dateien, die durchaus Größen von 100 MB und mehr annehmen können, wird dies aber eine sehr langsame Angelegenheit.

Die Lösung dieses Problems fand ich in einem MySQL-Befehl, der es erlaubt, große Dateien in Sekundenschnelle eine vorher erzeugte temporäre Tabelle zu transferieren (LOAD DATA (local) INFILE.

Hier ein Code-Beispiel:

```
tempfile = 'C:\\\y.csv'

sql = """LOAD DATA INFILE '%s' INTO TABLE `wka`.`temp`
            FIELDS TERMINATED BY ';' ENCLOSED BY '"'
            LINES TERMINATED BY '\r\n'""" % tempfile
```

Von der erzeugten Temp-Tabelle werden die Daten wiederum direkt mit einem SQL-Befehl (COPY TABLE) in die zentrale data-Tabelle geschrieben. In dieser zentralen Tabelle laufen alle Daten zusammen und werden mit einem Primärschlüssel, der den Namen der Windkraftanlage trägt, auseinander gehalten.

1.5 Zeitlicher Ablauf des Projektes

Der Ablauf stellt die grobe Reihenfolge dar, wie das Projekt begründet und vorangetrieben wurde.

1. intensives Einarbeiten in die Materie Python und PyQT mit den bereits genannten Büchern (ca. 30 h).
2. Implementierung der Datenbank-Anbindung mittels der Python-Bibliothek MySQLdb (ca.30 h)
3. Erstellen einer interaktiven Benutzeroberfläche. Dieser Projektabschnitt zog sich von Ende April bis zum Abschluss der Programmierarbeiten und war der Kern der eigentlichen GUI.

Dies musste zwangsläufig nebenläufig zu den anderen Schritten geschehen, da nur so neu hinzu gekommene Funktionalitäten zeitnah eingebunden werden konnten (ca. 90 h)

4. Es können Windpark-Betreiber angelegt werden. Nun müssen die csv-Dateien, die von den Betreibern der Anlagen kommen, diesen zugewiesen werden, ausgewählt und in die in die Datenbank eingepflegt werden können. Dabei kann es auch vorkommen, dass in einer csv-Datei bestimmte Zeitbereiche fehlen, die zu einem späteren Zeitpunkt nachfolgen können (ca. 100 h).
5. Implementierung der Datenvisualisierung mit matplotlib. Damit wurde die Erstellung von Histogrammen und Polarkoordinatenplots ermöglicht. (ca. 70 h)
6. Erstellen der Projektdokumentation (ca. 30 h)

2 Die Beschreibung der Elemente des Programmes

2.1 Grundsätzliche Funktionsweise

Das Programm ist in mehrere Teilmodule aufgesplittet. MainWin.py ist das Hauptprogramm; hier werden das Hauptfenster und die Dialoge graphisch umgesetzt.
Die Programmlogik ist hauptsächlich in der Datei model.py zu finden.
Desweiteren sind spezielle Funktionalitäten wiederum in eigene Programme aufgeteilt(import.csv, importWindparkCsvFile.py, createWindparkMapping.py , Plots.py) um so dem Model/ViewController-Ansatz gerecht zu werden.
Für meiner Meinung nach besonders wichtige Programmteile wurden eigene Dateien angelegt, um hier noch eine weitere Trennung zu implementieren.

2.2 Systemvoraussetzungen

Für den Betrieb von WindGUI müssen bestimmte Mindestvoraussetzungen erfüllt sein:

- PC mit Windows XP SP2
- Python(x,y) in der Default-Einstellung muss installiert sein. Die Standardinstallation umfasst alle relevanten Bibliotheken
- XAMPP muss installiert sein (oder alternativ MySQL selbst installiert werden). XAMPP umfasst alle datenbankrelevanten Programme sowie die automatische Installation und Konfiguration von MySQL-Server und phpMyAdmin. XAMPP kann von folgender Webseite heruntergeladen werden: http://www.apachefriends.org/de/xampp-windows.html#628

  Dort den Installer für die aktuelle Version (1.7.1) herunterladen; die Installation kann mit den Standardeinstellungen erfolgen und sollte keiner weiterer Erklärungen bedürfen.
- Nach der Installation von XAMPP in das Verzeichnis navigieren, in welchem XAMPP installiert wurde (C:\Programme\XAMPP). Dort auf xampp-control.exe klicken. Nun startet die XAMPP Control Panel Application. Hier bei Apache und MySQL auf den *Start*-Button klicken.
- Nun kann durch Eingabe von *localhost* in die Adressleiste des Webbrowsers geprüft werden, ob XAMPP (und damit auch MySQL) läuft.
- War die Installation erfolgreich, erscheint nun im Webbrowser die

Startseite von XAMPP (XAMPP für Windows)

Der Vorteil von XAMPP besteht darin, dass durch dessen einfache Installation sowohl MySQL als auch ein Apache-Webserver installiert werden und mit dem mitinstallierten phpMyAdmin sehr leicht zu Kontrollzwecken auf die Datenbanken zugegriffen werden kann.

2.3 Tutorial und Anleitung zur Einführung in WindGUI

Ich habe die Benutzerführung weitestgehend selbsterklärend aufgebaut, trotzdem möchte ich dieses Tutorial auch als „Bedienungsanleitung“ sehen, da alle wichtigen „Stationen“ tangiert werden.

Mit diesem Tutorial werden die umgesetzten Funktionalitäten des Programmes aufgezeigt und erlauben die Sichtung der Dialoge und Plots. Dazu wird ein Beispiel-Mapping für eine Windkraftanlage angelegt und daraufhin mittels dieses Mappings eine csv-Datei in die Datenbank gespielt.

Mittels der erfassten Daten werden dann im Anschluss mit der matplotlib-Bibliothek Histogramme und Polarkoordinaten-Plots angezeigt.

Vorgehensweise:

- Sicherstellen, dass XAMPP (respektive der MySQL-Server) läuft
- Starten der mainWin.py

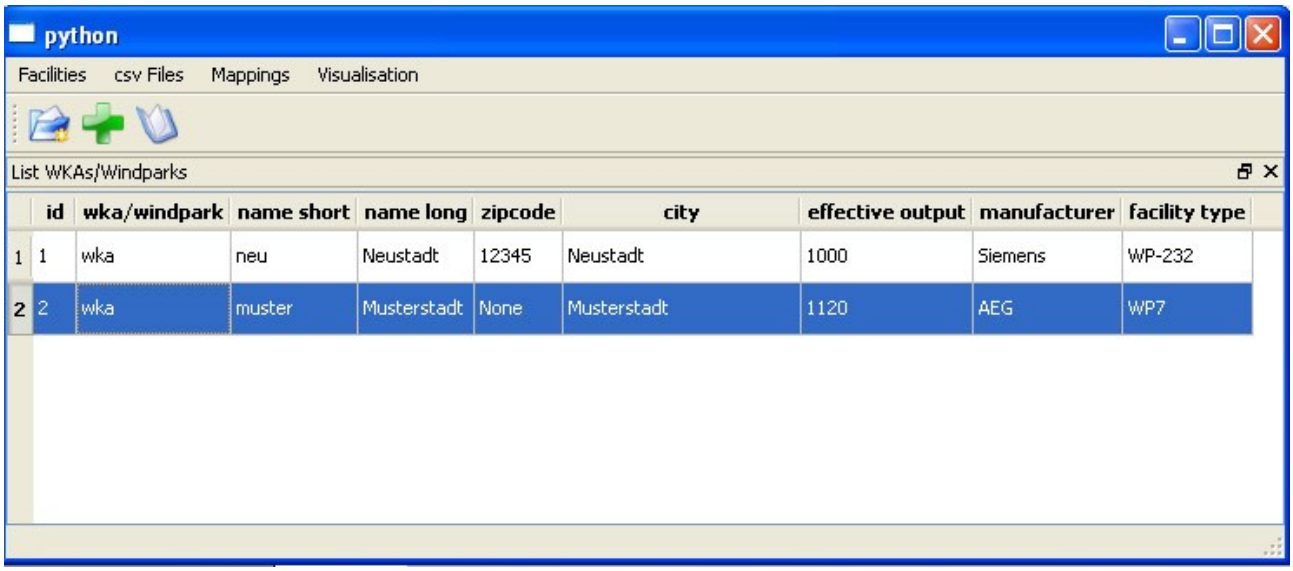

Abbildung 2: Hauptfenster der Anwendung

- daraufhin im erscheinenden Hauptfenster eine Windkraftanlage (WKA) anlegen: Durch einen Klick auf den Button „Add Facility“ (grünes Kreuz in der Toolbar) oder über das Menü *Facilities->Add new Facility* wird

eine neue Spalte in der Tabelle angelegt. Dann einfach die Felder mit den entsprechenden Werten füllen (Vorschlag: in der Spalte „Name short“ als Namen „TestWKA“ eintragen.

- Per Default wird beim 1.Start des Programmes nur der Kurzname der Windkraftanlage angezeigt, daher...
- ...können im Menü *Facilities->Choose Columns in facility list* die Spalten, die in der Tabelle angezeigt werden sollen, ausgewählt werden Wir klicken in diesem Dialog einfach ein paar Checkboxen an, klicken unten auf den Button „Apply Changes“ und sehen, dass die zugehörigen Spalten in der Tabelle angezeigt werden.
- Wir können nun durch Doppelklick (oder Mauszeiger auf die Zelle und F2) Tabellenzellen die Werte verändern. Nach Drücken der Enter-Taste werden diese direkt in die Datenbank übernommen.
- Nun muss ein Mapping angelegt werden. Mit dem „Buch“-Icon in der Toolbar oder mit dem Eintrag *Mappings->Create new WKA mapping* wird der Mapping-Dialog aufgerufen
- Im Mapping-Dialog nun durch Klick auf den Button „Open csv file“ die entsprechende csv-Datei (TestWKA.csv) der Windkraftanlage auswählen

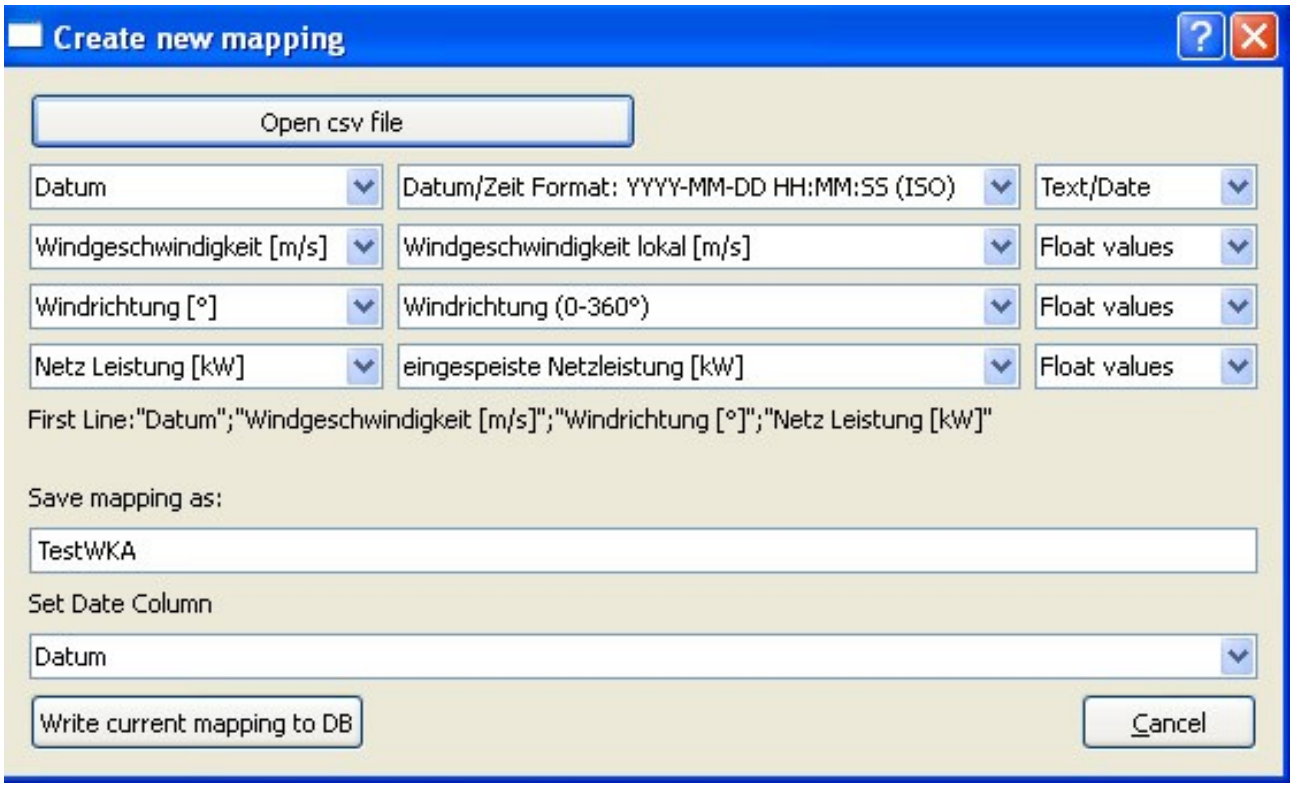

Abbildung 3: Mapping-Dialog

- Es erscheinen nun neue Auswahlmenüs (ComboBoxes) im Dialog – für jede „Spalte“ der csv-Datei wird ein Auswahlmenü angezeigt. Rechts sehen wir die Spalten der csv-Datei und links die Spalten der Datentabelle, welcher die Spalten zugeordnet werden sollen. In der 3. Auswahlmenü-Spalte sind die Auswahlmenüs für den Datentyp – Float für Zahlenwerte und Text/Date für Datum- und Textwerte – zu sehen.
- Nun erfolgt das eigentliche Mapping – dazu werden zu den Auswahlmenüs der 1.Spalte die Datenbank-Spalten zugeordnet. Der Einfachheit halber wählen wir hier die Werte so aus wie im hier gezeigten Screenshot. Als Namen des Mappings (im Textfeld „save mapping as:“) nehmen wir den Namen der vorher im Hauptfenster angelegten Anlage, also TestWKA, schliesslich wollen wir das Mapping ja für diese csv-Datei anlegen. Zuletzt muss noch der Name der Datumsspalte ausgewählt werden – dies erfolgt im untersten Auswahlmenü (hier „Datum“ auswählen – ist normalerweise schon auf Datum eingestellt).
- nach einem Klick auf den Button unten rechts „Write current mapping to DB“ wird das Mapping in der Datenbank unter dem Namen „TestWKA“ gespeichert und der Dialog wird automatisch geschlossen.
- Nun können wir auf Basis dieses Mappings die TestWKA.csv in die Datenbank einpflegen.

  Wir öffnen dazu das Menü *csv files->Add new csv file to DB* und werden daraufhin mit einem kleinen Dialog konfrontiert, auf welchem mit dem Button „Select file“ die bereits oben genante TestWKA.csv ausgewählt wird. Bei „Select mapping wählen wir das oben angelegte Mapping, welches wir „TestWKA“ genannt haben, aus.

  Nun erfolgt noch ein Druck auf den Button „Import“ und die Daten werden auf Basis der Mapping-Datei in die Datenbank eingespielt. Dies kann einige Zeit dauern, ja nach Größe des csv-Files.
- Da nun dynamische Daten (in unserem Fall z.B. Windgeschwindigkeiten) in der Datenbank vorliegen, können nun Plots der erfassten Daten angezeigt werden: Wir beginnen mit einem Histogramm. Dieses wählen wir in der Menüleiste im Main Window unter dem Menüpunkt „Visualization“ aus: *Visualization->Draw a histogram showing the prevailing windspeeds*.

  Hier werden nun Start- und Endzeitpunkte gesetzt. Durch Klick auf den Pfeil in den Auswahlboxen erscheinen Kalender, die die Eingabe erleichtern. Es sollte natürlich ein Zeitfenster ausgewählt werden, in welchem auch wirklich Daten vorhanden sind. Im unserem Fall sind durch das eingepflegte csv-File alle Daten zwischen dem 01.01.2007

und dem 31.12.2008 vorhanden. Durch Drücken des Buttons *„Draw Plot“* wird nun das Histogramm generiert und nach einiger Zeit ausgegeben.

Auf die gleiche Weise können im Visualization-Menü dann auch Polarkoordinaten-Plots erstellt werden, die aber auch je nach Datenmenge und Anzahl der Slices einige Zeit zur Darstellung benötigen.

Auf der nächsten Seiten sind 2 Darstellungen zu sehen – ein Histogramm sowie ein Polarkoordinatenplot.

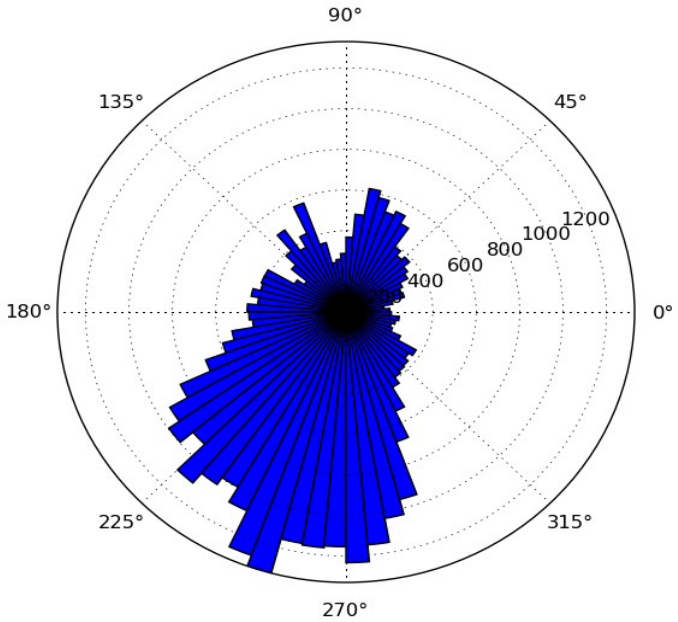

Abbildung 4: Beispiel: ein Polarkoordinatenplot

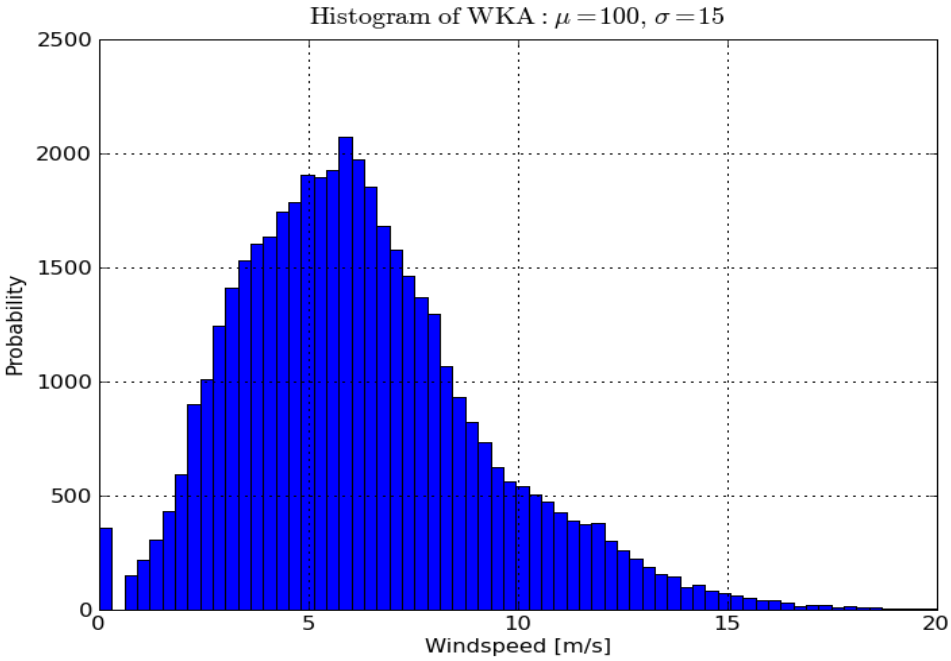

Abbildung 5: Beispiel: ein Histogramm

3. Abschließende Beurteilung der gewonnenen Erfahrung

3.1 Betrachtungen über Python

Die Erfahrung, ein Projekt als Einzelperson mit Hilfe von Projektmanagement-Tools durchzuführen hat mir gezeigt, wie wichtig es ist, auch bei Ein-Mann-Projekten stets eine konsequente Projektplanung einzuhalten. Es ist zwar - gerade zu Projektbeginn – relativ anspruchsvoll, die einzelnen Projektabschnitte zu definieren, aber durch eine gute Kommunikation mit den Projektbetreuern kann

- frühzeitig bei Änderungen im Design noch Rücksicht auf die weitere Entwicklung genommen werden
- durch einen klaren Anforderungskatalog schon von vornherein vieles entschieden werden und unnötige Programmierarbeiten für Funktionen, die im Endeffekt doch nicht benötigt werden, minimiert werden.

3.2 Persönlicher Lernerfolg durch das Projekt

Für mich persönlich war der wichtigste Lernerfolg das Kennenlernen und Erlernen einer neuen Sprache – Python – und wie einfach es in dieser Sprache ist, verschiedene Lösungsansätze vergleichend zu programmieren, um sich dann für das beste Resultat zu entscheiden.

Um es auf den Punkt zu bringen: Python ist - verglichen mit anderen Skriptsprachen wie Perl – sehr schnell und performant und darüberhinaus syntaktisch so logisch aufgebaut, dass auch nach kurzer Einarbeitungszeit schon relativ komplexe Problemstellungen gelöst werden können.

Ein weiterer Vorteil ist die Community, die beispielsweise im deutschen Python-Forum vertreten ist: www.python-forum.de . Hier habe ich so machen Tipp bekommen, wie ich mein Programm verbessern kann.

3.3 Fazit des Projekts und mögliche Weiterentwicklungen

Ein Projekt in diesem Umfang zu bearbeiten bedarf, wie schon erwähnt einiges an Aufwand und vor allem Planungsarbeit, insbesondere das Unterteilen des Projektes in Teilabschnitte. Aber als einmal die „Grundpfeiler“ abgesteckt wurden und ich mit dem Programmieren beginnen konnte, erwies sich Python als die richtige Entscheidung. Python erlaubt dem Programmierer, sehr schnell verschiedene Entwurfsmuster vergleichend auszuprobieren.

Mit PyQT und MySQLdb hat man sehr flexible Bibliotheken an der Hand, die sowohl den Datenzugriff als auch die Darstellung der GUI komfortabel umsetzen.

Als Erweiterungen kämen nun z.B. die Anpassung an andere Betriebssysteme wie Linux (Solaris) in Frage sowie die Implementierung neuer Visualisierungsarten. Hier bietet matplotlib eine Unzahl an Möglichkeiten.

Da mir die Entwicklung dieses Projektes sehr gefallen hat – auch durch die exzellente Betreuung durch Prof. Dr. Maucher und Dr. Martin Felder, die mir stets ein perfektes Verhältnis aus lenkender Führung und Freiheit in der Umsetzung boten, werde ich das Projekt auch nach dem „offiziellen“ Abschluss weiter entwickeln und zusätzliche Funktionen implementieren.

Ein anderer Punkt ist auch der Aspekt, einen (wenn auch kleinen) Beitrag zur Förderung erneuerbarer Energiequellen geleistet zu haben.