



HOCHSCHULE KONSTANZ TECHNIK, WIRTSCHAFT UND GESTALTUNG
UNIVERSITY OF APPLIED SCIENCES

Technische Dokumentation Bussimulation

Anwendung der Betrieblichen Systemforschung

HTWG Konstanz

**Prof. Dr. Grütz
Prof. Dr. Hedtstück**

Inhaltsverzeichnis

Inhaltsverzeichnis	I
Abbildungsverzeichnis	II
Tabellenverzeichnis	II
1 Programmbeschreibung	1
1.1 Java Pakete	1
1.2 Optimierung	3
1.3 Simulation	4
1.4 Kerneloptimierung	6
1.4.1 Ideenentwicklung	6
1.4.2 Technische Umsetzung	6
1.5 Business Logik	13
1.6 Präsentation	16
1.7 Speichern/Laden	17
1.7.1 Speichern	17
1.7.2 Laden	18
1.7.3 XML-Datei	18
1.8 GUI-Optimierung	26
1.8.1 Symbolleiste	26
1.8.2 Kontextmenü	28
1.8.3 Zeitmessung	29
1.8.4 Abbruchmöglichkeit der Simulation und Optimierung	29
1.8.5 Ergebnisdarstellung	31
1.9 Logger	33
1.10 Hilfsklassen	34
2 Projektstatus	35
2.1 Übersicht	35
2.2 Verbesserungsmöglichkeiten	39
2.2.1 Layout	39
2.2.2 Positionieren von Elementen	39
2.2.3 Klasse Strassennetz	39
2.2.4 Undo/Redo	40
2.2.5 Optimierung	40
2.2.6 Refactoring Business Klassen	40
2.2.7 Passagiergenerator	40
2.2.8 Umsteigen von Passagieren	41
2.2.9 Simulationsablauf steuern / Simulationsgeschwindigkeit	41
2.2.10 Anzeige des Simulationsablaufs	41
2.2.11 Eigenschaften von Bussen und Streckenabschnitten	41
2.2.12 Auswählen der Buslinie für einen Passagier	41
2.2.13 Manueller Personenzufluss an Haltestellen	42
2.2.14 Weitere Anforderungen	42
3 Anhang	45

Abbildungsverzeichnis

Abbildung 1-1: Java Pakete	2
Abbildung 1-2: Klassen für Optimierung	3
Abbildung 1-3: Klassen für Simulation	4
Abbildung 1-4: Transformation der Matrix.....	10
Abbildung 1-5: Listener Klassen	13
Abbildung 1-6: Klassen für Business Logik.....	15
Abbildung 1-7: Klassen für Präsentation.....	16
Abbildung 1-8: XML-Datei als Baumdarstellung	19
Abbildung 1-9 Schnellklickfunktion	26
Abbildung 1-10 Kontextmenü	29
Abbildung 1-11 Optimierung stoppen	30
Abbildung 1-12 bisherige Lösung der Ergebnisse	31
Abbildung 1-13 bisherige Ergebnis-Informationen.....	31
Abbildung 1-14 Auswertung der Ergebnisse in einer Tabelle	33

Tabellenverzeichnis

Tabelle 1: Implementierte Features	39
Tabelle 2: Weitere Features	44

1 Programmbeschreibung

Im folgenden Kapitel wird die Implementierung des Programms beschrieben. Das Programm ist in einer zweischichtigen Architektur implementiert, d.h. es gibt eine Schicht für die Business Logik und eine Schicht für die Präsentation. Als Programmiersprache wurde Java gewählt.

1.1 Java Pakete

Die Bussimulation 2005 v1.1 ist in mehrere Pakete unterteilt. Dies dient der Übersichtlichkeit und es ergibt sich auch eine bessere Trennung der Verantwortlichkeiten. In der Anwendung gibt es folgende Pakete:

- `de.fh_konstanz.simubus.controller`
- `de.fh_konstanz.simubus.model`
- `de.fh_konstanz.simubus.model.optimierung`
- `de.fh_konstanz.simubus.model.simulation`
- `de.fh_konstanz.simubus.model.simulation.entities`
- `de.fh_konstanz.simubus.model.simulation.events`
- `de.fh_konstanz.simubus.util`
- `de.fh_konstanz.simubus.view`

Das Paket `de.fh_konstanz.simubus.controller` enthält Listener-Klassen, d.h. deren Methoden werden bei bestimmten Aktionen wie z.B. das Drücken einer Schaltfläche ausgeführt. Alle Klassen für die Business Logik befinden sich im Paket `de.fh_konstanz.simubus.model`. In diesem Paket gibt es noch zwei Unterpakete: `optimierung` und `simulation`. Sie enthalten die jeweiligen Business-Klassen für die Optimierung bzw. Simulation. Im Unterpaket `simulation` gibt es wiederum zwei Pakete: `entities` und `events`. Das Paket `entities` enthält alle Einheiten, die für die Simulation benötigt werden. Alle Ereignisse für die Simulation befinden sich im Paket `events`. Klassen, die Hilfsmethoden zur Verfügung stellen, wurden in das Paket `de.fh_konstanz.simubus.util` ausgelagert. Alle Fenster, die durch die Anwendung dargestellt werden, befinden sich im Paket `de.fh_konstanz.simubus.view`.

Weiterhin wurden zur Implementierung der Bussimulation zusätzliche Bibliotheken verwendet. Diese wurden als JAR-Dateien in das Programm eingebunden. Sie befinden sich im lib Verzeichnis unter den Namen `desmoj.core_2.1.1.jar`, `Jama-1.0.2.jar` und `jgraph.jar` mit allen Unterpaketen. Das Paket `desmoj` enthält alle Klassen um eine ereignisorientierte Simulation entwickeln zu können. `Jama` enthält Klassen um relativ einfach

mit Matrizen arbeiten zu können und *org.jgraph* ist eine Bibliothek zur Darstellung von Kanten und Graphen in einer graphischen Oberfläche.

Die folgende Grafik soll veranschaulichen, wie die einzelnen Pakete miteinander verbunden sind:

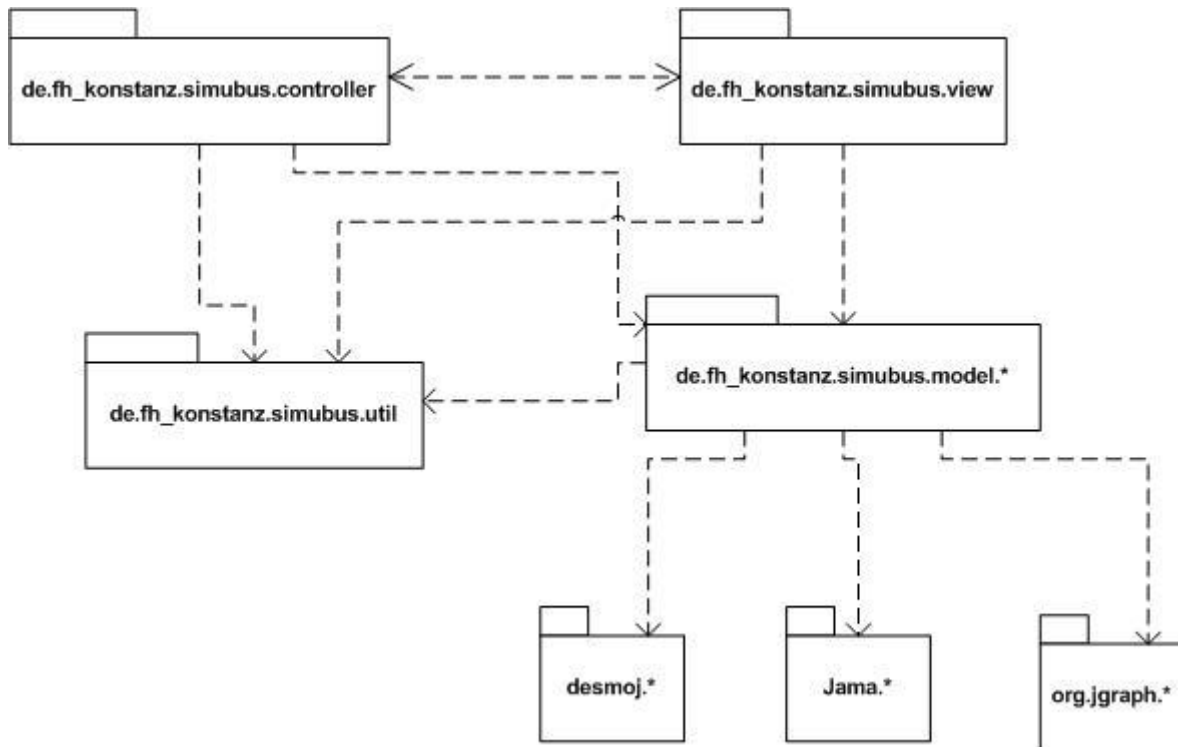


Abbildung 1-1: Java Pakete

1.2 Optimierung

Im folgenden UML-Diagramm sind alle Klassen, die für die Optimierung notwendig sind.

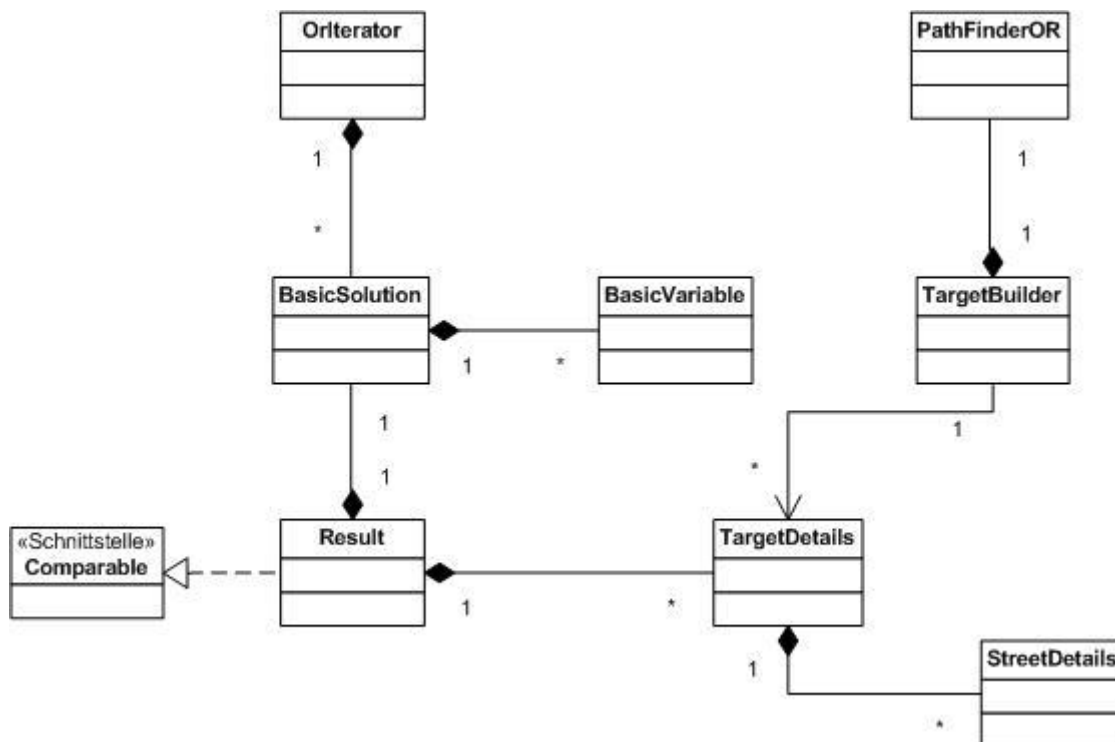


Abbildung 1-2: Klassen für Optimierung

Die Klasse `TargetBuilder` berechnet zu jedem Ziel die Entfernung zu den einzelnen Strassenstücken. Dafür verwendet sie die Klasse `PathFinderOR`, welche den A*-Algorithmus implementiert. Das Ergebnis wird dann in der Klasse `TargetDetails` gespeichert. Sie enthält die Information, welche Haltestellen-Positionen für das Ziel in Frage kommen. Dabei wird jedes Planquadrat mit der Information, wie weit es vom Ziel entfernt ist, in der Klasse `StreetDetails` gespeichert. Das endgültige Ergebnis der Optimierung wird in der Klasse `Result` gespeichert. Sie enthält alle nötigen Informationen zu einer gefundenen Lösung wie Positionen der Haltestellen, Informationen zu den Zielen oder die durchschnittliche Dauer. Zusätzlich implementiert die Klasse die Schnittstelle `Comparable`. Dies ist notwendig, um die Lösungen sortieren zu können. Um das Modell zu lösen wurde die Klasse `OrIterator` erstellt. Sie verwendet die Simplexmethode zur Umformung der Matrix. Als Ergebnis liefert die Klasse eine Menge von Basislösungen, welche die Klasse `BasicSolution` kapselt. Diese wiederum enthält eine Liste von Basisvariablen; sie werden durch die Klasse `BasicVariable` dargestellt.

1.3 Simulation

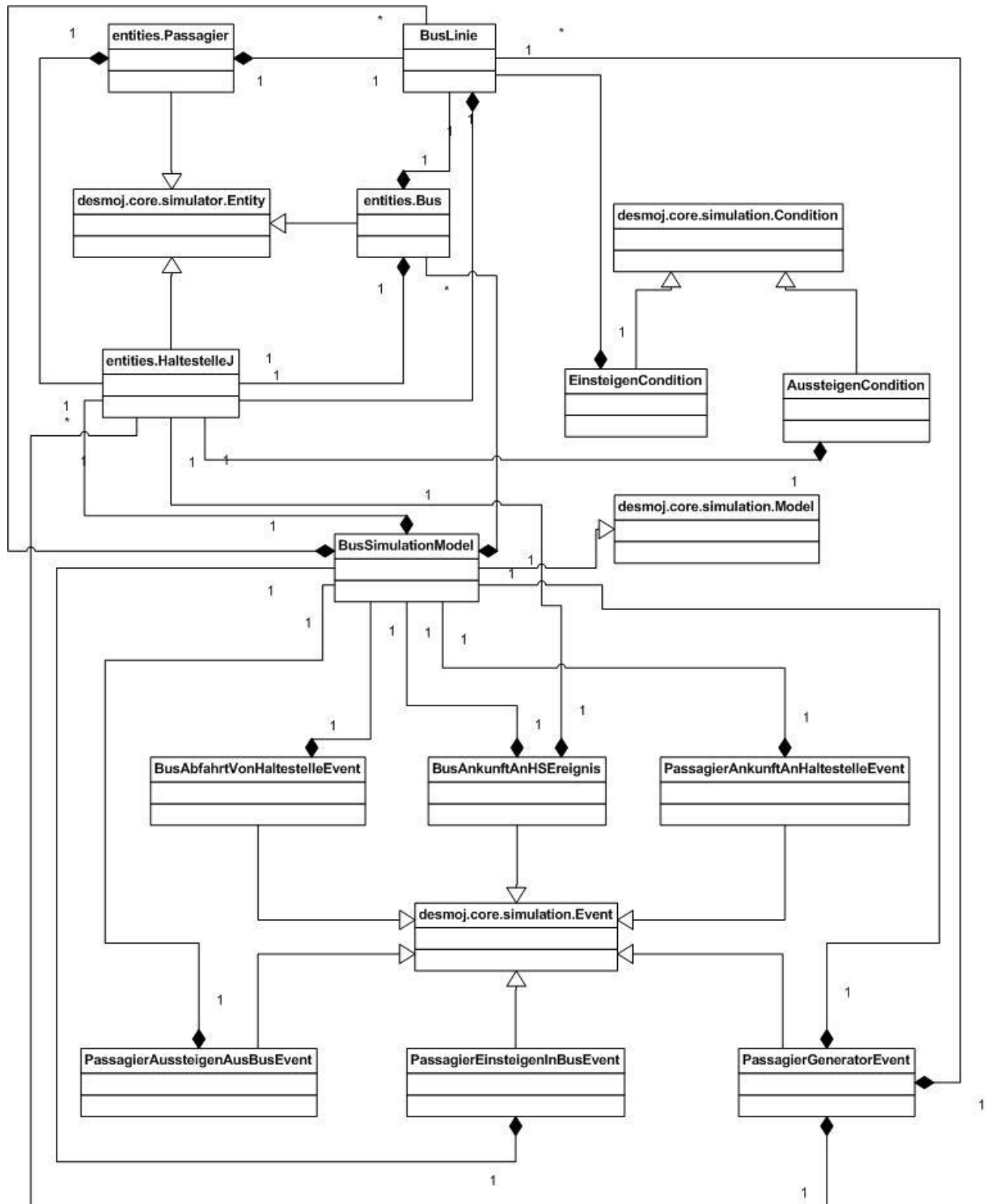


Abbildung 1-3: Klassen für Simulation

Die für die Simulation zuständigen Klassen befinden sich in folgenden Packages:

`de.fh_konstanz.simubus.model.simulation.events`

Die Klassen `BusAnkunftAnHSEreignis`, `PassagierAnkunftAnHaltestelleEvent`, `PassagierEinsteigenInBusEvent`, `PassagierAussteigenAusBusEvent`, `BusAbfahrtVonHaltestelleEvent` sind Ereignisklassen, welche entsprechende Ereignisroutinen bereitstellen. Sie müssen durch die Verwendung des Frameworks *DesmoJ* alle von `desmoj.core.simulation.Event` abgeleitet sein.

Die Klasse `PassagierGeneratorEvent` erzeugt Passagiere und plant `PassagierAnkunftAnHS` Ereignisse nach einer exponentialverteilten Zufallszeit. Des Weiteren plant das Ereignis sich selbst neu, damit der Passagierzufuss nicht abreißt.

Alle Ereignisklassen müssen die Methode `eventRoutine()`, welche ein *Entity* Objekt übergeben bekommt, implementieren. Diese Methode wird durch *DesmoJ* zur entsprechenden Simulationszeit aufgerufen.

`de.fh_konstanz.simubus.model.simulation.entities`

Hier befinden sich die Klassen der Entitäten.

Die Klasse `HaltestelleJ` kapselt alle Daten, die zu einer Haltestelle gehören. Da *DesmoJ* eine Vererbung von *Entity* fordert, wird die Klasse `HaltestelleJ` erstellt und im Code ein Mapping aller Haltestellen Objekte auf `HaltestellenJ` Objekte vorgenommen. Dies ist notwendig, da *Entity* Objekte erst zum Beginn der Simulation angelegt werden können. Das Mapping erfolgt in der Klasse `BusSimulationModel`.

Die Klasse `Bus` speichert die fahrenden Passagiere, die Buslinie auf der der Bus fährt sowie die maximale Kapazität des Busses.

Die Klasse `Passagier` speichert für jeden Fahrgast die Start- und Zielhaltestelle, sowie die verwendete Buslinie.

Eine Vererbung von *Entity* ist bei diesen Klassen erforderlich, da sie Warteschlangen (Queue) enthalten und diese nur durch die Vererbung in den *DesmoJ* Reports auftauchen.

`de.fh_konstanz.simubus.model.simulation`

Die Klasse `BusSimulationModel` ist die Hauptklasse der ereignisorientierten Simulation. Sie dient der Initialisierung der Bussimulation, der Verknüpfung mit einem *Experiment* und letztendlich zum Starten der Simulation.

Die Simulation wird in der Klasse `SimuButtonListener` im Controller-Package gestartet. Hier wird ein `BusSimulationModel` Objekt erzeugt und mit einem `Experiment` verknüpft. Das `Experiment` bzw. die Simulation wird hier gestartet und beendet.

Die `Condition` Klassen dienen der Suche von Elementen in einer Warteschlange, die bestimmte Kriterien erfüllen. In diesem Falle sind diese Kriterien Passagiere, die aus- bzw. einsteigen wollen.

Die Klasse `BusLinie` ist notwendig um die vorhandenen Buslinien abrufen zu können. Alle Buslinien werden in einem Array in der Klasse `BusSimulationModel` gespeichert.

1.4 Kerneloptimierung

Die Matrix zur Berechnung der optimalen Haltestellenpositionen enthält derzeit viele irrelevante Informationen, die die Rechenzeit des Programms erhöhen.

1.4.1 Ideenentwicklung

Die bisherige Matrix aus Haltestellen und Zielen (Zuordnungsproblem) ist sehr groß und enthält viele unnötige Informationen (Zeilen und Spalten, welche ausschließlich Nullen enthalten). Aufgrund der großen Dimensionen wird bei der Optimierung unnötig viel Rechenzeit benötigt.

Die grundsätzliche Idee besteht nun darin, aus der existierenden Matrix die notwendigen Informationen zu extrahieren, d.h. die Matrix wird komprimiert und das Optimaltableau der komprimierten Matrix wird durch den Simplexalgorithmus berechnet. Zur korrekten Darstellung muss die nun optimierte und komprimierte Matrix nun wieder aufgebläht werden, so dass die richtigen Lösungen auf der Oberfläche angezeigt werden.

1.4.2 Technische Umsetzung

Zur Komprimierung (Transformation) der Matrix muss vor der Berechnung der Lösungen eingegriffen werden, die Lösungsberechnung bleibt in ihrer bisherigen Weise erhalten. Nach dem Berechnen des Optimaltableaus muss wiederum eingegriffen werden, um die Lösungen korrekt anzuzeigen (Retransformation). Durch den Einsatz des Decorator- Musters wird die bisherige Klasse `Orlterator` (`de.fh_konstanz.simubus.model.optimierung`), die für die Optimierung zuständig ist, durch die neu erstellte Klasse `OrOptilterator` (`de.fh_konstanz.simubus.util`) umschlossen (dekoriert). Dementsprechend wird der Aufruf der Methode `starteOptimierung()` in der Klasse `OptiControlPanel` angepasst.

Bisherige Optimierung

Ursprüngliche Matrix

0	0	0	0	0	0	1	0	1	0
1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0
0	1	0	0	1	0	1	0	0	0

(8x10)

(Berechnung
Optimaltableau)



Lösungstableau groß

0	0	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0

Neue Optimierung

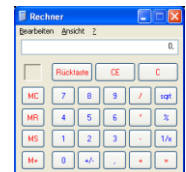
Ursprüngliche Matrix (8 x 10)

0	0	0	0	0	0	1	0	1	0
1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0
0	1	0	0	1	0	1	0	0	0

Verkleinerte Matrix (6x5)

0	0	0	1	1
1	0	1	0	1
0	0	0	0	1
1	1	0	0	1
1	0	0	0	1
0	1	1	1	0

(Berechnung
Optimaltableau)



Lösungstableau klein

0	0	0	1	0
0	0	1	0	0
0	0	0	0	1
1	0	0	0	0
0	0	0	0	0
0	1	0	0	0

Lösungstableau groß

0	0	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0

er scheinbar erhöhte Aufwand zur Transformation wird durch die verringerte Anzahl der Iterationen beim Lösen des Tableaus ohne Frage gerechtfertigt. Bei großen Modellen verringert sich die Berechnungszeit erfahrungsgemäß um 10%.

1.4.2.1 Verkleinerungsalgorithmus (Transformation)

Durch den Aufruf des Konstruktors

```
private OrOptiIterator( double[][] aMatrix )
```

werden alle zunächst notwendige Operationen ausgeführt.

Die übergebenen Matrix aMatrix hat die Dimensionen **Anzahl Ziele X (Gitterspalten * Gitterzeilen)**. Das heißt in den meisten Fällen besteht die Matrix aus mind. 10 Zeilen (für 10 Ziele) und 4096 (= 64 x 64) Spalten, wobei je nach Lage der Ziele ein Großteil der Zeilen und Spalten lediglich aus Nullen besteht.

Aus dieser großen Matrix aMatrix wird durch die Methode

```
public double[] createSumVector(double[][] sourceMatrix)
```

ein Vektor der Länge (**Gitterspalten x Gitterzeilen**) erstellt.

Dies geschieht durch die Aufsummierung aller Spaltenwerte durch diese Methode.

Anschließend wird aus diesem Vektor wieder eine Matrix erstellt, die zur weiteren Verarbeitung nötig ist. Die Methode

```
public double[][] createTwoDimMatrixFromRow(double[] sourceRow)
```

übernimmt diese Aufgabe.

Jetzt startet die eigentliche Minimierung der Matrix.

Die Methode

```
private double[][] minimizeMatrix(double[][] matrix)
```

durchläuft alle Spalten der übergebenen Matrix und überprüft, ob eine Spalte oder eine Zeile lediglich aus Nullen besteht. Sollte dies der Fall sein, so werden die Zeilen und Spalten für spätere Zwecke hinterlegt.

Nach dem Durchlauf der Spalten schneidet die Methode

```
private double[][] cutMatrix(double[][] sourceMatrix)
```

die irrelevanten (d.h. nur aus Nullen bestehende) Zeilen und Spalten aus der Matrix ab.

Die Matrix ist nun komprimiert worden, sie besteht lediglich aus den minimal möglichen Dimensionen, während der Informationsgehalt gleich geblieben ist.



Da der Orlterator keine Summenmatrix erwartet, sondern pro Ziel einen Vektor, werden im nächsten Schritt die Quellvektoren angepasst. Dies erfolgt unter Verwendung der zuvor gewonnen Informationen.

0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0

Vektorbildung

3	2	0	0	2	0	0	0	0
---	---	---	---	---	---	---	---	---

Bilden einer Matrix

3	2	0
0	2	0
0	0	0

0	0	0
0	0	0
0	0	0

1	0	0
0	1	0
0	0	0

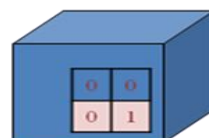
0	0	0
0	0	0
0	0	0

.....

0	1	0
0	1	0
0	0	0

Minimierte Matrix

0	0	0	0
1	0	1	0
0	0	0	0
0	0	0	0
1	1	0	0
0	0	0	0
1	0	0	0
0	1	0	1



Optimierer

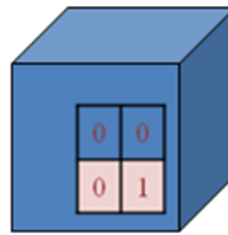
Abbildung 1-4: Transformation der Matrix

1.4.2.2 Retransformation

Nachdem die minimierte Matrix optimiert wurde, muss diese wieder in die ursprünglichen Dimensionen retransformiert werden. Dies ist notwendig, da die GUI-Elemente mit den Relationen der ursprünglichen Matrix positioniert werden. Bei Verwendung der optimierten Matrix würden die GUI-Elemente fehlerhaft angezeigt werden. Hierzu wird die Methode

```
public Matrix fillMinimalMatrix(Matrix aMatrix)
```

aufgerufen, dessen Übergabeparameter die optimierte Matrix darstellt. Die zu Beginn der Transformation abgeschnittenen unnötigen Informationen, d.h. Zeilen und Spalten, die ausschließlich aus Nullen bestehen, werden nun wieder an den passenden Stellen hinzugefügt.



Optimierer

Optimierte Matrix

0	0	0	0
0	0	1	0
0	0	0	0
1	0	0	0
0	0	0	1
0	1	0	0



Retransformation

0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0

1.5 Business Logik

Die Business Logik stellt die Programmsteuerung der Anwendung zur Verfügung. Sie ist das Bindeglied zwischen der Präsentation und den Geschäftsprozessen.

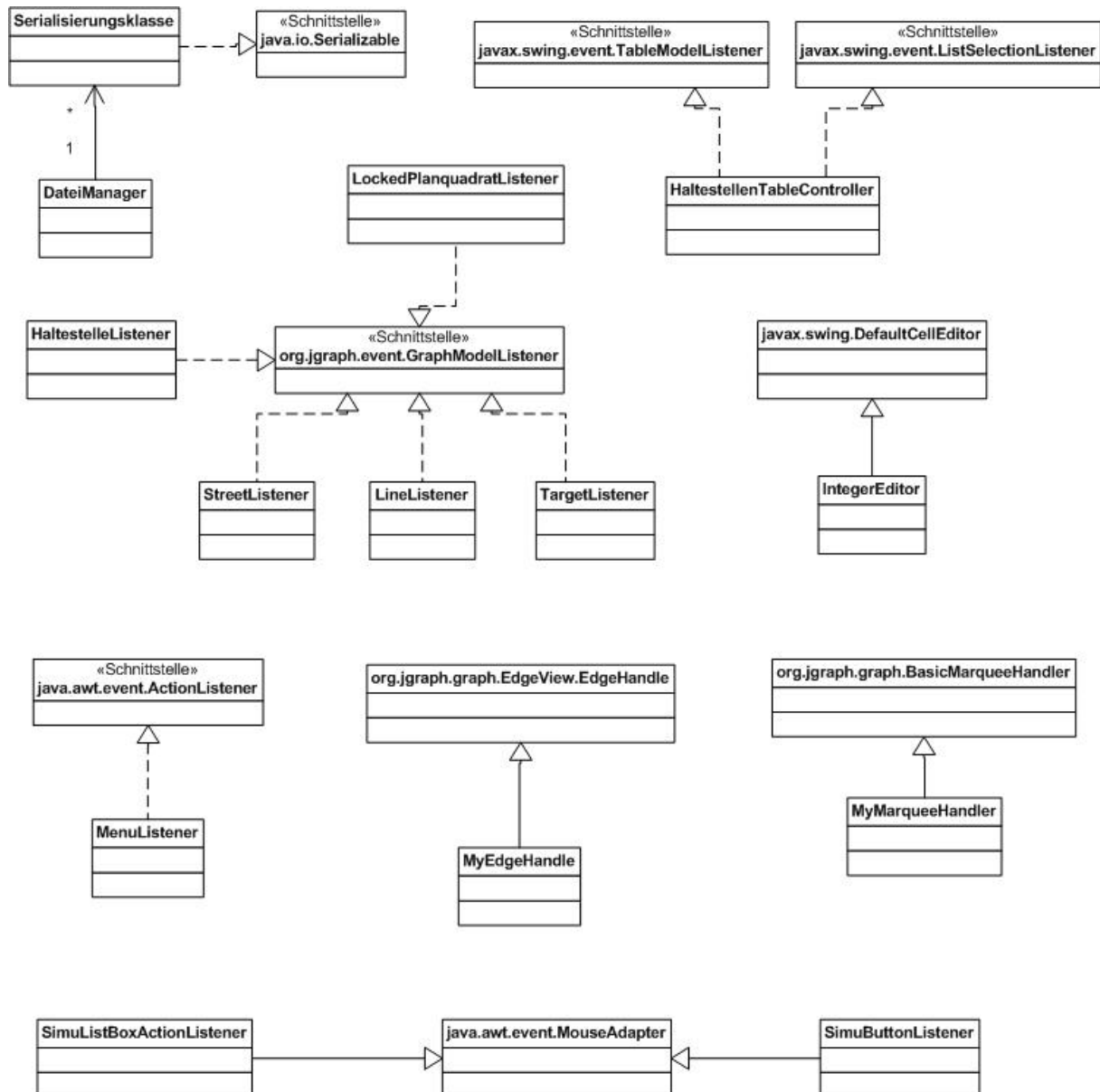


Abbildung 1-5: Listener Klassen

Die Klasse `DateiManager` ist für das Speichern und Laden von Szenarien verantwortlich. Diese hat eine Referenz auf die `SerialisierungsKlasse`, welche sämtliche Klassen die vom Speichern und Laden abhängig sind, bearbeitet. Um die Serialisierung vornehmen zu können, muss die `SerialisierungsKlasse` die Schnittstelle `Serializable` implementieren. Die Klassen `HaltestellenListener`, `LockedPlanquadratListener`, `StreetListener`, `LineListener` und `TargetListener` sind Klassen, die auf

Ereignisse aus der grafischen Benutzeroberfläche reagieren. Bei Änderungen wird die dementsprechende Listenerklasse aufgerufen und im Hintergrund die Klasse `Strassennetz` und `VirtualGrid` geändert. Die eben genannten Listenerklassen implementieren die Schnittstelle `GraphModelListener`. Die Klasse `HaltestellenTableController` reagiert zum einen auf Änderungen in der grafischen Oberfläche, wenn beispielsweise eine Haltestelle hinzugefügt oder entfernt wird. Zum anderen werden Eigenschaften der Haltestelle in der dazugehörigen Haltestellentabelle verarbeitet. Die Klasse implementiert die Schnittstellen `TableModelListener` und `ListSelectionListener`. Alle Ereignisse aus dem Hauptmenü werden in der Klasse `MenuListener` verarbeitet. Diese implementiert die Schnittstelle `ActionListener`. Die Klasse `IntegerEditor` fängt nicht zulässige Eingaben des Benutzers in der Haltestellen-Tabelle der graphischen Oberfläche ab. Somit kann der Benutzer nur ganzzahlige Werte zwischen 1 und 100 für die Kapazität einer Haltestelle eingeben. Die Klasse `IntegerEditor` erweitert die Klasse `DefaultCellEditor`. Die Klasse `MyMarqueeHandler` reagiert auf Maus-Ereignisse, wie das Selektieren von Knoten und Kanten. Ebenso reagiert diese Klasse auf Änderungen, wenn mit der Maus über eine Haltestelle gefahren wird, sowie das Verbinden zweier Haltestellen. Die Klasse `MyMaqueeHandler` erweitert die Klasse `BasicMarqueeHandler` des Frameworks `JGraph`. `MyEdgeHandler` reagiert auf Maus-Ereignisse, wenn der Benutzer einen Punkt auf eine Linie setzt bzw. löscht. Die Klasse `MyEdgeHandler` erweitert die Klasse `EdgeHandler` des Frameworks `JGraph`.

Die Klasse `SimuButtonListener` reagiert auf Ereignisse, wenn in der Klasse `SimuControlPanel` eine Schaltfläche gedrückt wird. Die Klasse `SimuListBoxActionListener` reagiert auf Ereignisse, wenn ein Listeneintrag der visuellen Ausgabe (Buslinie oder Teilstrecke) selektiert wird. Ändert sich beispielsweise die Selektion in der Buslinienliste, reagiert der `SimuListBoxActionListener` darauf, und ändert die visuelle Ausgabe der zugehörigen Teilstreckenliste.

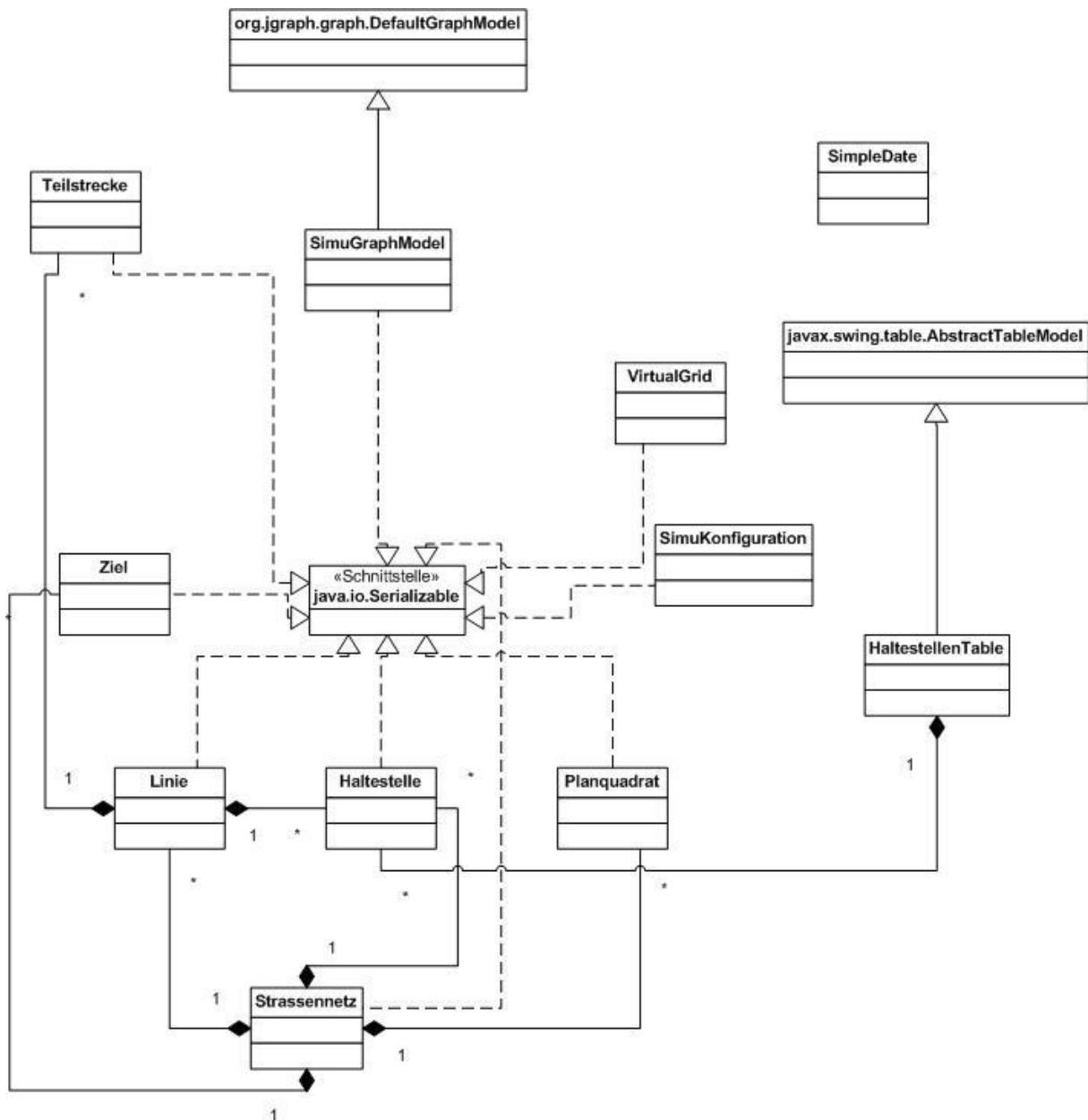


Abbildung 1-6: Klassen für Business Logik

Die Klasse `SimuGraphModel` implementiert das Model eines Graphen auf der Oberfläche. Sie erweitert die Klasse `DefaultGraphModel` des Frameworks `JGraph`.

Für Standardeinstellungen der Optimierung und Simulation ist die Klasse `SimuKonfiguration` verantwortlich. Im `Strassennetz` sind sämtliche Listen wie `Haltestellen`, `Strassen`, `Ziele`, `Buslinien`, `Planquadrate` sowie die Liste der gesperrten Felder enthalten. Die Klasse `VirtualGrid` beinhaltet eine nicht sichtbare Matrix, welche mit `Planquadraten` gefüllt ist. In der Klasse `Haltestelle` werden Merkmale wie die Koordinaten, der Name und die Kapazität einer Bushaltestelle gespeichert. Die `Linie` ist eine Buslinie, welche aus mehreren Linienabschnitten besteht. Ein Linienabschnitt kann wiederum aus mehreren `Teilstrecken` bestehen.

Die Klasse `Ziel` stellt ein Ziel mit deren Informationen auf der Oberfläche dar. Die Klasse `Planquadrat` beinhaltet Informationen, welche Eigenschaft (Strasse, Ziel, Haltestelle, gesperrtes Feld oder Leer) dieses Planquadrat hat. Sie ist essentieller Anteil der Optimierung. Das Model für die Darstellung der Daten in der Haltestellentabelle stellt die Klasse `HaltestellenTable` dar. Sie erweitert die Klasse `AbstractTableModel`. Die Klasse `SimpleDate` kapselt eine Uhrzeit. Mit Ausnahme der Klassen `HaltestellenTable` und `SimpleDate` implementieren alle Klassen der Business Logik die Schnittstelle `Serializable`.

1.6 Präsentation

Im Folgenden werden die für die Präsentation benötigten Klassen beschrieben. Sie stellt die Kommunikationsschnittstelle zum Benutzer dar, nimmt Benutzereingaben entgegen und ist verantwortlich für eine aktuelle Ausgabe von Inhalten auf dem Bildschirm.

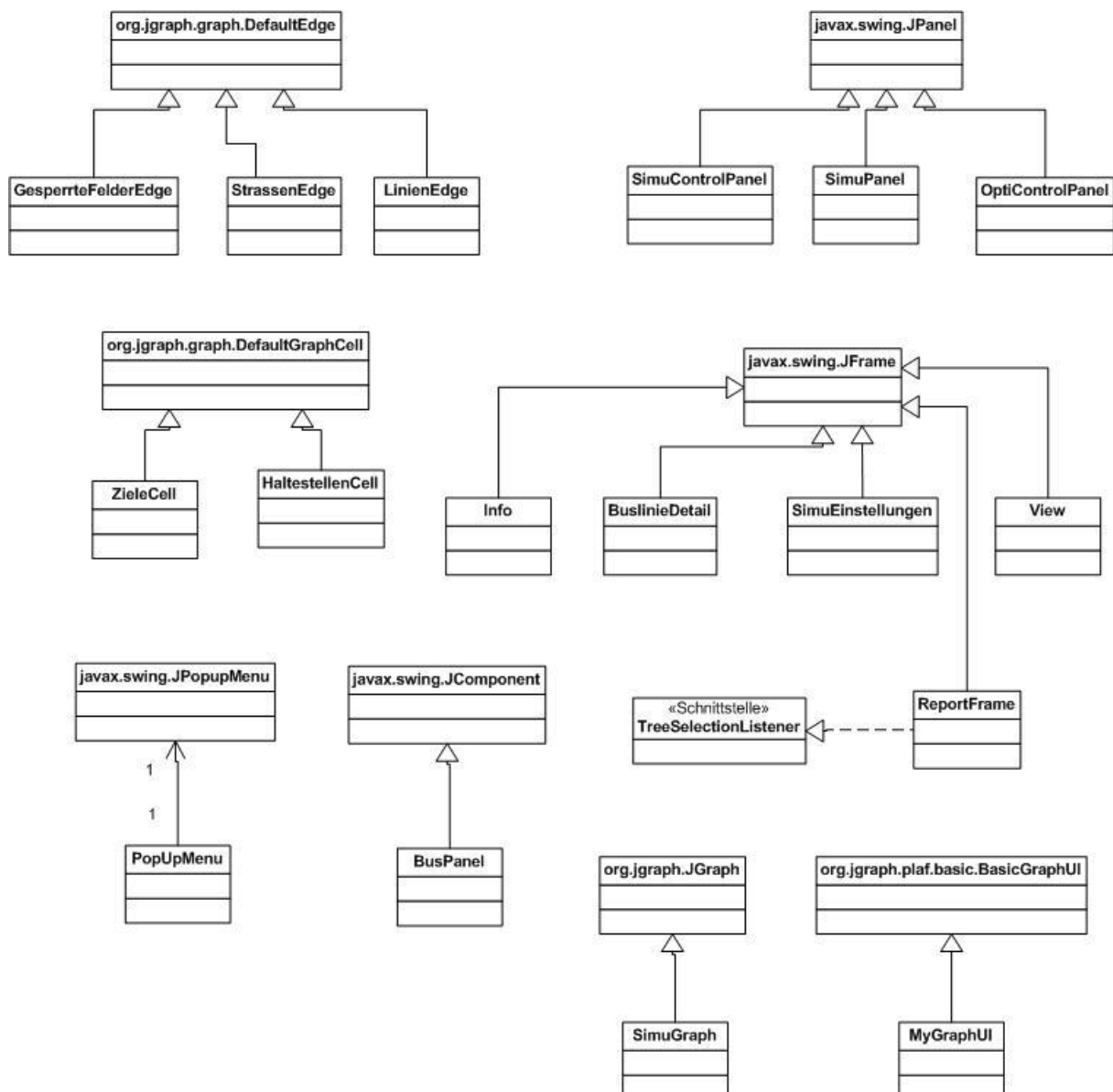


Abbildung 1-7: Klassen für Präsentation

`SimuPanel` ist die zentrale Klasse für das Einzeichnen und Löschen von Haltestellen, Strassen, Zielen, gesperrten Feldern und Linien. Über das `OptiControlPanel` kann der Benutzer Einstellungen für die Optimierung vornehmen. Nach der Optimierung werden die einzelnen Lösungen mit zusätzlichen Informationen angezeigt. Das `SimuControlPanel` zeigt die eingezeichneten Haltestellen, sowie die Buslinien mit den entsprechenden Teilstrecken. Des Weiteren kann der Benutzer Einstellungen bezüglich der Simulation vornehmen. Während der Simulation bekommt der Benutzer Informationen zum Ablauf. Zusätzlich besteht die Möglichkeit sich nach der Simulation diverse Auswertungen anzeigen zu lassen. Beide Klassen erweitern die Klasse `JPanel`.

`HaltestellenCell` und `ZieleCell` stellen die entsprechenden Knoten auf der Karte dar, die Klassen `GesperrteFelderEdge` und `StrassenEdge` die Kanten. Alle Planquadrate, durch die die Kanten und Knoten verlaufen, werden dementsprechend gesetzt. Die Klasse `LinienEdge` entspricht einem Linienabschnitt auf der Karte.

Das Hauptfenster der Anwendung wird durch die Klasse `View` repräsentiert. Die Klasse `SimuEinstellungen` ist ein Untermenü des Hauptfensters, welche allgemeine Einstellungen zur Optimierung und Simulation beinhaltet. Information über das Projekt und die Autoren sind in der Klasse `Info` enthalten. Die Klasse `BuslinieDetail` liefert detaillierte Informationen zu einer bestimmten Buslinie. Im `ReportFrame` werden die erzeugten Simulationsauswertungen angezeigt.

Das `PopupMenu` ist ein Kontext-Menü für die Karte, mit welchem Ziele, Haltestellen, Strassen und GesperrteFelder hinzugefügt und gelöscht werden. Die Klasse `BusPanel` wird für die Animation der Simulation benötigt, die Klasse `SimuGraph` für das Zeichnen eines Graphen. `MyGraphUI` wird zum Einbinden des Hintergrundbildes verwendet.

1.7 Speichern/Laden

Im Folgenden wird erklärt, wie die in der Präsentationsoberfläche angesiedelten Objekte zur Darstellung von Haltestellen, Strassen, Zielen, etc. in eine XML-Datei abgespeichert werden. Dies betrifft jedoch nur die Präsentationsobjekte und nicht die Ergebnisse, welche durch die Optimierung erzeugt wurden weil diese durch die gespeicherte Präsentationsoberfläche reproduziert werden können.

1.7.1 Speichern

Das Speichern von Präsentationsobjekten geschieht in der Java Klasse `XMLFileManager`, welche sich in dem Package `de.fh_konstanz.simubus.controller` befindet. Da die Klasse `XMLFileManager` auf einem Singleton Pattern aufbaut, ist eine Instanz dieser Klasse durch den Aufruf der Methode `getInstance()` zu erhalten. Wenn der Speichern Befehl ausgeführt wird, greift der `XMLFileManager` auf vier Instanzen zu, welche die Präsentationsobjekte zur Laufzeit verwalten. Diese vier Instanzen, sind jeweils Objekte vom Datentyp:

- Strassennetz,
- `SimuKonfiguration`,

- SimuGraphModel und
- VirtualGrid.

Die Präsentationsobjekte werden nacheinander und nach Klassenart geordnet ausgelesen. Nachdem ein Präsentationsobjekt ausgelesen wurde werden die Informationen, die sich in diesem Präsentationsobjekt befinden ausgelesen und in ein entsprechendes XML- Element geschrieben. Dieses wird dann in einen JDOM-Baum an die entsprechende Stelle eingehängt und verbleibt dort bis der gesamte Baum in eine XML-Datei geschrieben wird.

Der Klasse XMLFileManager wird beim Speichern ein Pfad mit Dateinamen übergeben unter diesem dann eine XML-Datei erstellt wird. Bevor der XMLFileManager eine XML-Datei erstellt, wird diese unter Zuhilfenahme einer XML Schema File auf Korrektheit überprüft. Die XML Schema File ist dem Anhang zu entnehmen.

Der JDOM-Baum , welcher für das Speichern verwendet wird, stammt aus der Bibliothek „jdom.jar“, die in das Projekt eingebunden wurde. Dieser JDOM-Baum ist eine für Java entwickelte Darstellung von XML.

1.7.2 Laden

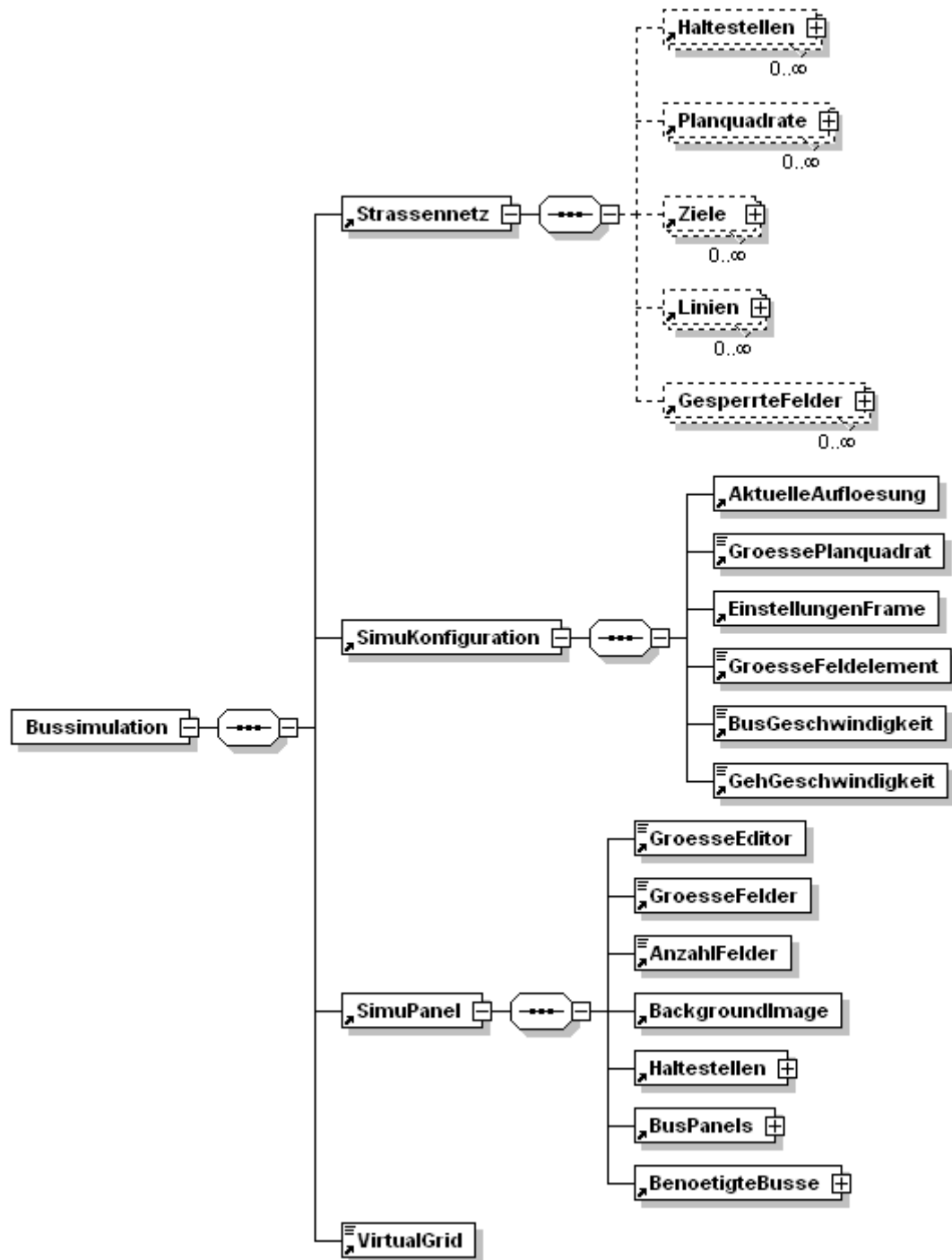
Das Laden von Präsentationsobjekten erfolgt analog zum Speichervorgang und erfolgt in der Java Klasse XMLFileLoader, welche sich in dem Package *de.fh_konstanz.simubus.controller* befindet. Dem XMLFileLoader wird der Pfad zu der vom Benutzer ausgewählten XML-Datei übergeben, welche geladen werden soll.

Für das Laden aus einer XML-Datei wird ein DOM-Baum verwendet, welcher mittels der JDOM Bibliothek „jdom.jar“ erzeugt wird, und in den die ausgelesenen Elemente der XML-Datei abgelegt werden. Zum Auslesen der XML-Datei wird ein SAX Parser verwendet.

Für die ausgelesenen Elemente, welche sich nun im DOM-Baum befinden, werden jeweils dem Datentyp entsprechende Präsentationsobjekte erzeugt und mit den Daten der Elemente aus dem DOM-Baum befüllt. Diese Präsentationsobjekte werden dann in die dafür vorgesehenen Verwaltungsobjekte Strassennetz, SimuKonfiguration, SimuGraphModel und VirtualGrid hinzugefügt. Die Applikation Bussimulation verwendet diese Verwaltungsobjekte zur Laufzeit, um auf die Präsentationsobjekte zuzugreifen und gegebenenfalls zu verändern.

1.7.3 XML-Datei

Im Folgenden wird eine XML-Datei dargestellt und exemplarisch erklärt (siehe Anhang 3).



Generated by XmlSpy

www.altova.com

Abbildung 1-8: XML-Datei als Baumdarstellung

```

<?xml version="1.0" ?>
<Bussimulation xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="xsd/busSimulation.xsd">

```

Listing 1-1: Rumpf der XML-Datei

Der in Listing 1-1 aufgeführte Code stellt den Rumpf der XML-Datei dar mit dem Root Element mit der Bezeichnung Bussimulation und dem Verweis auf das XML Schema

File. Das Root Element hat als direkte Kind-Elemente die Verwaltungselemente Strassennetz, SimuKonfiguration, SimuGraphModel und Virtual Grid.

In Listing 1-2 kann man die Hierarchie des Root Elementes und der Kind-Elemente nochmals klar erkennen.

```
<xs:element name="Bussimulation"><!-- Rootelement, Bussimulation -->
  <xs:complexType>
    <xs:sequence><!-- Child Elements, were written out -->
      <xs:element ref="Strassennetz"/>
      <xs:element ref="SimuKonfiguration"/>
      <xs:element ref="SimuPanel"/>
      <xs:element ref="VirtualGrid"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Listing 1-2: XML- Schema, das Root Element und seine Kinder

Das Element Strassennetz beinhaltet alle Elemente, welche die Daten der Präsentationsobjekte beinhalten.

```
<xs:element name="Strassennetz"><!-- Strassennetz -->
  <xs:complexType>
    <xs:sequence><!-- Child Elements -->
      <xs:element ref="Haltestellen" minOccurs="0"
        maxOccurs="unbounded"/><!-- optional Childs -->
      <xs:element ref="Planquadrante" minOccurs="0"
        maxOccurs="unbounded"/>
      <xs:element ref="Ziele" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="Linien" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="GesperrteFelder" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Listing 1-3: XML- Schema, Hierarchie Strassennetz

```
<Strassennetz>
```

Listing 1-4: Anfang Strassennetz

Das Element Haltestellen beinhaltet eine Vielzahl von Haltestellen und stellt somit ein Sequenzelement dar.

```
<Haltestellen>
  <Haltestelle id="1">
    <Name>Haltestelle 1</Name>
    <Kapazitaet>1</Kapazitaet>
    <Point xValue="266" yValue="305" />
  </Haltestelle>
  <Haltestelle id="2">
    <Name>Haltestelle 2</Name>
    <Kapazitaet>1</Kapazitaet>
    <Point xValue="396" yValue="435" />
  </Haltestelle>
  <Haltestelle id="3">
    <Name>Haltestelle 3</Name>
```

```
<Kapazitaet>1</Kapazitaet>
<Point xValue="370" yValue="617" />
</Haltestelle>
</Haltestellen>
```

Listing 1-5: Beispiel für eine Sequenz von Haltestellen

Das Element Planquadrante beinhaltet eine Vielzahl von PlanquadratStrasse Elementen, welche benötigt werden um die Daten für Strassen abzubilden.

```
<Planquadrante>
  <PlanquadratStrasse id="3">
    <Strasse id="3" art="isStreet">
      <GesamtKosten>0</GesamtKosten>
      <KostenStartBisPunkt>0</KostenStartBisPunkt>
      <KostenPunktBisZiel>0</KostenPunktBisZiel>
      <Point xStart="403" yStart="416" xEnd="364" yEnd="624" />
      <Feldgroesse width="13.0" height="13.0" />
    </Strasse>
  </PlanquadratStrasse>
  <PlanquadratStrasse id="2">
    <Strasse id="2" art="isStreet">
      <GesamtKosten>0</GesamtKosten>
      <KostenStartBisPunkt>0</KostenStartBisPunkt>
      <KostenPunktBisZiel>0</KostenPunktBisZiel>
      <Point xStart="312" yStart="299" xEnd="403" yEnd="416" />
      <Feldgroesse width="13.0" height="13.0" />
    </Strasse>
  </PlanquadratStrasse>
  <PlanquadratStrasse id="0">
    <Strasse id="0" art="isStreet">
      <GesamtKosten>0</GesamtKosten>
      <KostenStartBisPunkt>0</KostenStartBisPunkt>
      <KostenPunktBisZiel>0</KostenPunktBisZiel>
      <Point xStart="26" yStart="429" xEnd="182" yEnd="325" />
      <Feldgroesse width="13.0" height="13.0" />
    </Strasse>
  </PlanquadratStrasse>
  <PlanquadratStrasse id="1">
    <Strasse id="1" art="isStreet">
      <GesamtKosten>0</GesamtKosten>
      <KostenStartBisPunkt>0</KostenStartBisPunkt>
      <KostenPunktBisZiel>0</KostenPunktBisZiel>
      <Point xStart="182" yStart="325" xEnd="312" yEnd="299" />
      <Feldgroesse width="13.0" height="13.0" />
    </Strasse>
  </PlanquadratStrasse>
</Planquadrante>
```

Listing 1-6: Beispiel für eine Sequenz von Planquadranten

Das Element Ziele beinhaltet eine Vielzahl von einzelnen Ziel Elementen mit dem Namen und den Koordinaten des Ziels.

```
<Ziele>
  <Ziel id="0">
    <Name>Ziel [20, 21]</Name>
    <Point xValue="266" yValue="279" />
  </Ziel>
  <Ziel id="1">
```



```
<Name>Ziel [29, 33]</Name>
<Point xValue="383" yValue="435" />
</Ziel>
<Ziel id="2">
  <Name>Ziel [29, 47]</Name>
  <Point xValue="383" yValue="617" />
</Ziel>
</Ziele>
```

Listing 1-7: Beispiel für eine Sequenz von Zielen

Das Element Linien beinhaltet alle Buslinien als Unterelemente. Jede einzelne Buslinie beinhaltet wiederum eine Referenz auf die angefahrenen Haltestellen und eine Sequenz von Teilstrecken, welche die Teilstrecken zwischen den verschiedenen Haltestellen darstellen. Für diese Teilstrecken werden alle Daten, wie der Name der Teilstrecke Name, Start- und Endpunkt, als auch die Geschwindigkeit, welche auf dieser Strecke gefahren werden darf.

```
<Linien>
  <Linie id="Linie 1">
    <Haltestellen>
      <Haltestelle id="1" />
      <Haltestelle id="2" />
      <Haltestelle id="3" />
    </Haltestellen>
    <Teilstrecken>
      <Teilstrecke id="1">
        <Name>Teilstrecke 1 - 260.0;312.0/312.0;299.0
          53.600373133029585</Name>
        <StartPoint xValue="260.0" yValue="312.0" />
        <EndPoint xValue="312.0" yValue="299.0" />
        <Geschwindigkeit>50</Geschwindigkeit>
        <Breite>10</Breite>
        <Laenge>53.600373133029585</Laenge>
      </Teilstrecke>
      <Teilstrecke id="2">
        <Name>Teilstrecke 2 - 312.0;299.0/390.0;442.0
          162.88953311984167</Name>
        <StartPoint xValue="312.0" yValue="299.0" />
        <EndPoint xValue="390.0" yValue="442.0" />
        <Geschwindigkeit>50</Geschwindigkeit>
        <Breite>10</Breite>
        <Laenge>162.88953311984167</Laenge>
      </Teilstrecke>
      <Teilstrecke id="3">
        <Name>Teilstrecke 3 - 390.0;442.0/364.0;624.0
          183.84776310850236</Name>
        <StartPoint xValue="390.0" yValue="442.0" />
        <EndPoint xValue="364.0" yValue="624.0" />
        <Geschwindigkeit>50</Geschwindigkeit>
        <Breite>10</Breite>
        <Laenge>183.84776310850236</Laenge>
      </Teilstrecke>
    </Teilstrecken>
    <Color>-13315483</Color>
    <MaxKapazitaetPassagiere>50</MaxKapazitaetPassagiere>
    <Wiederkehrzeit>0.0</Wiederkehrzeit>
  </Linie>
```

```
</Linien>
```

Listing 1-8: Beispiel für eine Sequenz von Linien

In Listing 1-9 ist der schließende Tag des Elements Strassennetz zu sehen.

```
</Strassennetz>
```

Listing 1-9: Ende Strassennetz

Das Element SimuKonfiguration beinhaltet Konfigurationsdaten für die Applikation Bussimulation. Diese Daten sind z.B. die Bildschirmauflösung, die Busgeschwindigkeit oder die Gehgeschwindigkeit.

```
<SimuKonfiguration>
  <AktuelleAufloesung resWidth="1024" resHeight="768" />
  <GroessePlanquadrat>13</GroessePlanquadrat>
  <EinstellungenFrame width="330" height="380" />
  <GroesseFeldelement>65.0</GroesseFeldelement>
  <BusGeschwindigkeit>40.0</BusGeschwindigkeit>
  <GehGeschwindigkeit>6.0</GehGeschwindigkeit>
</SimuKonfiguration>
```

Listing 1-10: Beispiel SimuKonfiguration

Das Element SimuPanel ist das dritte große Element in der XML-Datei Bussimulation. In diesem Element werden die Daten verwaltet, welche beim Auslesen in das SimuGraphModel geschrieben werden. Die Haltestellen und Buslinien sind an dieser Stelle lediglich Referenzen auf die Präsentationsobjekte, welche bereits im Strassennetz Element der XML-Datei abgespeichert sind und nach dem Laden in einem Objekt vom Datentyp Strassennetz zur Laufzeit verwaltet werden.

```
<SimuPanel>
```

Listing 1-11: Start des SimuPanel

```
<GroesseEditor>640</GroesseEditor>
<GroesseFelder>13</GroesseFelder>
<AnzahlFelder>49</AnzahlFelder>
<BackgroundImage
  url="/C:/absf/sources/Bussimulation/trunk/bin/Eisenstadt.jpg"
  width="840" height="840" />
```

Listing 1-12: Allgemeine Konfigurationsdaten

In Listing 1-13 enthält das Element Haltestellen mehrere Referenzen auf die im XML-Element Strassennetz abgespeicherten Haltestellen.

```
<Haltestellen>
  <Haltestelle id="1" />
  <Haltestelle id="2" />
  <Haltestelle id="3" />
</Haltestellen>
```

Listing 1-13: Referenz auf Haltestellenobjekte

Das Element BusPanels enthält die Referenzen auf die im XML-Element Strassennetz abgespeicherten Buslinien mit denen von den Bussen angefahrenen Haltestellen und Teilstrecken. Zusätzlich werden hier Konfigurationsdaten

abgespeichert, wie z.B. die durchschnittliche Wartezeit für den jeweiligen Bus an dieser Haltestelle, etc.

```
<BusPanels>
  <BusPanel>
    <Image url="/C:/absf/sources/Bussimulation/trunk/bin/bus.png"
      width="18" height="21" />
    <Bus id="0">
      <BusLinie id="Linie 1">
        <HaltestellenJ>
          <HaltestelleJ>
            <BusWarteschlangeQueue>
              <Name>'Haltestelle Haltestelle1 BusQueue</Name>
              <DurchschnittsLaenge>0.0</DurchschnittsLaenge>
              <DurchschnittsWarteZeit>0.0</DurchschnittsWarteZeit>
              <WarteschlangenLimit>2147483647</WarteschlangenLimit>
              <WarteschlangenStrategie>FIFO</WarteschlangenStrategie>
            </BusWarteschlangeQueue>
            <PassagierWarteschlangeQueue>
              <Name>Haltestelle: Haltestelle1PassagierQueue</Name>
              <DurchschnittsLaenge>2.2755</DurchschnittsLaenge>
              <DurchschnittsWarteZeit>1.46944</DurchschnittsWarteZeit>
              <WarteschlangenLimit>2147483647</WarteschlangenLimit>
              <WarteschlangenStrategie>FIFO</WarteschlangenStrategie>
            </PassagierWarteschlangeQueue>
            <Point xValue="266.0" yValue="305.0" />
            <BusEinheiten>0</BusEinheiten>
          </HaltestelleJ>
          <HaltestelleJ>
            <BusWarteschlangeQueue>
              <Name>'Haltestelle Haltestelle2 BusQueue</Name>
              <DurchschnittsLaenge>0.0</DurchschnittsLaenge>
              <DurchschnittsWarteZeit>0.0</DurchschnittsWarteZeit>
              <WarteschlangenLimit>2147483647</WarteschlangenLimit>
              <WarteschlangenStrategie>FIFO</WarteschlangenStrategie>
            </BusWarteschlangeQueue>
            <PassagierWarteschlangeQueue>
              <Name>Haltestelle: Haltestelle2PassagierQueue</Name>
              <DurchschnittsLaenge>2.06084</DurchschnittsLaenge>
              <DurchschnittsWarteZeit>1.34403</DurchschnittsWarteZeit>
              <WarteschlangenLimit>2147483647</WarteschlangenLimit>
              <WarteschlangenStrategie>FIFO</WarteschlangenStrategie>
            </PassagierWarteschlangeQueue>
            <Point xValue="396.0" yValue="435.0" />
            <BusEinheiten>0</BusEinheiten>
          </HaltestelleJ>
          <HaltestelleJ>
            <BusWarteschlangeQueue>
              <Name>'Haltestelle Haltestelle3 BusQueue</Name>
              <DurchschnittsLaenge>0.0</DurchschnittsLaenge>
              <DurchschnittsWarteZeit>0.0</DurchschnittsWarteZeit>
              <WarteschlangenLimit>2147483647</WarteschlangenLimit>
              <WarteschlangenStrategie>FIFO</WarteschlangenStrategie>
            </BusWarteschlangeQueue>
            <PassagierWarteschlangeQueue>
              <Name>Haltestelle: Haltestelle3PassagierQueue</Name>
              <DurchschnittsLaenge>3.02966</DurchschnittsLaenge>
              <DurchschnittsWarteZeit>1.64658</DurchschnittsWarteZeit>
```

```

        <WarteschlangenLimit>2147483647</WarteschlangenLimit>
        <WarteschlangenStrategie>FIFO</WarteschlangenStrategie>
    </PassagierWarteschlangeQueue>
    <Point xValue="370.0" yValue="617.0" />
    <BusEinheiten>0</BusEinheiten>
</HaltestelleJ>
</HaltestellenJ>
<MaxKapazitaet>50</MaxKapazitaet>
<Teilstrecken>
    <Teilstrecke id="1">
        <Name>Teilstrecke 1 - 260.0;312.0/312.0;299.0
            53.600373133029585</Name>
        <StartPoint xValue="260.0" yValue="312.0" />
        <EndPoint xValue="312.0" yValue="299.0" />
        <Geschwindigkeit>50</Geschwindigkeit>
        <Breite>10</Breite>
        <Laenge>53.600373133029585</Laenge>
    </Teilstrecke>
    <Teilstrecke id="2">
        <Name>Teilstrecke 2 - 312.0;299.0/390.0;442.0
            162.88953311984167</Name>
        <StartPoint xValue="312.0" yValue="299.0" />
        <EndPoint xValue="390.0" yValue="442.0" />
        <Geschwindigkeit>50</Geschwindigkeit>
        <Breite>10</Breite>
        <Laenge>162.88953311984167</Laenge>
    </Teilstrecke>
    <Teilstrecke id="3">
        <Name>Teilstrecke 3 - 390.0;442.0/364.0;624.0
            183.84776310850236</Name>
        <StartPoint xValue="390.0" yValue="442.0" />
        <EndPoint xValue="364.0" yValue="624.0" />
        <Geschwindigkeit>50</Geschwindigkeit>
        <Breite>10</Breite>
        <Laenge>183.84776310850236</Laenge>
    </Teilstrecke>
</Teilstrecken>
</BusLinie>
<HaltestelleJ>
    <BusWarteschlangeQueue>
        <Name>'Haltestelle Haltestelle3 BusQueue</Name>
        <DurchschnittsLaenge>0.0</DurchschnittsLaenge>
        <DurchschnittsWarteZeit>0.0</DurchschnittsWarteZeit>
        <WarteschlangenLimit>2147483647</WarteschlangenLimit>
        <WarteschlangenStrategie>FIFO</WarteschlangenStrategie>
    </BusWarteschlangeQueue>
    <PassagierWarteschlangeQueue>
        <Name>Haltestelle: Haltestelle3PassagierQueue</Name>
        <DurchschnittsLaenge>3.02966</DurchschnittsLaenge>
        <DurchschnittsWarteZeit>1.64658</DurchschnittsWarteZeit>
        <WarteschlangenLimit>2147483647</WarteschlangenLimit>
        <WarteschlangenStrategie>FIFO</WarteschlangenStrategie>
    </PassagierWarteschlangeQueue>
    <Point xValue="370.0" yValue="617.0" />
    <BusEinheiten>0</BusEinheiten>
</HaltestelleJ>
<PassagierQueue>
    <Name>Passagierqueue in Bus</Name>
    <DurchschnittsLaenge>5.05134</DurchschnittsLaenge>

```

```
<DurchschnittsWarteZeit>1.05503</DurchschnittsWarteZeit>
<PassagierQueueLimit>50</PassagierQueueLimit>
<PassagierQueueStrategie>FIFO</PassagierQueueStrategie>
</PassagierQueue>
<MaxKapazitaet>50</MaxKapazitaet>
<Position xValue="5154" yValue="5661" />
</Bus>
</BusPanel>
</BusPanels>
```

Listing 1-14: Enthält Referenz auf Buslinien und Konfigurationsdaten

```
<BenoetigteBusse>
  <Bus id="0" />
</BenoetigteBusse>
```

Listing 1-15: Anzahl der eingesetzten Busse

```
</SimuPanel>
```

Listing 1-16: Ende des Elements SimuPanel

Das Element Virtual Grid ist das vierte große Element der XML-Datei, welches jedoch momentan noch keine Funktion hat. Es ist lediglich aufgrund einer möglichen Weiterentwicklung angegeben. Derzeit befinden sich die Daten für das Virtual Grid in dem Element SimuKonfiguration und müssen daher nicht gesondert und explizit abgespeichert werden.

```
<VirtualGrid>-- gets Data from SimuKonfiguration --</VirtualGrid>
```

Listing 1-17: Virtual Grid

```
</Bussimulation>
```

Listing 1-18: Ende des Elements Bussimulation und Ende XML-Datei

1.8 GUI-Optimierung

In der GUI gibt es viele Faktoren, die im Design nicht berücksichtigt wurden und sehr oft zu einer bedienerunfreundlichen Situation führen.

1.8.1 Symbolleiste

Eine erste Möglichkeit war die Einführung einer Symbolleiste, die so eine Art Schnellklickfunktion darstellt. Es sollte dazu beitragen, dass mit der linken Maustaste schnell Objekte eingetragen oder gelöscht werden können.



Abbildung 1-9 Schnellklickfunktion

Um diese Symbolleiste zu realisieren wurde die Klasse View erweitert.

```
static JToggleButton jtogleButton = null;
```

Die Methode

```
private JToolBar createToolBar()
```

wurde um die Buttons erweitert.

- Ziele einfügen,
- Haltestellen einfügen,
- Straßen einfügen
- Objekte löschen,
- Sperrfelder für Felder und Haltestellen

Das nachfolgende Beispiel zeigt einen Ausschnitt für den Button *Ziele einfügen* aus dem Quellcode.

```
// JToggleButton Ziele einfügen
URL targetUrl = ImageUtil.getImageUrl("ziele.gif");
ImageIcon zieleIcon = new ImageIcon(targetUrl);
jtoggleButton = new JToggleButton("", zieleIcon);
jtoggleButton.setActionCommand("Ziele einfügen");
jtoggleButton.addActionListener(new JToggleButtonListener());
jtoggleButton.setSelected(true);
jtoggleButton.setName("Ziele einfügen");
jtoggleButton.setToolTipText("Ziele einfügen");
toolbar.add(jtoggleButton);
```

Der erste Button bekommt den Status `setSelected(true)`, die restlichen Buttons werden auf `false` gesetzt. Jeder Button wird an den speziellen ActionListener `JToggleButtonListener` angehängt.

Es wurde dazu eine neue Klasse `JToggleButtonListener` erstellt, die vom `ActionListener` abgeleitet wird.

```
public class JToggleButtonListener implements ActionListener{
    public void actionPerformed(ActionEvent event) {
        View view = View.getInstance();
    }
}
```

Der `JToggleButtonListener` holt sich eine Instance der `View` und überprüft den Status des jeweiligen Buttons, ob dieser gedrückt wird. Dieses Event wird bei entsprechend gedrückten Button an die Methode in der Klasse `View` übergeben

```
view.ChangeJToggleButtons(event)
```

und dann in der `View` entsprechend ausgewertet.

Dieser Weg wurde deshalb so gewählt, da hier die Möglichkeit besteht den Button direkt bei der Auslösung des Events gleich auszuwerten. In diesem Fall war aber die sofortige Auswertung gar nicht gewünscht, sondern erst bei der Betätigung der linken Maustaste.

Die Methode *ChangeJToggleButtons*

```
public void ChangeJToggleButton(ActionEvent event)
```

holt sich mit der Zeile

```
JToggleButton zieleEinfuegen = findJToggleButtonWithName(View
    .getInstance().getContentPane(), "Ziele einfügen");
```

das Objekt des Buttons zurück. Danach wird über die Referenz des Events die Quelle ermittelt und durch die Methode `isSelected()` überprüft, ob dieser Button gedrückt wurde.

```
jtoggleButton = (JToggleButton) event.getSource();
boolean selected = jtoggleButton.getModel().isSelected();
```

Sobald ein Button gedrückt wird, werden die anderen Buttons auf nicht Aktiv gesetzt. Es hätte auch hier die Möglichkeit bestanden diese über eine `ButtonGroup` zu lösen, doch dann würden hier automatisch alle restlichen Buttons der `ButtonGroup` auf nicht aktiv gesetzt.

Das Problem in diesem Beispiel war es, das man den Button deaktivieren wollte und damit auch die Schnellklickfunktion deaktivieren möchte.

Im Falle eines nochmaligen Drückens wird der Button mit dieser Implementieren einfach auf Aktiv gesetzt ohne das die anderen Buttons davon betroffen sind.

Die Methode

```
private JToggleButton findJToggleButtonWithName(Container container,
    String jToggleButtonText)
```

wurde aus diesem Grunde implementiert um den Container nach `JToggleButtons` mit einem entsprechenden Namen zu durchsuchen und diesen zurückzugeben, da nicht genau ersichtlich ist, welcher Button eigentlich gedrückt wurde.

Um die Schnellfunktion auch zu realisieren muss noch der Event der linken Maustaste hinsichtlich des gedrückten Buttons abgefangen werden.

Dies passiert in der Klasse *MyMarqueeHandler* in der Methode

```
public void mousePressed(final MouseEvent e)
```

Dort wurde das Mouseevent der linken Maustaste abgefangen. Es werden danach wieder die entsprechenden Objekte der Buttons geholt und überprüft ob sie momentan gedrückt sind. Bei Übereinstimmung des aktiven Buttons werden die entsprechenden Methoden für die jeweiligen zugehörigen Funktionen ausgeführt. Es lässt sich spielend eine Schnellklickfunktion einfügen, die immer auf die linke Maustaste reagiert.

1.8.2 Kontextmenü

Das Kontektmenü wird um die gleichen Icons erweitert, das die Symbolleiste bereits enthält.

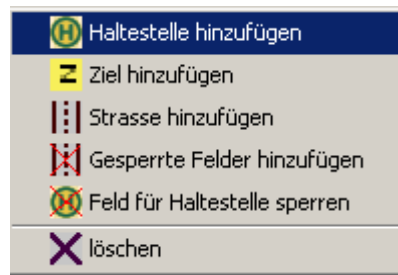


Abbildung 1-10 Kontextmenü

Es wird dazu in der Klasse *PopUpMenu* die Zeilen mit der Hilfsklasse *ImageUtil* eingefügt.

Hier z.B. für den Menüpunkt *Ziele einfügen* realisiert:

```
URL targetUrl = ImageUtil.getImageUrl( "ziele.gif" );
ImageIcon zieleIcon = new ImageIcon( targetUrl );
```

Es wird so beim Kontextmenü eine bessere Übersicht erreicht welcher Menüpunkt eigentlich jetzt genau zuständig ist. Die Icons unterstützen so die Ansicht.

1.8.3 Zeitmessung

Es hat sich sinnvoll erwiesen für die Optimierung und Simulation eine Zeitmessung zu implementieren.

Dazu wurde die Klasse *DateUtil* mit den Methoden

- `setStartTime`
- `setEndTime`
- `getTimeDiffAsString`
- `getTimeDiffExtended`

implementiert.

Die Zeitmessung für die Optimierung wird in der Klasse *OptiControlPanel* im Konstruktor bei der Anbindung des *MouseListener* aktiviert.

```
// Startzeit setzen
DateUtil du = DateUtil.getInstance();
du.setStartTime();
```

Die Zeitmessung wird nach dem Ende der Optimierung beendet, berechnet und über ein *JPanel* im Format(mm:ss) ausgegeben.

Bei der Simulation wird die Zeitmessung in der Klasse *SimuButtonListener* bei der Abfrage des *ButtonEvents* "startSimu" gestartet und nach der Simulation per *JPanel* ausgegeben.

1.8.4 Abbruchmöglichkeit der Simulation und Optimierung

Es soll die Möglichkeiten geben explizit auch über einen Button die Optimierung oder die Simulationen zu stoppen.

Es muss dazu in der Klasse *OptiControlButton* ein Button unter die Progressbar angebunden werden.

```
/** Stop Button */
private JButton stopButton = null;

// Stop Funktion einbauen
stopButton = new JButton("Optimierung stoppen");
stopButton.setSize(new Dimension(100, 20));
stopButton.setToolTipText("Optimierung stoppen");
stopButton.setName("Optimierung stoppen");
```

und mit

```
progressFrame.add(stopButton);
```

an den ProgressFrame angehängt werden.

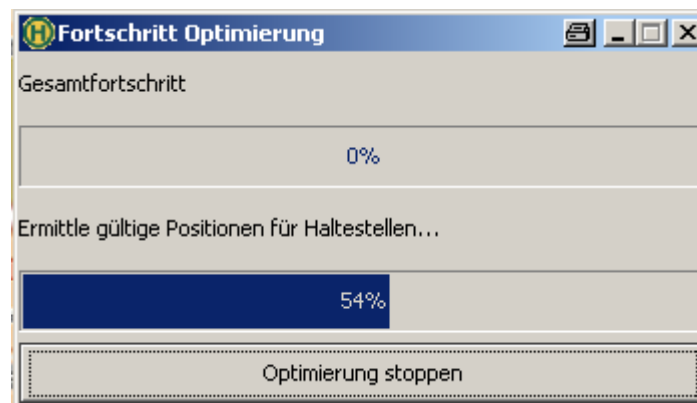


Abbildung 1-11 Optimierung stoppen

Über den ActionListener kann der Button im MouseListener dann abgefangen werden und dadurch den Thread der Optimierung beenden.

```
stopButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        ..
    }
})
```

Bei der Simulation muss der Button in der Klasse SimuButtonListener mit

```
progress.setLayout(new GridLayout(2,0));
progress.add(stopButton);
```

angehängt werden.

Der ActionListener kann den Button im MouseListener dann abgefangen und den Thread der Simulation beenden.

```
stopButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        ..
    }
})
```

1.8.5 Ergebnisdarstellung

Die Ergebnisdarstellung ließ in vielen Dingen zu wünschen. Es waren fehlerhafte Lösungen und auch doppelte Lösungen vorhanden.



Abbildung 1-12 bisherige Lösung der Ergebnisse

Die bisherige Lösung wurde verändert und in eine andere Darstellung gebracht.



Abbildung 1-13 bisherige Ergebnis-Informationen

Ziel war es die Darstellung der Ergebnisse deutlich zu verbessern. Es bot sich in diesem Fall an das ganze per Tabellen zu realisieren.

Für die technische Realisierung wurden dazu einige Klassen neu eingefügt und die GUI entsprechend angepasst.

- `OptiSelectListener.java` → Package: `de.fh_konstanz.simubus.controller`
- `TableListSelectionListener.java` → Package: `de.fh_konstanz.simubus.controller`
- `ColumnProperties.java` → Package: `de.fh_konstanz.simubus.view`
- `SelectTable.java` → Package: `de.fh_konstanz.simubus.view`
- `Table.java` → Package: `de.fh_konstanz.simubus.view`
- `TableColor.java` → Package: `de.fh_konstanz.simubus.view`
- `TableModell.java` → Package: `de.fh_konstanz.simubus.view`

Die Klassenbeschreibung zeigt die Vielfalt der kleinen Änderung in der Darstellung.

Die Klasse *OptimierteSelectListener* ist ein Listener, welche die optimierte Haltestelle bzw. Haltestellen auf der Karte markiert bzw. hervorhebt. Die Klasse bietet auch die Methoden an, dass die Ziele hervorgehoben werden können.

Die Klasse *TableListSelectionListener* beinhaltet die *LösungsInfoTabelle*. Sobald in der Lösungs-Tabelle auf eine Zeile geklickt wird, wird diese Klasse angelegt und die Lösungs Info Tabelle mit entsprechenden Werten gefüllt.

Die Klasse *ColumnProperties* beinhaltet die Spalten Breite und weitere Spalten Eigenschaften werden hier festgelegt.

Die Klasse *SelectTable* ist eine Event Klasse für die Spalten, Zellen, Zeilen. Es beinhaltet die Funktionalität des Kontextmenüs bei der Auswahl einer Zeile, Spalte oder Zelle.

Die Klasse *Table* präsentiert eine Tabelle. Mit dieser Klasse können weitere Tabellen angelegt werden. In diesen Tabellen werden die grundlegenden Eigenschaften, wie Spaltenbreite, Zeilen und Spaltenfarbe geändert in dem es auf die Klasse *TableColor* instanziiert.

Die Klasse *TableColor* ist für das Setzen der Tabellenfarben zuständig.

Die Klasse *TableModell* präsentiert das Modell für die *JTable(Table)* und wird von der *DefaultTableModel* abgeleitet.

Ergebnisse				
Lösung	Datum	Uhrzeit	Gesamtzeit	Durchschnittszeit
1	19.6.2007	14:42:13	12.35	2.06
2	19.6.2007	14:42:13	12.35	2.06

Ergebnis-Informationen	
Haltestellen	Ziele
[x: 318 y: 331]	[x: 266 y: 318]: 3.25 Minuten (größer max. Restgezeit)
[x: 487 y: 214]	[x: 487 y: 292]: 3.9 Minuten (größer max. Restgezeit)
[x: 682 y: 383]	[x: 682 y: 396]: 0.65 Minuten (kleiner max. Restgezeit)
[x: 617 y: 253]	[x: 604 y: 253]: 0.65 Minuten (kleiner max. Restgezeit)
[x: 448 y: 214]	[x: 448 y: 162]: 2.6 Minuten (größer max. Restgezeit)
[x: 500 y: 552]	[x: 500 y: 526]: 1.3 Minuten (größer max. Restgezeit)

Abbildung 1-14 Auswertung der Ergebnisse in einer Tabelle

Das Ergebnis bringt eine deutliche Verbesserung der Übersicht und einiges mehr an Funktionalität gegenüber der Vorversion.

1.9 Logger

Die Klasse „Logger“ wurde erstellt, um das Logging der Applikation zu vereinfachen. Bisher wurden Ausgaben wie Fehlermeldungen oder Debugausgaben einfach auf der Konsole ausgegeben. Dies war auch der Fall, wenn sich die Applikation nicht mehr in der Entwicklung, sondern schon im alltäglichen Alltag befand. In letzterem Fall wurde der Anwender der Software dann mit für ihn scheinbar unsinnigen, technischen Meldungen konfrontiert.

Um die Ausgabe der Logmeldungen zu vereinheitlichen, wurde eine Logger-Klasse erstellt, die für die Ausgabe von Fehler- und Debugmeldungen zu verwalten. Im Quellcode wird der Logger über einen Aufruf der Methode „getInstance()“ gestartet. Dies stellt eine Fabrikmethode des Singleton-Patterns dar. Hierdurch wird sichergestellt, dass es zur Laufzeit nur eine Instanz des Loggers gibt.

Möchte man Logmeldungen ausgeben, so erfolgt dies über die Methode „log()“. Dieser kann die Meldung als Parameter übergeben werden, der zweite, optionale Parameter „loglevel“ bestimmt den Fehlergrad der Meldung. Dies kann entweder „DEBUG“, „ERROR“ oder „FATAL ERROR“ sein.

Die Konfiguration des Loggers kann über die grafische Oberfläche der Anwendung vorgenommen werden. Hierfür muss der Eigenschaften-Dialog geöffnet werden. In diesem lässt sich nun sowohl die Granularität des Loggers, als auch das Ziel des Loggers angeben. Die Granularität unterscheidet zwischen den Werten „Debug, Fehler und „Fatale Fehler“, als Ziele sind sowohl die Konsole, als auch eine Datei wählbar.

1.10 Hilfsklassen

Um einen übersichtlichen Programmcode zu bekommen, wurden einige Hilfsklassen erstellt. Diese stellen statische Methoden zur Verfügung, die in einer oder mehrerer Klassen immer wieder benötigt werden. Folgende Hilfsklassen wurden erstellt:

- `OrUtil`
- `LayoutUtil`
- `ImageUtil`

Die Klasse `OrUtil` stellt Methoden zur Verfügung, die im Zusammenhang der Optimierung benötigt werden. Sie erstellt z.B. das OR-Modell oder rechnet Planquadrat-Koordinate in reale Koordinaten um und umgekehrt. Zur Vereinfachung der Arbeit mit Layout-Managern wurde die Klasse `LayoutUtil` erstellt. Sie enthält Methoden zum Einfügen einer Komponente in einen Container. Da an verschiedenen Stellen im Programm die URL zu Bildern benötigt wird, wurde die Klasse `ImageUtil` erstellt. Sie enthält eine Methode, die zu einem Bildnamen die entsprechende URL liefert.

2 Projektstatus

Die in Kapitel 2 beschriebenen Leistungsmerkmale der Software konnten aus Zeitgründen nicht alle komplett umgesetzt werden. Dieser Teil dient daher der Orientierung über den Projektfortschritt.

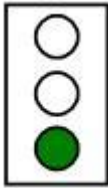
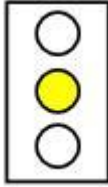
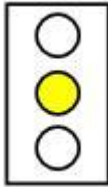
2.1 Übersicht

Die Grundfunktionalität der Software wurde vom Team realisiert. Die Optimierung wurde vollständig mit allen erforderlichen Eigenschaften umgesetzt, und ist damit abgeschlossen. Eine Verknüpfung von Optimierung und Simulation ist vorhanden, jedoch besteht noch erheblicher Entwicklungsbedarf im Simulationsteil.

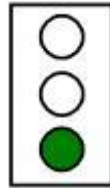
Folgende Features wurden bisher umgesetzt:

Die Ampel in der Status Spalte zeigt den Projektfortschritt jedes einzelnen Features.

- Eine grüne Ampel signalisiert die vollständige Umsetzung bzw. Implementierung eines Features.
- Die gelbe Ampel zeigt an, dass das Feature verbessert bzw. ergänzt werden soll/muss.
- Bei einer roten Ampel wurde nichts implementiert.

Feature	Status	Kommentar
Buslinien einzeichnen mittels Knoten und Kanten		Dies wurde vollständig implementiert.
Graphische Simulation		Kein flüssiger Ablauf. Anzeige eines Busses an der jeweiligen Haltestelle immer dann, wenn ein Bus Ankunftsereignis an Haltestelle stattfindet.
Personenzufluss		In der aktuellen Version ist der Zufluss zufällig und unabhängig von Tageszeiten bzw. bestimmten Haltestellen. Dies wurde mittels eines Passagiergenerators realisiert. Die Nachfolgeversion soll sowohl den manuellen Personenzufluss mit Bestimmung der Start- und Zielhaltestelle ermöglichen, als auch unterschiedliche Passagiergeneratoren bereitstellen die abhängig von der Tageszeiten verschiedene

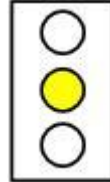
Feste/geplante
Haltestelle
implementieren



Ergebnisse liefern

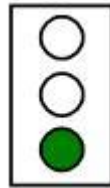
Dies wurde vollständig
implementiert.

Eigenschaften der
Busse



Lediglich die Kapazität und
Geschwindigkeit wird beachtet.
Die Ausmaße der Busse im Bezug
auf bestimmte Streckenabschnitte
wurden nicht beachtet.

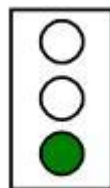
Kapazität Haltestellen



Falls die max. Kapazität einer
Haltestelle erreicht ist und ein
Passagier abgelehnt werden
muss, wird ein Hinweis in den
Report warnungen.txt
geschrieben.

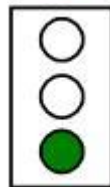
Bei Ablehnung des Passagiers
verschwindet er aus dem System
Zudem wird die Kapazität der
zugehörigen Haltebuchten
überprüft, d. h. bei maximaler
Auslastung der Haltebuchten und
Ankommens eines weiteren
Busses wartet er vor der HS und
eine Meldung erfolgt im Report
warnungen.txt

Berichterstellung nach
Ablauf der Simulation



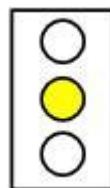
Es werden vier Reports vom
DesmoJ Framework erstellt, sowie
der implementierte Report
warnungen.txt, welcher Abweisung
von Passagieren und Wartezeiten
von Bussen vor HS protokolliert.

Dokumentation



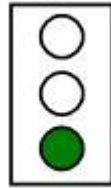
Wurde geschrieben.

Simulationseinstellungen



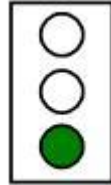
Die Oberfläche bietet die
Möglichkeit die Simulation in 3
verschiedenen
Geschwindigkeitsstufen ablaufen
zu lassen. Zudem kann der
Simulationsstart und das
Simulationsende eingestellt
werden und dadurch die Dauer der
Simulation festgelegt werden.

Knotenpunkte
verschiebbar machen



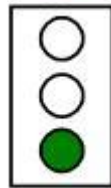
Dies wurde vollständig
implementiert.

Laden und Speichern



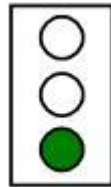
Das Laden und Speichern ist
vollständig implementiert. Die
Präsentationsobjekte werden in
ein XML File geschrieben.

Warnung bei
Überschreitung der
maximalen Restgezeit



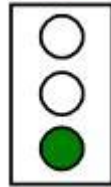
Wenn die angegebene maximale
Restgezeit nicht ausreicht um
von einem Ziel zu einer Straße zu
kommen, wird die Restriktion der
maximalen Restgezeit außer
Kraft gesetzt. In diesem Fall soll
eine Warnmeldung ausgegeben
werden.

Maximalen Restgezeit



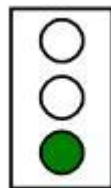
Wann wird die maximale
Restgezeit überschritten? Diese
Frage muss im Handbuch
beantwortet werden.

Reporterklärung



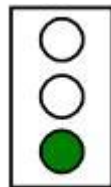
Im Handbuch muss (besser)
erklärt werden wie die
Auswertungen zu lesen sind. Im
Moment ist nicht klar ersichtlich
wie Busse fahren.

Sperrung einzelner
Planquadrante



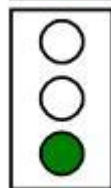
Es soll möglich sein einzelne
Planquadrante für Haltestellen zu
sperrern um somit Baustellen oder
ähnliches abbilden zu können. In
der Optimierung sollen dann keine
Haltestellen auf diese Quadrate
gesetzt werden.

Beschleunigen der
Berechnung der OR-
Matrix



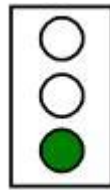
Das Lösen der OR-Matrix soll
optimiert werden durch das
entfernen der überflüssigen
Nullstellen.

Implementierung einer
Hupe



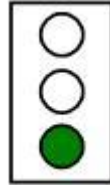
Durch das Implementieren des
Geräusches einer Hupe als Signal
für das erfolgreiche berechnen des
OR-Modells.

Schnellklicksymbolleiste



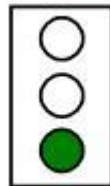
Die wichtigsten Präsentationsobjekte können durch die Schnellklicksymbolleiste einfach und schnell ausgewählt werden.

Zeitmessung und Ausgabe der Dauer der Optimierung und der Simulation



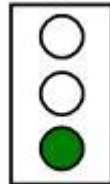
Die benötigte Zeit für die Dauern der Optimierung und der Simulation werden ausgegeben.

Erweiterung des Kontextmenüs



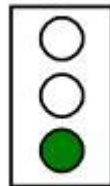
Das Kontextmenü wurde um die Symbole der Schnellstartsymbolleiste erweitert.

Abbruchbutton für die Optimierung und die Simulation



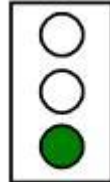
Damit die Optimierung, als auch die Simulation auf einfachem Wege abgebrochen werden können, wurden jeweils Abbruchbutton eingefügt.

Strassen bleiben erhalten



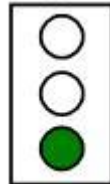
Die Strassen werden nach einer Optimierung nicht mehr verworfen, sondern bleiben erhalten.

Rundungen im Ergebnis



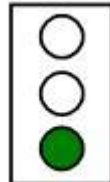
Die Ergebnisse einer Optimierung werden sinnvoll auf zwei Kommastellen gerundet.

Umstellen der Ergebnisse auf Tabelle



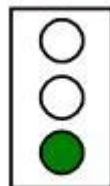
Die Ergebnisse einer Optimierung werden in der Darstellung übersichtlich in Tabellen gegliedert und ausgegeben.

Sortierung der Ergebnisse



Die Ergebnisse können ausgehend von den Ergebnissen nach jeder Spalte sortiert werden.

Ergebnis- kopieren



Einzelne Ergebnisse oder ganze Tabellen können ausgewählt werden und in die Zwischenablage kopiert werden.

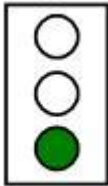
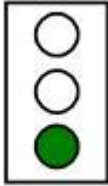
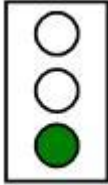
Ergebnisdarstellung		Ergebnisse werden durch verschiedene Farben hervorgehoben.
Ergebnis-Info		Ergebnisse werden bei Auswahl detailliert in der Ergebnis-InfoBox unterhalb der Ergebnis- Tabelle angezeigt.
Hervorheben der Präsentationsobjekte im Ergebnis		Beim auswählen eines Ergebnisses werden die zugehörigen Präsentationsobjekte in der Oberfläche hervorgehoben.

Tabelle 1: Implementierte Features

2.2 Verbesserungsmöglichkeiten

Die aktuelle Version der Bussimulation deckt die Grundfunktionalität ab. Es gibt jedoch noch viele Programmteile, die verbessert werden könnten bzw. müssen, um die Software benutzerfreundlicher zu gestalten. In diesem Kapitel werden Hinweise auf Programmteile gegeben, die nicht fertig gestellt wurden oder bei denen Verbesserungsbedarf besteht.

2.2.1 Layout

Da das Programm von der Vorgruppe übernommen wurde, wurde wenig Wert auf das Layout gelegt. Allerdings wurden sämtliche Fenster ohne einen entsprechenden Layout-Manager implementiert. Dies bedeutet, dass die Positionen und Größen der Elemente im Fenster fest einprogrammiert sind. Somit hat man nur bei einer Auflösung von 1280 * 1024 Pixel eine vernünftige Darstellung. Das ist natürlich nicht sehr anwenderfreundlich und sollte behoben werden. Dazu muss jedes Fenster mit einem Layout-Manager versehen werden und die Positionierung der Elemente durch ihn erledigt werden.

2.2.2 Positionieren von Elementen

Für die Optimierung der Haltestellen wird davon ausgegangen, dass eine Haltestelle nur auf einer Strasse positioniert werden kann. Ebenfalls wird eine feste Haltestelle nur dann berücksichtigt, wenn sie auf einer Strasse eingefügt wird. Auf der anderen Seite macht es keinen Sinn, dass ein Planquadrat ein Ziel und eine Haltestelle ist. Es muss also dafür gesorgt werden, dass außer Haltestellen und Strassen sich keine zwei Elemente auf einem Planquadrat befinden. Dies ist momentan noch möglich und sollte behoben werden.

2.2.3 Klasse Strassennetz

Die Vorgruppe hat eine Klasse `Strassennetz` verwendet, um alle Strassen, Haltestellen, etc. in einer globalen Klasse zu haben. Dadurch kann an jeder Stelle

des Programms auf diese zugegriffen werden. Da für die Strassen, etc. die Anforderung bestand diese als Kanten und Knoten einzeichnen zu können, musste ein virtuelles Grid erstellt werden, welches die Einteilung in Planquadrate übernimmt. Da hier ebenfalls die Informationen zur Verfügung stehen, welche Planquadrate welche Eigenschaft haben, sollte die Klasse `Strassennetz` gelöscht und deren Funktionalität in die Klasse `VirtualGrid` eingefügt werden. Momentan müssen die Informationen in beiden Klassen vorhanden sein, was bedeutet, dass bei jeder Änderung der Eigenschaften immer beide Klassen aktualisiert werden müssen. Dies ist eine potenzielle Fehlerquelle und sollte behoben werden.

2.2.4 Undo/Redo

Eine weitere Anforderung, welche nicht fertig implementiert werden konnte, ist die Undo/Redo Funktion. Es wurde geschafft, diese Funktion in der Oberfläche zu integrieren und soweit zu implementieren, dass entsprechende Änderungen auf der Karte gelöscht bzw. wiederhergestellt wurden. Allerdings werden diese Änderungen nicht in der Tabelle angezeigt oder auch die Klassen `Strassennetz` und `VirtualGrid` nicht aktualisiert.

2.2.5 Optimierung

Beim Testen der Optimierung ist aufgefallen, dass das Programm bei größeren Modellen (mehr als 8 Ziele) stehen bleibt. Die Ursache liegt wahrscheinlich darin begründet, dass, umso mehr Ziele vorhanden sind, die Lösungsalternativen exponentiell steigen, und somit der Algorithmus zum finden der Lösung abbricht. Es ist zu prüfen, ob der Algorithmus entweder so angepasst werden kann, dass dieser Fehler nicht mehr auftritt, oder das OR-Modell durch zusätzliche Restriktion deutlich weniger Lösungen bietet und somit der Algorithmus zuverlässig arbeitet.

2.2.6 Refactoring Business Klassen

Da der Simulations- und Optimierungsteil unabhängig voneinander entwickelt wurden, gab es Überschneidungen in den Business Klassen. Dies wurde leider zu spät bemerkt, sodass es zwei ähnliche Klassen für eine Buslinie und Haltestelle gibt. Für Buslinien gibt es die Klassen `Linie` und `BusLinie` und für eine Haltestelle gibt es die Klassen `Haltestelle` und `HaltestelleJ`. Für die Buslinie sollte es auf jeden Fall möglich sein diese zu vereinheitlichen, bei den Klassen für die Haltestelle könnte es evtl. Probleme geben. Da `HaltestelleJ` von der Klasse `Entity` abgeleitet ist, was für die Erstellung der Reports durch das DesmoJ-Framework notwendig ist, kann es sein dass hier beide Klassen bestehen müssen.

2.2.7 Passagiergenerator

Bisher sind die Einstellungen des Passagiergenerators wie in Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.** beschrieben statisch und nicht durch den Benutzer änderbar. Für eine möglichst reale Simulation ist es aber erforderlich, den Ankunftsstrom der Passagiere von mehreren Faktoren abhängig zu machen. Zu diesen zählen z.B. Tageszeit, Wochentag sowie die Priorität der Haltestellen.

Dazu sollte dem Benutzer eine Möglichkeit zur Eingabe dieser Daten gegeben werden, z. B. Festlegung des Ankunftsstroms von Passagieren in Minuten an eine Haltestelle zu einer gegebenen Tageszeit.

2.2.8 Umsteigen von Passagieren

Im realen Leben kommt es vor, dass ein Passagier sein Ziel nicht auf direktem Wege erreichen kann und sein Ziel nur durch Umsteigen auf eine andere Buslinie erreicht. Dieser Fall wurde in der aktuellen Version aus Zeitgründen nicht berücksichtigt, ist aber für eine realistische Simulation erforderlich. Beim Umsteigen ist auch noch zu beachten, dass manche Personen den nächsten Bus nehmen, welcher ihr Ziel anfährt. Andere Personen jedoch wählen den Bus bzw. die Buslinie aus, welche ihr Ziel in der kürzesten Zeit anfahren.

2.2.9 Simulationsablauf steuern / Simulationsgeschwindigkeit

Die Simulation kann bisher nur gestartet werden und läuft dann bis zum Ende der Simulation durch. Für eine komfortable Simulationssteuerung wäre ein Stopp- und Pausefunktionalität wünschenswert.

Da die Simulationsgeschwindigkeit abhängig von der Anzahl der Ereignisse im System ist, wäre es wünschenswert dem Benutzer abhängig vom Szenario mehrere sinnvolle Geschwindigkeitsstufen anzubieten.

2.2.10 Anzeige des Simulationsablaufs

Die Anzeige im Panel Simulationsablauf gibt Aufschluss, was im System gerade passiert. Es werden lediglich ein paar für den Benutzer interessante Daten angezeigt. Dazu gehört die Linie, die Haltestelle, die Anzahl der Personen im Bus, die Anzahl wartender Personen an der Haltestelle sowie die Anzahl der ein- und aussteigenden Passagiere. Je nach Simulationsgeschwindigkeit machen bestimmte Wertanzeigen keinen Sinn mehr, da sich die Werte zu schnell ändern. In der nächsten Version der Bussimulation könnte die Anzeige deshalb noch benutzerfreundlicher gestaltet werden.

2.2.11 Eigenschaften von Bussen und Streckenabschnitten

An Engstellen im Straßennetz kann es in der Realität zu Problemen kommen, da nicht zwei Busse gleichzeitig eine Engstelle passieren können oder Busse ab einer bestimmten Länge Kurven nicht mehr passieren können. Für diese Fälle ist es erforderlich, die Eigenschaften der Busse (Länge, Breite) und bestimmter Streckenabschnitte (Breite) im System festzuhalten. Die Simulation sollte dann auftretende Probleme erkennen und entsprechende Einträge im Report generieren.

2.2.12 Auswählen der Buslinie für einen Passagier

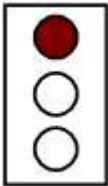
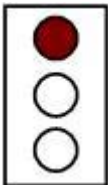
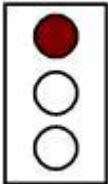
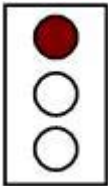
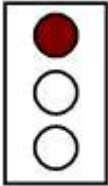
Bei der Suche nach einer zugehörigen Buslinie für den Passagier, wird aktuell die ERSTE Buslinie, welche sowohl Start-, als auch Zielhaltestelle anfährt genommen. Sollten mehrere Buslinien diese beiden Haltestellen bedienen, ist es wünschenswert die Auswahl nach gewissen Kriterien durchzuführen. Beispielsweise kürzeste Fahrzeit zwischen Start- und Zielhaltestelle unter Berücksichtigung einer maximal zumutbaren Umsteigeproblematik. (Beispiel: Wer steigt zehnmal um, wenn er dafür 5 Minuten früher am Ziel ist?!)

2.2.13 Manueller Personenzufluss an Haltestellen

Um die Realität tatsächlich abbilden zu können, muss die Möglichkeit bestehen den Personenzufluss an eine Haltestelle zu einer bestimmten Tageszeit manuell festzulegen. Dies soll in Intervallen von maximal 30 Minuten möglich sein.

2.2.14 Weitere Anforderungen

Die folgende Tabelle gibt einen Überblick über Anforderungen, die in Zusammenarbeit mit dem österreichischen Architekturbüro und Herrn Grütz erarbeitet wurden.

Bezeichnung	Status	Beschreibung
Engpässe		<p>Auf Streckenabschnitten sollen Engpässe definiert werden können. Für diese Engpässe soll protokolliert werden ob bzw. wann sich Busse dort treffen. Kollisionen von zwei Bussen sollen durch Vorschläge für verschiedene Taktungen der Linien verhindert werden.</p> <p>Trick: Durch das Einfügen einer Pseudo-Haltestelle könnte man testen, ob an dieser Haltestelle zwei Busse gleichzeitig ankommen und somit einen Engpass simulieren.</p>
Ratschläge für Taktung/Busgrößen		Das Programm soll auf Grund der Simulationsergebnisse Vorschläge zur Taktung der Linien bzw. zur Busgröße machen.
Überschrittene Restgezeit		Nach dem optimieren wird für jedes Ziel die Restgezeit angegeben. Dahinter steht eine Meldung die angibt, ob maximale Restgezeit überschritten wurde. Diese Meldung scheint nicht immer das richtige anzuzeigen. Das muss korrigiert werden.
Verschiedene Busgrößen		Die Simulation soll so erweitert werden, dass die verschiedenen Busgrößen in das Programm mit einbezogen werden und Szenarien mit verschiedenen Busgrößen durchsimuliert werden kann.
Gewichtung für Ziele		Es soll möglich sein, Ziele verschiedene Gewichtungen zuzuweisen. Diese Gewichtungen sollen angeben wie viele Menschen zu einem bestimmten Ziel wollen. Ziele mit größerer Gewichtung sollen in der Optimierung bevorzugt werden.

Einbahnstraßen		Es soll möglich sein Straßen als Einbahnstraßen zu markieren bzw. unterschiedliche Linienführung unterstützen. Das heißt, dass eine Linie in der Richtung von Ost nach West eine andere Strecke bedient als in der Richtung von West nach Ost
Eigenschaften von Straßenabschnitten		In der Simulation soll es möglich sein, Straßenabschnitten Eigenschaften wie Breite, Höchstgeschwindigkeit und Baustellen anzugeben. Diese Eigenschaften sollen bei der Simulation berücksichtigt werden.
Variable Anzahl von Lösungen		Es soll in der Optimierung möglich sein die Anzahl der zu berechnenden Lösungen anzugeben. Dem Benutzer soll vor der Optimierung die voraussichtliche Zeit für die Berechnung angegeben werden.
Stadtplan-Editor		Der größte Aufwand zu Beginn ist die Eingabe eines aussagekräftigen Datenmodells des Stadtgebietes. Ein möglichst einfacher und benutzerfreundlicher Stadtplan-Editor ist hier unbedingt erforderlich. Eine Doppeleingabe derselben Daten (einmal Straßennetz für die Haltestellensimulation, noch mal dasselbe Netz für die Buslinien) sollte vermieden werden.
Import für Fahrplan-Tabellen		Der nächste große Schritt aus Anwendersicht ist dann die Eingabe eines Fahrplanes. Eine Eingabemöglichkeit für besondere Fahrplanmerkmale wie zum Beispiel unregelmäßige Taktung, Sonn- und Feiertagsfahrpläne ist nötig. Des Weiteren wäre eine Import-Möglichkeit für die (digital vorhandenen) Fahrplan-Tabellen wünschenswert.
Schulbus-Management		Eine besondere Rolle für den Busverkehr nehmen in Eisenstadt die zahlreichen Schüler ein. Hierfür wäre ein Programmmodul, das ein intelligentes Schulbus-Management innerhalb der Simulation erlaubt wünschenswert. Hier für ist gutes Zahlenmaterial wie zum Beispiel woher die Schüler kommen, welche Schulen sie besuchen, Schulbeginn und Schulschluss vorhanden.
Kalibrierung		Um die Simulationsergebnisse mit der realen Situation in Zusammenhang zu bringen, soll es möglich sein nach einer Simulation durch Anpassung von Parametern die Simulation an die reale Situation anzupassen.

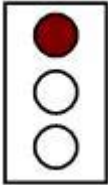
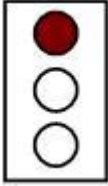
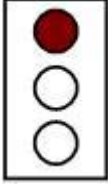
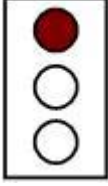
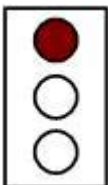
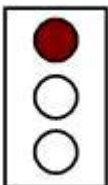
Minimalabstand für Haltestellen		Es ist sinnvoll für Haltestellen einen minimalen Abstand angeben zu können, da es bei der Optimierung durchaus auftreten kann, dass Haltestellen direkt neben einander platziert werden.
Frequenz der Buslinien		Bei der Simulation der Buslinie kann man zwar die Frequenz der Buslinie angeben, aber diese wird nicht korrekt abgespeichert.
Stellschraube für den Personenzufluss pro Haltestelle		Der Personenzufluss wird bisher mittels eines Passagiergenerators produziert. Es ist allerdings von Vorteil für jede Haltestelle den Personenzufluss explizit einstellen zu können.
Stellschraube für die Taktung der Busfrequenz		Die Taktung der Buslinien ist editierbar, es ist jedoch nicht möglich mehrere Busse auf derselben Linie hintereinander fahren zu lassen
Zuordnung von Zielen zu einzelnen Passagieren.		Implementierung einer Klasse Passagier für die Zuordnung von Zielen, Umsteigemöglichkeiten und Gewohnheiten von Passagieren. Das Ziel ist die bessere Simulation von Verhaltensweisen und Gewohnheiten der Passagiere, zudem können die Nötigen Daten aus Erfahrungswerten generiert werden.
Generierung von Erfahrungswerten		Durch das Sammeln von Erfahrungswerten für die einzelnen Passagiere, lassen sich Simulationen realitätsgetreuer gestalten. Die Erfahrungswerte könnten sich gemäß dem Umfang in eine Datei abspeichern lassen (z.B. CSV) oder in eine Datenbank.

Tabelle 2: Weitere Features


3 Anhang

Schema xml.xsd


Elements

[AktuelleAufloesung](#)
[AnzahlFelder](#)
[BackgroundImage](#)
[BenoetigteBusse](#)
[Breite](#)
[Bus](#)
[BusEinheiten](#)
[BusGeschwindigkeit](#)
[BusLinie](#)
[BusPanel](#)
[BusPanels](#)
[Bussimulation](#)
[BusWarteschlangeQueue](#)
[Color](#)
[DurchschnittsLaenge](#)
[DurchschnittsWarteZeit](#)
[EinstellungenFrame](#)
[EndPoint](#)
[Feldgroesse](#)
[GehGeschwindigkeit](#)
[GesamtKosten](#)
[Geschwindigkeit](#)
[GesperrteFelder](#)
[GesperrtesFeld](#)
[GesperrteStrasse](#)
[GroesseEditor](#)
[GroesseFeldelement](#)
[GroesseFelder](#)
[GroessePlanquadrat](#)
[Haltestelle](#)
[HaltestelleJ](#)
[Haltestellen](#)
[HaltestellenJ](#)
[Image](#)
[Kapazitaet](#)
[KostenPunktBisZiel](#)
[KostenStartBisPunkt](#)
[Laenge](#)
[Linie](#)
[Linien](#)
[MaxKapazitaet](#)
[MaxKapazitaetPassagiere](#)
[Name](#)
[PassagierQueue](#)
[PassagierQueueLimit](#)
[PassagierQueueStrategie](#)
[PassagierWarteschlangeQueue](#)
[Planquadrant](#)
[PlanquadratGesperrteStrasse](#)
[PlanquadratStrasse](#)
[Point](#)
[Position](#)
[SimuKonfiguration](#)
[SimuPanel](#)
[StartPoint](#)
[Strasse](#)
[Strassennetz](#)
[Teilstrecke](#)
[Teilstrecken](#)
[VirtualGrid](#)
[WarteschlangenLimit](#)
[WarteschlangenStrategie](#)
[Wiederkehrzeit](#)
[Ziel](#)
[Ziele](#)

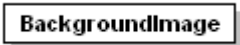
element AktuelleAufloesung

diagram						
properties	content	complex				
used by	element	SimuKonfiguration				
attributes	Name	Type	Use	Default	Fixed	annotation
	resHeight		required			
	resWidth		required			
source	<pre><xs:element name="AktuelleAufloesung"> <xs:complexType> <xs:attribute name="resHeight" use="required"/> <xs:attribute name="resWidth" use="required"/> </xs:complexType> </xs:element></pre>					

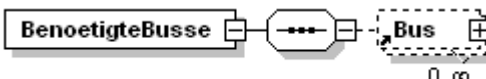
element AnzahlFelder

diagram					
used by	element	SimuPanel			
source	<xs:element name="AnzahlFelder"/>				

element BackgroundImage

diagram						
properties	content	complex				
used by	element	SimuPanel				
attributes	Name	Type	Use	Default	Fixed	annotation
	height		required			
	url		required			
	width		required			
source	<pre><xs:element name="BackgroundImage"> <xs:complexType> <xs:attribute name="height" use="required"/> <xs:attribute name="url" use="required"/> <xs:attribute name="width" use="required"/> </xs:complexType> </xs:element></pre>					

element BenoetigteBusse

diagram					
properties	content	complex			
children	<u>Bus</u>				
used by	element	<u>SimuPanel</u>			
source	<xs:element name="BenoetigteBusse">				

```
<xs:complexType>
  <xs:sequence>
    <xs:element ref="Bus" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

element Breite

diagram

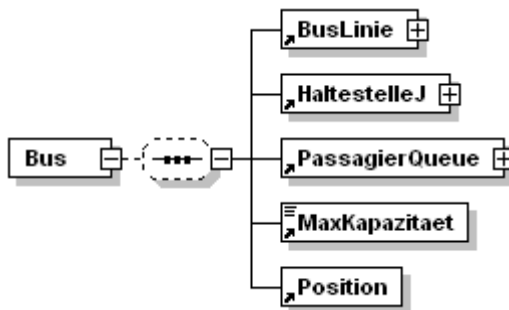


used by element [Teilstrecke](#)

source `<xs:element name="Breite"/>`

element Bus

diagram



properties content complex

children [BusLinie](#) [HaltestelleJ](#) [PassagierQueue](#) [MaxKapazitaet](#) [Position](#)

used by elements [BenotigteBusse](#) [BusPanel](#)

attributes	Name	Type	Use	Default	Fixed	annotation
	id		required			

source `<xs:element name="Bus">
 <xs:complexType>
 <xs:sequence minOccurs="0">
 <xs:element ref="BusLinie"/>
 <xs:element ref="HaltestelleJ"/>
 <xs:element ref="PassagierQueue"/>
 <xs:element ref="MaxKapazitaet"/>
 <xs:element ref="Position"/>
 </xs:sequence>
 <xs:attribute name="id" use="required"/>
 </xs:complexType>
</xs:element>`

element BusEinheiten

diagram

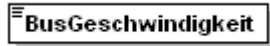


used by element [HaltestelleJ](#)

source `<xs:element name="BusEinheiten"/>`

element **BusGeschwindigkeit**

diagram

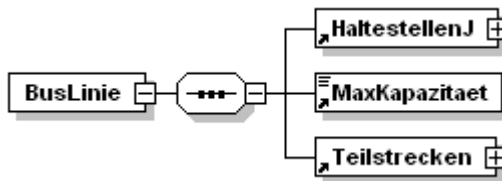


used by element [SimuKonfiguration](#)

source `<xs:element name="BusGeschwindigkeit"/>`

element **BusLinie**

diagram



properties content complex

children [HaltestellenJ](#) [MaxKapazitaet](#) [Teilstrecken](#)

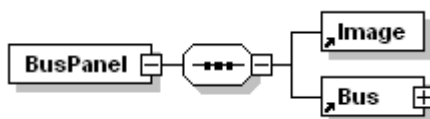
used by element [Bus](#)

attributes	Name	Type	Use	Default	Fixed	annotation
	id		required			

source `<xs:element name="BusLinie">
<xs:complexType>
<xs:sequence>
<xs:element ref="HaltestellenJ"/>
<xs:element ref="MaxKapazitaet"/>
<xs:element ref="Teilstrecken"/>
</xs:sequence>
<xs:attribute name="id" use="required"/>
</xs:complexType>
</xs:element>`

element **BusPanel**

diagram



properties content complex

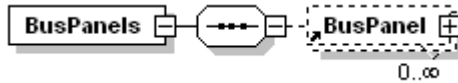
children [Image](#) [Bus](#)

used by element [BusPanels](#)

source `<xs:element name="BusPanel">
<xs:complexType>
<xs:sequence>
<xs:element ref="Image"/>
<xs:element ref="Bus"/>
</xs:sequence>
</xs:complexType>
</xs:element>`

element **BusPanels**

diagram



properties content complex

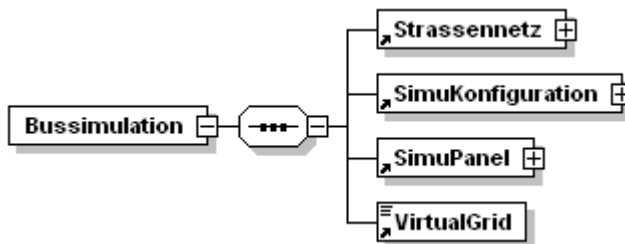
children [BusPanel](#)

used by element [SimuPanel](#)

```
<xs:element name="BusPanels">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="BusPanel" minOccurs="0" maxOccurs="unbounded"/>
      <!-- BusPanel -->
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element **Bussimulation**

diagram



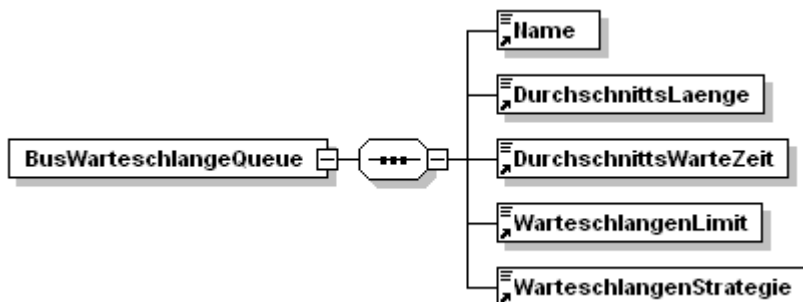
properties content complex

children [Strassennetz](#) [SimuKonfiguration](#) [SimuPanel](#) [VirtualGrid](#)

```
<xs:element name="Bussimulation">
  <!-- Rootelement, Bussimulation -->
  <xs:complexType>
    <xs:sequence>
      <!-- Child Elements, were written out -->
      <xs:element ref="Strassennetz"/>
      <xs:element ref="SimuKonfiguration"/>
      <xs:element ref="SimuPanel"/>
      <xs:element ref="VirtualGrid"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element **BusWarteschlangeQueue**

diagram



properties content complex

children [Name](#) [DurchschnittsLaenge](#) [DurchschnittsWarteZeit](#) [WarteschlangenLimit](#) [WarteschlangenStrategie](#)

used by element [HaltestelleJ](#)

```
<xs:element name="BusWarteschlangeQueue">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Name"/>
      <xs:element ref="DurchschnittsLaenge"/>
      <xs:element ref="DurchschnittsWarteZeit"/>
      <xs:element ref="WarteschlangenLimit"/>
      <xs:element ref="WarteschlangenStrategie"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```


element Color

diagram 

used by element [Linie](#)

```
<xs:element name="Color"/>
```

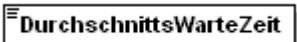
element DurchschnittsLaenge

diagram 

used by elements [BusWarteschlangeQueue](#) [PassagierQueue](#) [PassagierWarteschlangeQueue](#)

```
<xs:element name="DurchschnittsLaenge"/>
```

element DurchschnittsWarteZeit

diagram 

used by elements [BusWarteschlangeQueue](#) [PassagierQueue](#) [PassagierWarteschlangeQueue](#)

```
<xs:element name="DurchschnittsWarteZeit"/>
```

element EinstellungenFrame

diagram 


properties content complex

used by element [SimuKonfiguration](#)

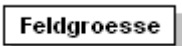
attributes	Name	Type	Use	Default	Fixed	annotation
	height		required			
	width		required			

```
<xs:element name="EinstellungenFrame">
  <xs:complexType>
    <xs:attribute name="height" use="required"/>
    <xs:attribute name="width" use="required"/>
  </xs:complexType>
</xs:element>
```

element EndPoint

diagram						
properties	content	complex				
used by	element	Teilstrecke				
attributes	Name	Type	Use	Default	Fixed	annotation
	xValue		required			
	yValue		required			
source	<pre><xs:element name="EndPoint"> <!-- End of a Point --> <xs:complexType> <xs:attribute name="xValue" use="required"/> <xs:attribute name="yValue" use="required"/> </xs:complexType> </xs:element></pre>					

element Feldgroesse

diagram						
properties	content	complex				
used by	elements	GesperrteStrasse Strasse				
attributes	Name	Type	Use	Default	Fixed	annotation
	height		required			
	width		required			
source	<pre><xs:element name="Feldgroesse"> <xs:complexType> <xs:attribute name="height" use="required"/> <xs:attribute name="width" use="required"/> </xs:complexType> </xs:element></pre>					


element GehGeschwindigkeit

diagram			
used by	element	SimuKonfiguration	
source	<pre><xs:element name="GehGeschwindigkeit"/></pre>		

element Gesamtkosten

diagram			
used by	elements	GesperrteStrasse Strasse	
source	<pre><xs:element name="Gesamtkosten"/></pre>		

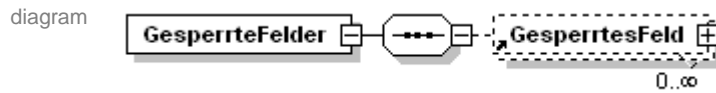
element Geschwindigkeit

diagram						
---------	---	--	--	--	--	--

used by element [Teilstrecke](#)

source `<xs:element name="Geschwindigkeit"/>`

element **GesperrteFelder**



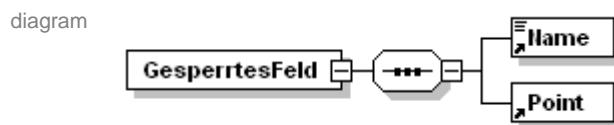
properties content complex

children [GesperrtesFeld](#)

used by element [Strassennetz](#)

source `<xs:element name="GesperrteFelder">
<xs:complexType>
<xs:sequence>
<xs:element ref="GesperrtesFeld" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>`

element **GesperrtesFeld**



properties content complex

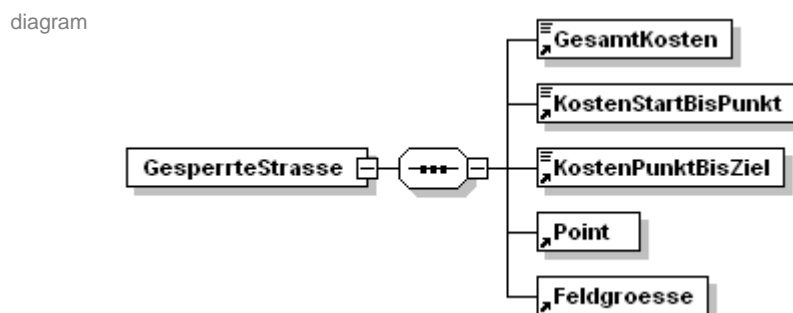
children [Name](#) [Point](#)

used by element [GesperrteFelder](#)

attributes	Name	Type	Use	Default	Fixed	annotation
id			required			

source `<xs:element name="GesperrtesFeld">
<!-- Gesperrte Haltestelle -->
<xs:complexType>
<xs:sequence>
<xs:element ref="Name"/>
<xs:element ref="Point"/>
</xs:sequence>
<xs:attribute name="id" use="required"/>
</xs:complexType>
</xs:element>`

element **GesperrteStrasse**



properties content complex

children [GesamtKosten](#) [KostenStartBisPunkt](#) [KostenPunktBisZiel](#) [Point](#) [Feldgroesse](#)

used by element [PlanquadratGesperrteStrasse](#)

attributes	Name	Type	Use	Default	Fixed	annotation
	art		required			
	id		required			

source

```
<xs:element name="GesperrteStrasse">
  <xs:complexType>
    <xs:sequence>
      <!-- Childs -->
      <xs:element ref="GesamtKosten"/>
      <xs:element ref="KostenStartBisPunkt"/>
      <xs:element ref="KostenPunktBisZiel"/>
      <xs:element ref="Point"/>
      <xs:element ref="Feldgroesse"/>
    </xs:sequence>
    <xs:attribute name="art" use="required"/>
    <xs:attribute name="id" use="required"/>
    <!-- Kind of the barred Way -->
    <!-- Id -->
  </xs:complexType>
</xs:element>
```


element **GroesseEditor**

diagram 

used by element [SimuPanel](#)

source `<xs:element name="GroesseEditor"/>`

element **GroesseFeldelement**

diagram 

used by element [SimuKonfiguration](#)

source `<xs:element name="GroesseFeldelement"/>`


element **GroesseFelder**

diagram 

used by element [SimuPanel](#)

source `<xs:element name="GroesseFelder"/>`

element **GroessePlanquadrat**

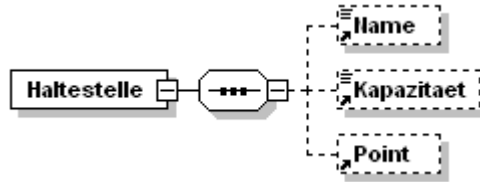
diagram 

used by element [SimuKonfiguration](#)

source `<xs:element name="GroessePlanquadrat"/>`

element Haltestelle

diagram



properties content complex

children [Name](#) [Kapazitaet](#) [Point](#)

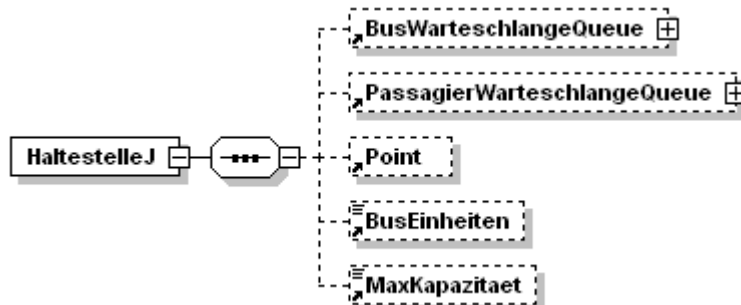
used by element [Haltestellen](#)

attributes	Name	Type	Use	Default	Fixed	annotation
id			required			

```
<xs:element name="Haltestelle">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Name" minOccurs="0"/>
      <xs:element ref="Kapazitaet" minOccurs="0"/>
      <xs:element ref="Point" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" use="required"/>
  </xs:complexType>
</xs:element>
```

element HaltestelleJ

diagram



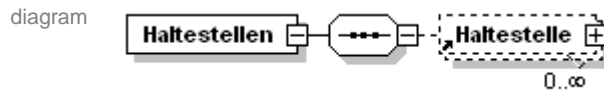
properties content complex

children [BusWarteschlangeQueue](#) [PassagierWarteschlangeQueue](#) [Point](#) [BusEinheiten](#) [MaxKapazitaet](#)

used by elements [Bus](#) [HaltestellenJ](#)

```
<xs:element name="HaltestelleJ">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="BusWarteschlangeQueue" minOccurs="0"/>
      <xs:element ref="PassagierWarteschlangeQueue" minOccurs="0"/>
      <xs:element ref="Point" minOccurs="0"/>
      <xs:element ref="BusEinheiten" minOccurs="0"/>
      <xs:element ref="MaxKapazitaet" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element Haltestellen



properties content complex

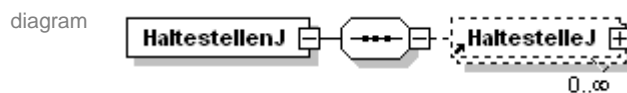
children [Haltestelle](#)

used by elements [Linie](#) [SimuPanel](#) [Strassennetz](#)

source

```
<xs:element name="Haltestellen">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Haltestelle" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element HaltestellenJ



properties content complex

children [HaltestelleJ](#)

used by element [BusLinie](#)

source

```
<xs:element name="HaltestellenJ">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="HaltestelleJ" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element Image



properties content complex

used by element [BusPanel](#)

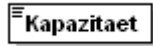
attributes	Name	Type	Use	Default	Fixed	annotation
	height		required			
	url		required			
	width		required			

source

```
<xs:element name="Image">
  <xs:complexType>
    <xs:attribute name="height" use="required"/>
    <xs:attribute name="url" use="required"/>
    <xs:attribute name="width" use="required"/>
  </xs:complexType>
</xs:element>
```

element **Kapazitaet**

diagram



used by element [Haltestelle](#)

source `<xs:element name="Kapazitaet"/>`

element **KostenPunktBisZiel**

diagram



used by elements [GesperrteStrasse](#) [Strasse](#)

source `<xs:element name="KostenPunktBisZiel"/>`

element **KostenStartBisPunkt**

diagram

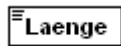


used by elements [GesperrteStrasse](#) [Strasse](#)

source `<xs:element name="KostenStartBisPunkt"/>`

element **Laenge**

diagram

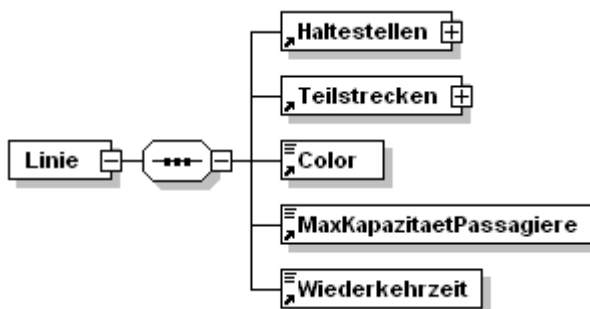


used by element [Teilstrecke](#)

source `<xs:element name="Laenge"/>`

element **Linie**

diagram



properties content complex

children [Haltestellen](#) [Teilstrecken](#) [Color](#) [MaxKapazitaetPassagiere](#) [Wiederkehrzeit](#)

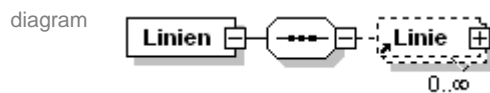
used by element [Linien](#)

attributes	Name	Type	Use	Default	Fixed	annotation
	id		required			

source `<xs:element name="Linie">
<xs:complexType>
<xs:sequence>`

```
<xs:element ref="Haltestellen"/>
<xs:element ref="Teilstrecken"/>
<xs:element ref="Color"/>
<xs:element ref="MaxKapazitaetPassagiere"/>
<xs:element ref="Wiederkehrzeit"/>
</xs:sequence>
<xs:attribute name="id" use="required"/>
</xs:complexType>
</xs:element>
```

element Linien



properties content complex

children [Linie](#)

used by element [Strassennetz](#)

source

```
<xs:element name="Linien">
  <!-- Linien -->
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Linie" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element MaxKapazitaet

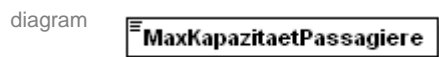


used by elements [Bus](#) [BusLinie](#) [HaltestelleJ](#)

source

```
<xs:element name="MaxKapazitaet"/>
```

element MaxKapazitaetPassagiere



used by element [Linie](#)

source

```
<xs:element name="MaxKapazitaetPassagiere"/>
```

element Name



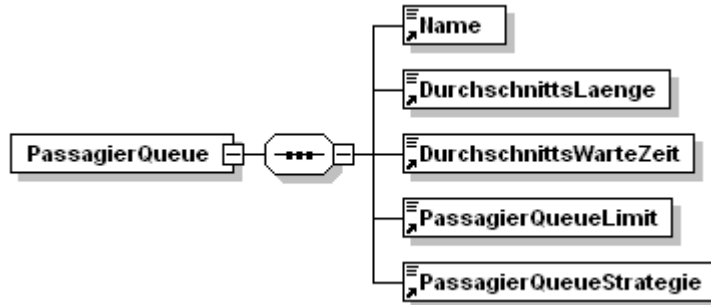
used by elements [BusWarteschlangeQueue](#) [GesperrtesFeld](#) [Haltestelle](#) [PassagierQueue](#) [PassagierWarteschlangeQueue](#) [Teilstrecke](#) [Ziel](#)

source

```
<xs:element name="Name"/>
```

element **PassagierQueue**

diagram



properties content complex

children [Name](#) [DurchschnittsLaenge](#) [DurchschnittsWarteZeit](#) [PassagierQueueLimit](#) [PassagierQueueStrategie](#)

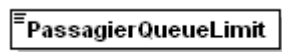
used by element [Bus](#)

source

```
<xs:element name="PassagierQueue">
  <!-- Warteschlange, Passagiere -->
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Name"/>
      <xs:element ref="DurchschnittsLaenge"/>
      <xs:element ref="DurchschnittsWarteZeit"/>
      <xs:element ref="PassagierQueueLimit"/>
      <xs:element ref="PassagierQueueStrategie"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element **PassagierQueueLimit**

diagram

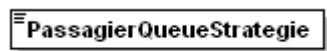


used by element [PassagierQueue](#)

source `<xs:element name="PassagierQueueLimit"/>`

element **PassagierQueueStrategie**

diagram

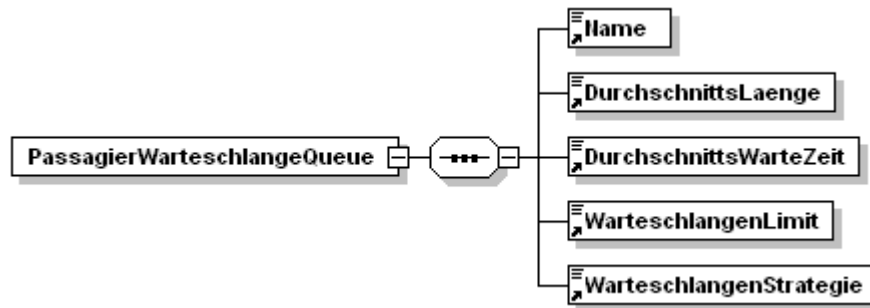


used by element [PassagierQueue](#)

source `<xs:element name="PassagierQueueStrategie"/>`

element **PassagierWarteschlangeQueue**

diagram



properties content complex

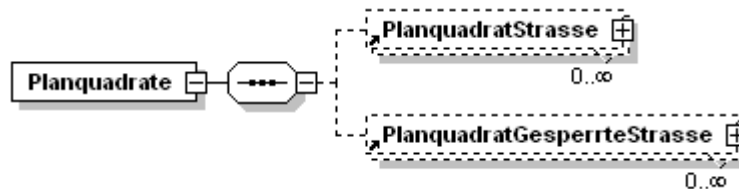
children [Name](#) [DurchschnittsLaenge](#) [DurchschnittsWarteZeit](#) [WarteschlangenLimit](#) [WarteschlangenStrategie](#)

used by element [HaltestelleJ](#)

```
source <xs:element name="PassagierWarteschlangeQueue">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Name"/>
      <xs:element ref="DurchschnittsLaenge"/>
      <xs:element ref="DurchschnittsWarteZeit"/>
      <xs:element ref="WarteschlangenLimit"/>
      <xs:element ref="WarteschlangenStrategie"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element **Planquadrat**

diagram



properties content complex

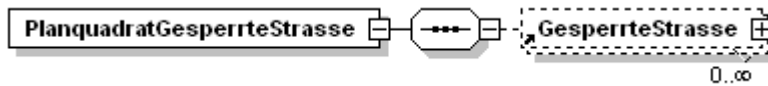
children [PlanquadratStrasse](#) [PlanquadratGesperrteStrasse](#)

used by element [Strassennetz](#)

```
source <xs:element name="Planquadrat">
  <!-- Planquadrat -->
  <xs:complexType>
    <xs:sequence>
      <!-- optional sequences -->
      <xs:element ref="PlanquadratStrasse" minOccurs="0" maxOccurs="unbounded"/>
      <!-- Strassen -->
      <xs:element ref="PlanquadratGesperrteStrasse" minOccurs="0" maxOccurs="unbounded"/>
      <!-- Gesperrte Strassen -->
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element PlanquadratGesperrteStrasse

diagram



properties content complex

children [GesperrteStrasse](#)

used by element [Planquadrat](#)

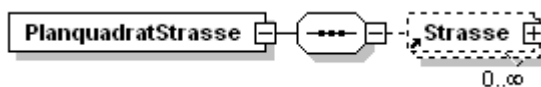
attributes	Name	Type	Use	Default	Fixed	annotation
	id		required			

source

```
<xs:element name="PlanquadratGesperrteStrasse">
  <!-- Gesperrte Strassen -->
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="GesperrteStrasse" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" use="required"/>
  <!-- Id -->
</xs:complexType>
</xs:element>
```

element PlanquadratStrasse

diagram



properties content complex

children [Strasse](#)

used by element [Planquadrat](#)

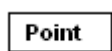
attributes	Name	Type	Use	Default	Fixed	annotation
	id		required			

source

```
<xs:element name="PlanquadratStrasse">
  <!-- Strassen -->
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Strasse" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" use="required"/>
  <!-- Id -->
</xs:complexType>
</xs:element>
```

element Point

diagram



properties content complex

used by elements [GesperrtesFeld](#) [GesperrteStrasse](#) [Haltestelle](#) [HaltestelleJ](#) [Strasse](#) [Ziel](#)

attributes	Name	Type	Use	Default	Fixed	annotation
	xEnd					
	xStart					
	xValue					
	yEnd					
	yStart					

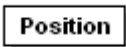
yValue

source

```
<xs:element name="Point">
  <!-- Point, with Coordinates -->
  <xs:complexType>
    <xs:attribute name="xEnd"/>
    <xs:attribute name="xStart"/>
    <xs:attribute name="xValue"/>
    <xs:attribute name="yEnd"/>
    <xs:attribute name="yStart"/>
    <xs:attribute name="yValue"/>
  </xs:complexType>
</xs:element>
```

element Position

diagram



properties content complex

used by element [Bus](#)

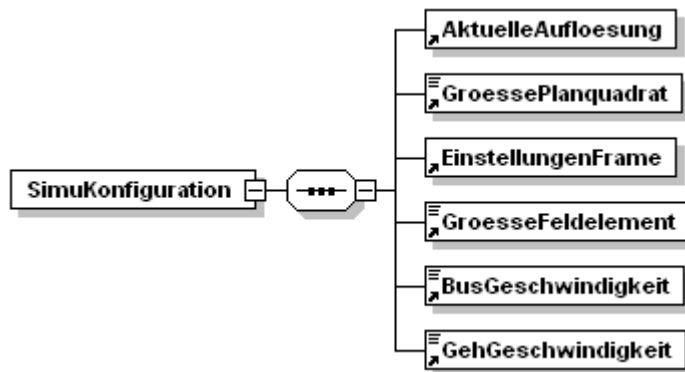
attributes	Name	Type	Use	Default	Fixed	annotation
	xValue		required			
	yValue		required			

source

```
<xs:element name="Position">
  <xs:complexType>
    <xs:attribute name="xValue" use="required"/>
    <xs:attribute name="yValue" use="required"/>
  </xs:complexType>
</xs:element>
```

element SimuKonfiguration

diagram



properties content complex

children [AktuelleAufloesung](#) [GroessePlanquadrat](#) [EinstellungenFrame](#) [GroesseFeldelement](#) [BusGeschwindigkeit](#) [GehGeschwindigkeit](#)

used by element [Bussimulation](#)

source

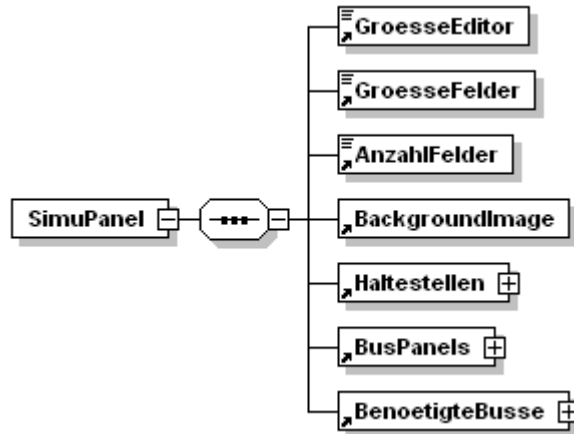
```
<xs:element name="SimuKonfiguration">
  <!-- Simulation Configuration -->
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="AktuelleAufloesung"/>
      <xs:element ref="GroessePlanquadrat"/>
      <xs:element ref="EinstellungenFrame"/>
      <xs:element ref="GroesseFeldelement"/>
      <xs:element ref="BusGeschwindigkeit"/>
      <xs:element ref="GehGeschwindigkeit"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



```
</xs:complexType>
</xs:element>
```

element **SimuPanel**

diagram



properties content complex

children [GroesseEditor](#) [GroesseFelder](#) [AnzahlFelder](#) [BackgroundImage](#) [Haltestellen](#) [BusPanels](#) [BenoetigteBusse](#)

used by element [Bussimulation](#)

```
source <xs:element name="SimuPanel">
  <!-- SimuPanel, important for the simulation part -->
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="GroesseEditor"/>
      <xs:element ref="GroesseFelder"/>
      <xs:element ref="AnzahlFelder"/>
      <xs:element ref="BackgroundImage"/>
      <xs:element ref="Haltestellen"/>
      <xs:element ref="BusPanels"/>
      <xs:element ref="BenoetigteBusse"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element **StartPoint**

diagram



properties content complex

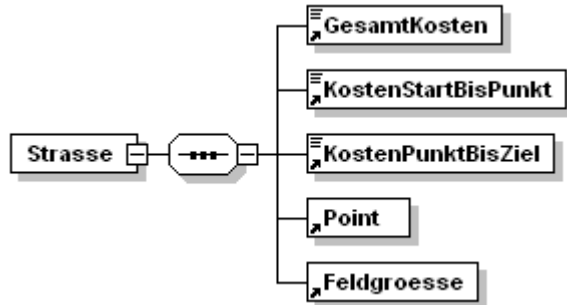
used by element [Teilstrecke](#)

attributes	Name	Type	Use	Default	Fixed	annotation
	xValue		required			
	yValue		required			

```
source <xs:element name="StartPoint">
  <!-- Start of a Point -->
  <xs:complexType>
    <xs:attribute name="xValue" use="required"/>
    <xs:attribute name="yValue" use="required"/>
  </xs:complexType>
</xs:element>
```

element **Strasse**

diagram



properties content complex

children [GesamtKosten](#) [KostenStartBisPunkt](#) [KostenPunktBisZiel](#) [Point](#) [Feldgroesse](#)

used by element [PlanquadratStrasse](#)

attributes	Name	Type	Use	Default	Fixed	annotation
	id		required			
	art		required			

source

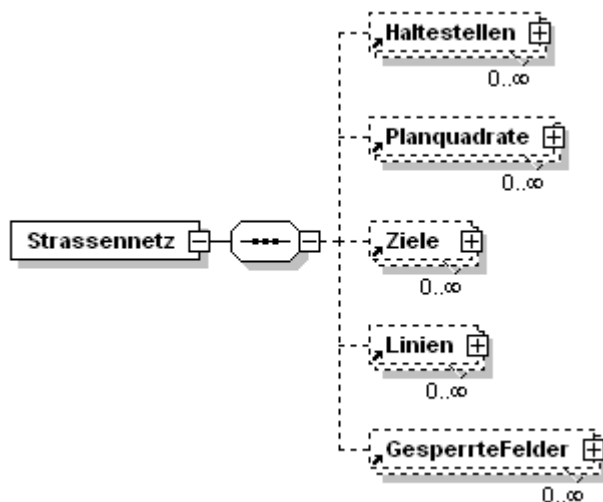
```

<xs:element name="Strasse">
  <!-- Element Strasse -->
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="GesamtKosten"/>
      <!-- Childs -->
      <xs:element ref="KostenStartBisPunkt"/>
      <xs:element ref="KostenPunktBisZiel"/>
      <xs:element ref="Point"/>
      <xs:element ref="Feldgroesse"/>
    </xs:sequence>
    <xs:attribute name="id" use="required"/>
    <xs:attribute name="art" use="required"/>
  <!-- Id -->
  <!-- Kind of Planquadrat -->
</xs:complexType>
</xs:element>

```

element **Strassennetz**

diagram



properties content complex

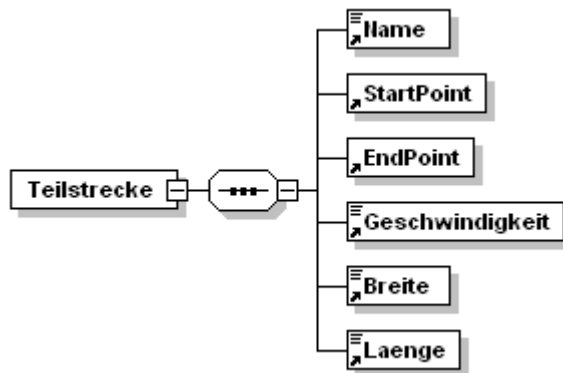
children [Haltestellen](#) [Planquadrante](#) [Ziele](#) [Linien](#) [GesperrteFelder](#)

used by element [Bussimulation](#)

```
source <xs:element name="Strassennetz">
  <!-- Strassennetz -->
  <xs:complexType>
    <xs:sequence>
      <!-- Child Elements -->
      <xs:element ref="Haltestellen" minOccurs="0" maxOccurs="unbounded"/>
      <!-- optional Childs -->
      <xs:element ref="Planquadrante" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="Ziele" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="Linien" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="GesperrteFelder" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element Teilstrecke

diagram



properties content complex

children [Name](#) [StartPoint](#) [EndPoint](#) [Geschwindigkeit](#) [Breite](#) [Laenge](#)

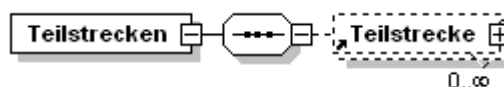
used by element [Teilstrecken](#)

attributes	Name	Type	Use	Default	Fixed	annotation
id			required			

```
source <xs:element name="Teilstrecke">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Name"/>
      <xs:element ref="StartPoint"/>
      <xs:element ref="EndPoint"/>
      <xs:element ref="Geschwindigkeit"/>
      <xs:element ref="Breite"/>
      <xs:element ref="Laenge"/>
    </xs:sequence>
    <xs:attribute name="id" use="required"/>
  </xs:complexType>
</xs:element>
```

element Teilstrecken

diagram



properties content complex

children [Teilstrecke](#)

used by elements [BusLinie](#) [Linie](#)

source

```
<xs:element name="Teilstrecken">
  <!-- Teilstrecken -->
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Teilstrecke" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

element VirtualGrid

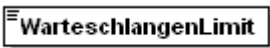
diagram 

used by element [Bussimulation](#)

source

```
<xs:element name="VirtualGrid"/>
```

element WarteschlangenLimit

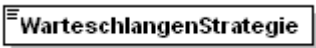
diagram 

used by elements [BusWarteschlangeQueue](#) [PassagierWarteschlangeQueue](#)

source

```
<xs:element name="WarteschlangenLimit"/>
```

element WarteschlangenStrategie

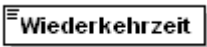
diagram 

used by elements [BusWarteschlangeQueue](#) [PassagierWarteschlangeQueue](#)

source

```
<xs:element name="WarteschlangenStrategie"/>
```

element Wiederkehrzeit

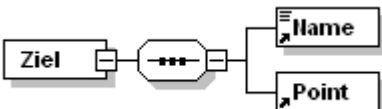
diagram 

used by element [Linie](#)

source

```
<xs:element name="Wiederkehrzeit"/>
```

element Ziel

diagram 

properties content complex

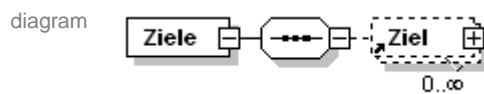
children [Name](#) [Point](#)

used by element [Ziele](#)

attributes	Name	Type	Use	Default	Fixed	annotation
id	id		required			

```
source <xs:element name="Ziel">
  <!-- goal -->
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Name"/>
      <!-- name of the goal -->
      <xs:element ref="Point"/>
      <!-- point of the goal, where is set up -->
    </xs:sequence>
    <xs:attribute name="id" use="required"/>
  </xs:complexType>
</xs:element>
```

element Ziele



properties content complex

children [Ziel](#)

used by element [Strassennetz](#)

```
source <xs:element name="Ziele">
  <!-- Ziele -->
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Ziel" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```