Implementierung

Praxis der Softwareentwicklung

Entwicklung einer Software zur Berechnung der Mandatsverteilung im Deutschen Bundestag

Gruppe 1

Philipp Löwer, Anton Mehlmann, Manuel Olk, Enes Ördek,
Simon Schürg, Nick Vlasoff

WS 2013 / 14

Inhaltsverzeichnis

1 Einleitung

Die dritte Phase unseres Projektes - die Implementierung. Unser Ziel in dieser Phase ist es, die bisherigen Errungenschaften in der Pflichten- und Entwurfsphase als ausführbares Programm umzusetzen und dabei möglichst wenig von den bisherigen Entwürfen auszuweichen. Hierbei ist uns jedoch aufgefallen, dass Veränderungen am Entwurf unumgänglich sind. Der Grund hierfür ist, dass bestimmte Sachen einfach nicht beachtet wurden, und somit übersehen worden sind.
Dieses Dokument wird das fertige Programm mit seinen Funktionen erläutern und alle Veränderungen mit den zugehörigen Entwurfsentscheidungen ausführlich erklären. Anschließend werden wir eine Einblick in die nächste Phase geben.

Dieses Dokument ist im Zuge der Implementierungsphase entstanden. Ziel dieser Phase, ist die Umsetzung der in der Pflichten- und Entwurfsphase festgelegten Strukturen und Prozessabläufe unter Berücksichtigung gegebener Rahmenbedingungen, Regeln und Zielvorgaben.
Da sich jedoch während der Implementierung Sachverhalte ergeben, die mit dem eigentlichen Entwurf nicht vollständig zu vereinbaren sind, ist es notwendig einige Änderungen bzw. Anpassungen, aber auch Erweiterungen vorzunehmen. Diese, vom eigentlichen Plan abweichenden Entscheidungen, werden im Folgenden erläutert.
Abschließend wird ein kurzer Ausblick auf die nächste Phase gegeben.

1.1 Notationshinweise

`Klassennamen` werden in diesem Dokument textuell hervorgehoben, indem sie **fett** und in einer anderen Schriftart geschrieben werden.
`Methodennamen` werden hervorgehoben, indem sie *kursiv* und ebenfalls in einer anderen Schriftart geschrieben werden.
Außerdem wird Bundestagswahl im gesamten Entwurfsdokument durch BTW abgekürzt.

2 Pakete

2.1 Datenmodell

Da die verwendeten Listen im Datenmodell durch die Berechnungen und Zuweisungen recht groß wurden, wurde zusätzliche Funktionalität in das Datenmodell gebracht. Dies erleichterte den Zugriff auf die benötigten Daten für verschiedene Komponente wie zum Beispiel **`Mandatsrechner`**, **`Wahlgenerator`** und **`GUI`**. Zudem wurde die Klasse **`BerichtDaten`** für die Klasse **`Sitzverteilung`** erstellt. Dies war notwendig, damit die Berichttabelle in der GUI korrekt befüllt werden kann. Dabei hält die Klasse **`BerichtDaten`** fünf Listen die jeweils die dazugehörigen Spalten befüllen. Kandidaten haben nun einen Namen und einen bestimmten Platz in der Landesliste.

2.2 Import/Export

Die Import-Export-Komponente wurde im Laufe der Implementierung stark angepasst. Anders als im Entwurf, haben wir das Exportieren vom Importieren getrennt. Da die Namen und der feste Platz in der Ladenliste mitgespeichert werden, muss eine zusätzliche .csv-Datei importiert werden. Mithilfe dieser Wahlbewerber-Datei werden die vorher ausgelesenen Kandidaten befüllt. Die Wahlbewerber-Datei für die Bundestagswahl 2013 wird im Programm mit übergeben und kann im Notfall für Bundestagswahl 2009 genutzt werden. Zudem wurde eine Config-Datei(ebenfalls im .csv-Format) hinzugefügt, die die Einwohnerzahl der Bundesländer und die Farben der Partei beinhaltet. Dadurch müssen diese Werte nicht mehr im Programmcode gespeichert und können durch das Editieren der Datei einfach angepasst werden.

2.3 GUI und GUI-Logik

2.3.1 Programmfenster

Das **`Programmfenster`** ist der eigentliche Eintrittspunkt in das Programm, d.h. es enthält die Main- Klasse und wird beim Start als Erstes ausgeführt. Dies bietet sich an, da das **`Programmfenster`** das Erste sein soll, was der Benutzer sieht, da er damit ja interagieren muss.
Wie bereits im Entwurf festgehalten, enthält das **`Programmfenster`** eine Liste von **`Wahlfenstern`**. Diese werden mithilfe einer **`TabLeiste`**, die ebenfalls vom **`Programmfenster`** gehalten wird, realisiert.
Zusätzlich besitzt es ein **`Menu`**, welches dem Benutzer ermöglicht, den gewünschten Befehl auszuwählen und ausführen zu lassen, ohne genaue Steuerbefehle kennen und anwenden zu müssen.

2.3.2 WahlFenster

2.3.3 Ansicht

Die **Ansicht** die Hauptkomponente des **WahlFenster**s. Sie enthält die im Späteren näher erläuterten **Tabellen-**, **Diagramm-** und **Kartenfenster**. Anders als im Entwurf festgelegt, haben wir uns entschieden nur eine Ansicht zu implementieren. Hauptgrund dafür war, dass eine **Ansicht** ausreichend ist, da bei Ansichtswechsel nur ein neues **DiagrammFenster** und ein neues **TabellenFenster** erzeugt werden müssen, das **KartenFenster** bleibt, dank JTree das selbe. Immer wieder das selbe KartenFenster-Objekt zwischen den drei verschiedenen Ansichten hin und her zu schieben wäre weit aus aufwendiger als einfach nur eine universale Ansicht einzuführen.
Die im Entwurf spezifizierte Methode *zeigeKomponenten()* wurde in zwei Methoden (*Initialisieren()* und *ansichtAendern()*) aufgespalten. Dies war von Nöten, weil bei der erstmaligen Ansichtserstellung alle drei Fenster erstellt werden müssen, bei einer Ansichtsänderung aber nur **Diagramm-** und **Kartenfenster** neu erstellt werden müssen.
Eine weitere Abweichung vom Entwurf ist die *berechnungNotwendig()*, welche festlegt, dass eine Stimme in einem Wahlkreis geändert wurde. Wurde eine Stimme geändert werden keine Diagramme angezeigt, sondern ein Berechne-Knopf an der **DiagrammFenster** Stelle angezeigt.
Der Hauptgrund für diese Änderung ist, dass es dadurch möglich ist mehrere Stimmen nacheinander zu ändern, ohne dass nach jeder Änderung eine neue Berechnung durchgeführt werden muss.

2.3.4 TabellenFenster

Das **TabellenFenster** ist das erste der drei Komponenten der **Ansicht**. Nicht wie im Entwurf vorgeschlagen in einer Klasse, haben wir das dieses in mehrere Klassen unterteilt. Da es drei Arten von Tabellen gibt (Land, Bundesland, Wahlkreis) gibt es zu jeder Art zwei Klassen, ein Mal die Daten-Klasse und eine TabelModel-Klasse. Dies hat die im Entwurf vorgeschlagene **Tabellenzellen**-Klasse zur Auslese von geänderten Stimmen abgelöst, da man dadurch viel leichte an die, in der Tabelle geänderten Stimmen kommt. Das **TabellenFenster** an sich erstellt die Tabellen wie im Entwurf vorgeschlagen mit der *tabellenFuellen()*-Methode, wobei diese für die drei Gebietsarten überladen ist. Die Erstellung der Klasse **GUIPartei** war notwendig, um Daten wie Sitze, Direktmandate, etc. festzuhalten.

2.3.5 DiagrammFenster

Das **DiagrammFenster** ist das zweite der drei Komponenten. Die Klasse an sich wurde fast genauso implementiert wie im Entwurfsdokument festgelegt. Das einzige was noch hinzugefügt wurde waren die verschiedene

Diagramm-Klassen, die die Diagramme darstellen sollen. Die Methode *erstelleDiagramm()* wurde überladen, weil die drei Arten von Diagrammen in den vorher genannten Klassen erstellt werden. Die Methode *zeigeSitzpla* öffnet das folgende **BerichtsFenster**.

2.3.6 BerichtsFenster

Im **BerichtsFenster** werden Daten visualisiert, die veranschaulichen sollen, woher Mandate der Abgeordneten kommen. Dieses wurde als Tabelle implementiert, ähnlich wie das **TabellenFenster**, um die hohe Menge an Daten möglichs übersichtlich zu halten. Zu dem **BerichtsFenster** gehören die Klassen **BerichtTableModel** und **BerichtDaten**.

2.3.7 KartenFenster

Das **KartenFenster** ist die letzte Komponente der **Ansicht**. Wie schon im Pflichtenheft festgelegt, ist es als Tabfenster implementiert und wie im Entwurf festgelegt gibt es die Methode *zeigeInformationen()*, die die Karte erstellt und eine Verzeichnisstruktur anlegt.
Das einzige was vom Entwurf abweicht ist das Weglassen des Zurück-Knopfes welches unnötig wurde, da man sich in der Verzeichnisstruktur von Ansicht zu Ansicht navigieren kann.

2.3.8 VergleichsFenster

Das **Vergleichsfenster** wurde wie im Pflichtenheft und im Entwurf beschrieben implementiert, wobei ein weiteres Diagramm hinzugefügt wurde, welches die Sitzdifferenzen der zwei Wahlen anzeigt. Dies fördert die Verdeutlichung der Unterschiede zwischen zwei Wahlen.

2.3.9 GUISteuerung

2.3.10 TabellenFenster

2.4 Mandatsrechner

Die Klasse **Mandatsrechner2009** berechnet die Sitzverteilung nach Sainte-Laguë/Schepers ohne Ausgleichsmandate und **Mandatsrechner2013** berechnet die Sitzverteilung ebenfalls nach Sainte-Laguë/Scheper, aber mit Ausgleichsmandate. Da der **Mandatsrechner2013** dadurch den **Mandatsrechner2009** zur Berechnung nutzt, fällt die Notwendigkeit der Oberklasse **Mandatsrechner** weg. Deswegen hält der **Mandatsrechner2013** ein Objekt der Klasse **Mandatsrecher2009**. Zudem

wurde noch in **Mandatsrechner2009** das Verteilungsverfahren nach d'Hondt implementiert, damit eine alternative Berechnung der Sitzverteilung möglich ist. Die Überladung der Methoden *bechne(Gebiet gebiet)* wurde aufgehoben, da die Berechnung nicht nach Gebieten sondern nach Ober- und Unterverteilung orientiert ist. Damit der **Mandatsrechner2013** möglichst viel wieder verwendet werden kann, wurden Bereiche die in beiden Berechnungsklassen Verwundung finden ausgelagert. Für die Implementierung des Entwurfsmuster Einzelstück wurden möglichst wenig globale Variablen, die vor jeder Berechnung neu initialisiert werden, verwendet, damit in dem Mandatsrechner nicht gewollte Zustände ausgeschlossen werden können.

2.5 Wahlgenerator

2.6 Wahlvergleich

2.7 Steuerung

2.8 Sonstiges

3 Vorschau auf die nächste Phase

3.1 Ideen und Ziele

3.2 Zeitplan