

Entwurf

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
1.1	Einleitung	1
2	Systemmodell	1
3	Klassendiagramm	1
3.1	GUI	1
3.2	Datenhaltung	4
3.3	Import/ Export	5
3.4	Wahlgenerierung	6
3.5	Chronik	8
3.6	Mandatsrechner	9
3.7	Meldung	10
4	Anwendungsfälle	11
5	Sequenzdiagramme	11
5.1	Import	11
5.2	Export	12
5.3	Wahlgenerierung	13
5.4	Paradoxe Wahlgenerierung und Vergleich	15
5.5	Vergleich	16
5.6	Chronik	17

Stimmen zu geben, die Hilfe, und ein Vergleichsfenster, wenn der Vergleich zweier Wahlen aktiv ist.

Ein Wahlfenster hat eine bestimmte Ansicht, diese ist entweder die Bundes-, Landes- und Wahlkreisansicht. Zu jeder von diesen gehört jeweils ein Tabellen-, Diagramm- und Kartenfenster.

Das Tabellenfenster zeigt die Daten der Bundestagswahl-Klasse an (Erst-, Zweitstimmen, Direktmandat,...).

Das Diagrammfenster visualisiert die Sitzverteilungs-Klasse.

Das Kartenfenster visualisiert die Deutschland-Klasse.

Das Tabellenfenster, in welchem Erst- und Zweitstimmen geändert werden können, und die dazugehörige ChangeListener-Klasse werden im Beobachter-Prinzip umgesetzt. Der ChangeListener hört das Tabellenfenster ab und wird bei Veränderungen informiert.

Gesteuert werden alle genannten Klassen durch die GUISteuerungs-Klasse, die die Ansicht und das Vergleichsfenster visualisiert und das gesamte Programmfenster beim Programmstart initialisiert.

Methoden

Tabellenfenster

- **senden(aenderungszeile : String, wert : Integer)**

Das Tabellenfenster informiert den ChangeListener darüber, welche Zeile im Tabellenfenster geändert wurde und was der neue Wert ist.

ChangeListener

- **aktualisiere()**

Der ChangeListener passt seine Attribute, nachdem er von dem Tabellenfenster informiert wurde, an, diese repräsentieren den zuletzt geänderte Tabelleneintrag.

- **uebergeben()**

Der ChangeListener übergibt, nachdem seine Attribute neu gesetzt wurden, diese an die Steuerung, sich dann um die Aktualisierung der internen Daten kümmert.

GUISteuerung

- **initialisiere()**

Der Programmfensterkonstruktor wird aufgerufen, alle Objekte erstellt und mit der Wahl 2013 befüllt.

- **visualisiere(bw : Bundestagswahl)**
Wurde eine neue Bundestagswahl geladen und berechnet, werden alle dazugehörigen Werte und Grafiken im Karten-, Diagramm- und Tabellenfenster geladen.
- **visualisiere(vergleich : Vergleich)**
Hat die Steuerung einen Vergleich zweier Wahlen errechnet und die Daten an die GUISteuerung übergeben, wird das Vergleichsfenster erstellt und mit den Daten des Vergleichsobjektes gefüllt.
- **zurueckSetzen(bw : Bundestagswahl)**
Wünscht der Benutzer einen zuvor geänderten Wert im Tabellenfenster zurückzusetzen, wird die Steuerung informiert.

3.2 Datenhaltung

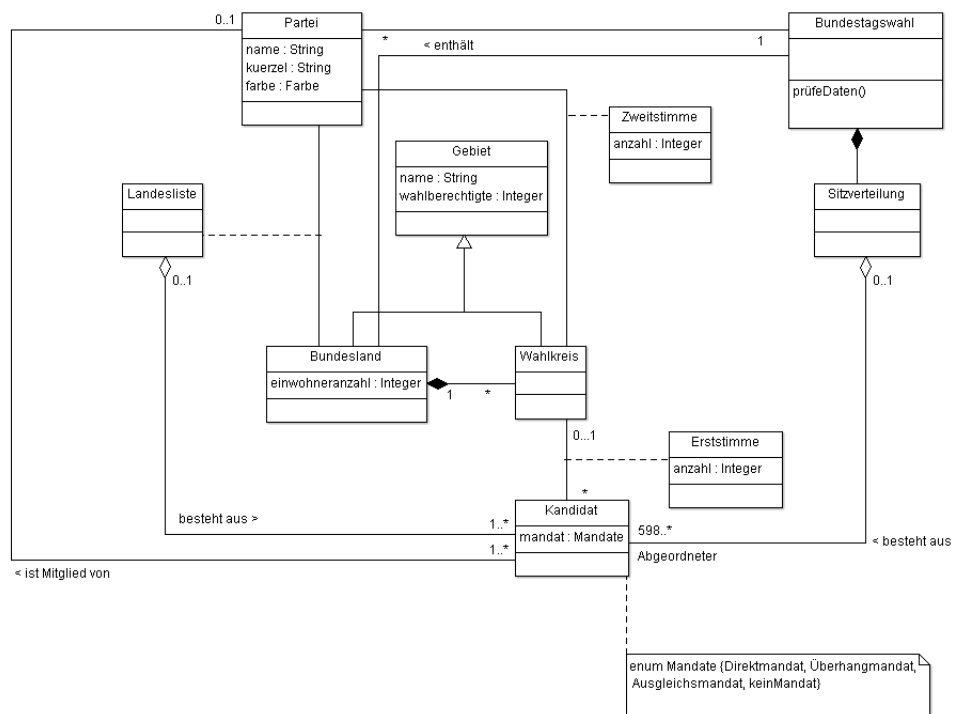


Abbildung 2: Datenhaltungs Komponente

Dieser Ausschnitt des Klassendiagramms stellt den Datenhaltungsteil dar.

3.3 Import/ Export

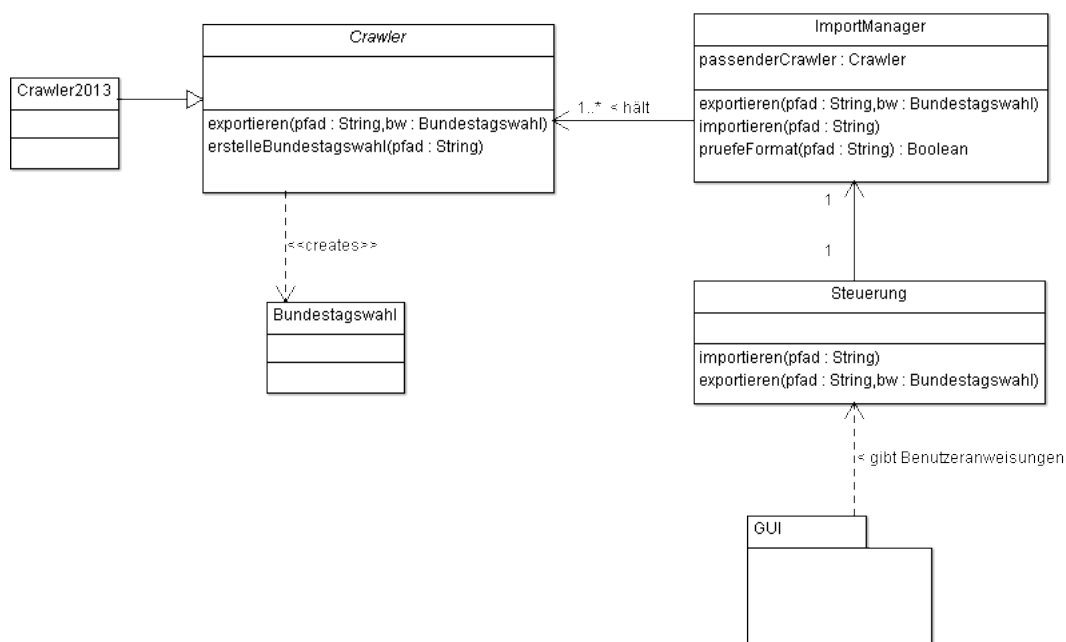


Abbildung 3: Import/Export Komponente

In diesem Klassendiagramm sieht man den Aufbau des Import-/Exportmoduls. Zur Übersichtlichkeit werden die zum Importieren/Exportieren nicht notwendigen Methoden in **Steuerung** und die genaue Struktur hinter **Bundestagswahl** und der GUI ausgeblendet.

Mit unserem Programm wird nur ein vorimplementierter Crawler mitgegeben, der .csv-Dateien, die dem Format der .csv-Datei zur Bundestagswahl 2013 der Bundeswahlleiter-Webseite entsprechen, auswerten kann - dies ist **Crawler2013**. Um jedoch die Möglichkeit zu garantieren, nachträglich weitere Crawler hinzuzufügen, haben wir uns dafür entschieden, eine abstrakte Oberklasse **Crawler** zu verwenden, von der **Crawler2013** erbt, und alle vorhandenen Crawler von der Klasse **ImportManager** halten zu lassen.

Ausgelöst wird der ganze Import- bzw. Exportvorgang durch eine Benutzerinteraktion (z.B. Betätigen des Laden- Knopfs im Menü), worauf die **GUISteuerung** die **Steuerung** anwählt, welche wiederum über den **ImportManager** den geeigneten Crawler bestimmt, der die Informationen ausliest die eigentliche Arbeit ausführt.

Methoden

exportieren(pfad : String, bw : Bundestagswahl)

Schreibt die Bundestagswahl bw in die .csv-Datei, deren Pfad in **pfad** gegeben ist.

importieren(pfad : String)

Liest die .csv-Datei, deren Pfad in pfad gegeben ist, ein, erstellt ein Bundestagswahlobjekt und füllt es.

pruefeFormat(pfad : String) : Boolean

Prüft das Format der Datei, deren Pfad in pfad gegeben ist. Wenn es sich um eine .csv-Datei handelt, die durch einen vorhandenen Crawler erfasst werden kann, wird dieser Crawler als passenderCrawler gesetzt und true zurückgegeben. Falls kein geeigneter Crawler gefunden wird, da die Datei keine .csv-Datei ist oder ihre Struktur nicht korrekt ist, wird false zurückgegeben.

3.4 Wahlgenerierung

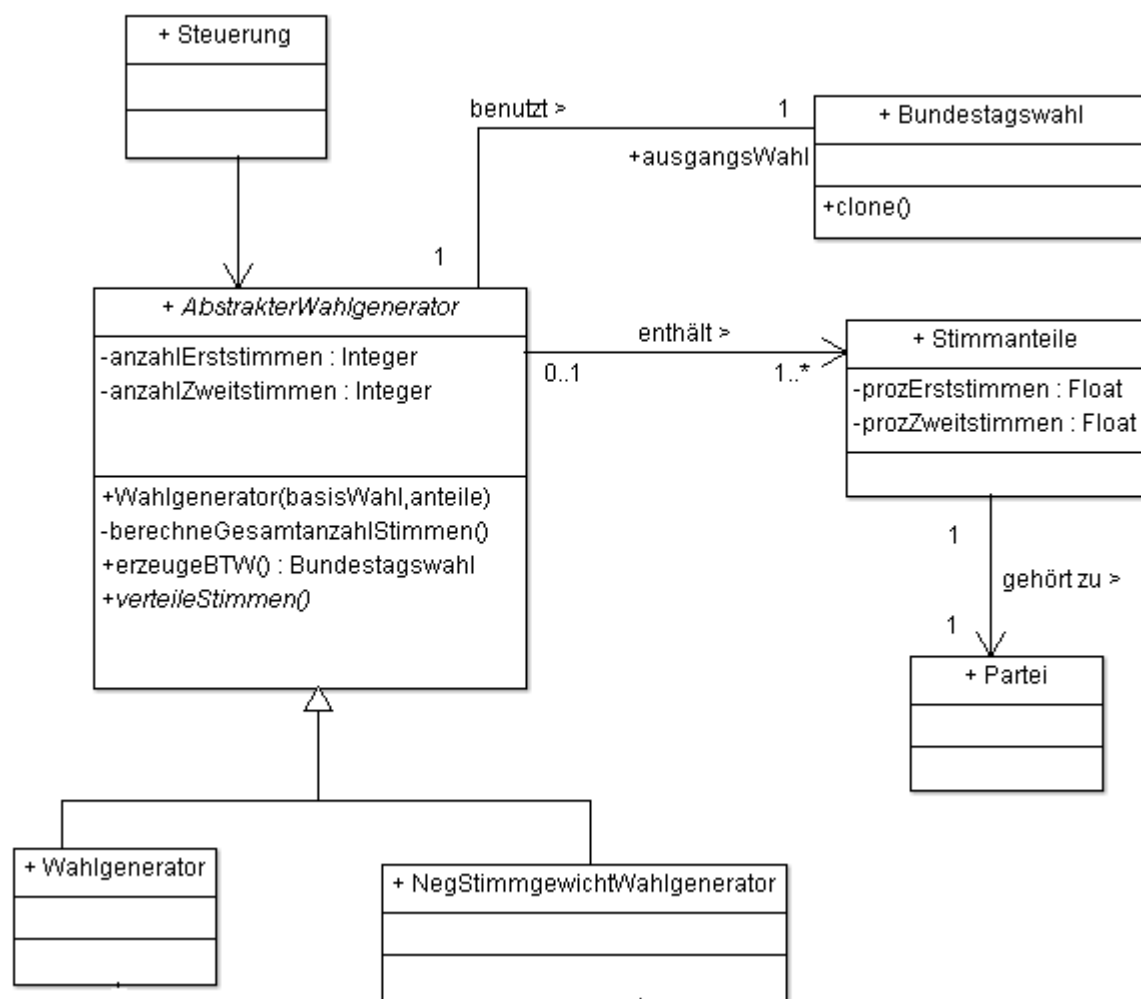


Abbildung 4: Wahlgenerierungs-Komponente

Der obige Ausschnitt des Klassendiagramms zeigt das Wahldatengenerierungs-Modul.

Mit diesem Modul können Bundestagswahlen anhand vorher definierten Stimmanteilen auf Bundesebene generiert werden. Bei diesen Stimmanteilen handelt es sich um eine Liste aller Parteien mit prozentualen Anteilen der Erst- und Zweitstimmen auf Bundesebene. Des weiteren benötigt der Wahlgenerator eine Basis-Bundestagswahl um Daten wie beispielsweise Bundesländer, Wahlkreise und Wahlberechtigte zur Verfügung zu haben.

Beim generieren einer Bundestagswahl werden die Stimmanteile mithilfe der Anzahl aller Wahlberechtigten in absolute Zahlen für Erst- und Zweitstimmen umgerechnet. Diese Stimmen werden dann zufällig, erst auf alle Bundesländer und dann auf die einzelnen Wahlkreise verteilt.

Dieser Vorgang ist im folgenden noch einmal als Sequenzdiagramm aufgezeigt.

3.5 Chronik

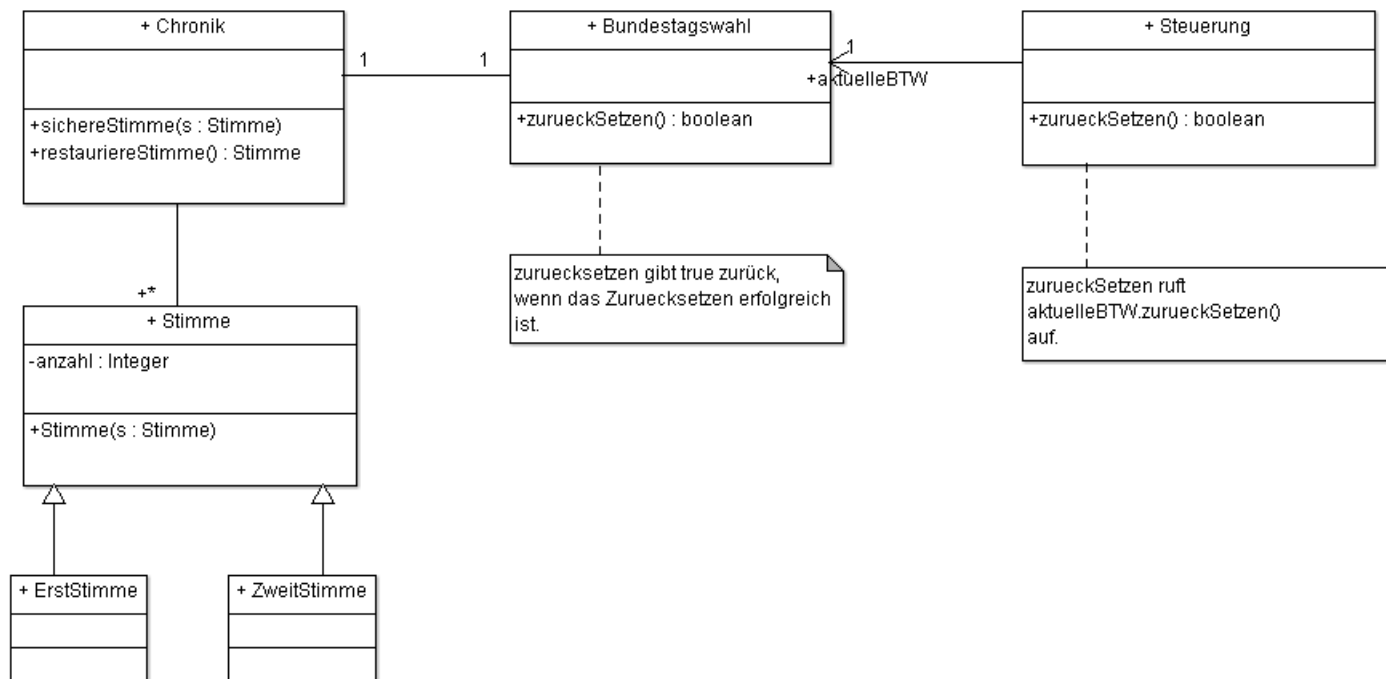


Abbildung 5: Chronik-Komponente

Die Klasse **Chronik** gibt dem Programm die Funktionalität, Veränderungen an den Stimmen rückgängig zu machen. Jede **Bundestagswahl** hat hierbei eine eigene Chronik. **Chronik** wird in dem Konstruktor von **Bundestagswahl** erzeugt, und ist daher in jedem **Bundestagswahl**-Objekt enthalten. Es besitzt eine Menge von **Stimmen**-Objekten. Bei jeder Veränderung wird ein neues **Stimmen**-Objekt angelegt, was die Veränderung widerspiegelt. Die Methode **sichereStimme** wird von dem **Bundestagswahl**-Objekt bei jedem Aufruf von **setzeStimme** aufgerufen.

Methoden

sichereStimme(s : Stimme)

Diese Funktion wird von dem assoziierten **Bundestagswahl**-Objekt innerhalb der *setzeStimme*-Funktion aufgerufen. Falls bereits fünf **Stimmen**-Objekte vorhanden sind, wird das älteste entfernt.

restauriereStimme() : Stimme

Wird von der **Steuerung** über die aktuelle Bundestagswahl mit der Funktion *zurueckSetzen()* aufgerufen und gibt die zuletzt hinzugefügte **Stimme** zurück. Die Bundestagswahl ersetzt dann die aktuelle Stimme mit der restaurierten Stimme.

3.6 Mandatsrechner

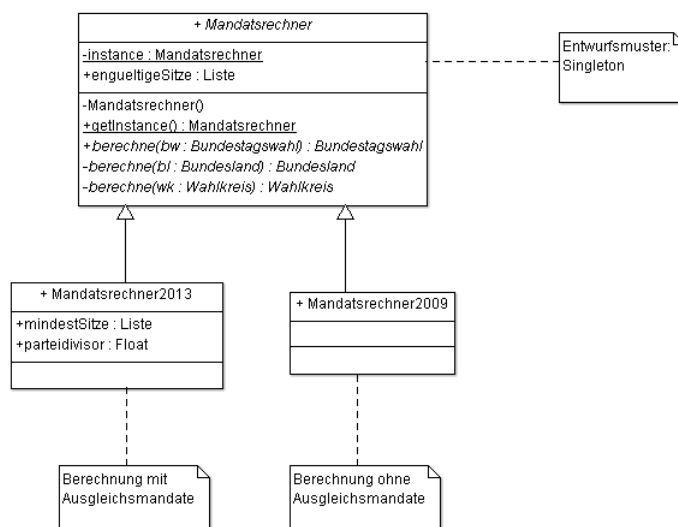


Abbildung 6: Mandatsrechner-Komponente

Die Berechnung der Wahl wird mit Hilfe des **Mandatsrechners** realisiert. Es stehen die Klasse

Mandatsrechner2013, die das Berechnungsverfahren von der Bundestagswahl 2013 benutzt und die Klasse **Mandatsrechner2009**, die das Berechnungsverfahren von der Bundestagswahl 2009 benutzt zur Verfügung. Beide Klassen erben von der abstrakten Klasse **Mandatsrechner**. Dadurch besteht die Möglichkeit, weitere Berechnungsverfahren in späteren Versionen zu dem Programm hinzuzufügen. Da nur ein Objekt von dem **Mandatsrechner** gebraucht wird, wird das Entwurfsmuster Einzelstück eingesetzt. Deswegen hält die Klasse einen privaten Konstruktor. Die abstrakten Methoden `berechne()` werden überladen, damit sie durch ihre Eingabeparameter spezifiziert werden. Diese werden dann in den Unterklassen je nach Wahlgesetz angepasst. Neben der Berechnung wird ein Bericht über die Sitzverteilung erstellt, der zum Nachvollziehen der Sitzverteilung helfen soll. Methoden

berechne(wk : Wahlkreis) : Wahlkreis

Es werden die Stimmen aus den jeweiligen Wahlkreis ausgewertet. Dabei wird der Wahlkreissieger bestimmt und die Anzahl der Zweitstimme von jeder Partei. Die Auswertung wird danach wieder in das Wahlkreis-Objekt geschrieben.

berechne(bl : Bundesland) : Bundesland

Um das Bundesland zu berechnen, müssen vorher alle Wahlkreise berechnet werden. Deswegen werden alle Wahlkreise, die ein Bundesland hält, neu berechnet. Die Berechnung der einzelnen Bundesländer erfolgt parallel. Nachdem die berechneten Wahlkreise im Bundesland gespeichert wurden, wird das Bundesland berechnet. Hier wird das Verhältnis der Parteien im Bundesland berechnet, damit später klar ist wie viele Sitze eine Partei in diesem Bundesland bekommt. Diese Ergeb-

nisse werden, wie beim Wahlkreis, im Bundeslandobjekt gespeichert und danach zurückgegeben.

berechne(bw: Bundestagswahl) : Bundestagswahl

Diese öffentliche Methode berechnet zuerst alle Bundesländer die sich in der Klasse befinden. Nachdem alle Bundesländer erfolgreich berechnet wurden, wird die endgültige Sitzverteilung nach dem jeweiligen Wahlgesetz berechnet. Die Sitzverteilung wird dann in dem Bundestagswahlobjekt gespeichert. Das Bundestagswahlobjekt wird danach wieder an die Steuerung zurückgegeben.

erstelleBericht(Zeile : String)

Während der Berechnung wird nebenbei eine Sitzverteilungsbericht verfasst, der beschreiben soll, wie die Sitzverteilung entstanden ist. Dabei wird die Methode immer aufgerufen, wenn eine Partei einen Sitz in der Sitzverteilung bekommen hat. Dies wird dann mit einer Zeile im Bericht protokolliert.

3.7 Meldung

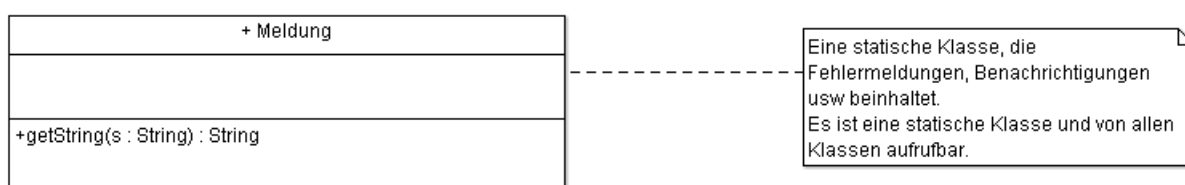


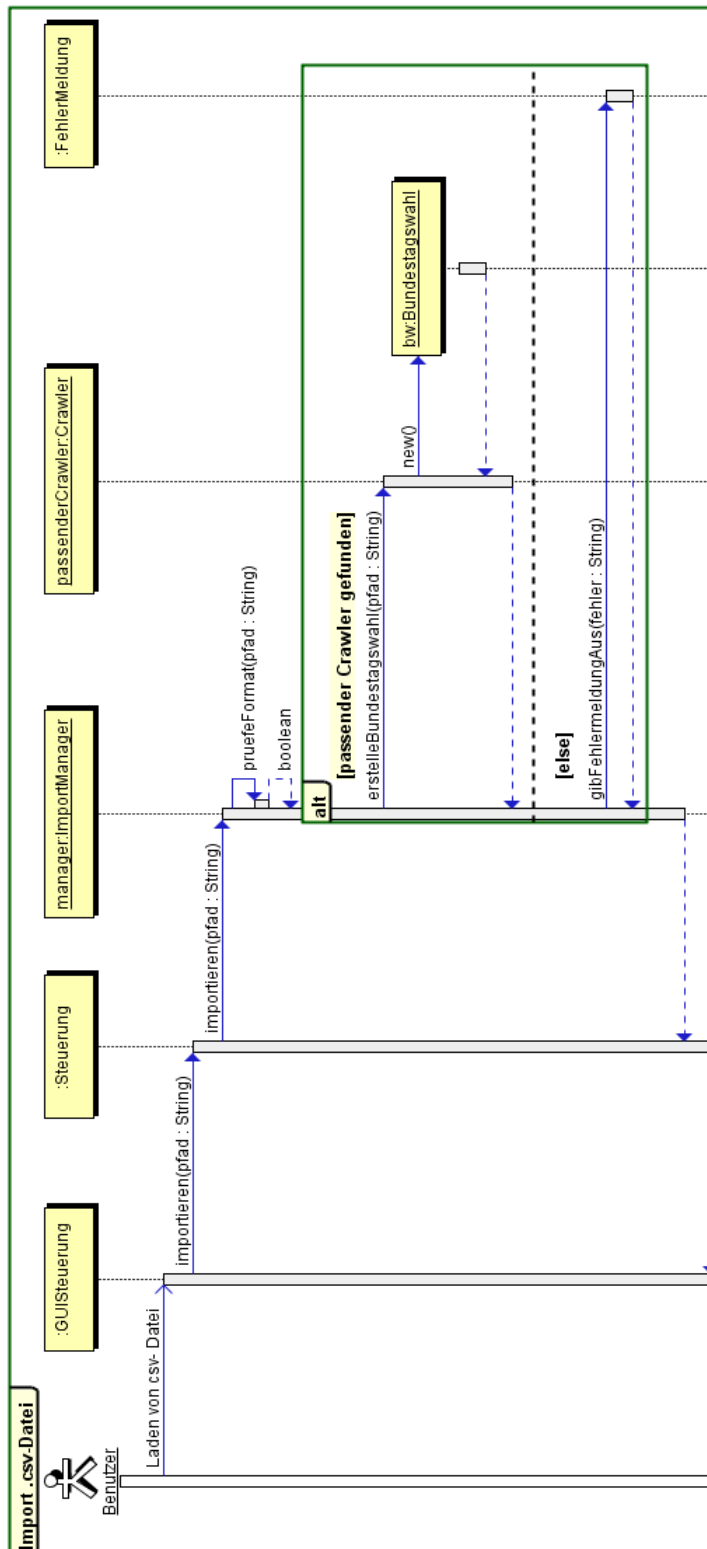
Abbildung 7: Meldungs-Klasse

Die Klasse **Meldung** ist verantwortlich für Fehlermeldungen, Benachrichtigungen und Fenstertexte. Es ist eine statische Klasse. Die Funktion *getString* gibt zu einem gegebenen Schlüssel ein String zurück. Die Strings dieser Klasse werden in einem externen Textdokument gelagert.

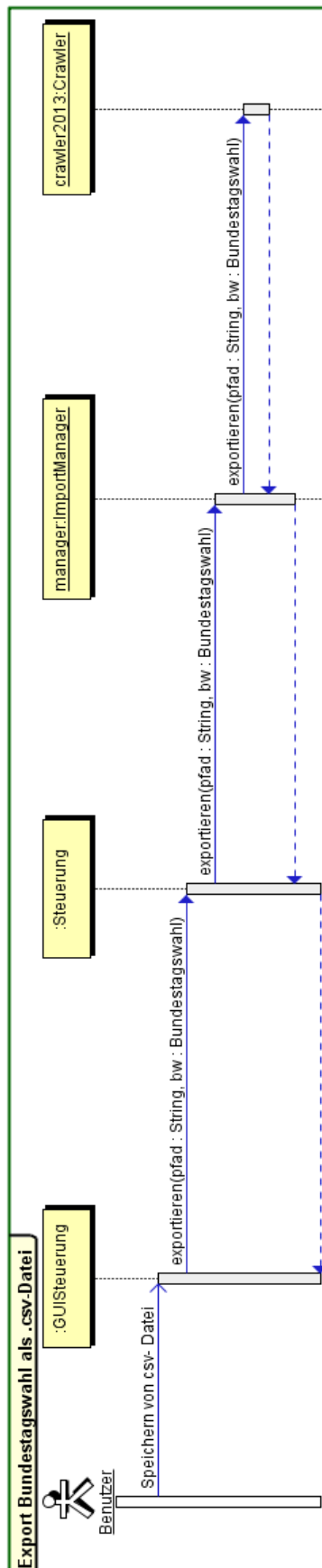
4 Anwendungsfälle

5 Sequenzdiagramme

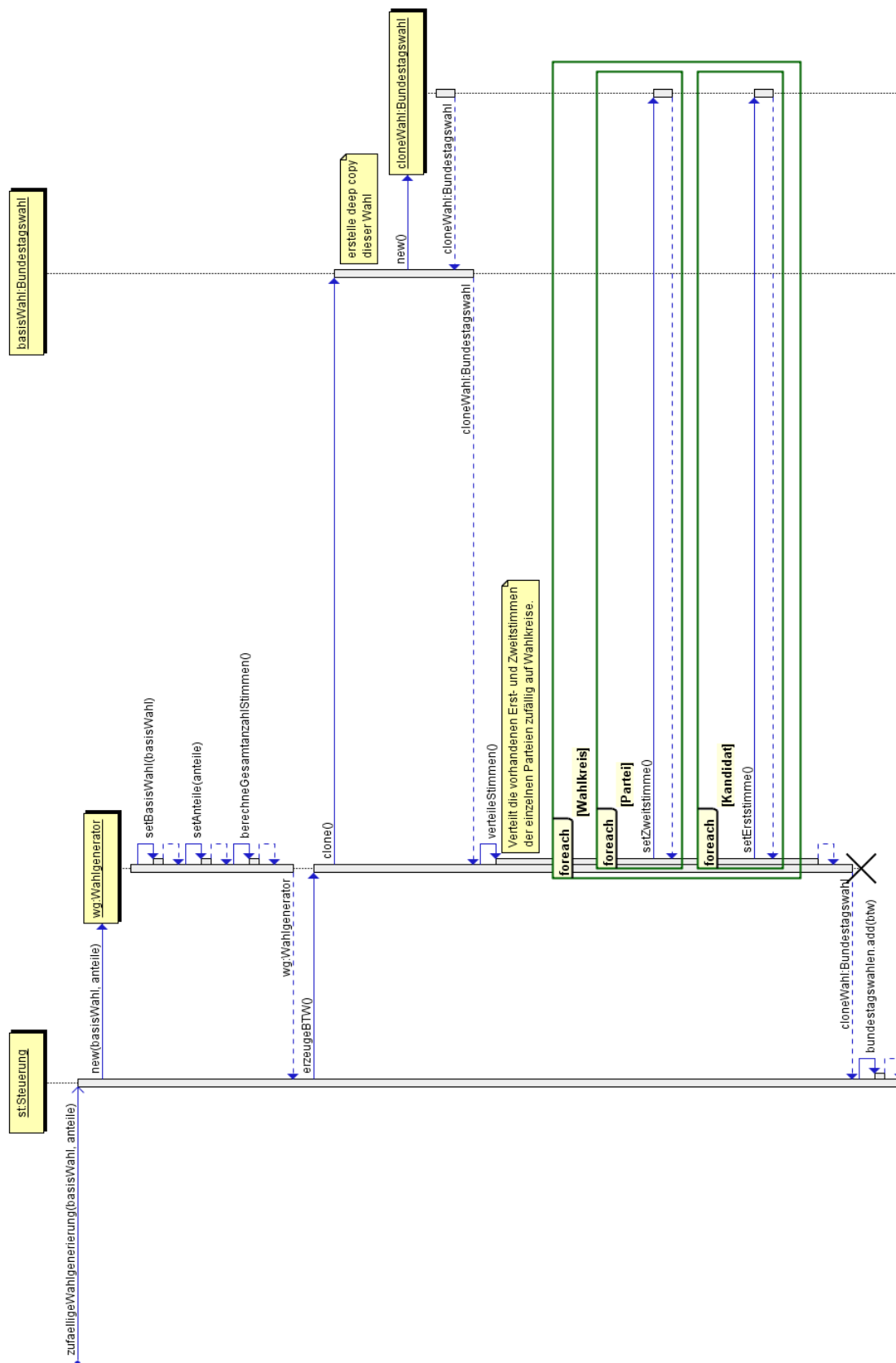
5.1 Import



5.2 Export



5.3 Wahlgenerierung

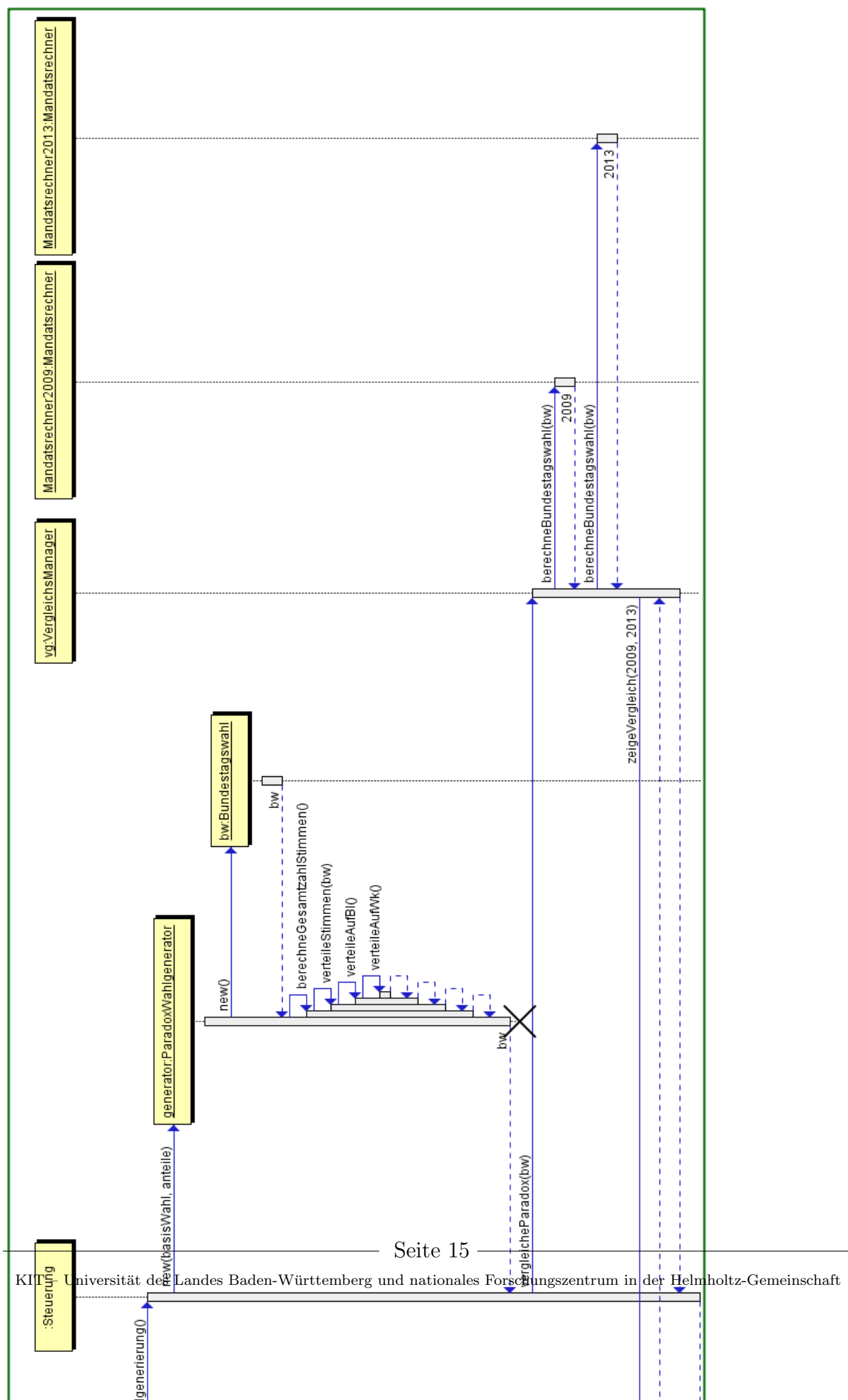


Beim Ge-

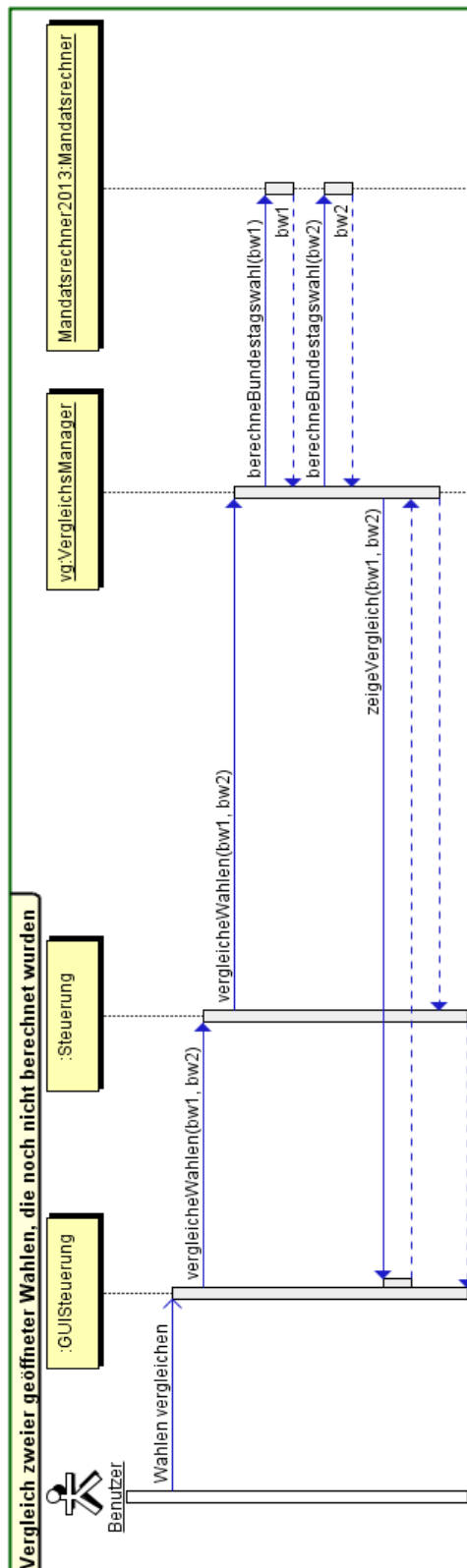
nerieren einer Bundestagswahl werden die Stimmanteile mithilfe der Anzahl aller Wahlberechtigten in absolute Zahlen für Erst- und Zweitstimmen umgerechnet. Diese Stimmen werden dann zufällig auf die einzelnen Wahlkreise verteilt.

Dieser Vorgang ist im folgenden noch einmal als Sequenzdiagramm aufgezeigt. Neben der zufälligen Verteilung von den Stimmen auf alle Bundesländer und Wahlkreise gibt es noch einen Wahlgenerator der Stimmverteilungen erzeugt, die nach dem Wahlgesetz von 2009 zu negativem Stimmgewicht durch Überhangsmandate führen. Diese werden benötigt um in einer Vergleichsansicht das negative Stimmgewicht zu verdeutlichen.

5.4 Paradoxe Wahlgenerierung und Vergleich



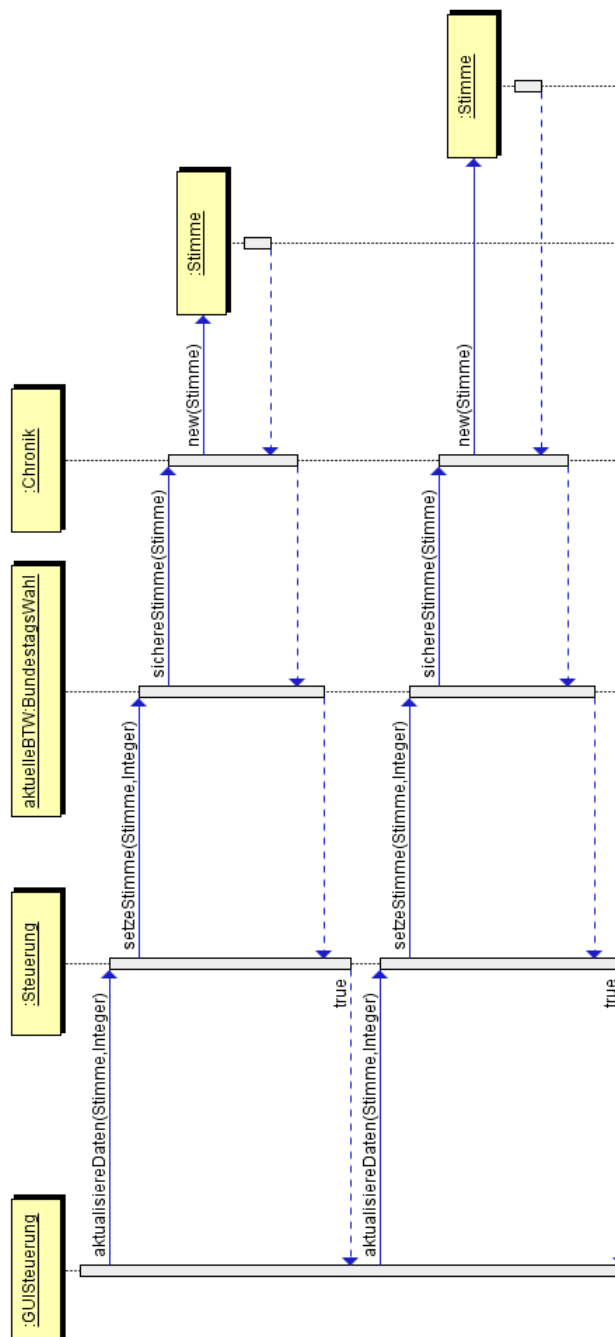
5.5 Vergleich



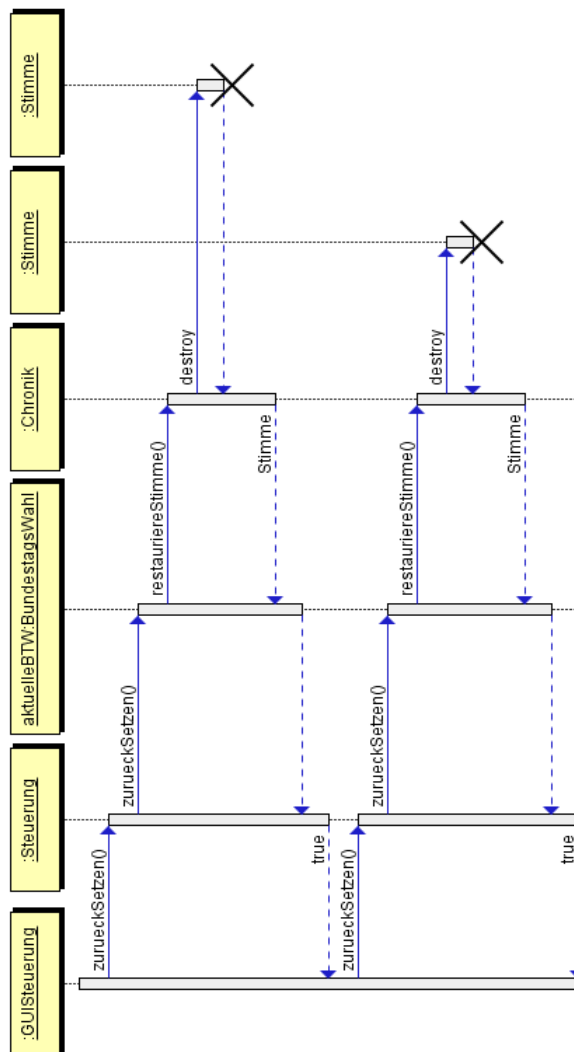
In diesem Sequenzdiagramm wird gezeigt, was passiert, wenn der Benutzer zwei Tabs geöffnet hat und einen Vergleich starten will.

5.6 Chronik

- Veränderung an den Stimmen



- Restaurieren einer Stimme



Stimmen werden in der Chronik Stack-artig zurückgegeben. Sobald eine Bundestagswahl rückgängig gemacht wurde, wird die neue Bundestagswahl als Rückgabewert zurückgegeben.

- Beispielszenario