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

Ziel der Validierungsphase war es das fertige Software-Produkt zu überprüfen, um die möglichst fehlerfreie Funktionsweise der Software bestmöglich zu gewährleisten. Dazu wurden sowohl die im Pflichtenheft bereits beschriebenen Testfälle, als auch zahlreiche weitere Tests sofern möglich, automatisch, mit JUnit durchgeführt und die dabei erreichte Testabdeckung mittels EclEmma ermittelt. Desweiteren wurde sowohl die GUI nach einem von uns verfassten und in diesem Dokument angegebenen Testplan getestet als auch die Funktion der Software auf verschiedenen Systemen sichergestellt und die Qualität des Produktes durch JUnit Tests überprüft.
Gruppenintern wurde der Name OpenBundestagswahl für die Entwickelte Software gewählt.

2 Globale Testfälle und Szenarien

Im Folgenden werden die im Pflichtenheft genannten Testszenarien durchgeführt und analysiert. Dabei werden die Ergebnisse, die am Anfang der Validierungsphase entstanden sind, mit abschließenden Testergebnissen verglichen.

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

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

Oben aufgeführte Ergebnisse wurden korrigiert. Der Test funktioniert jetzt fehlerlos.

/T0020/ Manuell einen negativen Wert als Stimmenanzahl eintragen:

- Exception wird geworfen, aber noch keine Fehlermeldung ausgegeben.

Oben aufgeführtes Ergebnis wurde korrigiert. Der Test funktioniert jetzt.

/T0030/ Manuell einen Buchstaben als Stimmenanzahl eintragen:

- Funktionierte bereits am Anfang der Validierungsphase ordnungsgemäß

/T0040/ Eine Fließkommazahl als Stimmenanzahl eintragen:

- Funktionierte bereits am Anfang der Validierungsphase ordnungsgemäß

/T0050/ Erststimme in der Wahlkreisansicht verändern:

- Wirft Exception im Model

Wurde korrigiert.

/T0060/ Die Funktion “Diagramm wechseln” testen:

- Funktion nicht in aktuellem Programm realisiert.

Funktion wurde verworfen, da sie nicht benötigt wurde.

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

/T0080/ Die Funktion “‘Wiederherstellen” testen:
Wiederherstellen funktioniert fehlerlos.

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
Dem Benutzer wird folgende Nachricht ausgegeben: ”Die .csv-Datei entspricht nicht dem gewünschten Format.”

/T0120/ Importieren, verändern, exportieren und das anschließende Importieren funktionieren.

/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).
Es wird ein Fehler ausgegeben.

/T0150/ Importdatei mit fehlerhaften Bundesländernamen: Test: Hamburg -¿ Hambe Exception im Mandatsrechner wurde gelöst. Es entspricht nicht dem erwünschten Format.

/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

Die Idee aus dem Pflichtenheft wurde verworfen, da bei 0 vergebenen Erststimmen zwischen allen Kandidaten ein Direktmandat verlost wird. Es können jetzt mehrere Erststimmen geändert werden und dann wird erst per Berechnen-Knopf eine neue Sitzverteilungsberechnung angestoßen. Diagramm wird ausgeblendet und Berechnen-Knopf wird passend angezeigt. Auch eine OutOfMemoryException wird nicht mehr geworfen.

/T0220/ Mehrere Wahlkandidaten haben gleich viele Stimme in einem Wahlkreis:
Es wird keine Meldung ausgegeben, es wird einfach so gelost.

/T0230/ Wurde endgültig verworfen.

/T0240/ Partei mit drei Direktmandate und 2.9 Prozent der Zweitstimmen
Partei ist richtigerweise mit 3 Sitzen im Bundestag vertreten.

/T0250/ Überhangmandat testen:

Wird richtig berechnet, es wird aber kein Hinweis ausgegeben.

/T0260/ Ausgleichsmandat testen:

Wird richtig berechnet, es wird aber kein Hinweis ausgegeben.

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 und wurden eingehalten.

- 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 dem Wahlgesetz entsprechen und exakte Ergebnisse liefern.
- Hilfreiche Fehlermeldungen
- Gespeicherte Daten müssen immer konsistent gehalten werden
- Kurze Einarbeitungszeit

4 Unit Tests

Wir haben Unit Tests geschrieben um die einzelnen Methoden dieser Anwendung automatisch und wiederholbar testen zu können.

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 Testen 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 wurde die Methode getErststimmeProPartei aus den Klassen entfernt, da sie riskanten Code enthielt und durch einfacheren Code in anderen Methoden ersetzt werden konnte.

- Partei:
  In der ParteiTest-Klasse wurden die Methoden der Partei getestet.
  Tests zu fundamentalen Sachen, wie dem Setzen einer Landesliste, Hinzufügen eines neuen Mitglieds oder des Mandates eines bereits vorhandenen Mitglieds, wurden verfasst und durchgeführt. Genauso das Setzen von Ausgleichs- und Überhangmandaten 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.

4.3 Mandatsrechner

- Mandatsrechner2009:
  Beim Mandatsrechner2009 wurde als erstes die Initialisierungen der Listen in der Sitzverteilung und die Mandatseinträge der Kandidaten getestet.
  Anschließend die Verteilung der Direktmandate. Dabei wurde einerseits in einem Wahlkreis abgefragt, ob der Kandidat mit den meisten Erststimmen das Direktmandat erhielt und andererseits ob die Vergabe des Direktmandats bei mehreren Kandidaten mit gleicher Stimmenanzahl zufällig ist.
  Bei den letzten Tests wurde die Sperrklausel getestet. Es wurde einer Partei über 5-Prozent der Zweitstimmen gegeben, einer weniger als 5-Prozent der Zweitstimmen aber 3 Direktmandate und einer weiteren weniger als 5-Prozent und weniger als 3 Direktmandate. Die ersten beiden zogen unter Berücksichtigung beider Stimmen in den Bundestag ein, bei der letzten Partei wurden nur die durch die Direktmandate erworbenen Sitze berücksichtigt.
  Bei diesen Tests ist uns aufgefallen, dass die Abstimmung zwischen abgegebenen Stimmen und Wahlberechtigten in diesem Wahlkreis nicht korrekt funktioniert. Es war möglich mehr Stimmen abzugeben als Wahlberechtigte vorhanden waren. Dieser Fehler konnte behoben werden.

- Mandatsrechner2013:
  Im Mandatsrechner 2013 wurde die korrekte Berechnung der Daten getestet, da der Mandatsrechner 2013 den Mandatsrechner 2009 benutzt und dort schon die genutzten Methoden getestet wurden. Es wurde die berechnete Sitzverteilung mit der aktuellsten Bundestagswahl 2013 verglichen. Dabei fiel uns auf, dass in der Sitzverteilung mehr Kandidaten waren als in dem offiziellen Ergebnis. Interessant war auch, dass es Kandidaten in der Sitzverteilung gab die kein Mandat hatten. Dies fiel uns aber in der GUI nicht auf, weil dort die Mandate der Kandidaten erfragt werden und danach die Summe in die entsprechende Tabelle eingetragen werden. Durch die Tests wurde klar, dass die Kandidaten nicht aus der Sitzverteilung entfernt werden. Der Grund für das Problem wurde dann in der Berechnung der Ausgleichsmandate gefunden. Nach der Änderung wurden die Kandidaten aus der Liste entfernt und der erste Test konnte fehlerfrei durchlaufen.
  Beim zweiten Test wollten wir anfangs selbst eine Berechnung der Wahl durchführen und das Ergebnis sozusagen als Orakel mit dem berechneten Ergebnis des Mandatsrechners vergleichen. Da der Rechenaufwand für eine ganze Bundestagswahl relativ groß ist und wir gleichzeitig den Import von anderen .csv-Dateien testen wollten, wurde die Bundestagswahl 2009 mit dem Wahlgesetz 2013 berechnet und mit dem Rechenbeispiel von Ulrich Wießner verglichen. Jedoch bemerkten wir danach, dass das Ergebnis von Ulrich Wießner teilweise falsch war/ist, denn laut seiner Berechnung hätte die CSU 47 Direktmandate in Bayern, obwohl es nur 45 Wahlkreise gibt.

Deswegen wurde bei diesem Test die Anzahl der Sitze der CSU nicht verglichen. Außerdem haben wir unser Ergebnis der Wahl auch mit dem amtlichen Ergebnis des Bundeswahlleiters verglichen. Dabei wurde unser Ergebnis bestätigt.
Der letzte Test berechnet wie der erste Test die Sitzverteilung von 2013. Dabei wird darauf geachtet, dass die ganze Berechnung nicht länger als 10 Sekunden dauert.

4.4 Wahlgenerator

Beim Wahlgenerator wurde als erstes das Verhalten des Konstruktors getestet, ob er sich bei der Übergabe von unterschiedlichen Stimmanteilen korrekt verhält. Dabei wird geprüft, ob er mit erwarteten Werten erzeugt wird, bzw. wie erwartet eine Exception wirft.

Dann wurde das zufällige Verteilen von übrigen Stimmanteilen getestet und die Stimmzahlen der Ergebniswahl überprüft.

Außerdem wurde ein Laufzeittest geschrieben, der feststellt, ob die Wahlgenerierung in unter 5 Sekunden ausgeführt wird.

Insgesamt wurden durch die Unit Tests mehrere Fehler entdeckt und vor allem die Validierung der Parameter robuster gestaltet.

4.5 Import/Export

Das Import-/Export Modul muss an erster Stelle zuverlässig mit den offiziellen Ergebnissen des Bundeswahlleiters arbeiten können. Daher gibt es einen Basistest, der die Wahlergebnis- und Wahlbewerber-Datei der Bundestagswahl 2013 importiert und exportiert. Als nächstes, ist es wichtig, dass das Import/Export Modul auch Veränderungen an dem importierten Objekt exportiert. Dies wird durch den doppelten Import/Export-Test abgedeckt. Hier wird ein Wahlergebnis importiert, exportiert, daraufhin wird die neu erstellte Datei erneut importiert und exportiert. Die relevanten Felder sind in allen drei Dateien gleich.

Anschließend ist zu überprüfen, wie das Modul auf fehlerhafte Dateien reagiert. Hierfür wird einmal eine fehlerhafte Wahlergebnis-Datei, einmal eine fehlerhafte Wahlbewerber-Datei und einmal sowohl eine fehlerhafte Wahlergebnis-, als auch Wahlbewerber-Datei importiert. In diesen Fällen, wird der Fehler in der GUI behandelt.

Zum Schluss wurde erfolgreich überprüft, ob generierte Wahlen exportiert und importiert werden können.

5 Testabdeckung

Die Testabdeckung unseres Codes haben wir durch das Eclipse-PlugIn EclEmma ermittelt. Getestet wurde jeglicher Code der nichts mit der Visualisierung des Programmes zu tun hat. Aus diesem Grund wurden das GUI-Paket und einige Teile anderer Pakete, wie

z.B. BerichtsFenster, nicht mit JUnit getestet.
Folgende Anweisungsüberdeckungen konnten erreicht werden (in Prozent):

- Chronik : 80,9
- Config: 46,9
- ImportExport: 93,2
- Mandatsrechner: 81,6
- Model: 84,9
- Wahlgenerator: 82,8
- Wahlvergleich: 65,8

6 Codeanalyse

Es wurde zum einen darauf geachtet, einheitlichen und gut lesbaren Code basierend auf bekannten Konventionen zu schreiben. Dafür wurde wie auch in der vergangenen Phase Checkstyle als Unterstützung verwendet.

Zum anderen haben wir das Tool FindBugs verwendet, dass Java Code auf Fehlermuster untersucht um gängige Fehler zu vermeiden.

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 allen Fällen unter 5 Sekunden.
- 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 sieht man 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.

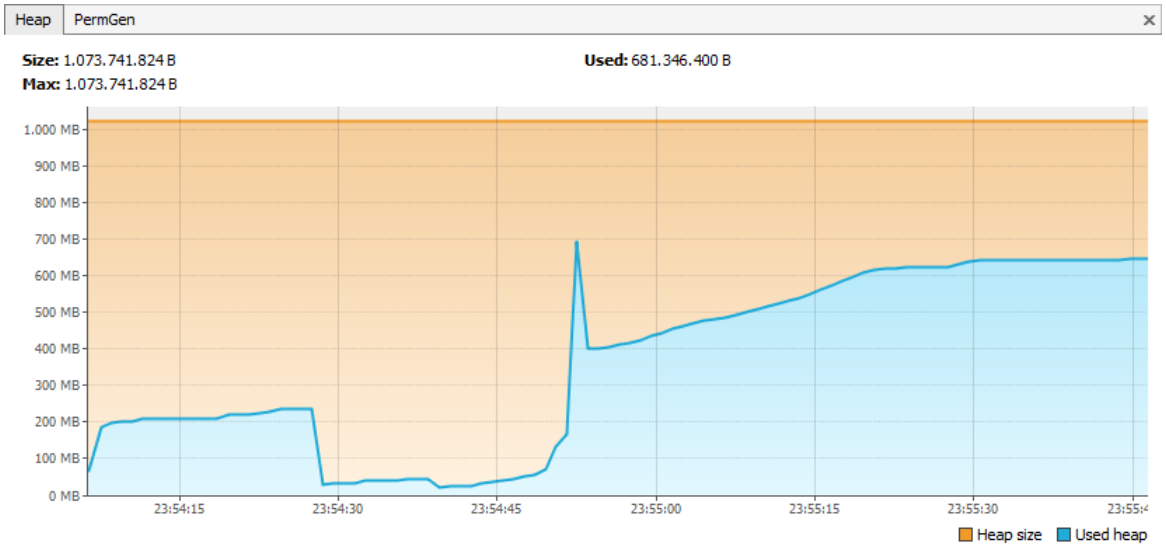

Abbildung 1: Auswertung der Performance mit Fehler

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:

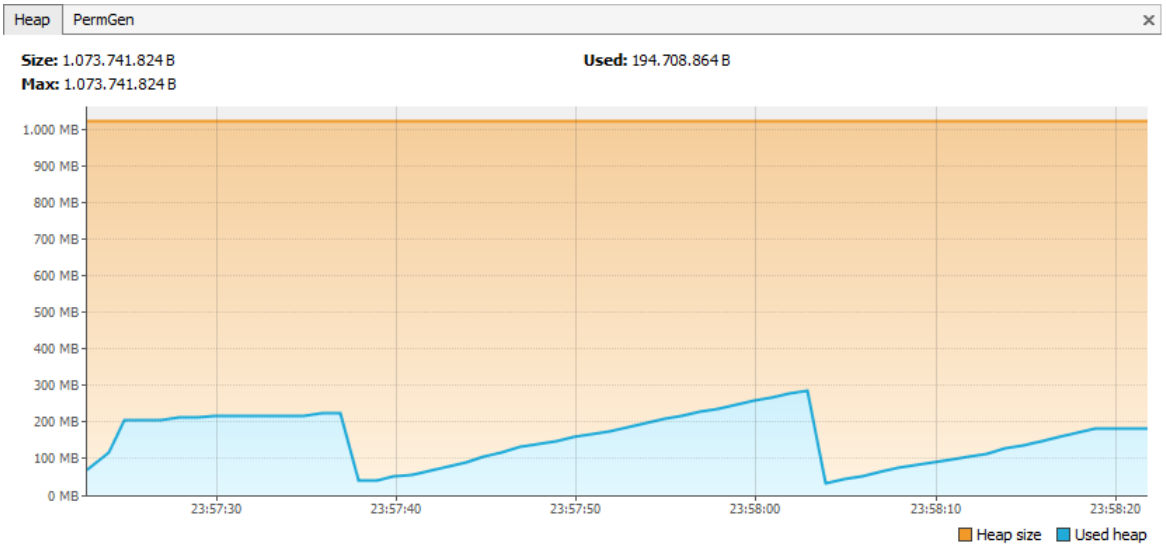

Abbildung 2: Auswertung der Performance ohne Fehler

8 Sonstiges

8.1 Zusätzliche Features

- **Sortierung**
  Ein Feature, wozu wir in der Implementierungsphase nicht gekommen sind, weil es uns zu aufwendig vorkam, ist die Sortierung der Tabellen durch Klicken auf die Köpfe. Dies wurde jetzt realisiert. Es wurde ein TableRowSorter im Wahlvergleichs-, Tabellen- und Berichtsfenster eingefügt und konfiguriert.

- **Debug Klasse**
  Während der Validierungsphase hielten wir es für notwendig strukturiert mit Debug Meldungen umzugehen. Aus diesem Grund wurde eine Debug Klasse implementiert, die dazu dient Debug Meldungen aktivieren und deaktivieren zu können. Es können außerdem unterschiedliche Debug Levels verwendet werden um den Überblick zu behalten.

8.2 Änderungen

- **Webview**
  Alle Webviews (About, Handbuch, Lizenz) wurden durch JFrames ersetzt, weil es unter Linux zu Komplikationen kam. Jetzt sind die Fenster auch unter Linux lauffähig.

- **Simulation des negativen Stimmgewichts**
  Die Simulation des negativen Stimmgewichts konnte während der Implementierungsphase nicht realisiert werden und die Entscheidung, ob eine Implementierung weiter verfolgt werden sollte, wurde auf die Validierungsphase verlegt. Nach erneuten Bemühungen wurde beschlossen, dass die Simulation des negativen Stimmgewicht nicht umgesetzt werden kann. Deswegen wird die Simulation weder weiter implementiert noch getestet.

- **Wahlgenerator**
  Es gibt nur noch einen Wahlgenerator. Es ist somit nicht mehr nötig eine abstrakte Oberklasse für diesen Wahlgenerator zu haben. Diese Oberklasse wurde daher entfernt.

- **Handbuch**
  Es wurde zusätzlich noch ein Handbuch erstellt, welches dem Benutzer helfen soll, sich in dem Programm zurecht zu finden. Die Texte wurden aktualisiert und es wurden Bilder von der aktualisierten Oberfläche hinzugefügt.

- **Chronik**

Die Chronik hat einige Veränderung erfahren. Durch JUnit-Tests wurden Entwurfsfehler festgestellt, die sich bis zur Validierungsphase unentdeckt blieben. Diese Fehler wurden erfolgreich behoben.

9 Ausblick auf die Endphase des Projekts

Die Entwicklung dieser Anwendung im Rahmen des Moduls Praxis der Softwareentwicklung ist nach dieser Phase abgeschlossen. Es folgt noch die interne Abnahme sowie die Abschlusspräsentation. Für diese Zeit wird Manuel Olk der Gruppenverantwortliche sein.

Diese Anwendung wird nach der Abschlusspräsentation auf einem Hosting-Dienst für Software-Entwicklungsprojekte im Internet unter der GPL Lizenz der Öffentlichkeit zur Verfügung gestellt.