Verteiltes Genom Browsing

Projektspezifikation

1. Dezember 2015

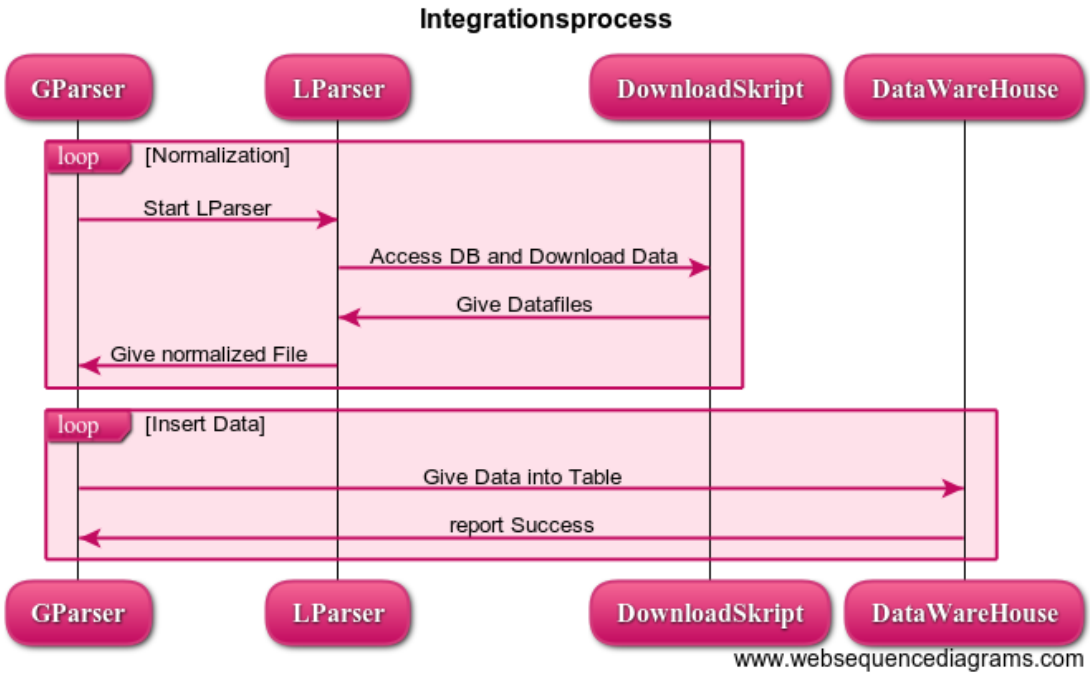
Inhaltsverzeichnis

1 Integration

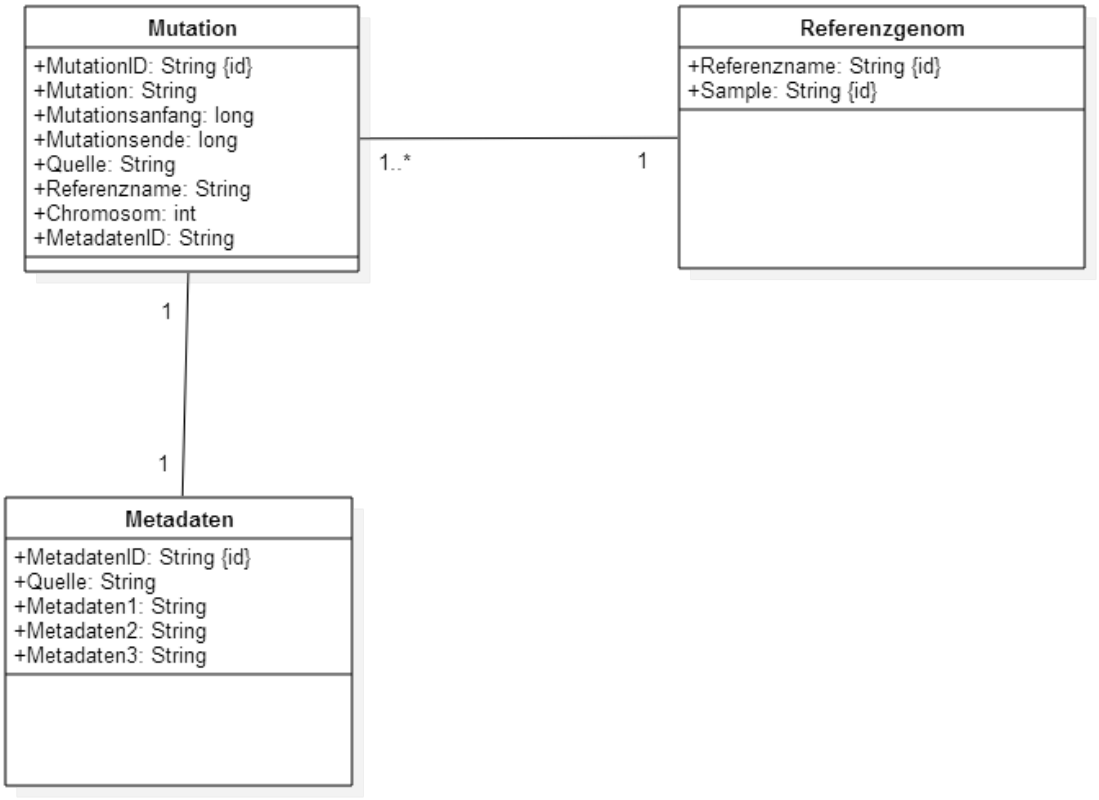
1.1 Integrationsprozess

1.1.1 Ablauf der Integration

1.1.2 Sequenzdiagramm

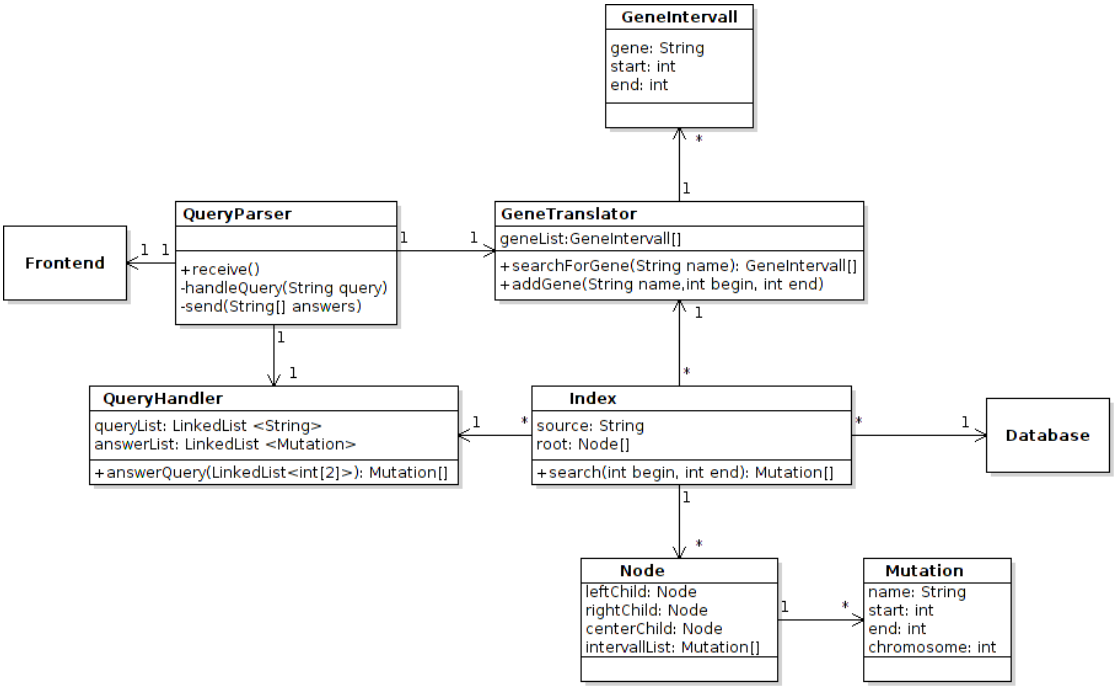


1.2 Datenbankentwurf



2 Middleware

2.1 UML-Diagramm



2.2 Schnittstellenspezifikation: Middleware - Frontend

2.3 Unit-Tests

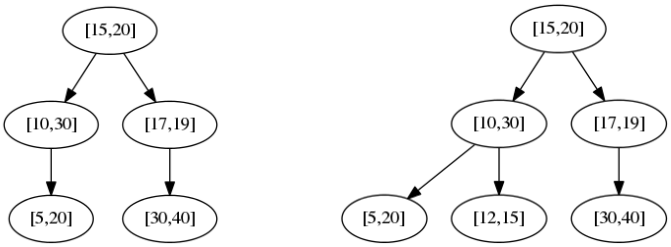
2.3.1 QueryHandler

2.3.2 GeneTranslator

2.3.3 Intervallbaum

1. Intervalle einfügen:
Das Intervall muss im Baum an der richtigen Stelle eingefügt werden und der Baum muss gegebenenfalls neu balanciert werden (z.B. wie ein AVL-Baum).

Bsp.: Einfügen des Intervalls [12,15]



2. Intervall mit Startpunkt > Endpunkt einfügen:
Wenn ein Intervall (S,E) mit S>E eingefügt wird, dann sollte unser Programm eine Fehlermeldung ausgeben und darauf hinweisen, dass die Grenzen für das Intervall nicht korrekt sind.

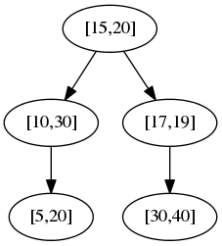
Bsp.: Einfügen des Intervalls [20,10] in einen beliebigen Baum.

3. Intervall mit Start- bzw. Endpunkt außerhalb des betrachteten Zahlenbereichs:
Wenn ein Intervall in dem Baum eingefügt werden soll, das teilweise oder vollständig außerhalb unseres Zahlenbereichs liegt (Länge des Genoms), dann muss es eine Fehlermeldung geben, die dem Nutzer mitteilt, dass der gültige Zahlenbereich überschritten wurde.

Bsp.: Einfügen des Intervalls [-5,7] in einen beliebigen Baum.

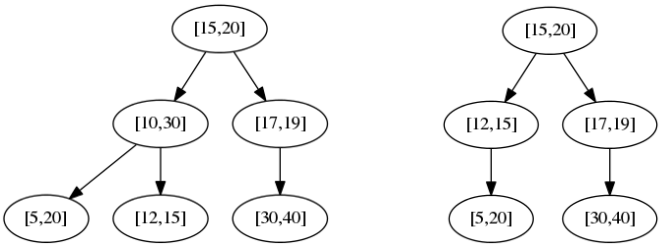
4. Schon vorhandenes Intervall einfügen:
Duplikate sollen von unserem Baum nicht gespeichert werden, d.h. es wird kein neuer Knoten hinzugefügt, sondern die Informationen (bei uns also Pointer auf Dateien) des neuen Knotens müssen im bereits vorhandenen Knoten mitgespeichert werden.

Bsp.: Einfügen des Intervalls [15,20] in den folgenden Baum



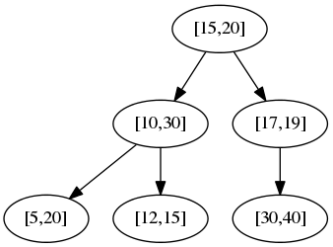
5. Intervalle löschen:
Das Intervall soll aus dem Baum gelöscht werden und der Baum muss falls nötig wieder neu balanciert werden.

Bsp.: Löschen des Intervalls [10,30]



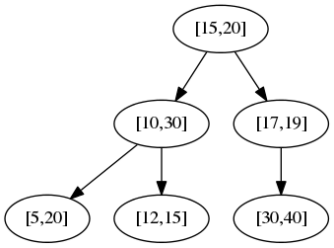
6. Nicht vorhandenes Intervall löschen:
Versucht der Nutzer ein nicht im Baum gespeichertes Intervall zu löschen, so muss der Baum unverändert bleiben. Es kann zusätzlich darauf hingewiesen werden, dass der Knoten nicht existiert.

Bsp.: Löschen des Intervalls [50,60] aus dem folgenden Baum



7. Suche nach vorhandenem Intervall:
Bei der Suche sollen alle Intervalle ausgegeben werden, die das gesuchte Intervall in irgendeinem Punkt überlappen.

Bsp.: Suche im folgenden Baum



Suche [4,5] ⇒ gib [5,20] aus
Suche [25,35] ⇒ gib [10,30] und [30,40] aus
Suche [20,20] ⇒ gib [15,20] und [5,20] aus

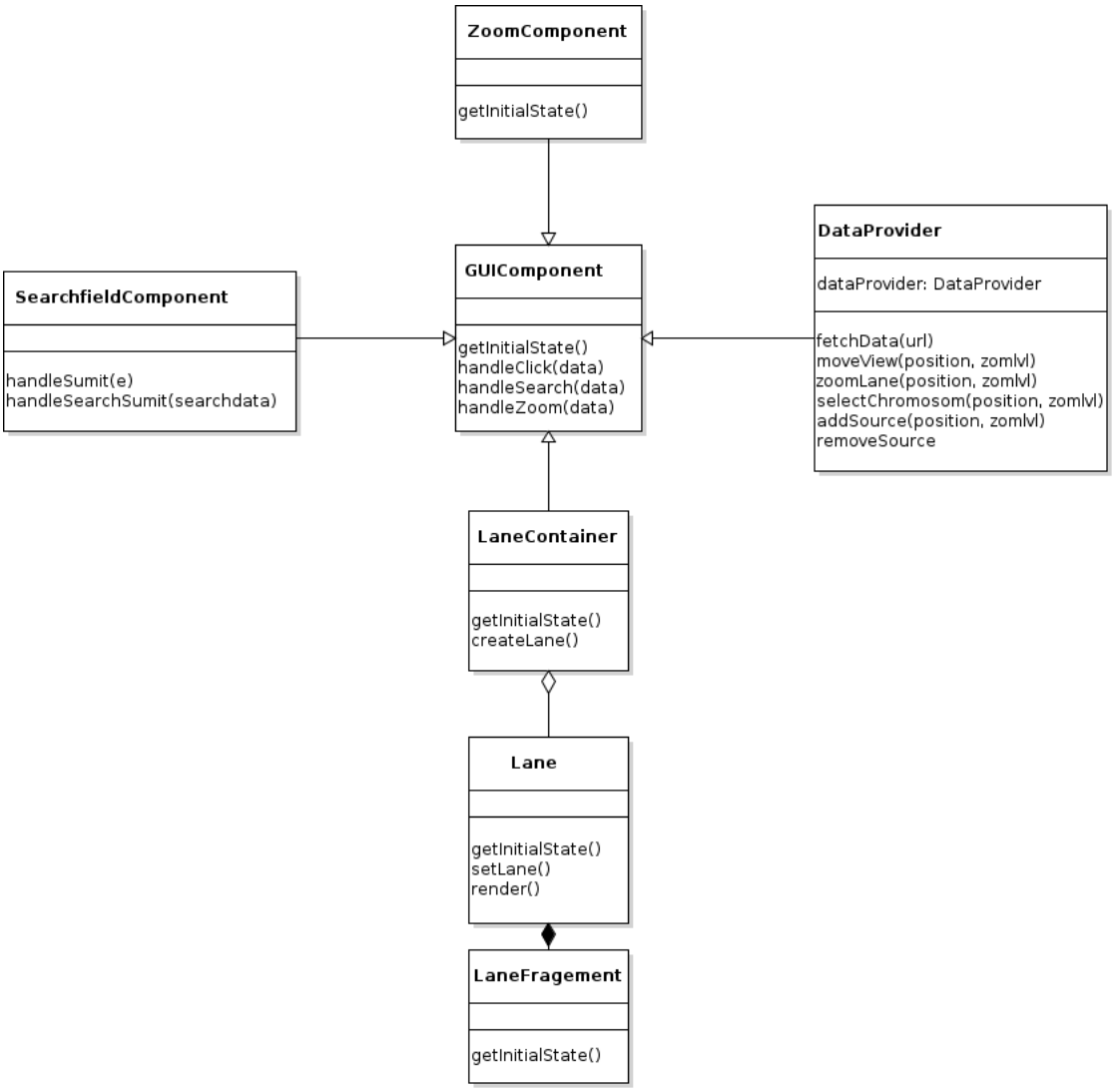
2.4 Stresstests

Beim Stresstest geht es darum das System so stark auszulasten wie mglich, um zu sehen, ob die Antwortzeit für einzelne Clients, die Anfragen senden bei oer Last merklich länger wird.
Hierfür werden Anfragen konstruiert, die so viel Arbeitslast wie mglich erzeugen. In diesem Fall soll jeder Client Anfragen stellen, die in allen Quellen suchen; Dadurch lastet jeder Client mit einer Anfrage alle VM's aus. Weiterhin soll jeder Client eine Gensuche anfragen, damit alle Anfragen den Extra-Schritt ber den GeneTranslator machen mssen, was dazu führt, dass alle System-Komponenten getestet werden. Auerdem kann so ein mglicher Flaschenhals in Form der searcForGene()-Funktion entdeckt werden.
Da die Software eine sehr spezialisierte Suchmaschine ist, wird der Kreis an Nutzern, die gleichzeitig den Webdienst in Anspruch nehmen relativ berschaubar bleiben.
Die Idee ist deshalb in Erfahrung zu bringen, was der Kunde fr eine Nutzer- menge für realistisch ählt. Wir setzen ersteinmal eine Zahl von 20 Clients fest. Diese kann so weit erhht werden, bis eine merkliche Verlangsamung des Systems eintritt.
Zusammenfassung:
20 Clients
Gennamen-Anfragen
Anfragen für alle Quellen

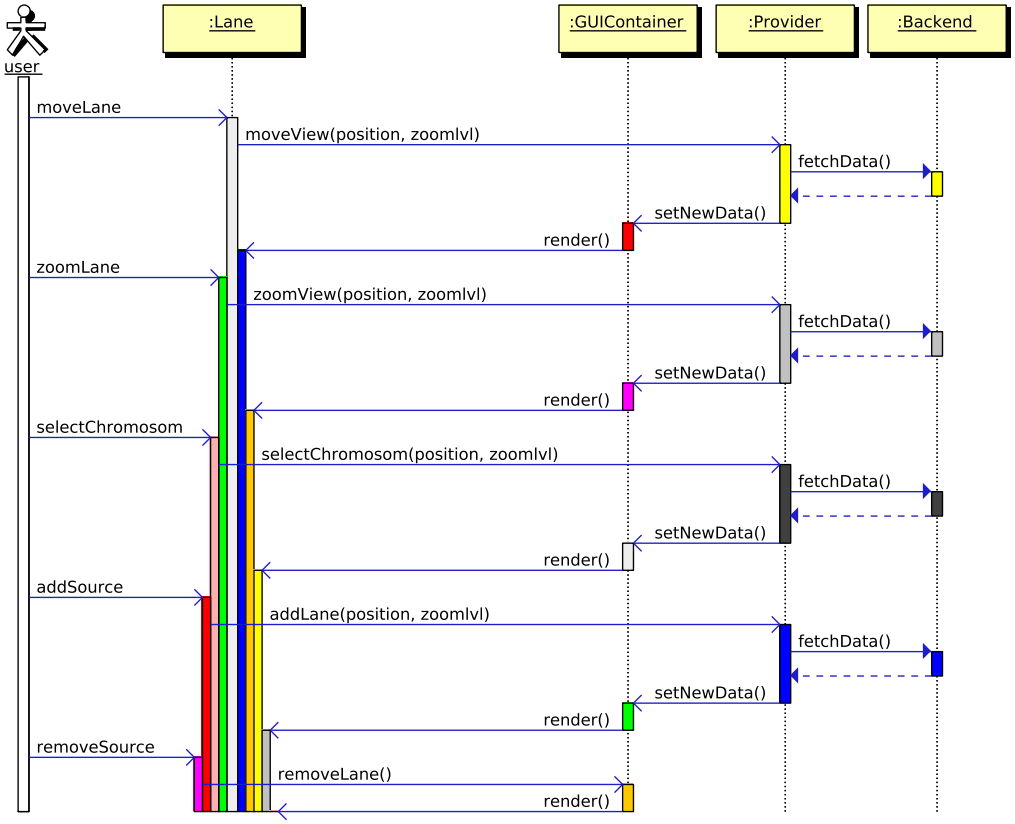
3 Frontend

3.1 Mock-Ups der Benutzerschnittstelle

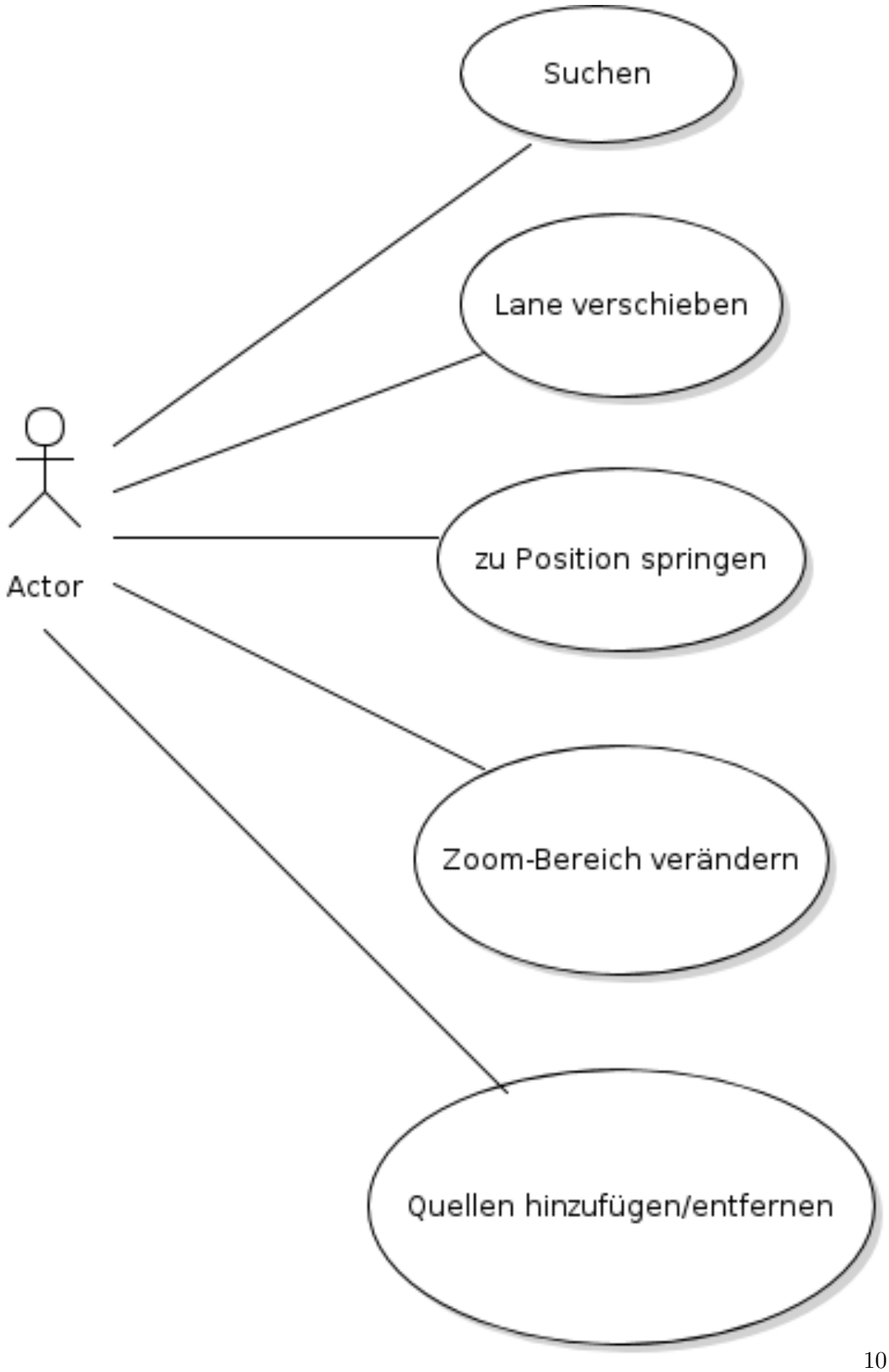
3.2 Klassen-Diagramm



3.3 Sequenzdiagramm



3.4 Use Cases


Suchen
Lane verschieben
zu Position springen
Actor
Zoom-Bereich verändern
Quellen hinzufügen/entfernen