From Design to Software 2

188.935

GedichteEditor

ausgeführt an der Technischen Universität Wien,

unter der Anleitung von

Ao.Univ.Prof. Mag.rer.soc.oec. Dr.rer.soc.oec.

Horst Eidenberger

von

Daniel Brandstetter

0627810, 935

1. Projektbeschreibung

1.1 Aufgabenstellung

Thema dieses Projektpraktikums ist die Erstellung eines Editors für Gedichte. Der Benutzer der Software soll also bei der Erstellung seiner Gedichte, bzw. bei der Analyse von bestehenden Gedichten unterstützt werden.

Zuerst soll ein geeigneter Texteditor erstellt werden, der alle benötigten Standardfunktionen bereitstellt, wie z.B. öffnen und speichern von Gedichten. Weiteres gilt es, ein Wörterbuch zu implementieren, welches später für die Analyse der Gedichte, bzw. der Reimbildung benötigt wird. Das Wörterbuch soll zusätzlich noch eine Silbentrennung enthalten.
Eine der Hauptaufgaben ist die Erkennung von Silben und Versfüßen, und das daraus entstehende Versmaß, bzw. die Metrik bestimmter Textzeilen. Dabei soll die Analyse entsprechend visualisiert und dargestellt werden.
Weiters soll der Editor Vorlagen für Reimpaare anbieten, und in Kombination damit auch Reimwörter bereitstellen. Der Benutzer soll möglichst interaktiv Reime für ein entsprechendes Wort finden können.

Das vorliegende Dokument beschreibt die Erstellung des gesamten Programmes, vom grafischen User Interface bis hin zur Implementierung der wichtigsten Funktionen. Auch etwaige Probleme, welche während des Projektes aufgetreten sind, werden hier beschrieben.

2. Implementierung

2.1 Systemumgebung

Das Programm wurde in Java geschrieben, also Programmierumgebung diente Eclipse 4.2.2.

2.2 Grafisches User Interface

Das grafische User Interface stellt die Schnittstelle zwischen dem Benutzer und dem Programm selbst dar. Beim Design wurde darauf geachtet, dass die Daten gut strukturiert und einfach zu erkennen sind.

Grundsätzlich besteht das User Interface aus einem Hauptfenster, welches vor allem aus einem Bereich zum Schreiben besteht.

Ein Screenshot des Programmes ist in der nachfolgenden Abbildung zu sehen.

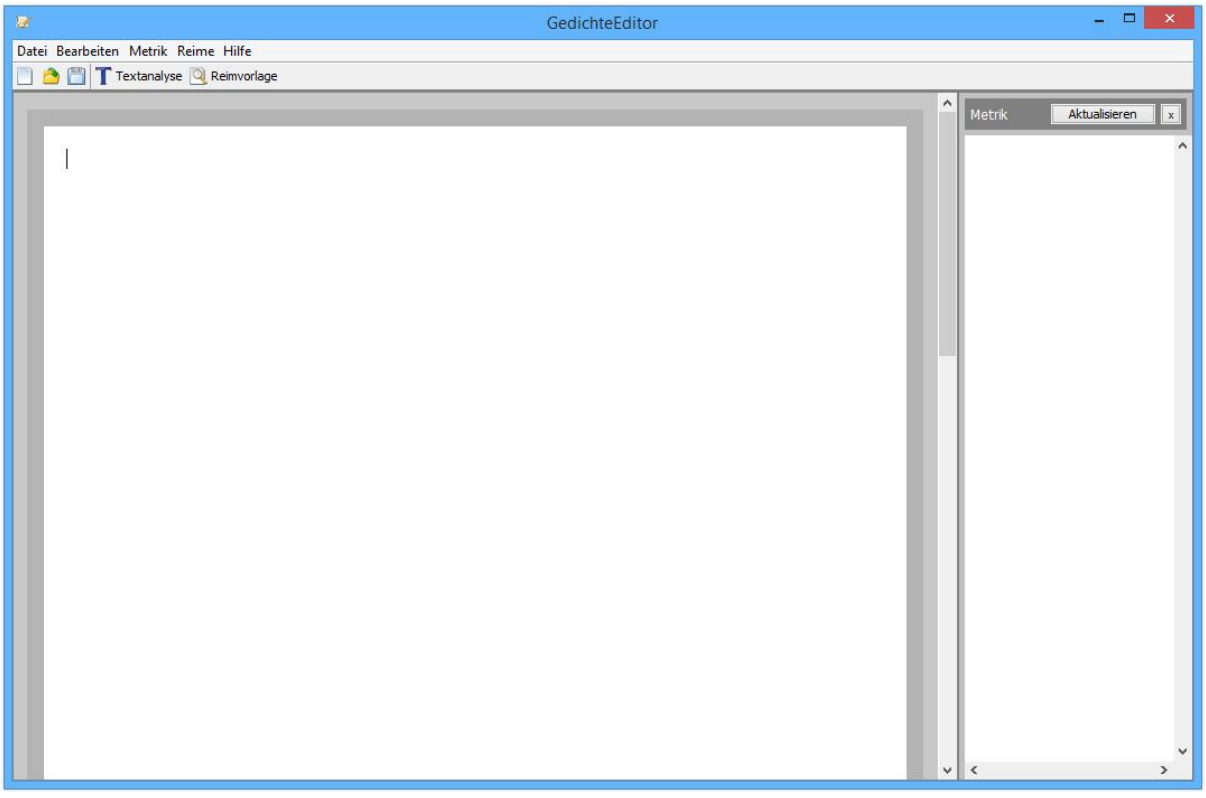


Abbildung 1: Grafisches User Interface des GedichteEditors

Bei der Erstellung des Interfaces wurde auch darauf geachtet, dass sämtliche Aktionen mittels Tastaturbefehle erreichbar sind. Das Programm kann somit auch mittels Tastatur gesteuert werden, vor allem mit Hilfe der Alt Taste.

Für die Darstellung der Reimwörter wurde ein externes UI Element verwendet, dabei handelt es sich um die Klasse „MenuScroller“. Da die Liste an Reimwörtern durchaus sehr lang werden kann, wurde für eine übersichtlichere Darstellung auf diese Klasse zurückgegriffen. Diese bietet Scrolling

Funktionen für lange Menü Dropdowns bzw. Popup Menüs. Die nachfolgende Abbildung zeigt den „MenuScroller“.

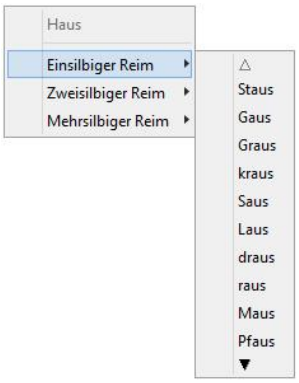


Abbildung 1: MenuScroller

Die verwendete Klasse darf frei verwendet werden und kann ohne Einschränkungen beliebig verändert werden. (http://tips4java.wordpress.com/2009/02/01/menu-scroller/)

Sprachunterstützung

Das Programm wurde darauf vorbereitet, mehrere Sprachen zu unterstützen. Standardmäßig sind die Sprachen Deutsch und Englisch vorhanden.

Die Zeichenfolgen selbst sind in einer externen properties Datei gespeichert (language.properties), wodurch die Sprache geändert werden kann.

Zu beachten ist hier, dass die Sprachänderung nur jeweils durch einen Programm Neustart möglich ist.

Technisch realisiert wird die Änderung der Sprach grundsätzlich mittels eines „ResourceBundle“ Objekt, hinter dem die entsprechende Lokalisierungsdatei steht. Die passende Datei wird dabei von Java selbst herausgesucht.

Hilfesystem

Der GedichteEditor verfügt über ein Hilfesystem. Erstellt wurden die Hilfedateien mittels HTML Editor, welche anschließend mit Hilfe des „HTML Help Workshop“ Tools kompiliert wurden. Die daraus resultierende .chm Datei kann dann problemlos in das Programm eingefügt werden. Die Hilfedatei selbst ist nur in deutscher Sprache vorhanden.

2.3 Technische Implementierung

Als ersteres wurde das grafische Interface umgesetzt, danach wurden die grundlegenden Editorfunktionen implementiert.

Da sowohl für die Reimbildung, als auch für die Textanalyse ein Wörterbuch benötigt wird, war die Implementierung eines solchen der nächste Schritt. Beim Wörterbuch handelt es sich um eine deutsche Wortliste (http://repo.or.cz/w/wortliste.git), welche Informationen zur Silbentrennung beinhaltet. In einer eigenen Klasse wird die Wortliste beim Programmstart einmal ganz durchlaufen, und in einer „Hashmap“ gespeichert. Die Struktur dieser „Hashmap“ besteht aus einem „key“ und einem „value“ Wert, wobei „key“ das Wort, und „value“ die zugehörige Silbentrennung ist. Dadurch kann für jedes Wort die passende Silbentrennung abgerufen werden.

Reime

Um passende Reimwörter für ein gegebenes Wort zu finden, muss einmal die gesamte Wortliste durchlaufen werden. Bei der Reimbildung wurde zwischen einsilbigen, zweisilbigen und mehrsilbigen Wörtern unterschieden. Um ein passendes Reimwort zu finden, wurde das gegebene Wort anhand dessen Silben untersucht. Vor allem wenn die letzte Silbe eines Wortes gleich mit einem anderen Wort ist, reimt es sich in den meisten Fällen. Auch die Position der Selbstlaute (a, e, i, o, u, ä, ü, ö) spielen eine wichtige Rolle.

Da der Durchlauf der gesamten Wortliste doch erheblich Zeit in Anspruch nimmt, wird jeweils ein neuer Thread gestartet, um weitere Eingaben des Benutzers nicht zu blockieren. Somit läuft das Programm ohne Verzögerungen weiter.

Textanalyse

Für die Analyse des eingegebenen Textes wird dieser grundsätzlich in dessen Zeilen aufgespalten. Danach wird jede Zeile einzeln untersucht.

Um die Anzahl der Silben zu bestimmen wird die entsprechende Textzeile Wort für Wort durchlaufen, wobei für jedes Wort deren Silben bestimmt werden. Da sich die Wortliste in einer „Hashmap“ befindet, werden das Wort und dessen zugehörige Silbentrennung äußerst schnell gefunden. Somit kann die Anzahl der Silben problemlos visualisiert werden.

Um die Metrik einer Textzeile zu bestimmen, muss der Benutzer dessen Betonung angeben. Diese wird einfach in einem Array gespeichert. Anhand der Anzahl der Silben und deren Betonung kann anschließend die Metrik durch einfaches Vergleichen bestimmt werden.

Der gesamte Vorgang der Reimbildung bzw. Textanalyse beschäftigt sich somit ausschließlich aus dem Zerlegen des Textes in Zeilen, und diese wiederum in Wörter und Silben, und einer anschließenden Auswertung und Visualisierung der zerlegten Elemente. Technisch ist dazu nicht allzu viel zu sagen.

2.4 Known Bugs und Probleme

Das Versmaß einer Textzeile wird anhand der Anzahl der Silben und deren Betonung bestimmt. Dabei kann eine Silbe betont oder unbetont sein. Es gibt leider keine genauen Regeln, wann eine Silbe betont, bzw. unbetont ist. Da die Betonung auch abhängig vom jeweiligen Satz ist, kann auch nicht die Betonung einzelner Wörter festgelegt werden. Anfangs sollte die Bestimmung des Versmaßes automatisch geschehen, mittlerweile muss der Benutzer die Betonung aber selbst angeben.

Die Sprache des Programmes kann nur dann geändert werden, wenn das Programm auch neu gestartet wird. Dies liegt daran, dass zu Beginn alle Komponenten erzeugt werden, und Texte in der angegebenen Sprache anzeigt werden. Will man die Sprache zur Laufzeit ändern, wäre ein Algorithmus nötig, der sämtliche Komponenten neu durchläuft und die Texte entsprechend ändert.

2.5 Weiterentwicklungen

Grundsätzlich könnte man versuchen, dass das Versmaß selbstständig erkannt wird. In wie weit dies möglich ist, ist schwer zu sagen, dafür wären fundierte Kenntnisse der Metrik nötig. Auch die Anzahl an erkennbaren Versmaßen könnten erhöht werden.

Weiteres könnte man noch Versuchen, bessere und genauere Reime für beliebige Wörter zu finden. Auch die Vorlagen an Reimtypen könnte man noch beliebig erweitern.

2. Zusammenfassung

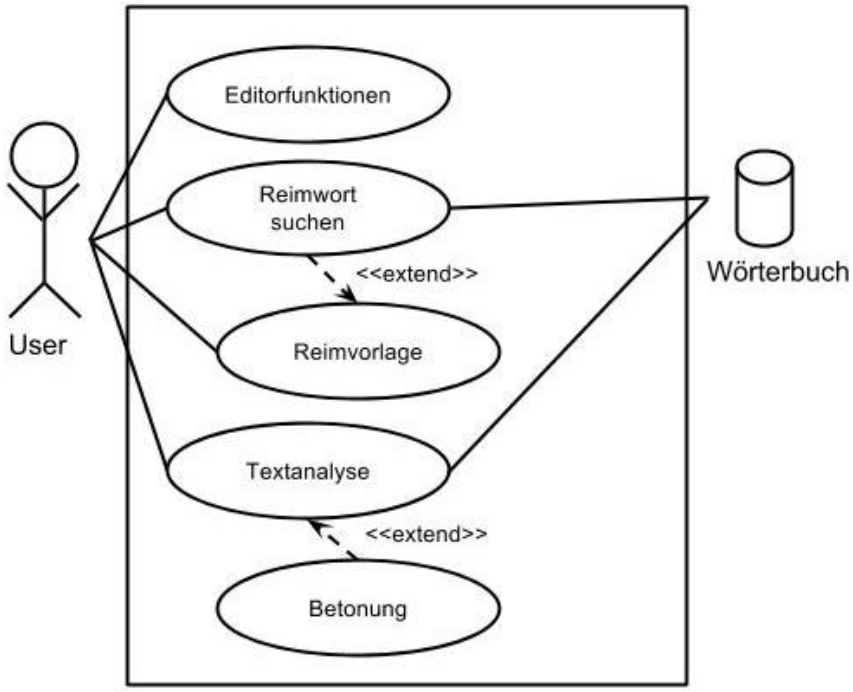
Zu Anfang des Projektes war es vor allem mühsam, sich mit der Erstellung und dem Aufbau von Gedichten auseinander zu setzen. Nach kurzer Zeit des Einlesens wurde die Materie aber durchaus interessanter als gedacht.

Eine Hürde war sicherlich das Finden einer geeigneten Wortliste, da diese auch über eine entsprechende Silbentrennung verfügen sollte. Nach längerer Suche wurde schließlich eine geeignete Liste gefunden.

Abschließend muss ich sagen, dass es ein durchaus interessantes und lehrreiches Projekt war.

3. Anhang

3.1 Anwendungsfall Diagramm



3.2 Klassendiagramm