Fachhochschule der Wirtschaft
-FHDW-
Bielefeld

Ausarbeitung
zur
Vorlesung

Praxis der Softwareentwicklung 1

Thema:

Grafische Oberflächen mit Java Foundation Classes / Swing

Verfasser:

Andreas Gerlach
Andreas Pörtner

5. Semester
Studiengang Bachelor of Science
Schwerpunkt: Wirtschaftsinformatik

Eingereicht am:

22. Dezember 2010

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Pflichtenheft

1.1 Zielbestimmung

Folgende Kriterien muss die Software, laut Aufgabenstellungen erfüllen:

- das Einlesen von Texten (unformatierter Rohtext)
- das Anzeigen des eingelesen Textes (ohne Formatierung)
- das Führen von Statistiken über
    - die Anzahl aller Wörter
    - die Anzahl aller Zeichen (inklusive Satzzeichen und Leerzeichen)
    - die Anzahl aller alphanumerischen Zeichen
    - die Anzahl aller Konsonanten
    - die Anzahl aller Vokale
    - die Häufigkeit der am meisten vorkommenden Wörter
- eine gewichtet Darstellung der vorkommenden Wörter in Form einer „Tag-Cloud“

Abbildung 1: Tag-Cloud

Quelle: http://www.muenstergass.ch/blog/?p=1679

Eine Tag-Cloud, zu deutsch Schlagwortwolke, bietet eine Visualisierung von einer Liste oder einem Ranking. In einer Tag-Cloud werden auf einen Blick die wichtigsten oder am höchsten platzierten Begriffe deutlich. In Abbildung 1 sind es die orange gefärbten Begriffe, welche im Ranking am höchsten stehen. In diesem Fall wird eine Tag-Cloud eingesetzt um die Häufigkeit der Wörter des eingelesenen Textes darzustellen.

Die nachfolgenden Kriterien können laut Aufgabenstellung erfüllt werden:

- das Einlesen von formatierten Texten
- bspw. RTF, DOC, DOCX, PDF
- die Ausgabe der Anzahl aller Substantive (nur bei deutschen Texten)
- eine farbliche Hervorhebung der Wörter im Text, bei Selektion des Wortes in der Tag-Cloud

1.2 Produkteinsatz

Das Produkt dient in erster Linie nur dem Projekt und soll nicht darüber hinaus verwendet werden. Der Dozent soll an Hand des Produktes die geforderten Leistungen überprüfen können. In ihrer derzeitigen Form dient die Anwendung als Machbarkeitsstudie. Für einen produktiven Einsatz im universitären oder redaktionellen Bereich müssten jedoch weitere Funktionen hinzugefügt werden.

1.3 Produktübersicht

Das Produkt mit den Namen „ASTA – A Students Text Analyzer“ ist ein kleines Hilfsprogramm, in dem der Nutzer eine Textdatei einlesen und die Analyse des Textes starten kann. Während des Prozesses analysiert die Anwendung die eingelesene Textdatei nach verschiedenen Kriterien und baut Statistiken über die Zeichen- und Wortverwendung auf. Diese werden nach Abschluss der Analyse übersichtlich und graphisch aufbereitet präsentiert. Über die sogenannte Tag-Cloud werden die Worthäufigkeiten sofort visuell sichtbar gemacht. Beim Selektieren eines Wortes in der Tag-Cloud werden die Vorkommnisse im Textbereich hervorgehoben.

1.4 Produktfunktionen

Am oberen Fensterrand befinden sich die Bedienungsicons in der so genannten Toolbar.

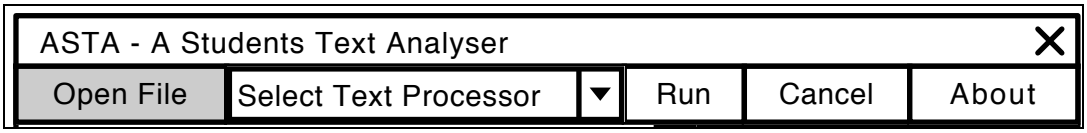

Abbildung 2: Toolbar

Quelle: Eigene Darstellung

Ganz links befindet sich das Icon zum Öffnen des Dateiauswahlfensters, um die zu analysierende Datei anzugeben. Es wird der systemspezifische Datei-Auswahl-Dialog benutzt.
Um mehrere verschiedene Weisen der Textuntersuchung und –analyse zu unterstützen, kann in der nachfolgenden Auswahlschaltfläche die zu benutzende Text Analyze Strategie ausgewählt werden. Hierbei wird aktuell nur die „Standard Text Analyse“ angeboten. Durch die „Run“ Schaltfläche wird der Analysevorgang gestartet. Mit dem dadurch aktivierten „Cancel“ Befehl kann die Analyse jederzeit abgebrochen werden. Durch einen Klick auf die „About“ Schaltfläche wird der Informationsdialog geöffnet. In diesem wird der Name des Programms, das Erstellungsjahr und die Autoren angezeigt.
Mit dem ASTA können Textdateien mit der Dateiendung *.txt eingelesen werden. Nach dem Einlesen wird eine Analyse des Textes durchgeführt. Bei dieser Analyse werden folgende statistischen Daten erfasst:

- Anzahl Wörter,
- Anzahl Zeichen,
- Anzahl Satzzeichen,
- Anzahl alphanumerischer Zeichen,
- Anzahl Vokale und
- Anzahl Konsonanten.

Des Weiteren wird eine grafische Auswertung in Form eines Tortendiagramms ausgegeben. Ebenso wird nach der Analyse eine „TOP 10“-Liste der 10 häufigsten Wörter angezeigt:

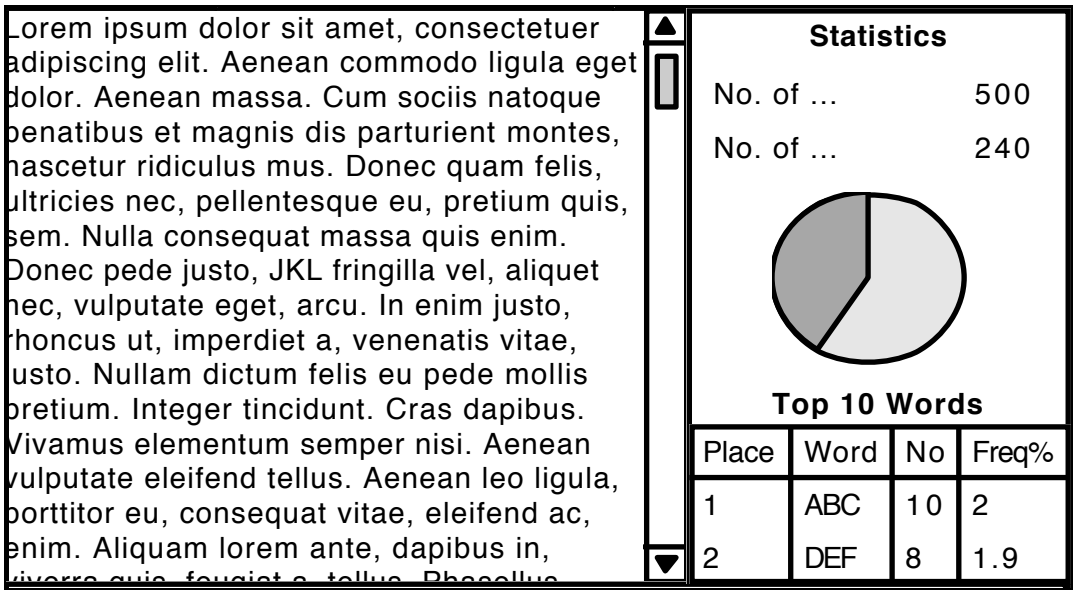

Abbildung 3: Screenshot nach einer Analyse

Quelle: Eigene Darstellung

Am unteren Fensterrand wird während der Analyse ein Fortschrittsbalken angezeigt, an dem der Verlauf der Analyse abgelesen werden kann. Sobald die Analyse abgeschlossen ist, wird die Statusanzeige ausgeblendet und an derselben Stelle erscheint die „Tag-Cloud“. In der „Tag-Cloud“ werden ebenfalls die 10 häufigsten Wörter je nach Gewichtung unterschiedlich stark hervorgehoben angezeigt. Zudem kann man beliebige Begriffe aus der „Tag-Cloud“ entfernen bzw. filtern. Dabei wird das Wort nicht nur aus der „Tag-Cloud“ sondern auch aus der „Top 10“ entfernt. Diese Funktion wurde implementiert, um Füllwörter wie „und“, „oder“, „der“, „die“ und „das“ selektiv aus der Liste der häufigsten Wörter ausschließen zu können. Durch einen Mausklick auf einen Begriff, kann dieser entweder ignoriert oder im eingelesen Text farblich hervorgehoben werden.

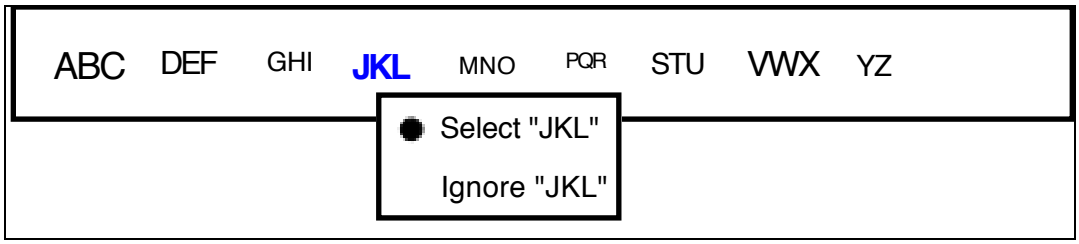

Abbildung 4: "Tag-Cloud" und Filterfunktion

Quelle: Eigene Darstellung

1.5 Produktdaten

Es werden keine Daten persistent gespeichert. Die Daten des Programms existieren lediglich flüchtig im Hauptspeicher und dienen der Anzeige der statistischen Werte wie:

- die Anzahl aller Wörter
- die Anzahl aller Zeichen (inklusive Satzzeichen und Leerzeichen)
- die Anzahl aller alphanumerischen Zeichen
- die Anzahl aller Konsonanten
- die Anzahl aller Vokale
- die Häufigkeit der am meisten vorkommenden Wörter

Nach Auswahl einer neuen Textdatei und Start der Analyse sind die Daten des vorherigen Laufes ungültig und nicht mehr zugreifbar.

1.6 Benutzungsoberfläche

Die Benutzeroberfläche des Programms wird mit Java Swing umgesetzt. Sie benutzt das native Look-n-Feel der Zielplattform. Es existiert ein Hauptfenster, welches über die Toolbar im Kopfbereich den Zugriff auf die wesentlichen Steuerungsfunktionen der Anwendung:

- Quelldatei öffnen,
- Textprozessor auswählen,
- Textanalyse starten / stoppen oder
- den Informationsdialog mit Angaben zu den Verfassern aufrufen

anbietet.

Im Hauptteil des Anwendungsfensters erscheint der eingelesene Text. Ist der Text länger als der sichtbare Bereich im Fenster erscheint ein vertikaler Scrollbalken. Das Textfenster füllt sich nach der Textanalyse. Diese findet im Hintergrund statt und ein Fortschrittsbalken im unteren Bereich des Fensters informiert über den Bearbeitungszustand. Die Schaltfläche zum Starten der Analyse ist für die Dauer einer Analyse deaktiviert. Stattdessen wird der Abbruch Knopf aktiviert, um hierüber den Abbruch der Analyse jederzeit zu gewährleisten.

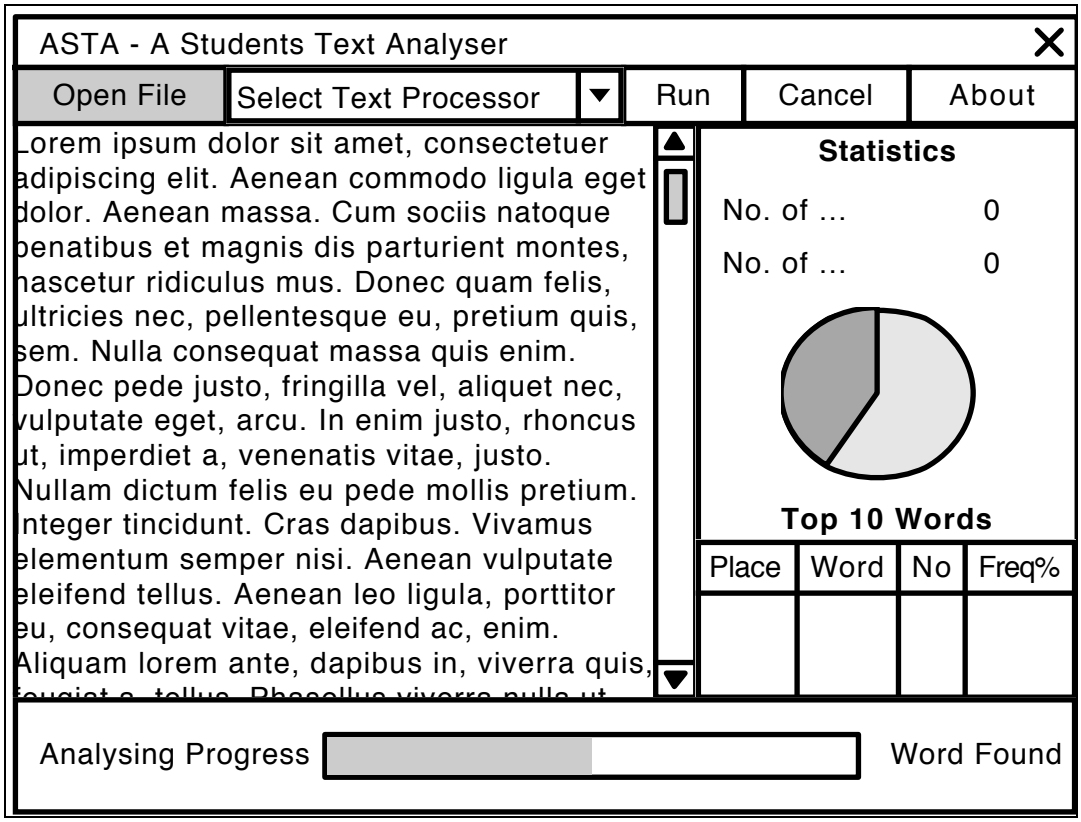

Abbildung 5: Anwendungsfenster während der Analyse

Quelle: Eigene Darstellung

Ist die Analyse des Textes erfolgreich abgeschlossen worden, erscheint im rechten Drittel des Anwendungsfensters die Ausgabe der statistischen Werte. Hierbei werden die Kennzahlen aus der Anforderungsanalyse zunächst tabellarisch aufgeführt. Anschließend erfolgt eine graphische Darstellung der prozentualen Anteile der Konsonanten, Vokale und Steuerzeichen im Text. Der Statistikbereich wird mit der Auflistung der Top-10-Wörter inklusive deren prozentualen Anteil am Text abgeschlossen. Im unteren Bereich des Anwendungsfensters wird die Fortschrittsanzeige durch eine Tag-Cloud ersetzt, die je nach Gewichtung (relative Häufigkeit) der Wörter die Schriftgrößen der Texte setzt. Die Texte ermöglichen das Auswählen eines Wortes, wodurch dessen Vorkommnisse im Text optisch hervorgehoben werden.

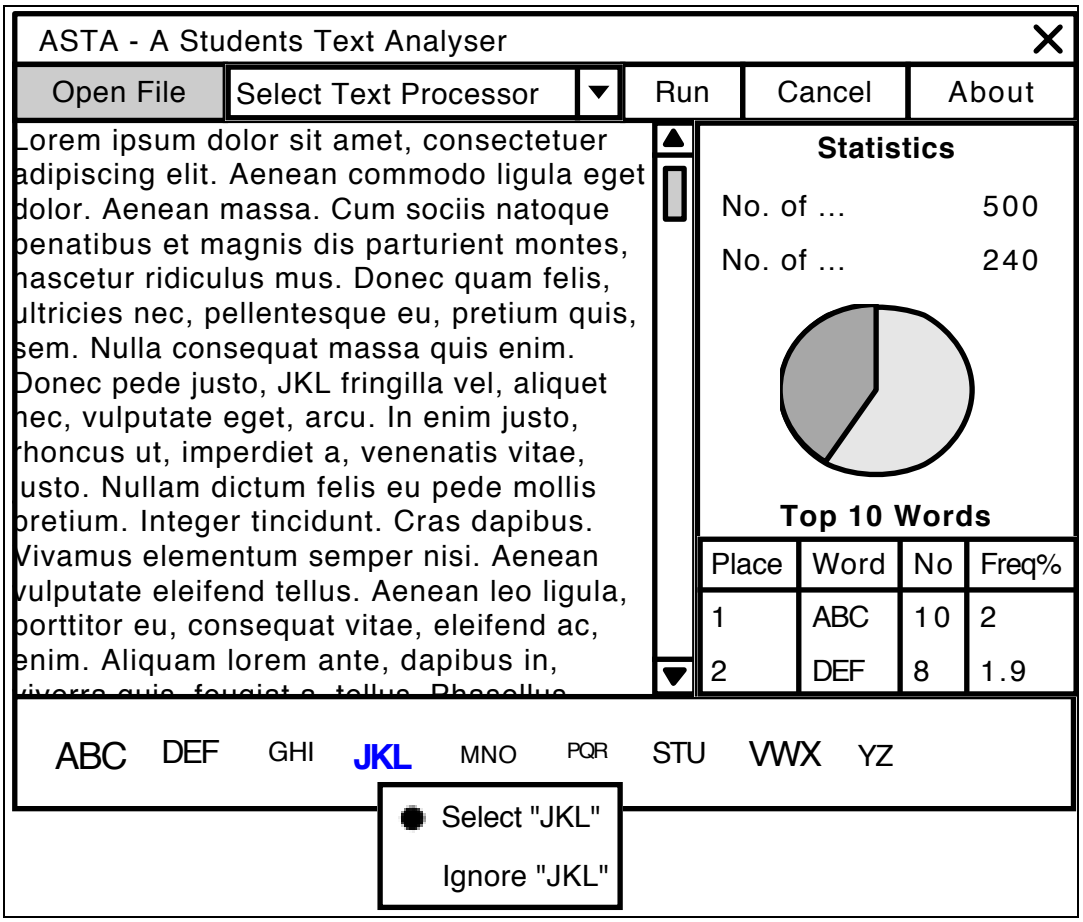

Abbildung 6: Anwendungsfenster nach Abschluss der Analyse

Quelle: Eigene Darstellung

1.7 Nicht-funktionale Anforderungen

Die Anwendung ist in Java geschrieben und somit plattform-übergreifend einsetzbar. Ein Funktionstest der Anwendung wird unter den Betriebssystemen Windows, Linux und Mac OS X durchgeführt. Die Anwendung integriert sich weitestgehend in die existierenden Oberflächenstrukturen der Betriebssysteme ein. Drag-&-Drop wird jedoch nicht unterstützt. Es werden derzeit lediglich Textdateien unterstützt. Die Anwendung ist jedoch daraufhin konzipiert worden, auch andere Dateitypen und Analyse-Strategien zu unterstützen. Die einzulesenden Textdateien müssen im UTF-8 Format vorliegen.

1.8 Technische Produktumgebung

Die Anwendung wurde auf Java 1.6 und der Produktversion 1.6.0_22 erstellt. Es wird geraten keine älteren Versionen zu nutzen, da diese nicht getestet wurden und daher auch keine Freigaben bekommen haben.
Für die Anwendung wurden folgende Bibliotheken integriert:

- jCharts-0.7.5.jar — wird für die Darstellung des Tortendiagramms benötigt
- swingx-core-1.6.2.jar — wird für spezielle Steuerelemente auf der grafischen Benutzeroberfläche benötigt

1.9 Spezielle Anforderungen an die Entwicklungsumgebung

Als Entwicklungsumgebung wurde Eclipse SDK Version 3.6.1 verwendet. Eclipse war bereits aus früheren Vorlesungen und ersten Schritten in Java bekannt, daher bot es sich an dieses weiter zu nutzen. Zudem ist der Funktionsumfang sehr groß und bringt alle Merkmale mit, die für dieses Projekt benötigt wurden. Aufgrund der Erweiterbarkeit der Entwicklungsumgebung und den zahlreichen, teilweise kostenlos verfügbaren Komponenten lässt sich Eclipse auf die verschiedensten Anforderungen hin anpassen. Im ASTA Projekt wurde für das UI-Prototyping auf die WireframeSketcher Erweiterung (http://wireframesketcher.com) zurückgegriffen.

2 Fachkonzept

Die Anwendung wurde mit Hilfe des Model-View-Controller Konzeptes modelliert, um eine strikte Trennung der Oberflächen- und Geschäftslogik-Schicht zu ermöglichen.
Im Model-View-Controller Konzept wird die Geschäftslogik im Model-Bereich konzentriert. Im View-Bereich sind hingegen die Oberflächenelemente und graphischen Sichten zu finden. Um die lose Verbindung der beiden Elemente zu ermöglichen, agiert der Controller als Steuereinheit, welche Interaktionen des Benutzers an der Oberfläche empfängt und auf Aktionen im Model weiterleitet. Sind durch die Aktionen Änderungen im Model erfolgt, informiert der Controller die Anwendungsoberfläche mit den neuen Daten, damit diese sich dem Anwender ggü. entsprechend aktualisiert.
Diese eindeutige Trennung der Bereiche im Model-View-Controller Konzept findet sich in den ASTA Quellen über die Paketnamen der Java Klassen wieder:

- asta.controller — beinhaltet den Controller Code der Anwendung
- asta.model — beinhaltet die verschiedenen Model Klassen der Anwendung
- asta.view — beinhaltet die diversen View Klassen der Anwendung

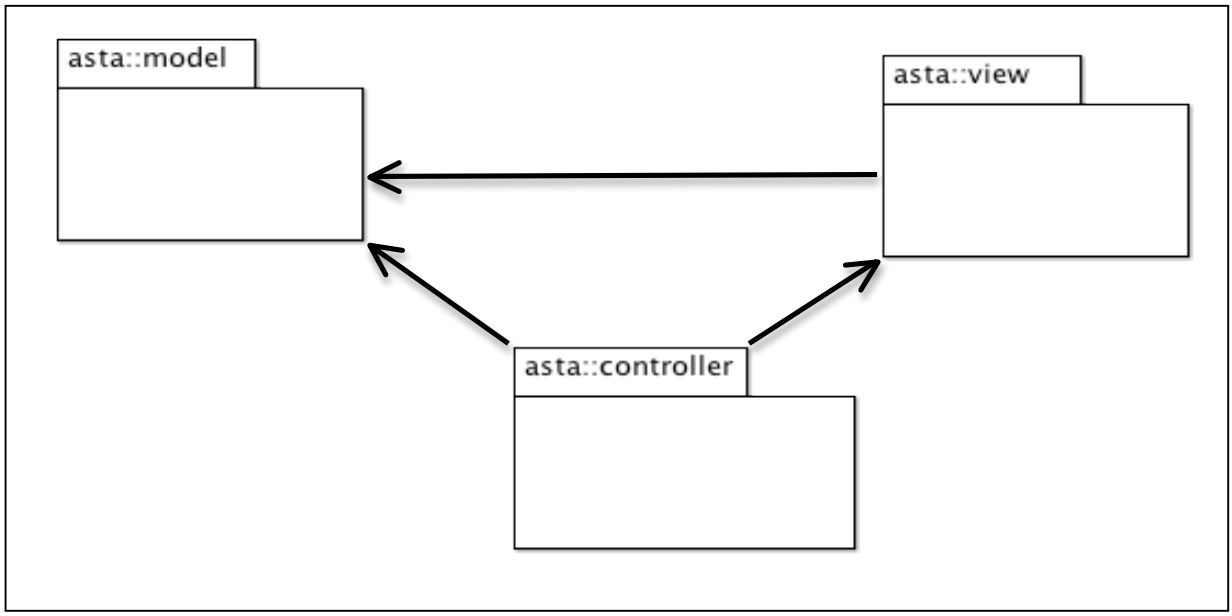

Abbildung 7: Überblick Model-View-Controller Konzept

Quelle: Eigene Darstellung

2.1 Klassen im Model Paket

Im Model Paket finden sich die Klassen, welche die Geschäftslogik der Anwendung abbilden. Die Hauptentität im Paket „asta.Model“ ist die Klasse TextMetaData, welche die Datenstrukturen mit den Ergebnissen der Text-Analyse beinhaltet. Diese Klasse wird über den Anwendungscontroller weiter an die Oberflächenelemente des View Paketes gereicht, damit die Steuerelemente, welche für die Anzeige der Werte zuständig sind, einen Zugriff auf die Analyseergebnisse erhalten. Hierin finden sich nicht nur die Zähler, die die Anzahl der Zeichen, Anzahl der alphanumerischen Zeichen, Anzahl der Satzzeichen usw. widerspiegeln, sondern auch die für jedes eindeutig gefundene Wort gespeicherte Häufigkeit sowie die jeweiligen Positionen der Wortvorkommnisse im Text. Die notwendigen Informationen werden über die beiden öffentlichen Methoden addWord() und addPuntuation() der TextMetaData Klasse von außen mitgeteilt (siehe auch JavaDoc).
Damit die Anwendung mit verschiedenen Dateiformaten und Datei-Analysemethoden umgehen kann, wurden im Paket „asta.Model“ öffentliche Schnittstellen für die Abstraktion des Dateizugriffs und der Textanalyse entworfen. Die Aufgabe einer Klasse, die die Schnittstelle ITextProcessingStrategy unterstützt, liegt hierbei im Parsen des Dateiinhaltes. Dabei wird beim Starten der Analyse die vom Endanwender in der Oberfläche ausgewählte Text Processing Strategie mit der IFileHandler Implementierung aufgerufen, die zur jeweils aktuell ausgewählten Textdatei passt. Über den IFileHandler kann die Text Processing Strategie an den textuellen Inhalt der Datei gelangen und ihn mittels der eigenen Logik in Wörter, Satzzeichen, Markup-Befehle u.ä. aufteilen.
Beim erfolgreichen Erkennen eines Wortes, Satzzeichens usw. ruft die Text Processing Strategie die TextMetaData Klasse auf, um die gefundene Information in den Text Meta Daten zu hinterlegen.

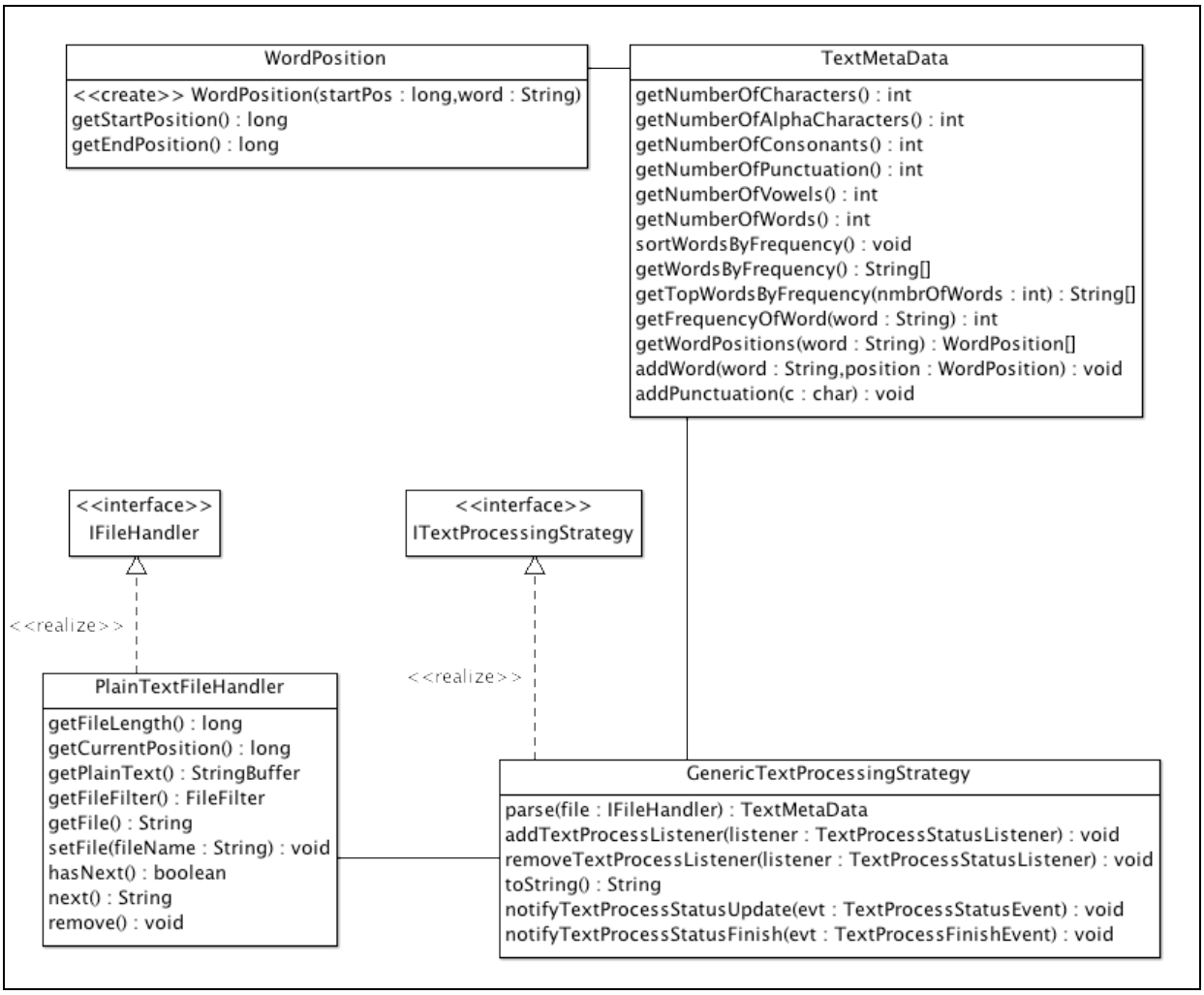

Abbildung 8: wichtigste Klassen im Paket „Model“

Quelle: Eigene Darstellung

Mit dieser Version der Anwendung wird ein Datei-Handler für normale, unformatierte Textdateien und eine Analyse-Strategie, die den Text nach gängigen Satzzeichen – bspw. Komma, Punkt, Ausrufe- und Fragezeichen untersucht und anhand dessen den Text in Wörtern unterteilt, freigegeben. Als ein Beispiel für eine Sonderfallbehandlung in der Text Analyse Strategie wurde die aktuelle Text Analyse Logik um den Sonderfall Versionsnummern bzw. allgemeiner Text, indem bestimmte Satzzeichen wie bspw. der Punkt im Wort vorkommen erweitert. Diese Vorkommnisse werden als ein Wort von der Text Analyse Logik erkannt. Somit werden die Zeichenketten „V1.0.1“, „u.a.“ und „z.Bsp.“ korrekt als jeweils ein Wort erfasst.

Eine sinnvolle Erweiterung des ASTA Programmes wäre bspw. die Erstellung einer XHTML Text Analyse Strategie, welche die im HTML vorkommenden Markup-Befehle erkennt, herausfiltert und somit die Text Meta Informationen auf den textuellen Nutzinhalt ohne Berücksichtigung der Markups aufbaut.

2.2 Klassen im View Paket

Die Hauptentität im View Paket ist die MainWindow Klasse, welche den Rahmen für das Anwendungsfenster aufbaut. Dieses Hauptfenster wird anhand des Oberflächen-Prototypings in folgende Regionen unterteilt:

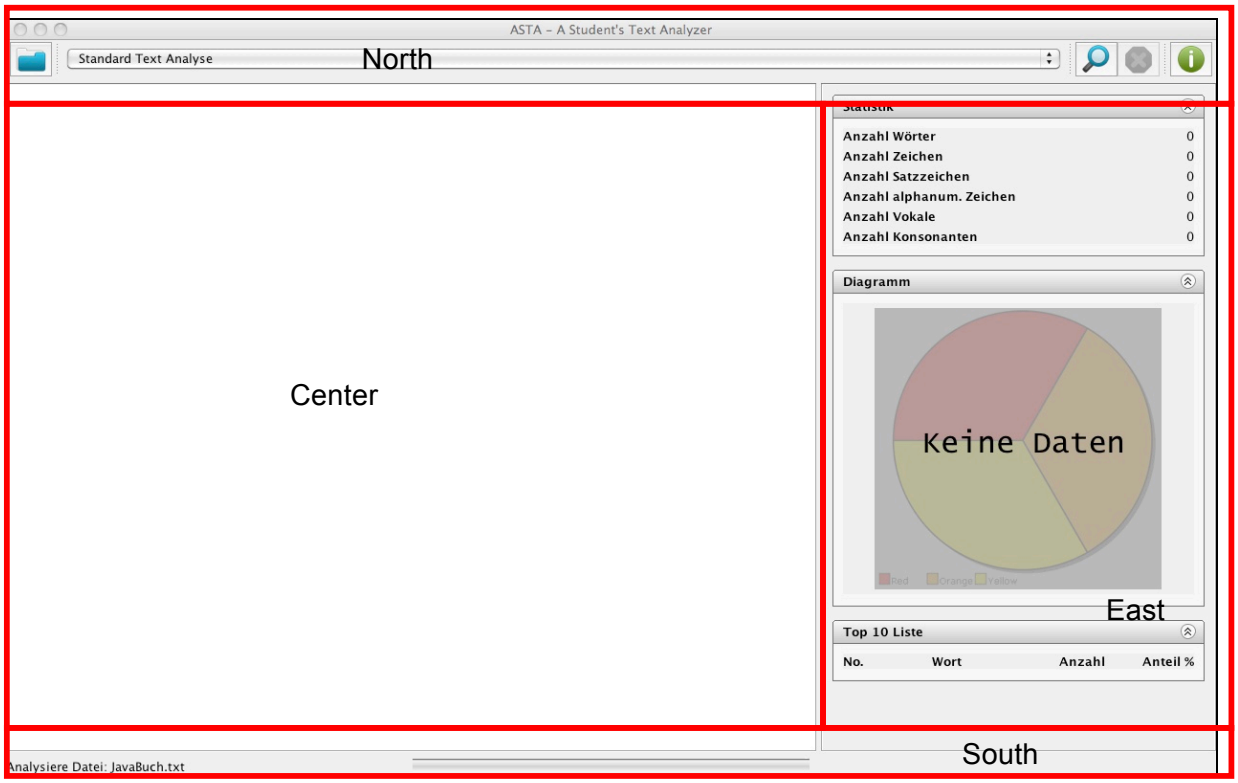

Abbildung 9: Bildschirmfoto des Hauptfensters

Quelle: Eigene Darstellung

Hierbei wird ein BorderLayout Manager genutzt, um die Steuerelemente zur Laufzeit in die einzelnen Bereiche automatisch zu positionieren. Jeder Bereich wiederum wird durch eine eigene von JPanel abgeleitete Klasse repräsentiert.
Im North-Bereich des BorderLayouts findet sich die ToolbarView, die die verschiedenen Aktionen der Anwendung über Schaltflächen und einer Auswahlbox für die zu nutzende Text Analyse Strategie zur Verfügung stellt.
Im Center-Bereich befindet sich die TextView Komponente. Diese nimmt nach Abschluss der Text Analyse den Rohtext der Datei auf und stellt diesen in einer Editor Komponente dar. Ist der Text länger als der sichtbare Bereich, so erscheinen automatisch Scroll Leisten, die ein Navigieren durch den Text ermöglichen.

Im East-Bereich sind drei ausklappbare Anzeigen angeordnet. Die erste Anzeige ist standardmäßig ausgeklappt und beinhaltet die Statistiken über die Anzahl Wörter, Anzahl der Zeichen usw.

Abbildung 10: East-Border [Statistik]

Quelle: Eigene Darstellung

Die zweite Anzeige beinhaltet ein Tortendiagramm, in welchen die Verhältnisse zwischen Vokalen, Konsonanten, Satzzeichen und sonstige Zeichen sichtbar gemacht wird. Diese Anzeige ist standardmäßig eingeklappt und wird mit einen grau hinterlegten Standarddiagramm initialisiert.

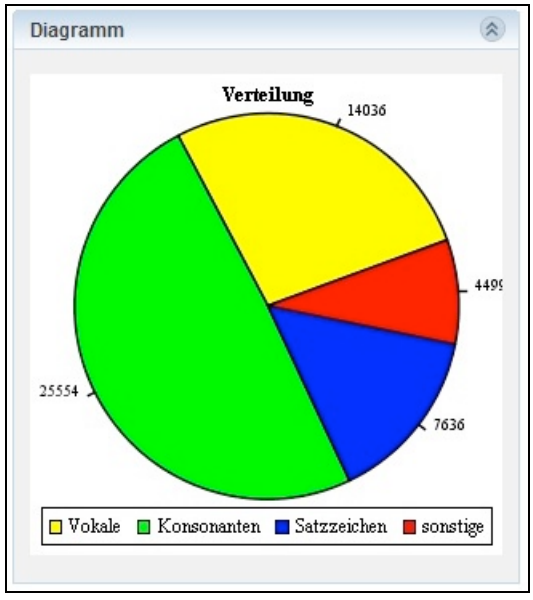

Abbildung 11: East-Border [Diagramm]

Quelle: Eigene Darstellung

Die Dritte und letzte Anzeige beinhaltet die TOP 10 Liste. In dieser Liste werden die 10 am häufigsten vorkommenden Wörter aufgelistet. Zusätzlich zu ihrer Anzahl wird ihr prozentualer Anteil an der Gesamtanzahl aller Wörter ausgegeben. Auch diese Anzeige ist standardmäßig eingeklappt und wird auch nicht mit Standardwerten initialisiert.


Top 10 Liste

No.	Wort	Anzahl	Anteil %
1.	und	285	4,35
2.	mit	129	1,97
3.	Die	120	1,83
4.	Java	75	1,14
5.	der	75	1,14
6.	von	70	1,07
7.	die	60	0,92
8.	Zum	59	0,90
9.	in	57	0,87
10.	Klasse	53	0,81

Abbildung 12: East-Border [TOP 10 Liste]

Quelle: Eigene Darstellung

Der South-Bereich wechselt zwischen 2 Sichten je nach Programmstatus. Vor und während des Einlesens und Analysierens eines Textes wird dort ein Statusbalken angezeigt, siehe Abbildung 8. Nach einer erfolgreichen Analyse wird der Statusbalken ausgeblendet und an seiner Stelle wird eine Tag-Cloud angezeigt. Wie bereits in Kapitel 1.1 erwähnt, werden in einer Tag-Cloud die 10 häufigsten Begriffe mit einer grafischen Gewichtung dargestellt.

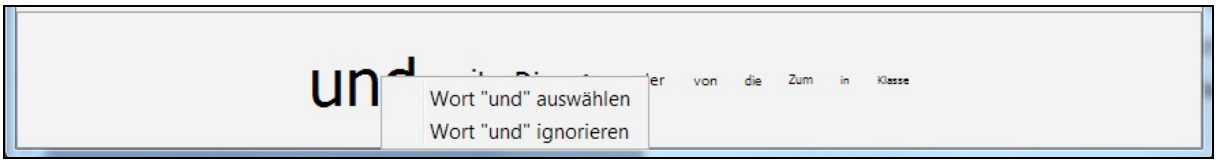

Abbildung 13: South-Boarder [Tag-Cloud]

Quelle: Eigene Darstellung

Des Weiteren kann man durch einen Mausklick auf einen Begriff in der Tag-Cloud, diesen aus der Tag-Cloud und TOP 10 Liste entfernen oder ihn im Text-Fenster farblich hervorheben lassen.

2.3 AppController Klasse

Die AppController Klasse dient als Bindeglied zwischen den Klassen der Pakete „asta.model“ (hier vor allem der TextMetaData Klasse) und „asta.view“ (insbesondere der MainWindow Klasse).

Die AppController Klasse ist als Singleton implementiert. Dies bedeutet, dass es während der Laufzeit der Anwendung zu jedem Zeitpunkt immer nur eine Instanz der Klasse gibt. Diese wird über die statische Methode getInstance() nach außen hin zur Verfügung gestellt.

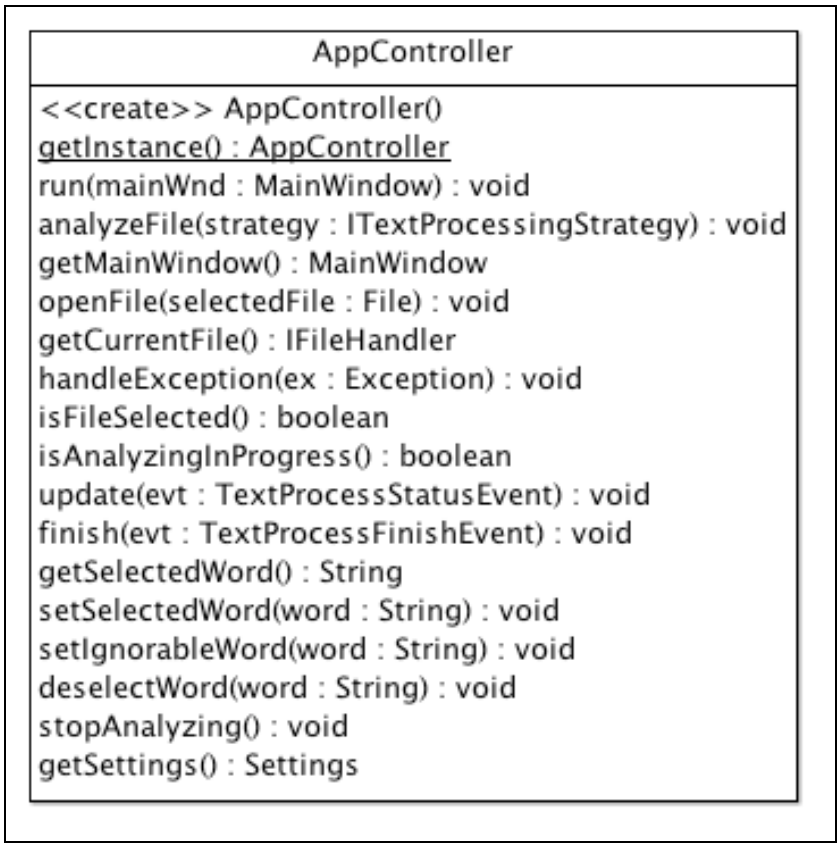

Abbildung 14: öffentliche Schnittstelle der AppController Klasse

Quelle: Eigene Darstellung

Die von der zentralen Instanz des AppControllers angebotene Funktionalität dient für die Steuerung der Programmablauf Logik. So nimmt die AppController Klasse Eingaben der Benutzeroberfläche entgegen und führt Aktionen auf dem Modell entsprechend aus. So wird bspw. beim Öffnen einer Datei über die Schaltfläche auf der Oberfläche die AppController Methode openFile() aufgerufen. Diese sucht anhand der Dateiendung den passenden IFileHandler aus der Liste der verfügbaren IFileHandler heraus und initialisiert diesen mit dem ausgewählten Dateinamen. Zusätzlich nimmt der AppController die Ereignisse bei der Abarbeitung der Analyse vom Modell entgegen (Methoden: update() und finish()) und aktualisiert die Oberfläche durch den Aufruf geeigneter Operationen auf der MainWindow Klasse.

3 Projektplan

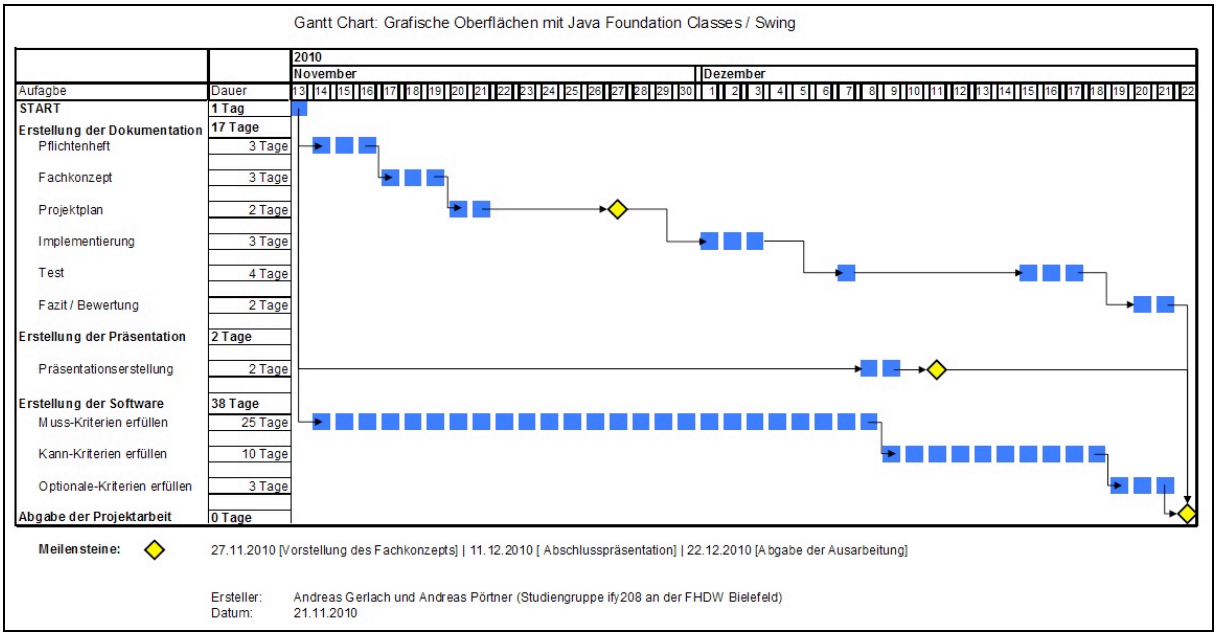

Abbildung 15: Projektplan

Quelle: Eigene Darstellung

Wie in obiger Abbildung dargestellt besteht der zeitliche Ablauf aus drei Phasen. Phase 1 ist die Erstellung des Fachkonzepts, dies beinhalte alle Punkte aus dem Inhaltsverzeichnis. Ebenso beinhaltet Phase 1 einen Meilenstein (Vorstellung: Fachkonzept), wenn diese abgeschlossen ist kann Phase 2 starten.
In Phase 2 wird eine Präsentation erstellt, welche als Meilenstein zur dieser Phase am 11. Dezember vorgestellt wird. Die Präsentation beinhalte eine kurze Einführung in das Thema, anschließend die Lösungsansätze und einen grafischen Entwurf der Software. Je nach Fortschritt der Phase 3 wird es innerhalb der Präsentation eine Live-Demo der bisher erstellen Software geben. Anzumerken ist an dieser Stelle das die Live-Demo kein Pflichtelement der Präsentation ist und nur den aktuellen Stand der Softwareentwicklung darstellen kann.
Phase 3 läuft während des gesamten Projekts, sie dient der Erstellung der Software und läuft neben Phase 1 und 2 parallel ab. Die letzte Phase ist die Hauptphase, während der gesamten Laufzeit wird die Entwicklung der Software voranschreiten.

Die Sollwerte der Projektplanung sind die bereits erwähnten Meilensteine, folgende Liste stellt diese dar:

- Vorstellung des Fachkonzept
- Abschlusspräsentation des Projekts
- Abgabe der entwickelten Software

Der letzte Meilenstein (Abgabe der entwickelten Software) setzt voraus das alle anderen Phasen und Meilensteine abgeschlossen wurden.

4 Test

Testsystem 1: (Dell XPS 16)

Prozessor: Intel® Core™ 2 Duo P8700 @ 2,53GHz
Arbeitsspeicher: 4,00 GB
Betriebssystem: Windows 7 Professional 64 Bit
Klassifikation: 5,9 Windows-Leistungsindex

Testfile's:

testfile_bibel.txt

Größe:	4.766.158 Bytes
Anzahl Wörter:	763680
Anzahl Zeichen:	4766158

testfile_insel.txt

Größe:	53.140 Bytes
Anzahl Wörter:	6555
Anzahl Zeichen:	53140

Zeitmessung: (Zeit für den Einlese- und Auswertungsvorgang)

testfile_bibel.txt:	34,4 Sekunden
testfile_insel.txt:	1,1 Sekunden

Lasttest: (Auslastung der CPU und des Arbeitsspeicher)

testfile_bibel.txt:

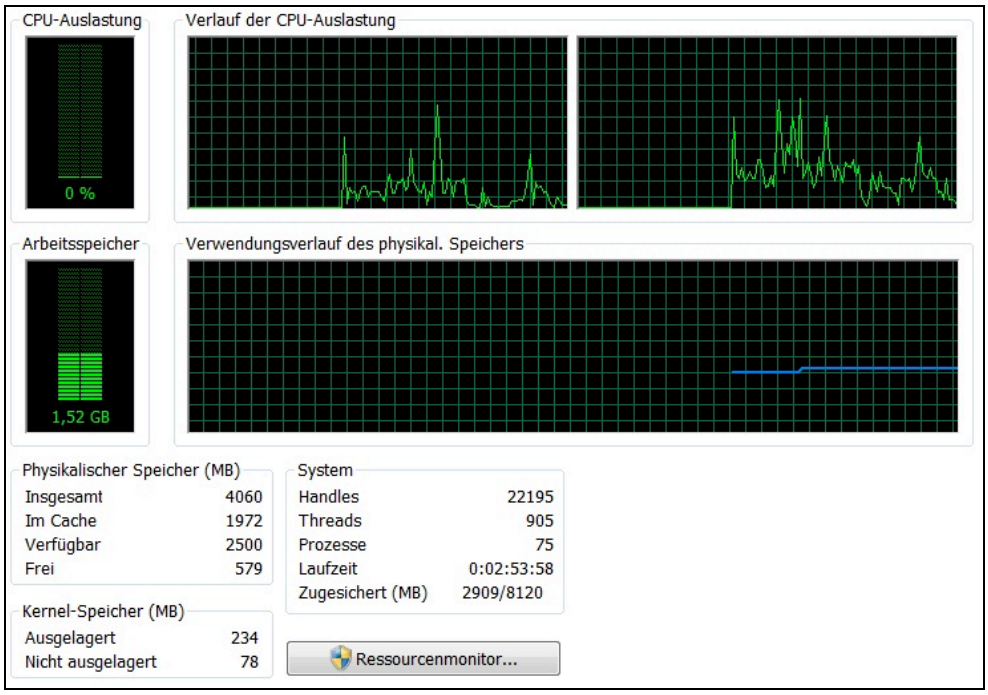

Abbildung 16: Im Ruhezustand

Quelle: Eigene Darstellung

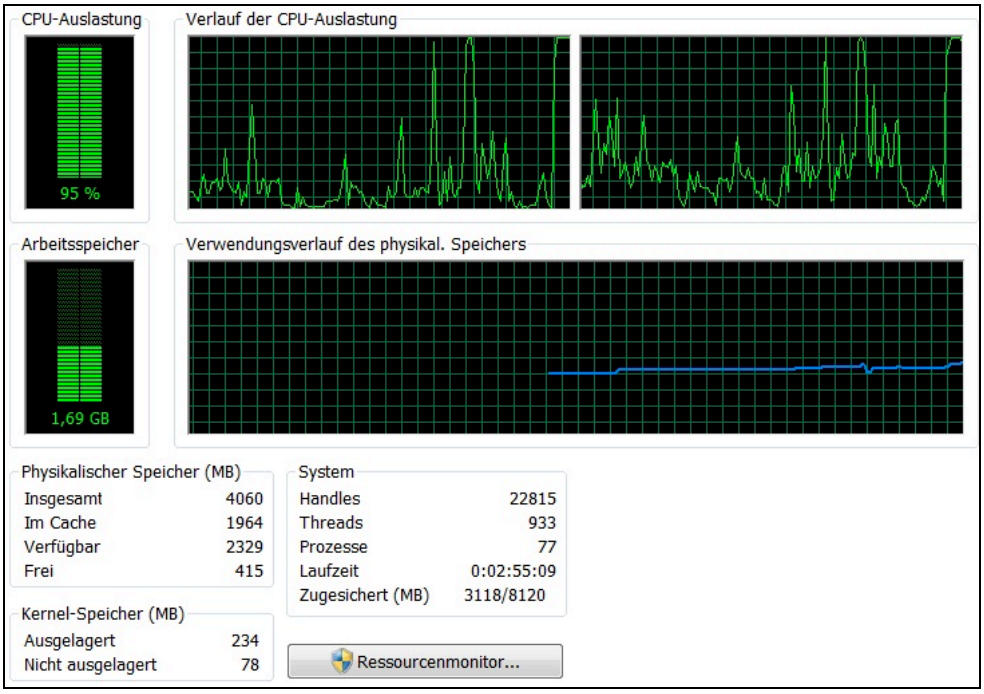

Abbildung 17: Während des Einlesens und der Auswertung

Quelle: Eigene Darstellung

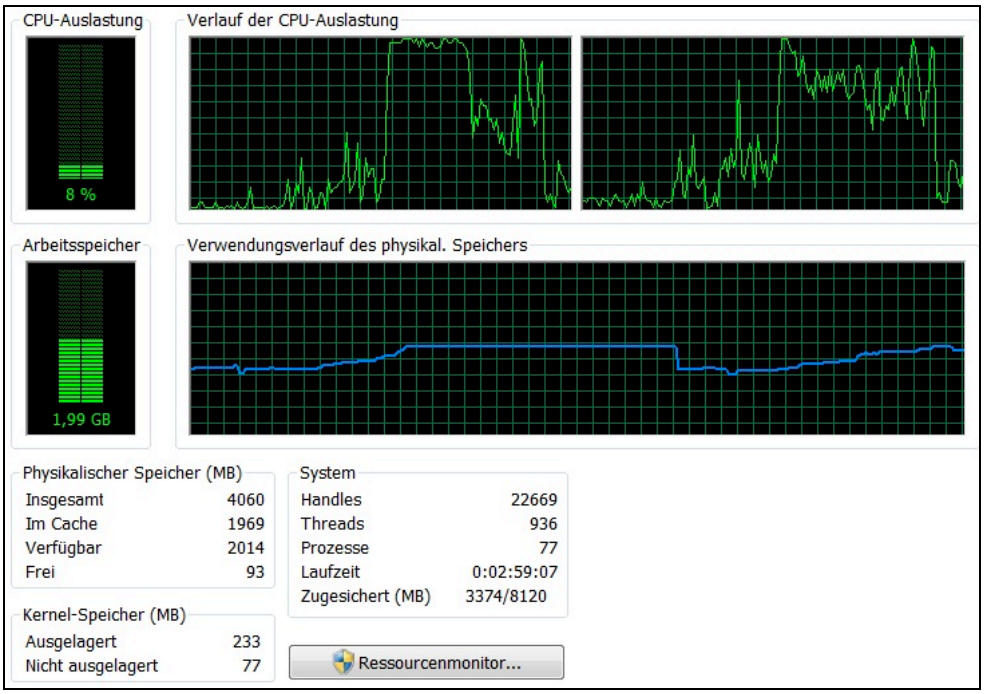

Abbildung 18: Nach dem Analysevorgang

Quelle: Eigene Darstellung

An Hand der Bildschirmausdrucke lässt sich eine Zunahme der CPU-Auslastung von 95% erkennen. Diese Auslastung fällt direkt nach der Analyse und Auswertung wieder zurück bis auf 8%. Im Arbeitsspeicher hingegen werden bis zum Schließen der ASTA-Anwendung ca. 470 MB reserviert.

Testsystem 2: (MacBook pro)

Prozessor:	Intel® Core™ 2 Duo P8700 @ 2,5GHz
Arbeitsspeicher:	4 GB
Betriebssystem:	Mac OS X 10.6.5 64-bit

Zeitmessung: (Zeit für den Einlese- und Auswertungsvorgang)

testfile_bibel.txt:	24,5 Sekunden
testfile_insel.txt:	1,19 Sekunden

5 Fazit / Bewertung

Das „ASTA“ Programm ist für beide Software-Autoren das erste größere Projekt mit Java Swing. Aufgrund der geringen Erfahrung bzgl. der UI Programmierung mit Java haben die Autoren mit einer flachen Lernkurve und vielen zeitraubenden Stolpersteinen gerechnet. Nachdem die ersten Ideen und Gedanken zu Papier gebracht waren, wurde recht schnell der UI Prototype mittels WireframeSketcher erstellt. Dieses war im Anschluss auch Diskussionsgrundlage und Basis für die Entwicklung der verschiedenen UI Panels und Steuerelemente.

Es stellte sich heraus, dass auf Basis der guten Vorarbeit mittels UI Prototyping ein Umsetzen der graphischen Oberfläche unter Zuhilfenahme der diversen Layoutmanager von Java doch recht angenehm und einfach vonstatten ging – selbst ohne Einsatz eines UI WYSIWYG Designers. Mit diesem Ergebnis haben wir zu Beginn der Arbeit an dem Projekt ebenso wenig gerechnet wie mit dem Resultat, dass sich unsere recht ambitionierte Oberflächengestaltung während des UI Prototypings wirklich 1:1 mit Java Swing umsetzen lässt.

Auch das durch die Sprache Java angebotene höhere Abstraktionsniveau kam uns während der Entwicklung der Software zugute – wurde es dadurch doch möglich, auf recht einfache Art und Weise ein PlugIn Mechanismus für die Anwendung zur Unterstützung verschiedener Dateitypen und Text Analyse Strategien verfügbar zu machen.

Einzig die Verlagerung der Analyse in einen Background-Thread und die dadurch notwendige Synchronisation der Oberfläche mit dem Status des Threads wurde durch einige „Trial-‚n’-Error“ Schritte begleitet. Hier stellte sich heraus, dass sich unter gewissen Umständen die Oberfläche nicht aktualisierte. Eine Größenänderung des Rahmens führte dann die notwendigen Aktualisierungen in der Tag-Cloud View durch. Hier sind wir auch bisher auf keine saubere Lösung gekommen – die „Brut-Force“ Methode: beim Aufnehmen eines neues Wortes in die Black-Liste der ausgeschlossenen Wörter die komplette Tag-Cloud zu zerstören und durch eine neue Instanz zu ersetzen, ist nicht die optimale Lösung, die die beiden Autoren beim Konstruieren der Anwendung im Sinn gehabt haben.