Validierung

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

Dieses Dokument ist im Zuge der Validierungsphase entstanden.

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 Globale Testfälle und Szenarien

Im folgenden werden die im Pflichtenheft genannten Testszenarien durchgeführt und analysiert.

/T0010/ Zwei Wahlen miteinander vergleichen:
Das Vergleichsfenster öffnet wie vorgesehen, aber es gibt noch einige Ungereimtheiten.

- Wahl kann mit sich selbst verglichen werden, was keinen Sinn macht, wenn man in der Vergleichsansicht keine Stimmen ändern kann
- Wahlfenster öffnet sich im Hintergrund
- Berichtsfenster können nicht angezeigt werden

/T0020/ Manuell einen negativen Wert als Stimmenanzahl eintragen:
Exception wird geworfen, aber noch keine Fehlermeldung ausgegeben.

/T0030/ Manuell einen Buchstaben als Stimmenanzahl eintragen: Folgende Meldungen werden ausgegeben.

- Nur positive ganze Zahlen erlaubt.
- Stimme konnte nicht geändert werden.

Daraufhin wird die Stimme wieder zurückgesetzt

/T0040/ Eine Fließkommazahl als Stimmenanzahl eintragen:
siehe **/T0030/**

/T0050/ Erststimme in der Wahlkreisansicht verändern:
funktioniert.

/T0060/ Die Funktion “Diagramm wechseln” testen:
Funktion nicht in aktuellem Programm realisiert.

/T0070/ Die Funktion “Rückgängig machen” testen:
Zurücksetzen funktioniert noch nicht.

/T0080/ Die Funktion “‘Wiederherstellen” testen:
Da rückgängig machen noch nicht funktioniert, kann wiederherstellen nicht angewählt werden.

2.0.1 Import-/Exportverhalten

Die folgenden Testfälle testen das Import-/Exportverhalten des Programms. Dabei wird vorausgesetzt, dass das Programm gestartet wurde und sich im Startzustand befindet.

/T0110/ Struktur einer Importdatei verändern:
Test: Gebiet gelöscht
Exception in Crawler (”Kein geeigneter Crawler gefunden”) wurde geworfen, aber keine Fehlermeldung an Benutzer ausgegeben

/T0120/ ??

/T0130/ Test: SPD zu CDU
Keine Exception und keine Fehlermeldung. Programm funktioniert wie gewöhnlich

/T0140/ Nur eine Partei befindet sich in der Importdatei:
Test: Alle Parteien außer CDU entfernt, Bundesländer und Wahlkreise für die Übersichtlichkeit entfernt, Stimmzahlen angepasst(z.B. Gesamtzweitstimmenanzahl in D)
Endlosschleife

/T0150/ Importdatei mit fehlerhaften Bundesländernamen: Test: Hamburg -¿ Hambe
Exception im Mandatsrechner

/T0160/ Eigenen Wahlausgang erstellen:
funktioniert.

2.0.2 Korrekte Berechnung der Sitzverteilung

Die folgenden Testfälle testen die korrekte Berechnung der Sitzverteilung. Dabei wird vorausgesetzt, dass das Programm gestartet wurde und erfolgreich eine Importdatei geladen wurde.

/T0210/ Ein Direktmandat fehlt:

- Es kann immer nur eine Erststimmenanzahl geändert werden, dann muss berechnet werden
- Es können alle Erststimmenanzahlen auf 0 gesetzt werden, aber Direktmandat im Bundesland wird immer noch angezeigt
- Berechne- Knopf erscheint immer über Diagramm
- Exception: java.lang.OutOfMemoryError: Java heap space

/T0220/ Mehrere Wahlkandidaten haben gleich viele Stimme in einem Wahlkreis:

- Es wird kein Hinweis hinsichtlich einer Auslosung ausgegeben
- bleibt zu testen, ob überhaupt gelost wird

/T0230/ Ein negatives Stimmgewicht in einer Wahl provozieren: fällt im Moment weg.

/T0240/ Partei mit drei Direktmandate und 2.9 Prozent der Zweitstimmen:

- Bei 3 Direktmandaten passiert nichts
- Wahlgenerierung und Partei 3 Prozent der Zweitstimmen gegeben, sie war aber nicht im Bundestag vertreten

/T0250/ Überhangmandat testen:

- Überhangmandate werden in Landesansicht nicht angezeigt
- Hinweis meiner Meinung nach überflüssig

/T0260/ Ausgleichsmandat testen:

- Hinweis meiner Meinung nach nicht wichtig

weitere :

- leere Bewerberliste NullPointer in Mandatsrechner

2.1 GUI-Test-Plan

Folgender Ablauf sollte alles testen, was ein Benutzer mit unserem Programm machen kann:
Als aller Erstes sollten alle kleinen Dinge getestet werden, dies sind:
- Handbuch
- About
- Lizenz
Als nächstes sollte man eine neue Wahl importieren und daraufhin die Bundestagswahl 2013 schließen. In dieser neuen Wahl kann man durch verschiedene Bundesländer und Wahlkreise hin und her wechseln. Anschließend sollte man in einem der Wahlkreise jeweils eine Erst- und eine Zweitstimme ändern. Nun sollte der 'Berechne' Knopf betätigt werden.
Änderung einer weiteren Stimme und betätigen der 'Rückgängig'-Funktion erprobt die Funktion, eine Eingabe zu ändern. Hier kann auch als Eingabe ein Buchstabe, eine negative Zahl oder eine Zahl, die über die Anzahl der Wahlberechtigten geht, probiert werden. Anschließend kann man mit dem 'Wiederherstellen' Knopf die Stimme, die man zurückgesetzt hat, wieder herstellen.
Als nächstes kann man den 'Bericht' aufrufen, in diesem Tabellen verschieben und sortieren und wieder schließen.
Zum Schluss sollte man die veränderte Wahl exportieren und wiederum importieren, um zu testen, ob der Import veränderter Wahlen funktioniert.
Nun kann man sich eine zufällige Wahl generieren lassen, deren Auswertung überprüft werden sollte.
Diese kann man mit der vorher importierten Wahl vergleichen. Auch dort ist das sortieren und verschieben von Tabellenspalten möglich.
Zum Schluss sollte das Programm durch 'Schließen' beendet werden.
Dieser Plan umfasst alle Funktionen des Programmes.

2.1.1 Fehler der GUI

- Generiere-Knopf blieb aktiv, nachdem man den Namen löschte
  Das Problem bei diesem Fehler war, dass ein ActionListener verwendtet wurden. Aus diesem Grund wurde die Abfrage, ob der aktuelle Name gesetzt ist, nur abgefragt, sobald der Benutzer den Enter-Knopf betätigte. Durch den neuen KeyListener wird bei kleinster Änderung abgefragt, ob sich noch Text in dem JTextField befindet.

- Berichtsknopf
  Bei der Präsentation der Implementierungsphase ergab sich, dass die Einführung eines Knopfes der den Bericht erscheinen lässt handlicher wäre. Vorher musste man auf das Diagramm klicken, was nicht gerade intuitiv war.
  Wegen dem neuen Button waren Änderungen am Layout von Nöten. Die Klasse DiagrammFenster enthält jetzt das GridBagLayout, wobei unter dem Diagramm der Knopf angezeigt wird.

- Änderungen von Erst- und Zweitstimmen
  Nach der Implementierung war es noch möglich negative Stimmwerte einzutragen. Dies wurde behoben, indem eine einfache Abfrage eingeführt wurde, sobald der Wert unter 0 ist wird eine NumberFormatException geworfen.
  Außerdem war es möglich mehr Stimmen abzugeben als es Wahlberechtigte im Land gab. Auch dies wurde durch eine Extra-Abfrage behoben.
  Folgender Test wurde ausgeführt, um die Korrektheit der Stimmenänderung zu testen:
  - Nach .csv-Datei sind im Wahlkreis Karlsruhe-Stadt noch 57624 Wahlberechtigte übrig. Nun ändern wir die Erststimmen einer Partei, die 0 bekommen hat, auf 57625. Wie erwartet, wird angezeigt, dass nur noch 57624 Wahlberechtigte übrig sind.
  - Eine Änderung um den Wert 57624 ist ohne Probleme möglich.
  - Ein ähnlicher Test wurde auch für die Zweitstimmen erprobt.

- Tabwechsel
  Das Wechseln der Tabs enthielt den Fehler, dass die Bundestagswahl, die die Steuerung hält, nicht zu der geändert wurde, die zum neuen Tab gehört.
  Dies wurde korrigiert, indem der TabLeiste zu jedem Tab ein MouseListener hinzugefügt wurde.

- Letzten Tab schließen
  Wurde der letzte Tab geschlossen, war es nicht mehr möglich, eine neue Wahl zu importieren, das Programm wurde unbenutzbar. Durch eine neue if-Abfrage im Listener des 'x'-Buttons ist es jetzt nur noch möglich einen Tab zu schließen, wenn noch mindestens zwei Tabs offen sind.

3 Im Pflichtenheft genannte Qualitätsanforderungen

Die im folgenden genannten Qualitätsanforderungen aus dem Pflichtenheft wurden überprüft.

- Starten des Programms: unter 10 Sekunden
- Laden eines Zustandes: unter 10 Sekunden
- Berechnung der Sitzverteilung: unter 10 Sekunden
- Speichern eines Zustandes: unter 10 Sekunden
- Exportieren/Importieren von Daten: unter 10 Sekunden
- Beenden des Programms: unter 3 Sekunden

Die Genauigkeit des Algorithmus zur Sitzberechnung muss demWahlgesetz entsprechen und exakte Ergebnisse liefern.
Wie genau habe wir das getestet?

- Hilfreiche Fehlermeldungen
- Kein Datenverlust (auch nach Programmabstürzen)
- Gespeicherte Daten müssen immer konsistent gehalten werden
- Kurze Einarbeitungszeit

4 Unit Tests

4.1 Wahlvergleich

Der Wahlvergleich wurde auf zwei Arten getestet. Einmal haben wir zwei gleiche Wahlen verglichen und überprüft, ob die übergebenen Werte gleich waren und außerdem, ob die Differenzen 0 betrugen.
In der zweiten Testklasse wurden die Bundestagswahl 2013 und die Bundestagswahl 2009 miteinander verglichen.
Orakel für die Differenzen der Erst- und Zweitstimmen als auch für die prozentualen Differenzen waren wir selbst. Auch hier zeigte sich der korrekte Ablauf des Wahlvergleichs.

4.2 Model

Das Teste des Models hat uns am meisten Zeit gekostet, da es der Hauptteil unseres Codes ist.
In Klassen wie der Sitzverteilung, Gebiet, Mandat, Stimme brauchten nahezu gar nicht getestet zu werden, da sie kaum relevanten Code, bis auf Getter-Setter-Methoden, enthielten.
Eine größere Herausforderung waren Klassen wie Partei, Wahlkreis, Bundesland,...

- Wahlkreis, Bundesland, Deutschland:
  In diesen Klassen wurden Methoden wie getAnzahlErst-,Zweitstimmen getestet. Dank dieser Tests wurden einige Methoden entfernt, die das gleiche wie andere machten. Außerdem wurden die Methoden getStimmeProPartei aus den Klassen entfernt, da sie riskanten Code enthielten und durch einfacheren Code in anderen Methoden ersetzt.

- Partei:
  In der ParteiTest-Klasse wurden die Methoden der Partei getestet.
  Tests zu fundamentalen Sachen wie dem setzen einer Landesliste, hinzufügen eines neuen oder des Mandates eines bereits vorhandenen Mitglieds wurden verfasst und durchgeführt. Genauso das Setzen von Ausgleichs- und Überhangsmandaten musste überprüft werden.
  Grobe Fehler sind uns dabei nicht untergekommen, bis auf das fehlen einiger equals-Methoden, welche hinzugefügt wurden.

- Erst- und Zweitstimme Bei den Stimmen wurde hauptsächlich die Änderung und Kopie dieser getestet. Gravierende Fehler wurden bei den Tests nicht gefunden.

Auch bei den restlichen Klassen des Models wurden nur kleine Fehler, wie eine Methoden-Löschung oder einfaches Code-Refactoring, verbessert.

5 Testabdeckung

EclEmma Welche Art von Abdeckung haben wir erreicht?

6 Codeanalyse

FindBugs, Checkstyle

7 Performanceanalyse

Zu der gesamten Performance ist einzig zu sagen, dass wir die Grenzen, die wir uns im Pflichtenheft gesetzt haben, erfüllt haben.
- Die Berechnung der Sitzplatzverteilung erfolgt auf allen unseren Rechnern in unter 3 Sekunden.
- Der Import der Bundestagswahl 2013 dauerte in fast allen Fällen unter 5 Sekunden, nur in einzelnen etwas länger.
- Der gesamte Programmstart, inklusive Import und Berechnung, benötigt keine 10 Sekunden und ist somit unter unseren Anforderungen.

Das einzige Problem, das uns bis zur letzten Woche begleitete, war das Volllaufen des

Heaps, das ausgelöst wurde, wenn man schnell zwischen verschiedenen Gebieten schaltete. Hier sehen Sie eine Grafik unserer ersten Analyse des Problems mit dem Programm JVisualVM.

Den Fehler fanden wir in der Ansichtsklasse. Dort wurde die Größe des Tabellenfensters bei jeder Änderung um das Zweifache erhöht. Dies hatte keinerlei Auswirkungen auf die GUI, doch die immer größer werdenden double-Werte sprengten den Rahmen des Heaps, verlangsamten das Programm stark und machten es fast unbenutzbar.
Gelöst wurde das Problem, indem die GridHeight des GridBagConstraits der Ansicht bei Diagramm- und Kartenfenster auf 1 und bei Tabellenfenster auf 2 gesetzt wurde. So sieht die Analyse der JVisualVM nach dem beheben des Fehlers aus, wenn man schnell Gebiete wechselt:

8 Ausblick auf die Endphase des Projekts