

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	1
2	Globale Testfälle und Szenarien	1
2.0.1	Import-/Exportverhalten	2
2.0.2	Korrekte Berechnung der Sitzverteilung	2
2.1	GUI-Test-Plan	3
2.1.1	Fehler der GUI	4
3	Im Pflichtenheft genannte Qualitätsanforderungen	5
4	Unit Tests	5
4.1	Wahlvergleich	6
4.2	Model	6
4.3	Mandatsrechner	7
4.4	Wahlgenerator	8
4.5	Import/Export	8
5	Testabdeckung	8
6	Codeanalyse	9
7	Performanceanalyse	9
8	Sonstiges	10
8.1	Zusätzliche Features	10
8.2	Änderungen	11
9	Ausblick auf die Endphase des Projekts	11

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 EcEmma 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.

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

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. Gesamtweitstimmennanzahl in D)

Es wird ein Fehler ausgegeben.

/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 Erststimmennanzahl geändert werden, dann muss berechnet werden
- Es können alle Erststimmennanzahlen 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 Auslösung ausgegeben
- bleibt zu testen, ob überhaupt gelöst 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 NullPointerException 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 verwendet wurde. 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 dem Wahlgesetz 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

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 betrug.

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 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.

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 Wahl des Direktmandate 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 in den Bundestag ein, die letzte nicht.

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 die 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 anfangs selbst eine Berechnung der Wahl durchführen und das Ergebnis sozusagen als Orakel mit dem berechneten Ergebnis des Mandatsrechner 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. 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. Und dann entweder 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. Daher gibt es den *doubleImportExportTest()*. Hier wird ein Wahlergebnis Importiert, Exportiert, daraufhin wird die neu erstellte Datei erneut importiert und exportiert. Die relevanten Fehler 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 eine fehlerhafte Wahlergebnis, als auch Wahlbewerber-Datei importiert. In diesen Fällen, wird eine Exception geworfen.

Zum Schluss muss überprüft werden, ob generierte Wahlen exportiert und importiert werden können. Dies wird mit *exportGenerierteWahl()* getestet.

5 Testabdeckung

Die Testabdeckung unseres Codes haben wir durch den Eclipse-PlugIn EclEmma gemacht. Getestet wurde jeglicher Code der nichts mit der Visualisierung des Programmes zu tun haben. Aus diesem Grund wurde das GUI-Paket und einige Teile anderer Pakete, wie z.B. Berichtsfenster, nicht mit JUnit getestet.

Folgende Blockabdeckungen konnten erreicht werden (in Prozent):

- Chronik : 80,9

- Config: 46,9
- ImportExport: 90,5
- Mandatsrechner: 83,6
- Model: 85,4
- 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 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 GridBagConstraints 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:

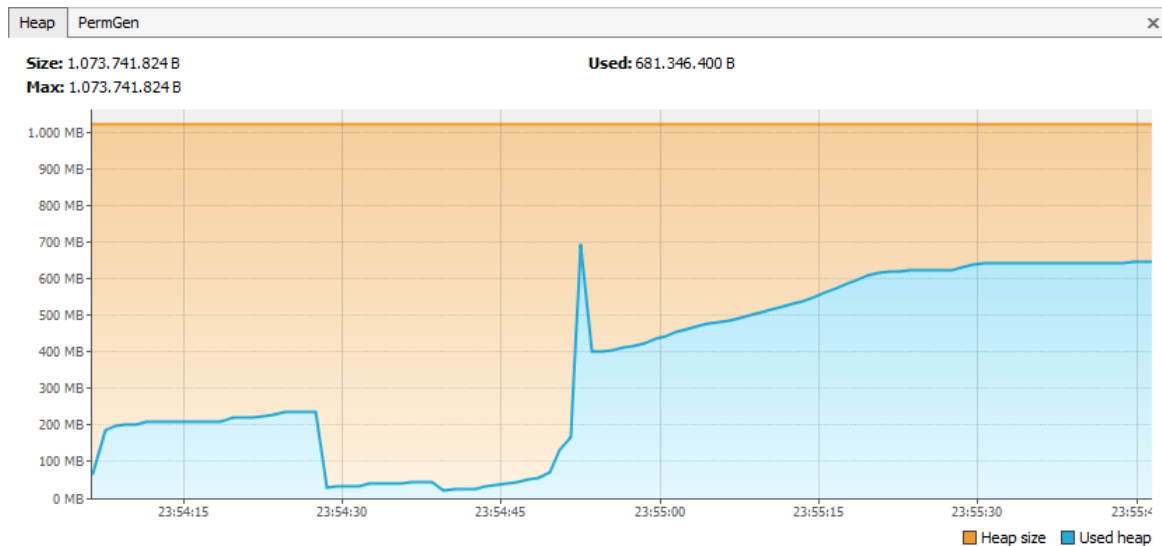


Abbildung 1: Auswertung der Performance mit Fehler

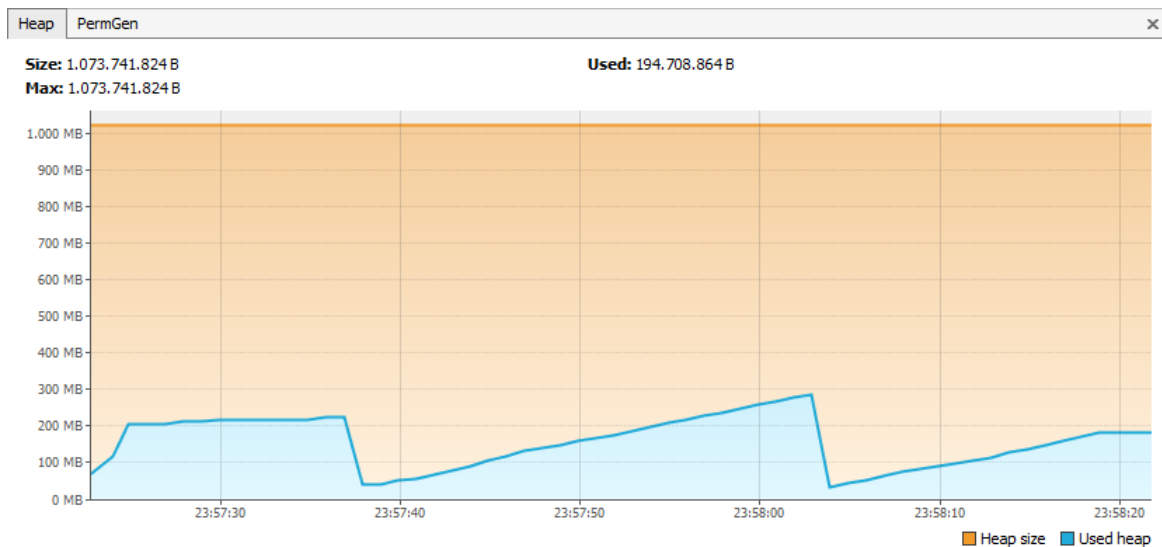


Abbildung 2: Auswertung der Performance ohne Fehler

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, da es sich als doch nicht so schwer herausstellte. 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**

Das negative Stimmgewicht konnte in der Implementierungsphase nicht provoziert werden und wurde deswegen auf die Validierungsphase verlegt. Es wurde dann in der Validierungsphase beschlossen, dass die Simulation des negativen Stimmgewicht komplett fallen gelassen werden soll. Deswegen wird die Simulation weder weiter implementiert noch getestet.

- **Wahlgenerator**

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

- **Handbuch**

Es wurde zusätzlich noch ein Handbuch erstellt, welches den 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 eine Veränderung erlebt. Es wurden Entwurfsfehler festgestellt, die sich bis zur Validierung durch-gepflanzt haben.

Die Entwurfsfehler wurden dank den JUnit Tests festgestellt.

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.