Grafisches Editieren von Straßen- und ÖV-Netzmodellen

Für die Verkehrssimulation

Bachelorarbeit

für die Prüfung zum Bachelor of Science
im Studiengang Verkehrswesen
Vertiefungsrichtung Planung und betrieb
an der Technischen Universität Berlin
Fakultät V – Verkehrs- und Maschinensysteme

Berlin, 10.10.2014

Erstgutachter: Prof. Dr. Kai Nagel
Zweitgutachter: Dipl.-Inf. Michael Zilske

Vorgelegt von
Nico Kühnel, geb. 18.03.1992
Matr.-Nr. 332634

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den 09.10.2014

……

Unterschrift

I. Inhaltsverzeichnis

I. Inhaltsverzeichnis I
II. Abbildungsverzeichnis III
III. Tabellenverzeichnis IV
IV. Abkürzungsverzeichnis V
1. Einleitung 1
2. Grundlagen 3
2.1. Netzmodelle zur Darstellung der Infrastruktur 3
2.2. OpenStreetMap als Datengrundlage 6
2.2.1. Datentypen in OpenStreetMap 7
2.2.2. Typisierung und Eigenschaften der Elemente durch Tags 8
2.2.3. Abbildung des Straßen- und Liniennetzes in OpenStreetMap 10
2.3. Visuelles Editieren von OpenStreetMapDaten mit JOSM 15
2.3.1. Prinzip 16
2.3.2. Plugin Interface 17
2.4. MATSIM 18
2.4.1. IV-Datenformat 19
2.4.2. ÖV-Datenformat 21
2.5. Bisheriges Erstellen und Bearbeiten von MATSim-Netzmodellen 22
2.6. Graphischer Editor für MATSim – Motivation und Anforderung 25
3. Graphischer Editor für MATSim Netze auf Basis von JOSM 29
3.1. Datenhaltung der OSM und MATSim Daten 29
3.2. Konvertierung 30
3.2.1. IV / Straßenmodell 31
3.2.2. ÖV / Routen 37
3.2.3. Networklistener / Online Prinzip 43
3.3. Visualisierung 46

3.3.1. Toggle Dialog 46
3.3.2. MapRenderer 48
3.4. Validierung 49
3.5. Speichern von Netzwerken und ÖV-Routen 50
3.6. Importieren von Netzwerken 51
3.7. Leeres Netzwerk erstellen 53
3.8. Preferences / Einstellungsoptionen 54
4. Anwendung Beispielszenario 56
5. Zusammenfassung 62
V. Literaturverzeichnis VI
VI. Glossar X
VII. Anhang XII

II. Abbildungsverzeichnis

Abbildung 1: a) einfache, b) parallele, c) Schleifen- sowie d) Folgen von Kanten 5
Abbildung 2: stark zusammenhängender, bewerteter Graph 5
Abbildung 3: Übersicht des OSM-Datenmodells 9
Abbildung 4: Elemente der transit schedule 22
Abbildung 5: In den Daten der Layers enthaltene Informationen 30
Abbildung 6: Flussdiagramm des Konvertierungsablaufs für Straßen 33
Abbildung 7: Filtern der Nodes – Verbindungen davor (links) und danach (rechts) 35
Abbildung 8: MATSim Tagging Preset für Ways 37
Abbildung 9: U-Bahnhaltestelle Olympia-Stadion ist Element entgegengesetzter Routen 38
Abbildung 10: Unstetigkeiten in der Route des RE 5 39
Abbildung 11: Flussdiagramm des Konvertierungsablaufs für ÖV-Routen 42
Abbildung 12: MATSim Toggle Dialog 47
Abbildung 13: Ways mit MATSim Informationen, vom neuen Renderer dargestellt 49
Abbildung 14: Flussdiagramm der Importfunktion 52
Abbildung 15: Anpassen der Converting Defaults 54
Abbildung 16: Verwendetes Stadtgebiet Ulms 56
Abbildung 17: Ulm mit OSM Tagging Schema 57
Abbildung 18: Auszug aus der Relationsliste der Linie 21 in Ulm 58
Abbildung 19: Ulm mit MATSim Tag Schema 59
Abbildung 20: Lage der neuen Brücke 60

III. Tabellenverzeichnis

Tabelle 1: etablierte Werte für das ‚highway' Tag und deren Bedeutung 11
Tabelle 2: Für Straßen verwendete Tags 12
Tabelle 3: etablierte Werte für das railway Tag 13
Tabelle 4: Tag-Werte mit dem puclic_transport Schlüssel 14
Tabelle 5: Tags zur Beschreibung von Routen 14
Tabelle 6: Elemente eines Network und deren Attribute 20
Tabelle 7: MATSim Tagstruktur 32
Tabelle 8: erfasste events für die Echtzeitkonvertierung 44

IV. Abkürzungsverzeichnis

.jar	*Java Archive*
GPS	*Global Positioning System*
ID	*Identifizierung*
IV	*Individualverkehr*
JOSM	*Java OpenStreetMap (Editor)*
MATSim	*Multi Agent Transport Simulation*
OSM	*OpenStreetMap*
ÖV	*öffentlicher Verkehr*
POI	*Point of Interest*
XML	*eXtensible Markup Language*

1. Einleitung

In der modernen Verkehrsplanung kommen komplexe Verkehrssimulationsprogramme zum Einsatz. Sie verwenden Modelle des Verkehrs, die die Realität abbilden. Um gut kalibrierte Modelle zu erhalten, die den realen Verkehr möglichst treffend vorhersagen, werden - neben anderen Voraussetzungen - Netzmodelle der zugrundeliegenden Verkehrswege benötigt.

Schon seit die Menschheit ortsgebunden wirtschaftete und sesshaft wurde, entwickelten sich zeitgleich mit dem Tauschhandel erste Verkehrsnetze (vgl. Müller, 1998 S.4). Auch Heute verlaufen die meisten Ortsveränderungen auf relativ festgelegten Wegen: einer Infrastruktur. Aufgrund der Komplexität und Menge an Informationen ist es nicht sinnvoll, diese eins-zu-eins auf ein (Computer-)Modell zu übertragen. Eine Lösung besteht darin, sie durch Netzwerke darzustellen, die aus Kanten und Knoten bestehen. Dabei reicht eine rein topologische Beschreibung für eine aus ökonomischer Sicht agierende Modellierung nicht aus. Es werden zusätzliche Kosten-Charakteristika benötigt, damit ein Netzwerk das Angebot für eine bestimmte (Verkehrs-)Nachfrage darstellen kann (vgl. Nagurney, 2007, S. 3).

Die Netzwerke sollten für realitätsnahe Simulationsergebnisse bestmögliche Abbildungen der jeweiligen Infrastruktur sein. Damit menschliche Anwender das zuverlässig und intuitiv beurteilen können, bietet es sich an, die räumlichen Komponenten der benötigten Simulationsgrundlagen visuell darzustellen und zu bearbeiten. Dabei sollte der Anwender stets wissen, was die jeweiligen Darstellungen abbilden und welche Auswirkungen von ihm durchgeführte Interaktionen auf das zugrundeliegende Modell haben. Neben der Bearbeitung von Netzen für die Simulation kann eine Visualisierung auch sinnvoll sein um Netzwerke zu analysieren.

Im Rahmen dieser Arbeit wurde ein Programm entwickelt, dass diese Ideen aufgreift und für die Verkehrssimulation MATSim (‚Multi Agent Transport Simulation‘) umsetzt. MATSim ist ein Java-basiertes Programm, bei dem einzelne Agenten simuliert werden und sich auf einem Netzwerk fortbewegen. Dabei können Agenten neben Individualverkehr (IV) auch den öffentlichen Verkehr (ÖV) nutzen. Für beide Verkehrsarten werden räumliche Informationen zu Verkehrswegen, Kreuzungen oder auch Haltestellen benötigt. Bisher wurden Netzwerke vor allem durch Java-Code aus OpenStreetMap (OSM) Daten konvertiert

(siehe Abschnitt 2.5). Die Ausgabe erfolgt in einer XML-Datei, die zumeist manuell als Textdatei bearbeitet wird.

Die erwähnten Daten stammen vom OSM-Projekt, das frei verfügbare Informationen zu verschiedensten geographischen Information anbietet. Da sie bereits umfangreich für viele Orte der Erde gesammelt wurden, bieten sie eine gute Grundlage für eine simple Abbildung der Realität auf einem Computermodell. Ein weiterer Vorteil ist, dass es bereits einen Editor gibt, der die OSM Daten visuell darstellen und bearbeiten kann. Der in Java geschriebene, JOSM (Java OpenStreetMap) genannte Editor bietet außerdem die Möglichkeit, eigene Funktionalität durch das Programmieren von Plugins hinzuzufügen.

Ziel dieser Arbeit ist es, eine Anwendung zu entwerfen, die, auf Grundlage der verfügbaren OSM Daten und der in JOSM vorhandenen Plugin Schnittstelle, ein graphisches Editieren von Straßen- und ÖV-Netzmodellen ermöglicht. Zunächst werden die Grundlagen und Anforderungen für ein solches Programm beschrieben. Daraufhin folgt eine Erläuterung des Aufbaus und der Funktionsweise des entworfenen JOSM Plugins. Dieses wird anschließend in einem Beispielszenario getestet. Probleme, Erkenntnisse und weitere Entwicklungen werden am Schluss zusammengefasst.

2. Grundlagen

In den folgenden Abschnitten werden zu Beginn die theoretischen Grundlagen für Netzmodelle des Verkehrs erläutert. Anschließend werden die verwendeten Datenmodelle von OSM, JOSM und MATSim vorgestellt. Es wird ein Überblick über die bisherige Bearbeitung von MATSim Netzmodellen gegeben, woraufhin sich eine Übersicht über die sich ergebenden Ansprüche und Vorüberlegungen für ein graphisches Editierprogramm anschließen.

2.1. Netzmodelle zur Darstellung der Infrastruktur

Die verkehrliche Infrastruktur ist räumlich immer in ihre Umwelt eingebettet. Sie stellt geographische Beziehungen, wie zum Beispiel Handelsrouten, zwischen verschiedenen Orten dar. Erst durch die Herauslösung aus ihrer Umwelt verbleibt ein Netz, das für die Verkehrsplanung relevant ist. Die für die rechnergestützte Bearbeitung von Verkehrsnetzen notwendige Digitalisierung des Netzes geht mit einem Informationsverlust einher. Außerdem erschwert sie eine sofortige Erkennung einer Struktur für das menschliche Auge. Eine graphische Visualisierung der Netze kann indes die Bearbeitung der Netze vereinfachen. (vgl. Clausen, 1981 S. 23)

Auch in der Verkehrssimulation werden digitalisierte Netzmodelle genutzt, die die vorhandene oder geplante Infrastruktur vereinfacht abbilden. Oft basieren diese Modelle auf einem Netzwerk (engl.: network), das nur die für die Simulation relevanten Informationen beinhaltet und eine übersichtliche Darstellung der Infrastruktur ermöglicht.

Netz(werk)

Grundsätzlich können Netzwerke als Systeme beschrieben werden, „deren zugrundeliegende Struktur sich mathematisch als Graph modellieren lässt und die über Mechanismen zu ihrer Organisation verfügen. Der Graph besteht aus einer Menge von Elementen (Knoten), die mittels Verbindungen (Kanten) miteinander verbunden sind.“ (“Netzwerk” 2014) Diese Definition lässt sich auf Verkehrsnetze übertragen. Clausen versteht unter einem Verkehrsnetz „ein System aus vielen miteinander verbundenen, sich kreuzenden und voneinander abzweigenden, dem Verkehr dienenden Linien“ (Clausen, 1981 S.10). Dieses Netz solle bevorzugt durch einen Graphen beschrieben werden, der die

Relationen einzelner Elemente zueinander beschreibt. Die Graphentheorie „entspricht damit weitgehend der Abbildung von Verkehrsnetzen in abstrahierter Form“ (Clausen, 1981 S.24).

Zusammenfassend lässt sich sagen, dass ein Verkehrsnetz aus verbundenen Elementen besteht, die in bestimmen Beziehungen zueinander stehen und einer Transportfunktion dienen. Das Netz kann durch einen Graphen dargestellt werden, der die einzelnen Elemente als Kanten, Knoten und deren Beziehungen zueinander veranschaulicht.

Graph

Graphen werden von Clausen wie folgt definiert:

„Ein Graph besteht aus einer Menge N von Knoten, einer Menge K von Kanten, die mit N elementefremd ist: $N \cap K = Ø$, und einer Abbildung φ, […] durch die jedem Element aus K, also jeder Kante $e_{ij} \in K$, eindeutig zwei Elemente v_i und v_j aus N zugeordnet werden. Das Tripel $G = (N, K, φ)$ ist ein Graph.“ (Clausen, 1981 S. 12)

Es ist also festzuhalten, dass ein Graph aus Knoten und Kanten besteht, wobei Kanten Verbindungen zwischen zwei Knoten darstellen. In einem Verkehrsnetz entsprechen die Knoten den Verbindungsstücken, zum Beispiel Kreuzungen oder Bahnübergänge oder auch geometrischen Stützknoten die Richtungsänderungen einleiten, während die Kanten die Verkehrswege (Straßen, Schienen) zwischen den Kreuzungen repräsentieren.

Graphen können visualisiert werden, indem die Knoten als Punkte in einer Ebene und die Kanten als Linien zwischen diesen Knoten dargestellt werden (Clausen, 1981 S. 12). Dies bietet sich auch für Verkehrsnetze an, bei denen zum Beispiel Straßen zwischen verschiedenen Orten als zweidimensionale Linien veranschaulicht werden. Auf diesem Prinzip beruhen auch die meisten Karten der Straßenatlanten oder die Liniennetzpläne öffentlicher Verkehrsbetriebe, bei denen die Linien als Aneinanderreihung von Kanten zwischen den einzelnen Halteknoten dargestellt werden.

Die Graphentheorie ermöglicht weitergehende Beschreibungen, von denen nur einige, für diese Arbeit relevante, genannt und in Abbildung 1 veranschaulicht sind. Von einem *endlichen Graphen* ist die Rede, wenn Knoten sowie Kanten aus einer endlichen Menge bestehen. Das trifft für Graphen von Verkehrsnetzen zu. Auch *Schleifen* kommen in Verkehrsnetzen vor (z.B. Wendeschleifen). Sie entstehen durch Kanten, die einen Knoten mit sich selbst verbinden, deren zwei Knoten also dieselben sind. Mehrere Kanten, die dieselben zwei Knoten verbinden, sind *parallel* und können nach Richtungen oder Verkehrsmittel getrennte Fahrbahnen beschreiben. Kanten, die weder parallel noch

Schleifen sind, heißen *einfache Kanten*. Eine Reihe von benachbarten Kanten, die mindestens einen gemeinsamen Knoten besitzen ist eine *Kantenfolge*. Diese wiederum ist ein *Kantenzug*, wenn keine Kante doppelt in ihm auftaucht. Taucht außerdem auch kein Knoten doppelt auf, wird von einem *Kantenweg* oder einer *Route* gesprochen. Bei *Zusammenhängenden Graphen* ist jedes beliebige Knotenpaar über eine Kantenfolge verbunden.

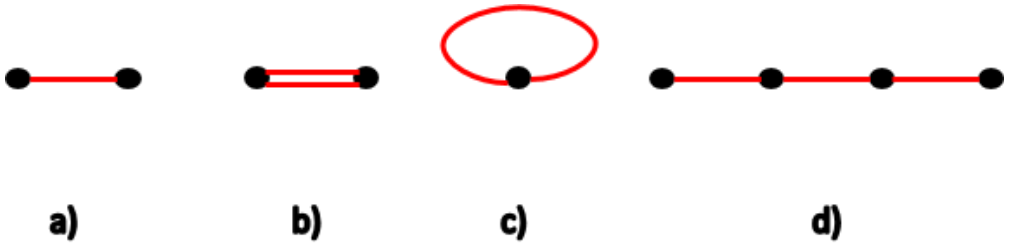

Abbildung 1: a) einfache, b) parallele, c) Schleifen- sowie d) Folgen von Kanten
Quelle: eigene Darstellung

Die Kanten eines Graphen können außerdem *gerichtet* sein (nur in eine Richtung befahrbare Straßen). Dabei sind die zugehörigen Knotenpaare geordnet – es gibt einen Anfangs- sowie einen Endknoten für jede Kante. Ein Graph, der nur aus gerichteten Kanten besteht, ist ein *gerichteter Graph*. Ist ein gerichteter Graph zusätzlich *zusammenhängend*, besteht also zwischen zwei beliebigen Knoten eine Kantenfolge entlang der Kantenrichtungen, dann ist der Graph *stark zusammenhängend*. Letztlich gibt es auch *bewertete Graphen*. In diesen sind den Kanten beliebige Werte zugeordnet. In Verkehrsnetzen sind dies üblicherweise die Kapazität, Länge und erlaubte Geschwindigkeit einer Straße (vgl.Clausen, 1981 S. 12 ff.). Abbildung 2 veranschaulicht einen stark zusammenhängenden, bewerteten Graphen. Jeder Knoten kann von jedem anderen Knoten erreicht werden. Die sechs Kanten sind jeweils gerichtet und mit einem Wert C_i gewichtet.

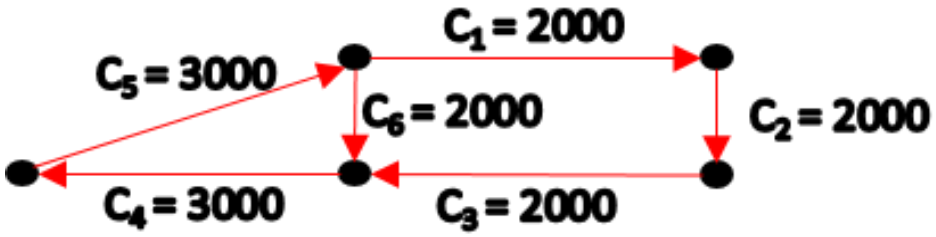

Abbildung 2: stark zusammenhängender, bewerteter Graph
Quelle: eigene Darstellung

Mithilfe dieser Grundlagen können Verkehrsnetze und ihre Elemente beschrieben werden. Dabei verfolgen verschiedene Datenmodelle unterschiedliche Herangehensweisen. So ist beispielsweise nicht festgelegt, ob in zwei Richtungen befahrbare Straßen durch je eine gerichtete Kante für jede Richtung oder durch eine Kante ohne Richtung repräsentiert werden sollen. Außerdem gibt es unterschiedliche Anforderungen an den Detailgrad des Graphen. Eine exakte geographische Abbildung einer Straße würde in zunehmender Auflösung aufgrund der vielen geometrischen Stützknoten einen Graphen mit unendlich vielen Knoten und unendlich kleinen Kanten zwischen diesen erfordern. Gleichwohl kann der reale Verlauf einer Strecke bei der Visualisierung stark verzerrt werden, wenn der eigentliche Verlauf uninteressant ist und nur eine lineare Verbindung zwischen Anfang und Ende der Verbindung gezogen wird.

Die Darstellung durch Graphen ist unabhängig davon, ob ein Straßennetzmodell oder ein Modell für den ÖV verdeutlicht werden soll. Auch die Linien des ÖV können durch Kanten (Gleise, Busspuren) und Knoten (Kreuzungen der Route, Haltestellen) dargestellt werden. Während der ÖV zumeist durch ein restriktives Liniennetz beschrieben wird, ist die Straßeninfrastruktur für den IV durch ein ausgeprägtes Erschließungsnetz charakterisiert, das eine flächendeckende Anbindung bietet. Für die Modellierung des ÖV werden zudem die Linien der Verkehrssysteme (Bus, Bahn) abgebildet. Diese können durch Kantenfolgen beschrieben werden (vgl.Clausen, 1981 S. 32).

2.2. OpenStreetMap als Datengrundlage

OpenStreetMap (OSM) ist ein 2004 in Großbritannien gestartetes, gemeinnütziges Projekt, das es sich zur Aufgabe gemacht hat, frei verfügbare Geodaten zu sammeln, die ohne urheberrechtliche Beschränkungen weiterverwendbar sind, da sie einer *GNU General Public License* unterliegen (vgl. “gnu.org”). Dabei werden die Daten zumeist von freiwilligen Mitgliedern der OSM-Community gesammelt und in einer Datenbank bereitgestellt. Dabei kann, ähnlich wie bei *Wikipedia,* jeder mitarbeiten. Die Informationen werden mithilfe von GPS-Spuren, Datenimporten vorhandener Quellen, Luftbildern und auch einfacher Ortskenntnis zusammengetragen. Das Projekt hat bereits viele Städte detaillierter kartographiert als vergleichbare Web-Angebote von *Google*, *Yahoo* oder *Microsoft*. Während es ausdrücklich keine Vorschriften bezüglich der Vorgehensweise beim

Aufnehmen der Daten gibt, hat sich im Laufe der Zeit ein übliches Vorgehen etabliert (vgl. Ramm und Topf, 2009 S. 3f.).

Das Projekt wurde anfangs vornehmlich mit dem Zweck Straßen(-netze) zu kartieren ins Leben gerufen. Heutzutage werden jedoch wesentlich mehr Informationen gesammelt. So werden unter anderem auch Gebäude, *Points of Interests* (POI; bspw. Tankstellen, Ärzte, etc.), Gewässer oder Trassenverläufe aufgezeichnet. Neben diesen physischen Informationen werden jedoch auch abstrakte Zusammenhänge, wie Ländergrenzen, Routenverläufe oder Abbiegebeziehungen in die Datenbank aufgenommen (vgl. Bennett, 2010 S. 8).

2.2.1. Datentypen in OpenStreetMap

Die Datenstruktur von OSM ist so einfach wie möglich gehalten („the simplest thing that will work“ (Bennett, 2010, S.54)). Zur Darstellung der Informationen werden zwei geometrische Grundelemente (Nodes, Ways) und Beziehungen zwischen diesen (Relations) verwendet. Es ist ein abstraktes, offenes Modell, in dem alle Einträge in der OSM Datenbank auf diesen drei Elementen basieren. Für unterschiedliche Repräsentationen der Realität werden dieselben Elementtypen genutzt. Erst durch das Hinzufügen von Attributen (siehe Abschnitt 2.2.2*)* wird die eigentliche Bedeutung des Elements definiert. So werden kleine Flüsse ebenso durch Ways dargestellt wie auch Straßen. Erst durch die Identifizierung mittels zusätzlicher Attribute können diese unterschieden werden. Alle Grundelemente erhalten bei ihrer Erzeugung eine vom System verwaltete, innerhalb des Elementtyps eindeutige Identifizierung (ID) oder Kennung (vgl. Ramm und Topf, 2009 S.49f.).

Node

Nodes, oder auch Knoten, sind geometrische Grundelemente, die durch eine Punktkoordinate (Längen- und Breitengrad) räumlich exakt festgelegt werden. Sie bilden die Grundlage für komplexere, durch Ways dargestellte Linien oder können auch - bei Verwendung von Attributen - einzelne Objekte und POI beschreiben. Typische Anwendungen sind zum Beispiel die Kennzeichnung von Hausnummern und Hauseingängen, Bäumen oder auch Hydranten (vgl. ebd. S.50). Insgesamt gibt es mehr als 2,5 Milliarden Knoten in der OSM-Datenbank (Stand 10/2014, vgl. “OpenStreetMap Taginfo,” 2014).

Way

Ways, oder auch Linien, sind Verbindungslinien zwischen zwei oder mehreren Nodes (maximal 2.000), die physische oder abstrakte Verläufe graphisch darstellen. Dabei werden zwei aufeinanderfolgende Punkte durch eine gerade Linie verbunden. Sie werden genutzt um Straßen, Trassen und andere linienhafte Verläufe abzubilden. Sie besitzen eine bestimme Richtung, d.h. die Reihenfolge ihrer Nodes ist geordnet. Bei identischen Nodes am Anfang und Ende eines Ways bildet dieser eine geschlossen Fläche. Somit können Wälder, Landnutzungen oder auch Gebäude dargestellt werden (vgl. Ramm und Topf, 2009, S.51). Während das zugrundeliegende Modell es nicht verbietet, dieselben Nodes beliebig oft in einem Way unterzubringen, wird diese Möglichkeit vor allem durch die vorhandenen Editiermöglichkeiten eingeschränkt, da es wenige realistische Anwendungsfälle dafür gibt. Die meisten Editoren erlauben maximal einen gleichen Anfangs- und Endknoten. Über 255 Mio. Ways sind in der Datenbank vorhanden (Stand 10/2014, vgl. "OpenStreetMap Taginfo," 2014).

Relation

Relations (Relationen, Abhängigkeiten) sind keine geometrisch abbildbaren Elemente. Sie beschreiben lediglich Beziehungen zwischen Nodes, Ways oder auch anderen Relationen. Demnach enthält eine Relation immer eine sortierte Liste mehrerer anderer Elemente. Die Elemente einer Relation werden *Member* der Relation genannt. Sie können zusätzlich zu ihrer Position innerhalb der Relationsliste einer *Rolle* zugewiesen werden. Oft genutzte Relationen sind beispielsweise Routenverläufe, die aus aufeinanderfolgenden Ways der Strecke und Nodes als Haltestellen bestehen (vgl. ebd., S.51). Die OSM Datenbank enthält über 2,8 Mio Relationen (Stand 10/2014, vgl. ebd.).

2.2.2. Typisierung und Eigenschaften der Elemente durch Tags

Die Attribute der zuvor beschriebenen Elemente werden über sogenannte *Tags* definiert. Ein Tag besteht aus einem Schlüssel (*key*) und einem Wert (*value*). Beide können bis zu 255 beliebige UTF-8-kodierte Zeichen enthalten. Während der Schlüssel gesetzt sein muss, kann der zugehörige Wert auch leer sein. Es ist nicht erlaubt, einem Objekt mehrere Tags mit demselben Schlüssel zuzuweisen. Festgelegte Vorschriften über den Inhalt der Tags gibt es

nicht. Es gibt jedoch Richtlinien die sich in der OSM Community etabliert haben. So werden Tags, die einen Wahrheitswert beschreiben mit ‚*yes*' oder ‚*no*' gekennzeichnet. Spezifische Namensräume für zusammengehörende Schlüssel werden in der Regel durch einen Doppelpunkt vorangestellt. Das Tag hat dann die Form ‚*Namensraum*:*Schlüsselname*=*Wert*'. (vgl. ebd., S.52f.)

Eine Übersicht des Aufbaus des OSM-Datenmodells ist auf Abbildung 3 zu sehen. OSM-Daten können in einem eigenen Dateiformat mit der Endung *.osm* gespeichert werden, das auf dem *XML*-Standard beruht. In diesem werden die einzelnen Objekte und ihre Eigenschaften als *XML*-Elemente nacheinander aufgeführt. Eine beispielhafte *.osm*-Datei ist im Anhang A angefügt.

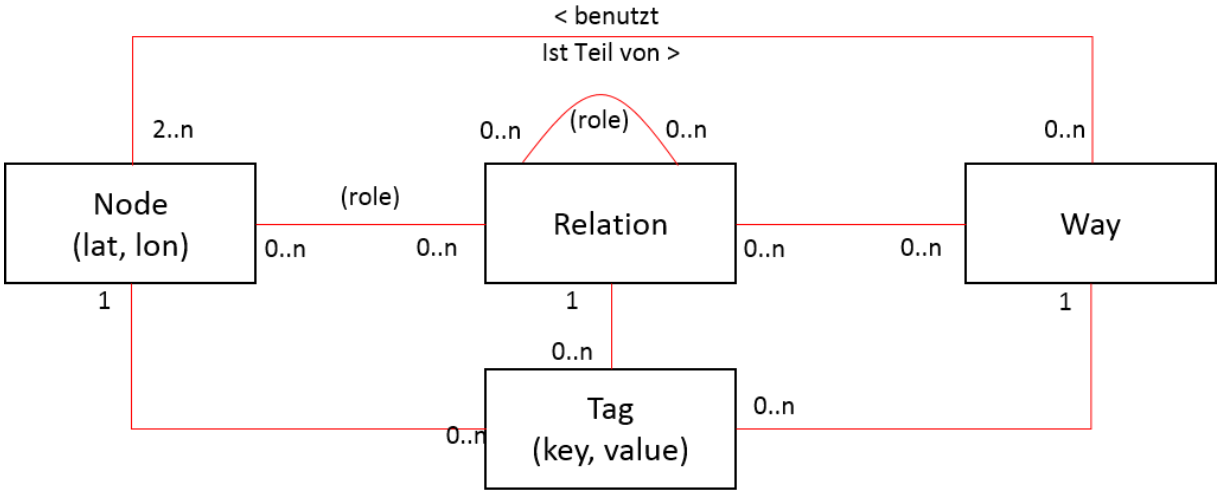

Abbildung 3: Übersicht des OSM-Datenmodells
Quelle: eigene Darstellung nach Ramm und Topf, 2009

Während Nodes und Wayss an sich nur geographische Konturen abbilden, definieren erst die ihnen zugewiesenen Tags, was genau durch sie dargestellt werden soll. Der Bearbeiter kann beim Einpflegen der Elemente in die Datenbank alle Informationen hinzufügen, die er für wichtig hält. Etablierte Tagstrukturen mit empfohlenen Richtlinien existieren nur für die am häufigsten verwendeten Tags. Dabei haben die verschiedenen Kartenrenderer, die aus den Daten eine visuelle Karte generieren, einen großen Einfluss. Nur was den Anforderungen der Renderer entspricht wird von diesen auch so gezeichnet wie gewünscht (Ramm und Topf, 2009, S.59).

2.2.3. Abbildung des Straßen- und Liniennetzes in OpenStreetMap

Zunächst wird die Dokumentation von Straßen vorgestellt. Da Straßen von Anfang an das Hauptaugenmerk des Projekts sind, hat sich für diese schnell eine relativ einheitliche Tag-Struktur etabliert. Straßen werden durch Ways definiert, die mit einem *highway* Tag versehen sind (nachfolgend Highway). Der Schlüssel des Tags enthält demnach immer das Wort ‚*highway*', während der Wert des Tags die Straße genauer definiert. Ein Highway ist hier jedoch nicht nur als befestigte Straße für den Kfz-Verkehr zu verstehen. Auch Fahrradwege, Reitwege, Fußwege oder auch Treppen werden durch *highway* Tags definiert. Die gewählte Klassifizierung gilt dann jeweils für den gesamten Way. Tabelle 1 zeigt gängige Werte, die dem *highway* Tag zugeordnet werden können und ihre ungefähre (deutsche) Entsprechung. Besonders zu beachten ist dabei, dass mit Ausnahme der Autobahnen und deren Verbindungen die Straßen grundsätzlich unabhängig von der Zeichenrichtung des Ways sind. Während Autobahnen im Regelfall die Richtung repräsentieren, in der der entsprechende Way gezeichnet ist, beschreiben Ways aller anderen Straßentypen zumeist beide Richtungen der Straße. Zusätzlich zu der Straßenklassifizierung gibt es weitere Tags, die eine genauere Beschreibung der Straßen ermöglichen. Genannt seien die Tags:

- *maxspeed*, das die zulässige Höchtgeschwindigkeit angibt
- *name*, der Name der Straße
- *ref*, die Nummer bzw. offizielle Abkürzung der Straße
- *junction*, beschreibt Kreuzungen, wird zumeist für Kreisverkehre verwendet
- *oneway* (nicht bei *motorway* und *motorway_trunk*), gibt an, ob die Straße nur in eine Richtung befahren werden darf, wobei die Zeichenrichtung des Ways entscheidend ist

Es gibt noch weitere Straßenspezifische Tags. Durch die nicht vorhandenen Restriktionen bei der Straßenmodellierung gibt es auch viele Tags, die im Extremfall nur einmalig vorkommen. Die erwähnten Tags zur Straßenmodellierung werden in Tabelle 2 zusammengefasst (vgl. Ramm und Topf, 2009, S.62ff.).

Tabelle 1: etablierte Werte für das ‚highway' Tag und deren Bedeutung
Quelle: eigene Darstellung vgl. Ramm und Topf, 2009

Wert	Bedeutung
motorway	Autobahn, beschreibt eine Richtung
motorway_link	Autobahnauffahrt oder –verbindung, beschreibt eine Richtung
trunk	Kraftfahrstraße o.ä.,
trunk_link	Kraftfahrstraßenauffahrt oder -verbindung
primary	wichtige überregionale Straße, Bundesstraße
primary_link	Bundesstraßenauffahrt oder –verbindung
secondary	gut ausgebaute Straße, Verbindungsstraße außer Orts oder wichtige, verkehrsreiche Straße innerorts
tertiary	kleinere regionale Straße, Kreisstraße
residential	Straße im Wohngebiet
living_street	verkehrsberuhigte Straße
unclassified	Nebenstraße; Straße die nicht in einer der o.g. Klassifikationen fällt
service	Zugangswege die nicht für den allgemeinen öffentlichen Verkehr (nicht im Sinne des Verkehrsmodus) gedacht sind
track	Feld- oder Waldweg
bridleway	Reitweg
cycleway	Fahrradweg
footway	Fußweg
path	alternativ für tracks, die nicht breit genug sind, um vierrädrige Fahrzeuge aufzunehmen
pedestrian	Fußgängerzone
steps	Treppe
road	Straße mit unbekannter Klassifizierung

Tabelle 2: Für Straßen verwendete Tags
Quelle: eigene Darstellung vgl. Ramm und Topf, 2009

Schlüssel	**Wert**	**Beispiel**
highway	Klassifizierung der Straße	highway=primary
maxspeed	Geschwindigkeit in Kilometern pro Stunde	maxspeed=50
name	Straßenname	name=Berliner Straße
ref	Straßennummer / -kürzel	ref=B 96
junction	Kreuzungstyp, zumeist für Kreisverkehr genutzt: roundabout	junction=roundabout
oneway	• ‚yes' oder ‚true' oder ‚1' (nur in Richtung des Ways befahrbar) • ‚reversed' / ‚-1' (nur in entgegengesetzter Richtung des Ways befahrbar)	oneway=yes

Die Abbildung der Elemente des ÖV geschieht auf ähnliche Weise. Die Strecken werden auch hier durch Ways dargestellt. Busse verkehren auf den bereits beschrieben Highways. Bahngleise werden durch Tags mit dem Schlüssel ‚*railway'* identifiziert. Dabei gibt es unterschiedliche Werte für verschiedene Schienensysteme, die in Tabelle 3 zusammengestellt sind. Straßenbahnkörper, die kein eigenständiges Gleisbett besitzen, können auch durch Ways repräsentiert werden, die sowohl Straße als auch Schiene definieren. Für das Setzen von Haltestellen gibt es weitere Tags. Alle werden mit dem Schlüssel ‚*public_transport'* beschrieben. Dabei wird der Wert ‚*stop_position'* für den eigentlichen Halt des Fahrzeugs auf der Strecke verwendet, während ‚*platform'* die Stelle beschreibt, an der die Fahrgäste warten. Gebäude, beziehungsweise größere Bereiche, werden mit ‚*station'* definiert. Zum Zusammenfassen zu einen Haltestellenbereich kann eine Relation mit dem Tag ‚*public_transport=stoparea'* genutzt werden, die mehrere der zuvor genannten Elemente beinhalten kann. Eine Übersicht der erklärten Tags ist in Tabelle 4 verdeutlicht. Diese Handhabung ist für alle öffentlichen Verkehrssysteme gleich (vgl. "Public transport - OpenStreetMap Wiki," 2014 sowie "Proposed features/Public Transport," 2014) .

Auch wenn eine richtungsgetrennte Modellierung der einzelnen Bahntrassen nicht vorgeschrieben ist, ist es auch nicht unüblich, parallel verlaufende Gleisstrecken einzeln abzubilden. Es ist daher nicht allein aus der abgebildeten Infrastruktur ersichtlich, ob ein Gleis in eine oder in zwei Richtungen verwendet wird. Dafür werden weitere betriebliche

Informationen benötigt. Die eigentliche Nutzung der Strecken durch öffentliche Verkehrsmittel wird in Linien und deren Routen beschrieben. Diese abstrakten Informationen werden in Relationen erfasst. Eine Relation, die eine ÖV-Route darstellen soll, wird zuerst mit einem ‚*type=route*' Tag versehen. Damit werden allgemein Routen, also z.B. auch Rad- und Wanderwege beschrieben. Die genauere Spezifikation erfolgt durch ein Tag mit dem Schlüssel route, dass die Art der Route, beispielsweise Bus- oder Bahnstrecke, festlegt. Ein zugehöriger Linienname der Route kann durch ein Tag mit dem Schlüssel ‚*ref*' eingepflegt werden. Die Routen-Tags sind in Tabelle 5 zusammengetragen.

Eine OSM-Route beschreibt immer genau eine Richtung. Mehrere Routen, die einer Linie zugehörig sind, können über eine Elternrelation zusammengefasst werden. Eine Elternrelation beinhaltet andere zusammengehörende Relationen als Mitglieder. Die Strecke der Routen wird durch eine sortierte Liste von aufeinanderfolgenden Ways als Mitglieder der Relation definiert. Die Halte der Routen werden ebenfalls in sortierter Reihenfolge durch die Nodes mit Halte-Tags (siehe Tabelle 4) repräsentiert. In der Regel wird dafür die eigentliche Halteposition auf dem Fahrweg genutzt, was jedoch nicht immer der Fall ist. Generell ist das recht aufwändige Einpflegen von ÖV-Routen problembehaftet, da die vorgestellte Vorgehensweise nicht bindend ist und auch viele Fehler - mangels systematischer Überprüfung vor der Übernahme in die Datenbank - möglich sind. Die empfohlenen Richtlinien (*proposals*) unterliegen außerdem steten Wandlungen.

Tabelle 3: etablierte Werte für das railway Tag
Quelle: eigene Darstellung vgl. "De:Key:railway – OpenStreetMap Wiki," 2014

railway Wert	**Bedeutung**
rail	gewöhnliche Vollbahnstrecke
light_rail	Stadtbahnstrecken
tram	Straßenbahnstrecke
subway	U-Bahnstrecke

Tabelle 4: Tag-Werte mit dem puclic_transport Schlüssel
Quelle: eigene Darstellung vgl. "Public transport -OpenStreetMap Wiki," 2014

public_transport Wert	**Beschreibung**	**Angewendet auf**
stop_position	bezeichnet Halteposition des Fahrzeugs auf der Strecke	Knoten, der auf einem Way der Strecke liegt
platform	Ort, an dem Passagiere warten bzw. Haltestellenmast	Knoten neben der Strecke
station	Station im Sinne von Bahnhof(sgebäude) oder Busbahnhof	geschlossenem Way (Fläche) oder Knoten in der Fläche
stop_area	fasst die anderen Elemente zu einem Halt zusammen	Relation, die Elemente mit den vorherigen Werten enthält

Tabelle 5: Tags zur Beschreibung von Routen
Quelle: eigene Darstellung nach "Public transport -OpenStreetMap Wiki," 2014

Schlüssel	**Wert**	**Bedeutung**
type	*route*	Definiert die Relation als Route
route	*train; rail; bus; subway; tram*	Legt den Routentyp fest (Bahn, Bus, U-Bahn, Straßenbahn etc…)
ref	Freitext	Linienkürzel (z.B. ICE3)
name	Freitext	Name der Linie

Bei der Anwendung des Graphenmodells auf das Straßen- und Schienennetz von OSM, können die Nodes der Ways als Knoten des Graphen aufgefasst werden. Die Ways jedoch lassen sich nicht direkt als Kanten verstehen. Laut Definition sind Kanten lineare Verbindungen zwischen zwei Knoten. Ways hingegen können mehr als zwei Nodes verbinden. Demnach ist ein Way eher eine *Kantenfolge* (oder ein *Kantenzug*). Da die Linienrouten Folgen von Ways sind, sind auch die Routen Kantenfolgen. In früheren Datenmodellen von OSM gab es die sogenannten Segmente (*segments*). Diese bestanden aus

genau zwei Nodes und entsprachen damit eher einer Kante im Sinne des Graphenmodells. Da die Daten auch für Routingzwecke nutzbar sein sollen, wird ein *zusammenhängender Graph*, d.h. ein zusammenhängendes Netz, bei dem jede Straße / Schiene an das Netz angeschlossen ist, angestrebt. Dennoch ist es nicht verboten (und auch in der Realität oft vorhanden), Straßen oder Schienen zu definieren, die nicht an den übrigen des Netzelementen angeschlossen sind. Autobahnen und Straßen mit einem gesetzten *oneway* Tag sowie eigenständige Schienen sind nur in eine Richtung befahrbar. Die jeweiligen Kanten der Kantenfolge sind demnach gerichtet. Alle anderen Straßen repräsentieren hingegen zwei entgegengesetzte Richtungen. Der Graph ist deshalb nicht gerichtet. Zu beachten ist, dass diese Einschätzung unabhängig von der eigentlichen Zeichenrichtung des Ways erfolgt. Die Klassifizierung der Straßen und Schienen können als Gewichtung aufgefasst werden. Das Gesamtnetz in OSM ist also ein nicht zusammenhängender, gewichteter Graph, der einzelne gerichtete Kanten enthält.

Die OSM-Daten eignen sich daher gut für eine visuelle Darstellung von Verkehrsnetzen. Zudem ist die freie Verwendung der Daten ein wichtiger Punkt. Es können nicht nur abgeschlossene Datensätze benutzt werden, sondern auch alle zugrundeliegenden Daten, die frei editier- und manipulierbar sind. Das ermöglicht neben dem reinen Visualisieren auch ein Anpassen und Bearbeiten der Daten.

Die Editiermöglichkeiten werden von verschiedenen Programmen unterstützt. Eines davon wird im folgenden Kapitel vorgestellt.

2.3. Visuelles Editieren von OpenStreetMapDaten mit JOSM

Das Programm JOSM – Java OpenStreetMap Editor – ist ein auf der Programmiersprache Java basierender Editor, der vielfältige Interaktionen mit den OSM Daten ermöglicht. Nach Art des OSM Projekts ist er kostenlos und frei verfügbar. Zudem ist auch der Quellcode frei erhältlich und manipulierbar unter Angabe der ursprünglichen Lizenz. Daher gibt es bereits zahlreiche Erweiterungen, die zusätzliche Funktionen bieten. Für diese *Plugins* ist eine eigene Pluginstruktur vorgesehen, die es Außenstehenden erleichtert, eigene Funktionalität ohne Ändern des eigentlichen Codes einzubringen. Die neueste Version des Programms ist unter *http://josm.openstreetmap.de/* verfügbar. Exakte Installationshinweise und Anleitungen sind auf *https://wiki.openstreetmap.org/wiki/JOSM* zu finden und werden in

dieser Arbeit nicht weiter aufgeführt. Es wird lediglich ein kurzer Einblick in das Programm gegeben.

2.3.1. Prinzip

Der Hauptbestandteil des Editors ist die in der Mitte des Ansichtsfensters positionierte Kartenansicht, die sich auf ein globales Koordinatensystem bezieht (i.d.R. *WGS84*). Dort werden alle geöffneten Daten visuell dargestellt. Nodes werden durch Punkte und Ways durch Linien dargestellt. Die Darstellung kann also als Graph aufgefasst werden. Für die Darstellung gibt es umfangreiche Einstellungsmöglichkeiten. Zusätzlich zu den visualisierten Daten können auch Hintergrundbilder, wie gerenderte Karten oder Satellitenaufnahmen eingestellt werden. Die eigentlichen Daten werden in Ebenen (nachfolgend Layer) geöffnet, die übereinander gelegt werden können.

JOSM erlaubt es zum einen, verschiedene Dateiformate (darunter *.osm*-Dateien) lokal zu öffnen und abzuspeichern. Zum anderen besteht auch die Möglichkeit, innerhalb des Programms mit dem OSM Server zu interagieren. Dabei lassen sich anhand einer Karte interaktiv Ausschnitte auswählen, deren Daten heruntergeladen und geöffnet werden sollen. Außerdem können eigene Bearbeitungen und Änderungen nach der Sitzung auf den Server geladen werden. Neben dem Öffnen vorhandener Daten können auch komplett neue Layer erstellt und bearbeitet werden.

Die Bearbeitung lässt sich grob in das geometrische Bearbeiten der Nodes und Ways und in das beschreibende Bearbeiten durch Relationen und Tags einteilen. Für die geometrische Bearbeitung stehen einige Zeichenfunktionen zur Verfügung, mit denen detaillierte Darstellungen geographischer Gegebenheiten möglich sind. Knoten und Wege lassen sich mithilfe der Maus zeichnen. Interessant ist hierbei, dass JOSM komplexe Ways intern auf *WaySegments* abbilden kann. Graphentheoretisch sind *WaySegments* einzelne Kanten der Kantenfolge. Vorhandene Elemente können verschoben, angeordnet, entfernt oder zu Relationen hinzugefügt werden. Die Relationen werden als Listen dargestellt, die die Elemente auflisten und deren Rolle und Reihenfolge anzeigen. Die Tags selektierter Elemente werden in einem separaten Dialog angezeigt, der eine direkte Bearbeitung erlaubt. Gemäß dem OSM Prinzip gibt es hierbei keine Einschränkungen bei der Tag-Erstellung bzw. Bearbeitung. Schlüssel und Werte lassen sich tabellarisch eintragen. Um eine einheitliche Verwendung bei den Tags zu erleichtern, stellt JOSM sogenannte *Presets* bereit. *Presets*

sind vordefinierte Tag-Gruppen, die auf Nodes, Ways oder Relationen anwendbar sind. Dabei wird dem Benutzer das Kennzeichnen bestimmter Objekte erleichtert, indem die benötigten und optionalen Informationen für die von ihm gewünschte Klassifizierung in einem Dialog abgefragt werden. Die Darstellung bestimmter Elemente wird über die gesetzten Tags differenziert. Zum Beispiel werden Straßen mit *oneway* Tag mit Pfeilen entlang der Linien versehen, die die Richtung der Einbahnstraße symbolisieren.

Um Daten auf Plausibilität zu prüfen, bietet JOSM verschiedene *Validatoren* an, die die Elemente auf Konsistenz und Fehler prüfen. Unstimmigkeiten können dann in einem separaten Error Layer angezeigt und teilweise auch automatisch repariert werden.

Weitere Funktionalitäten sind unter anderem eine Filterfunktion, mit der sich Elemente ausblenden lassen sowie eine Suchfunktion, die Objekte anhand von Tags oder aber auch anhand von Metadaten wie der OSM ID durchsucht.

2.3.2. Plugin Interface

Die Funktionalität von JOSM kann durch Plugins erweitert werden. Dabei gibt es vorgesehene Schnittstellen, die das Erstellen und Integrieren der Erweiterungen vereinfachen. Im Programm selbst können die Plugins über die Einstellungen gesucht und aktiviert werden. Dafür wird eine kompatible *.jar*-Datei des Plugins benötigt, die den Java Code inklusive zusätzlicher Metadaten beinhaltet (vgl. “Java Archive,” 2014). Bereits existierende Plugins können auf einem zentralen Server gehostet werden, über den JOSM sie automatisch bezieht. Eine Liste aller auf dem offiziellen Server verwalteten Plugins befindet sich unter:

https://josm.openstreetmap.de/wiki/Plugins

Der eigentliche Code benötigt einen Einstiegspunkt für das Plugin mittels einer Klasse, die vorzugsweise von der JOSM-Klasse `Plugin` erbt und einen Konstruktor mit einer `PluginInformation` als Parameter bereitstellt. Diese Klasse kann mit dem Hauptprogramm interagieren, indem es folgende drei Funktionen implementiert:

- `public void mapFrameInitialized(MapFrame old, MapFrame new)`
    - wird aufgerufen, wenn eine neue *MapFrame*, also eine neue Kartenansicht geöffnet wird (beim Starten von JOSM wird zuerst eine Startseite angezeigt)

- `public PreferenceSetting getPreferenceSetting()`
    - ermöglicht dem Plugin, eigene Einstellungen bereitzustellen, indem es beim Aufrufen dieser Methode einen eigenen Einstellungseditor zurückgibt
- `public void addDownloadSelection(List<DownloadSelection> list)`
    - ermöglicht dem Plugin, eine eigene Download-Methode beim Herunterladen von Daten hinzuzufügen

Zusätzlich dazu stellt JOSM eigene *Listener* bereit, die von anderen Klassen abonniert werden können, um Änderungen im Programmablauf und in den geöffneten Daten mitzuschneiden. Zentrale Elemente wie die momentane Kartenansicht, Einstellungen oder das Hauptfenster können über statische Methoden adressiert werden (vgl. "DevelopersGuide/DevelopingPlugins – JOSM,").

2.4. MATSIM

Die Verkehrssimulation MATSim (Multi Agent Transport Simulation) ist ein auf Java basierendes Open Source Projekt, mit dem große Verkehrsszenarien untersucht werden können. Sie nutzt eine mikroskopische Herangehensweise, das heißt, dass alle Verkehrsteilnehmer, die sogenannten *Agenten*, einzeln modelliert werden. Die Agenten bilden die *Population* der Simulation und stellen die Verkehrsnachfrage dar. Die Nachfrage entsteht dabei durch *Aktivitäten*, die die Agenten zu bestimmten Zeiten an bestimmten Orten ausführen wollen (*‚activity based demand'*). Strategien zur Lösung dieser Nachfrage werden für jeden Agenten in Plänen festgehalten, die neben den geplanten Aktivitäten auch jeweils eine geplante Route zwischen den Aktivitätenorten, ein sogenanntes *leg*, enthält. Durch die gleichzeitige Simulation aller Agenten und ihrer Pläne entstehen Interferenzen, die sich in verlängerten Reisezeiten oder sogar Stau äußern können. Eine iterative Herangehensweise ermöglicht es, jeden ausgeführten Plan eines Agenten am Ende eines Durchlaufs nach vordefinierten Kriterien zu bewerten. Nun können verschiedene Strategien aktiviert werden, die es einzelnen Agenten erlaubt, ihre geplanten Aktivitäten und Ortsveränderungen anzupassen und im nächsten Durchlauf auszuführen. Daraufhin wird wieder bewertet und adjustiert. Die am besten bewerteten Pläne werden dann beibehalten. Das wird *Co-evolutionärer Algorithmus* genannt. Für die Auswertung der Simulation werden zum einen vorgefertigte Statistiken herausgeschrieben. Zum anderen können verschiedene Ereignisse,

die während des Ablaufs auftreten, analysiert und interpretiert werden. Diese *Events* werden nach jeder Iteration herausgeschrieben, sodass eine zeitliche Entwicklung der Anpassungsstrategien der Agenten nachvollziehbar ist. Erfasst werden unter anderem das Betreten und Verlassen eines Links (siehe auch Abschnitt 2.4.1) oder die Ankunft eines Agenten an einem Aktivitätenort.

Während die eigentliche Simulation über eine Konsole gesteuert wird, können die Ergebnisse in dem Visualisierungsprogramm *Via* des Unternehmens *Senozon* dargestellt werden. Für einzelne Iterationen können dort die Agenten bei ihrem Weg durch das Verkehrsnetz beobachtet werden. Zusätzlich gibt es den sogenannten *OTFVis (On the fly visualizer),* der eine Darstellung der sich bewegenden Agenten zur Simulationszeit erlaubt (vgl. senozon AG, 2014).

Die Agenten können verschiedene Verkehrsmodi nutzen, sodass sowohl der IV als auch der ÖV simuliert werden können. Beide benötigen für die Routenwahl ein zugrundeliegendes Netz, das für die Simulation bereitgestellt wird. Die benötigten Informationen für beide Modi werden im Folgenden erläutert.

2.4.1. IV-Datenformat

Für das Simulieren des IV (in erster Linie des Kfz-Verkehrs) wird ein für den IV zugelassenes Straßennetz benötigt. Das Netz besteht dabei aus je einer Liste von Nodes (Knoten) und Links (Kanten). Alle Elemente des Netzwerks besitzen eine eindeutige Kennung (ID). Die Nodes, die die Verbindungselemente der Links darstellen, haben eine feste Koordinate, die durch einen X und einen Y Wert beschrieben wird. Die allgemeingehaltenen XY-Koordinaten können in verschiedenen Koordinatensystemen genutzt werden. Geeignete Konvertierungsfunktionen stellt MATSim dazu bereit. Links verbinden genau zwei Nodes in einer bestimmten Richtung miteinander und stellen die Straßen- oder Bahnverbindungen dar. Sie besitzen eine *fromNode* und eine *toNode*. Es sind daher gerichtete Kanten. Über diese geometrischen Informationen hinaus verfügen Links über verkehrliche Informationen, die für die Simulation von Bedeutung sind. Dazu zählen die Länge, Kapazität, Geschwindigkeit im unbelasteten Netz, die Anzahl der Fahrspuren sowie die erlaubten Verkehrsmodi. Eine Zusammenfassung der Elemente und deren Attribute ist in Tabelle 6 aufgelistet. Im Gegensatz zu den meisten Straßen im OSM Datenformat sind die Kanten hier also immer *gerichtet*. Es gibt außerdem keine

Kantenfolgen in der Art von OSM-Ways, die mehr als zwei Knoten umfassen. Das MATSim-Netz ist ein stark zusammenhängender Graph. Die Notwendigkeit des Zusammenhangs besteht in der Simulation, die für jede Knotenrelation eine funktionierende Route berechnen können muss. Das Netzwerk (*Network*) wird vor dem Start der Simulation in Form einer einer XML-Datei bereitgestellt. Eine beispielhafte network.xml Datei ist im Anhang B beigefügt (vgl. Rieser et al., 2014, S.18).

Tabelle 6: Elemente eines Network und deren Attribute
Quelle: eigene Darstellung

	Merkmal	**Erklärung**
Network	nodes	Liste aller Nodes.
	links	Liste aller Links.
	capacity period	Zeitintervall in Sekunden, auf das sich die Kapazitätsangaben der Links beziehen. Üblicherweise 3.600 Sekunden (bzw. 1 Stunde).
	effective lane width	Räumliches Maß, das die Breite einer Fahrspur in Metern angibt. Üblicherweise 3,75m.
Node	id	Alphanumerische, innerhalb aller Nodes eindeutige Kennung.
	coord	XY-Koordinate, die die Position des Knoten festlegt, vorzugsweise in kartesischem Koordinatensystem.
Link	id	Alphanumerische, innerhalb aller Links eindeutige Kennung.
	originalId	Alphanumerische Kennung, die vor allem Rückschlüsse auf die konvertierte Datengrundlage gibt.
	from	Anfangsknoten / Startknoten.
	to	Endknoten / Zielknoten.
	length	Länge des Links in Metern.
	capacity	Kapazität des Links. Gibt an, wie viele Fahrzeuge den Link innerhalb der capacity period des Network passieren können. Üblicherweise in Fahrzeuge je Stunde.
	freespeed	Maximal erlaubte Geschwindigkeit in Metern je Sekunde, mit der Fahrzeuge den Link passieren können.

	number of lanes	Anzahl der Fahrspuren.
	modes	Eine Liste aller erlaubten Verkehrsmodi, die den Link benutzen dürfen.

2.4.2. ÖV-Datenformat

Der ÖV bewegt sich innerhalb desselben Netzes wie der IV. Dabei kann er eigenständige Links benutzen, deren Verkehrsmodi nur ÖV zulassen und nicht mit dem restlichen IV-Netz verbunden sind (z.B. Bahnstrecken) oder auch das Netz des IV verwenden, sofern es öffentliche Verkehrsmodi zulässt. Da öffentliche Verkehrsmittel zumeist vordefinierte Linien bedienen, werden weitere Informationen benötigt. Dafür wird ein sogenannter *transit schedule* (nachfolgend TSchedule) verwendet. In einem ersten Teil werden dort Informationen zu den vorhandenen Haltestellen im Netz, den *stop facilities* (nachfolgend StopFacility), gegeben. StopFacilities besitzen ähnlich wie Nodes und Links eine ID. Außerdem werden sie wie Nodes durch Koordinaten räumlich bestimmt. Anders als OSM Haltestellen werden StopFacilities nicht als Knoten zwischen zwei Kanten aufgefasst. Vielmehr hängen sie an einem definierten Link. Eine StopFacility benötigt daher zusätzlich einen Link, über den sie erreichbar ist. Bei der eigentlichen Simulation wird die Haltestelle bedient, wenn das Ende des zugehörigen Links erreicht wird. Für eine bessere Zuordnung zu den realen Haltestellen können zusätzlich Namen vergeben werden, die auf die Simulation keinen Einfluss haben.

Nach der Deklaration der Haltestellen werden die *transit lines* (nachfolgend TLines) beschrieben, die die Linien des ÖV abbilden. Eine TLine besteht aus einer ID und kann mehrere *transit routes* (nachfolgend TRoutes) besitzen. TRoutes sind die einzelnen Streckenverläufe der Linie (z.B. Hin- und Rückrichtung). Auch sie besitzen eine ID. Außerdem wird ein *transport mode* für jede TRoute festgelegt, der das Verkehrssystem bestimmt (Bus, Bahn etc.). Der eigentliche Verlauf der Routen wird durch zwei Auflistungen beschrieben. Die erste, das *route profile*, listet den geordneten Ablauf aller StopFacilities als *transit route stops* auf und wird für das Routing der Fahrgäste genutzt. Alle StopFacilities können also in beliebig vielen TRoutes vorkommen. Die zweite Auflistung beschreibt die *network route*, die Abfolge aller Links, die das öffentliche Verkehrsmittel abfährt. Die Abfahrtszeiten (*departure*) sowie die verwendeten Fahrzeugtypen (die in einer separaten Datei spezifiziert werden können) der Route sind ebenfalls Bestandteil einer TRoute, werden aber hier nicht weiter erklärt (vgl. Rieser, 2010, S.64f.). Die Relationen der einzelnen

Elemente zueinander sind in Abbildung 4 dargestellt. Eine beispielhafte TSchedule *XML*-Datei ist im Anhang C beigefügt.

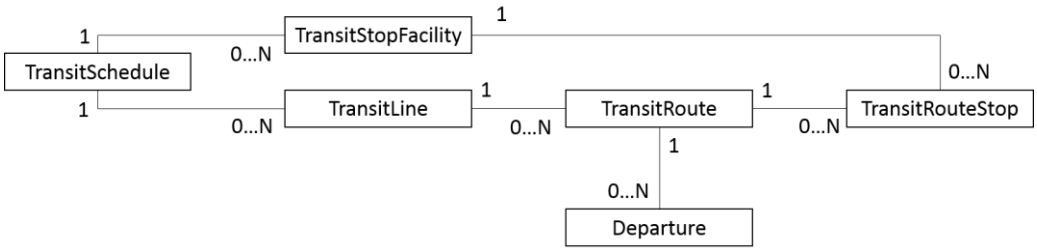

Abbildung 4: Elemente der transit schedule
Quelle: Rieser, 2010, S. 65

2.5. Bisheriges Erstellen und Bearbeiten von MATSim-Netzmodellen

Die (Straßen-)Netzwerke für MATSim-Simulationen werden bisher oft aus OSM Daten generiert. Auf der offiziellen Website wird das Vorgehen erläutert. Für größere Gebiete müssen die OSM Daten von der Website *Geofabrik* (http://download.geofabrik.de/) heruntergeladen werden. Geofabrik stellt Ausschnitte der OSM Datenbank für verschiedene Teile der Erde in unterschiedlichen Detailstufen zur Verfügung. Danach können die Daten mittels des Kommandozeilenprogramms *Osmosis* (vgl. "Osmosis – OpenStreetMap Wiki," 2014) weiter gefiltert werden. Kern der Konvertierung der OSM Daten zu einem MATSim-Netzwerk ist der *OsmNetworkReader*, eine Java-Klasse, die von MATSim bereitgestellt wird. Dieser nimmt die OSM Daten und generiert eine MATSim Netz nach vordefinierten Einstellungen. Alle Einstellungen müssen jedoch manuell im Code vorgenommen werden. Das sind neben den Dateipfaden der Ein- und Ausgangsdateien auch Parameter, die die Konvertierung der Highway Klassen zu Attributen der MATSim Links definieren. Dies ist notwendig, da die meisten verkehrlichen Eigenschaften der Straßen, die von MATSim benötigt werden, nicht oder nur bedingt direkt durch Tags ersichtlich sind (Balmer, 2011).

Einmal konvertiert, können Änderungen nur noch manuell vorgenommen werden, indem die XML-Datei bearbeitet wird oder eine neue Konvertierung mit veränderten Parametern durchgeführt wird. Die Vorgehensweise des *OsmNetworkReader* kann vereinfacht wie folgt beschrieben werden:

1. Initialisieren, Setzen der Parameter
2. Einlesen der OSM Datei im angegebenen Dateipfad
3. Herausfiltern aller Nodes und Ways
4. Entfernen der Ways, deren Nodes nicht vollständig bekannt sind
5. Entfernen der Nodes und Ways, die nicht zu einem unterstützten Highway gehören
6. (Optional) Entfernen der Nodes, die nur geometrische Bedeutung besitzen und keine Kreuzung zweier Ways darstellen
7. Erzeugen äquivalenter MATSim Nodes aus den OSM Nodes
8. Erzeugen äquivalenter Links aus den Ways
9. Herausschreiben der Network Datei

Standardmäßig werden folgende Highways konvertiert (siehe auch Abschnitt 2.2.3):

- *motorway*
- *motorway_trunk*
- *primary*
- *primary_trunk*
- *secondary*
- *tertiary*
- *residential*
- *living_street*
- *unclassified*

Alle anderen Wege, Pfade usw. werden nicht konvertiert, da sie für den Kfz-Verkehr i.d.R. keine Bedeutung besitzen. Die meisten Straßen werden *ungerichtet* abgebildet, während MATSim gerichtete Links verwendet. Daher werden in zwei Richtungen befahrbare Straßen in zwei entgegengesetzte MATSim Links konvertiert (parallele Kanten).

Neben dem ‚*highway*' Tag gibt es weitere Tags, die Einfluss auf die Konvertierung einer Straße nehmen:

- *junction*: wenn ‚*junction=roundabout*' beschreibt der Way einen Kreisverkehr, der nur in eine Richtung befahren werden kann. Es werden daher nur Links in eine Richtung erstellt.
- *oneway*: wenn auf „wahr" gesetzt (‚*1*' oder ‚*true*') stellt der Way eine Einbahnstraße dar. Es werden dann auch hier nur Links in eine Richtung erstellt.

- *maxspeed*: wenn angegeben, überschreibt der Wert den Standardwert für diese Highway Klasse. Wichtig ist dabei, dass MATSim in Metern pro Sekunde rechnet während *maxspeed* zumeist in Kilometern pro Stunde getaggt wird.
- *lanes*: beschreibt die Anzahl aller Fahrpuren der Straße. Fälschlicherweise wird diese Zahl beim Konvertieren für Hin- und Rückrichtung einer zweibahnigen Straße genutzt, obwohl sie eigentlich beide Richtungen zusammenaddiert (vgl. "De:Key:lanes – OpenStreetMap Wiki," 2014).

Die Länge der Links wird beim Konvertieren durch die Addition der Länge der einzelnen Abschnitte eines Ways, die durch den Link repräsentiert werden sollen, errechnet. Die IDs der MATSim Nodes werden gleich der jeweiligen OSM ID der OSM Knoten gesetzt. Da die Ways eine *ungerichtete Kantenfolge* darstellen, die oft durch mehr als einen Link repräsentiert werden, ist eine direkte Übertragung der IDs nicht möglich, da ansonsten mehrere Links mit derselben ID existieren würden. Stattdessen werden die IDs durch eine inkrementelle Nummer vergeben. Die ursprüngliche OSM ID des zugrundeliegenden Ways wird als *original ID* (nachfolgend OriginalId) der Links gespeichert, sodass eine Rückverfolgung möglich ist.

Auf Grundlage dieser Konvertierung gibt es bereits den ‚networkEditor', der als *contribution* bzw. *Zusatzmodul* seitens MATSim angeboten wird. Das Programm wird aus einer Entwicklungsumgebung oder über die Konsole gestartet und erlaubt graphisches Darstellen sowie Bearbeiten von Networks. Die Darstellung erfolgt als Graph. Netze können eingelesen und exportiert werden. Auch OSM Datensätze können importiert werden und werden vor der Visualisierung mit dem standardmäßigen *OsmNetworkReader* konvertiert. Die Konvertierungsparameter können nicht angepasst werden. Beim Bearbeiten können Elemente gezeichnet und hinzugefügt oder entfernt werden. Außerdem kann der MATSim *NetworkCleaner* angewendet werden, um lose Kanten und Knoten zu entfernen. Zusätzlich können *Counts* der Links, also Zählungen der Verkehrsmengen editiert werden. Das Bearbeiten und Visualisieren von ÖV Daten ist nicht möglich. Die Konvertierung aus OSM Daten geschieht nicht interaktiv und kann nicht nachvollzogen werden (vgl. Rieser, 2014b).

Eine weitere Möglichkeit, ein MATSim Netz zu erhalten, ist die Konvertierung eines bestehenden *VISUM* Netzwerkes. Ähnlich wie bei der OSM Konvertierung wird auch dies manuell über Programmcode realisiert, der kein interaktives, visuelles Bearbeiten ermöglicht. Diese Herangehensweise wird in dieser Arbeit nicht näher erläutert.

Auch proprietär verfügbare Daten können herangezogen werden. Aber auch diese benötigen eine Anpassung an das MATSim Format, da die benötigten Informationen, zum Beispiel die Kapazität einer Straße, nicht immer komplett vorliegen. Standardwerte anhand von Straßenklassifikationen sind eine Möglichkeit der Datenaufbereitung. Außerdem werden topologische Details des Straßennetzes (ähnlich wie bei den mehrknotigen Ways bei OSM) nicht benötigt und können ausgedünnt werden (vgl. Meister et al., 2010, S. 7).

Für eine mit MATSim durchgeführte Studie für eine Modellierung des Verkehrs in Tel Aviv wurde auf bereits vorhandene Daten der lokalen Behörden zurückgegriffen. Diese lagen in Form von *EMME/2* Netzwerken vor. *EMME/2* ist eine kommerzielle Verkehrsplanungssoftware der Firma *iNRO*. Dabei konnten viele Informationen ohne größeren Aufwand übernommen werden (vgl. Bekhor et al., 2011, S.9).

Komplett neue Netzwerke, die auf keinen bestehenden Daten aufbauen, können nur über die manuelle Definition ihrer einzelnen Elemente im Code oder in einer Network Datei erstellt werden. Dies ist jedoch nur schwer bildlich vorstellbar und fehleranfällig.

Für Netzmodelle des ÖV gibt es bisher keine standardmäßige Konvertierungsroutine. Sie können auch nicht graphisch editiert werden. Viele Einstellungen müssen bislang manuell in Network und TSchedule angepasst werden.

2.6. Graphischer Editor für MATSim – Motivation und Anforderung

Die bisherigen Netzwerke für MATSim können zwar aus mehreren Datenquellen gewonnen werden. Jedoch erfolgt dies zumeist durch manuelles Konvertieren, bei dem der Benutzer Programmcode anpassen und ausführen muss. Das ist für Anwender ohne Programmierkenntnisse anspruchsvoll und auch fehleranfällig, da viele Schritte während des Konvertierens nicht nachvollzogen werden können. Das Konvertieren gleicht einem Black-Box-Verfahren, bei dem nur das endgültige Resultat ersichtlich ist. Hinzu kommt, dass das MATSim Netz im Anschluss nicht visuell und damit schwer zu bearbeiten ist. Bei Änderungen der zugrundeliegenden Daten muss auch das MATSim Netz neu konvertiert werden, wobei damit alle Änderungen nach der Konvertierung verloren gehen. Für den ÖV gibt es derzeit kein standardisiertes Verfahren zur Erzeugung des Infrastrukturmodells. Ein graphischer Editor, der den Nutzern bei den beschriebenen Problemstellungen hilft, bietet sich daher an.

Um auf eine große Auswahl an Rohinformationen zurückgreifen zu können, ist es von Vorteil, wenn der Editor mit frei verfügbaren Daten, die möglichst vollständig sind, arbeiten kann. Dafür kommen die OSM Daten in Frage. Ein weiterer Vorteil ist, dass es bereits den Editor JOSM für OSM gibt, der Erweiterungen in Form von Plugins erlaubt und eine Netzdarstellung in Graphenform ermöglicht. Das im Rahmen dieser Arbeit entwickelte Programm wurde daher als Plugin für JOSM entwickelt. Dieses soll eine interaktive Bearbeitung der OSM und MATSim Daten erlauben.

Eine interaktive Bearbeitung hat „den Vorteil, daß gewisse Entscheidungen nicht grundsätzlich im Voraus, sondern dann gefällt werden können, wenn schon Zwischeninformationen vorhanden sind“ (Budelmann, 1977, S. 5). Bei der bisherigen Konvertierung gibt es keine Zwischeninformationen und alle Parameter müssen im Voraus gesetzt werden. Für die interaktive Bearbeitung spricht außerdem, dass nicht zufriedenstellende Versuche und Fehler der Netzkonvertierung oder -bearbeitung früher erkannt und vermieden werden können (vgl. ebd., S.5f.). Das Ändern der Parameter sollte daher in Echtzeit zu Auswirkungen führen.

Wichtig ist außerdem, dass das Programm nachvollziehbare Ergebnisse produziert, die nach klaren Regeln und Anweisungen erstellt werden. Eine Aktion, die zweimal unabhängig auf die gleichen Daten angewendet wird, muss die gleichen Resultate hervorbringen. Damit wird auch die Reproduzierbarkeit des Netzes sichergestellt, die für etwaige Wiederholungs- oder Neuberechnungen mit leicht veränderten Daten benötigt wird (vgl. Hölsken und Ruske, 1977, S.87f.).

Visuelles Editieren der Netze hilft außerdem bei der Vermeidung von Fehlern im Modell. Hölsken und Ruske (vgl. ebd., S. 82) beschreiben allgemein vier Fehlerarten, die bei digitalen Netzmodellen auftreten können:

1. Fehlen einer Verknüpfung:

   Eine eigentlich vorgesehene Verknüpfung ist im Netz nicht enthalten. Dies tritt im MATSim Netz auf, wenn ein Link zwischen zwei Nodes nicht vorhanden ist.

2. Falsche Verknüpfung:
   Zwischen zwei Knoten ist fälschlicherweise eine Verknüpfung vorhanden. Dies tritt im MATSim Netz auf, wenn zwei Nodes durch einen Link verbunden sind, der nicht vorgesehen ist.

3. Fehlen der Qualitätsangabe
   Einzelne Informationen zu den Elementen werden vergessen. In MATSim könnten wichtige Link Attribute fehlen.
4. Falsche Qualitätsangabe
   Einzelne Informationen zu den Elementen sind falsch. In MATSim können beispielsweise falsche Längenangaben gesetzt sein.

Das Erfassen dieser Fehler kann durch eine überschaubare Darstellung der Elemente verbessert werden. Des Weiteren können automatische Prüfverfahren integriert werden.

Für die Anwendbarkeit in tatsächlich auftretenden Szenario-Größen muss das Programm große Mengen von Kanten und Knoten darstellen können. Das Tel Aviv Szenario nutzt ein Netzwerk mit 7.879 Nodes und 17.118 Links (vgl. Bekhor et al., 2011, S9). Für die gesamte Schweiz wurde einmal Planungsnetz mit 24.180 Nodes und 60.528 Links verwendet (vgl. Meister et al., 2009, S.9). Ein detaillierteres Netz der Schweiz beinhaltet 472.819 Nodes und 1.035.305 Links (Meister et al., 2010, S.7). Wichtig ist hierbei der Detailgrad der Netze. Oft ist es sinnvoll, kleine, eher unbedeutende Straßen zugunsten der Rechenzeit nicht aufzunehmen. Eine Klassifizierung der OSM Highways bietet sich als Grad zur Detailliertheit an und sollte berücksichtigt werden.

Wichtig ist weiterhin, dass Netze ohne Informationsverlust herausgeschrieben und wieder eingelesen werden können. Auch neue Netzwerke ohne Datengrundlage sollen erstellbar sein. Eine weitere Forderung ist, dass neben dem Straßennetz auch das öffentliche Verkehrsnetz abgebildet werden soll. Dabei sollen Linien- bzw. Routenverläufe und deren Haltestellen modellierbar sein.

Zusätzlich könnte ein graphischer Editor auch dazu genutzt werden, um Netzwerke zu analysieren und zu bewerten. Vergleichbar wäre dabei zum Beispiel die Erweiterung „*Network Analyst*“ des Geodaten-Programms „*ArcGIS*“. Diese ermöglicht unter anderem die Beantwortung folgender Fragen für Netzwerke (vgl. “Desktop Help 10.0 - Was ist Network Analyst?”):

- „Was ist die schnellste Möglichkeit, von Punkt A zu Punkt B zu gelangen?“
- „Welche Häuser liegen im Abstand von fünf Minuten von XYZ?“
- „Wo kann ein Unternehmen ein Geschäft eröffnen, um den Marktanteil zu erhöhen?“

Auch wenn diese Fragen nicht in genau dieser Form für MATSim Netze beantwortet werden müssen, wären ähnliche Fragestellungen denkbar. Zum Beispiel könnte die Erreichbarkeit eines bestimmten Aktivitätenortes beurteilt werden.

Die Verkehrsplanungssoftware Visum des Unternehmens PTV (vgl. “PTV Visum,” 2014) bietet bei seiner Netzdarstellung die Verwendung von Hintergründen an. Diese werden hinter den Graphen angezeigt und ermöglichen eine Orientierung an Satellitenbildern, politischen Grenzen oder Ähnlichem. Charakteristika der Kanten können durch auf ihnen angewendete Farben und Texte verdeutlicht werden. Die Vorteile, die sich aus den Visualisierungsmöglichkeiten ergeben, können auch bei der Bearbeitung von MATSim Netzen dienlich sein.

3. Graphischer Editor für MATSim Netze auf Basis von JOSM

Der graphische Editor wurde als Plugin für JOSM entwickelt. Der Pfad zum aktuell veröffentlichten Plugin der zum Einbinden in JOSM genutzt werden kann lautet:

http://svn.vsp.tu-berlin.de/repos/public-svn/dist/plugin

Sobald das Plugin installiert ist, steht es zur Anwendung bereit. Aufbau und Funktionsweise werden in den folgenden Abschnitten erläutert. Dabei beziehen sich die Ausführungen auf den noch nicht veröffentlichten Code im MATSim SVN Repository unter:

https://svn.code.sf.net/p/matsim/source/playgrounds/trunk/nkuehnel/ **(Revision 30539)**

Für eine konsistente Beschreibung der beiden Datenformate sei folgendes festgelegt:

- OSM Kanten werden weiterhin als Ways bezeichnet
- MATSim Kanten werden als Links bezeichnet
- Die OSM Knoten werden nachfolgend mit $\text{Node(s)}_{\text{OSM}}$ beschrieben
- MATSim Knoten erhalten die Bezeichnung $\text{Node(s)}_{\text{MATSim}}$

3.1. Datenhaltung der OSM und MATSim Daten

Die OSM Datensätze werden von JOSM mithilfe von Layers dargestellt. Jeder dieser Layer besitzt einen Datensatz (*data set*) und kann die darin enthaltenen Elemente anzeigen. Einzelne Elemente können auch versteckt werden, etwa wenn sie als gelöscht oder unsichtbar markiert werden. Für das Plugin wurde ein eigenes MATSim Layer entwickelt. Diese ist eine Erweiterung des OSM Layer und enthält zusätzlich zu einem OSM Datensatz ein MATSim Szenario samt Network und TSchedule. Diese Erweiterung ermöglicht dem Programm beim Speichern zu unterscheiden, ob eine OSM Datei oder MATSim Daten gespeichert werden sollen (siehe Kapitel 3.5). Des Weiteren werden die konvertierten MATSim Daten und Referenzierungen zwischen beiden Datenformaten auch bei einem zwischenzeitlichen Wechsel der Layer beibehalten.

Die in den Datensätzen beider Layer enthaltenen Information sind in Abbildung 5 veranschaulicht. Die in den grauen Feldern dargestellten Daten entsprechen den OSM Primitiven. In einem MATSim Layer sind zusätzlich die Elemente aus dem blauen Feld vorhanden. Ein *Scenario* Element wird von MATSim für die Simulation eines Szenarios

verwendet und fasst die einzelnen für die Simulation relevanten Bestandteile zusammen. Für das entworfene Programm sind dabei das Network und die TSchedule von Bedeutung. Zusätzlich gibt es drei verschiedene Datenmappings, die die beiden Datenmodelle miteinander verbinden und in den folgenden Abschnitten als *Referenzmaps* bezeichnet werden:

- Way2Links: Weist 1…n Links einem Way zu. Der Way stellt zur Laufzeit diese Links dar
- Link2Segments: Weist 1…n WaySegments einem Link zu. Der Link wird von den entsprechenden Segmenten des Ways dargestellt.
- Relation2Route: Weist einer Relation eine TRoute zu. Die TRoute wird von der Relation dargestellt

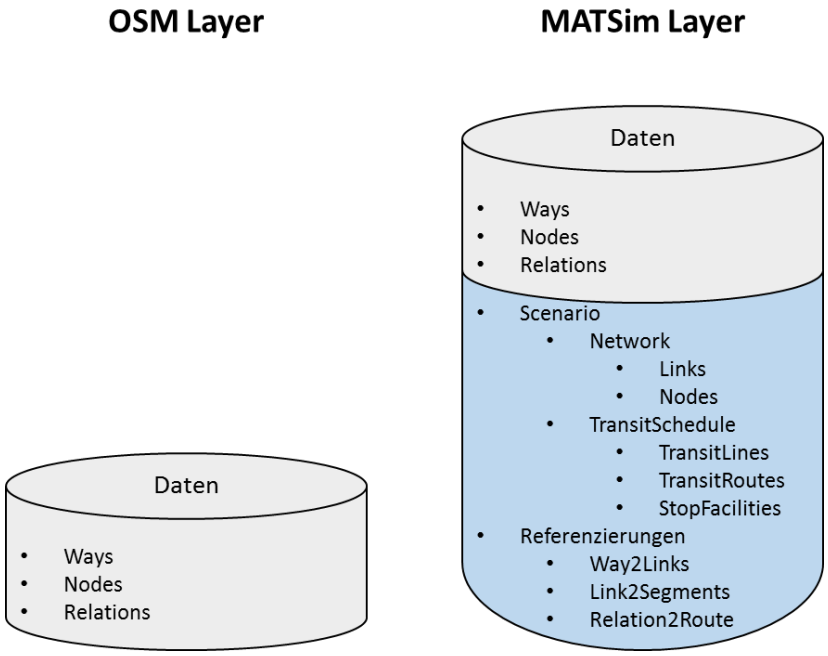

Abbildung 5: In den Daten der Layers enthaltene Informationen
Quelle: eigene Darstellung

3.2. Konvertierung

Da die MATSim Daten mithilfe von OSM Datentypen repräsentiert und bearbeitet werden sollen, ist es wichtig, diese konsistent miteinander zu halten und eine entsprechende Übersetzung zwischen den Formaten zu schaffen. Die Konvertierungsfunktion ist daher das Herzstück des Plugins und wird im Folgenden ausführlich erläutert.

Die Konvertierung von OSM Daten ist an mehreren Stellen des Programms möglich. Dabei kann unterschieden werden zwischen einer vollständigen Konvertierung, bei der der gesamte Datensatz der aktuellen Ebene konvertiert wird, und einer Konvertierung einzelner Elemente. Es ist wichtig, dass diese beiden Konvertierungsformen konsistent miteinander sind. Das heißt, dass ein konvertiertes Element immer in dem gleichen MATSim Äquivalent resultieren muss, unabhängig davon, ob nur dieses Element allein oder der gesamte Datensatz konvertiert wurde. Zu diesem Zweck wurde ein modularer Aufbau der konvertierenden Java-Klasse angestrebt, bei der die vollständige Konvertierung eines Datensatzes auf ein mehrmaliges Aufrufen der Umwandlungsfunktion für einzelne Elemente beruht. Die umgewandelten Daten werden daraufhin entweder temporär behalten oder als Information zu einer MATSim Datenebene hinzugefügt.

3.2.1. IV / Straßenmodell

Für das Straßennetzmodell wurde auf die Funktionalität des bestehenden *OsmNetworkReader* (siehe Kapitel 2.5) in abgewandelter Form zurückgegriffen. Bevor die Vorgehensweise erklärt wird, ist es wichtig zu wissen, welche Daten konvertiert werden. Damit Ways in Links umgewandelt werden können, benötigen sie die richtigen Tags. Die Straßen aus OSM werden mittels des ‚*highway*' Tags identifiziert und mit den jeweiligen Standardwerten konvertiert, sofern der Highway Typ unterstützt wird (siehe Kapitel 2.5). Die Ways resultieren dann in der Regel in mehreren Links, je nach Anzahl der Segmente des Ways und ob eine oder zwei Fahrtrichtungen dargestellt werden. Zusätzlich dazu wurde ein eigenes Tagging Schema für MATSim eingeführt. Grund dafür ist, dass es eine Möglichkeit geben soll, Links mit unterschiedlichen Attributen darstellen zu können, die nicht immer auf einen vordefinierten Highway Typ passen. Das ist vor allem beim Importieren vorhandener Netze wichtig, da nicht immer eine Zuordnung eines Links zu einem Highway anhand seiner Attribute möglich ist. Das Schema basiert darauf, dass die Attribute für jeden Way frei editiert werden können. Alle Links, die aus einem solchen Way konvertiert werden, erhalten dann die in den Tags beschriebenen Attributwerte. Die folgenden, in Tabelle 7 mit Beispielen dargestellten, Tags wurden definiert:

Tabelle 7: MATSim Tagstruktur
Quelle: eigene Darstellung

Schlüssel	Bedeutung	Wert
id (optional)	setzt die Ids der vom Way dargestellen Links auf den angegebenen Wert, einer aufsteigenden Nummer sowie ggf. einem Kürzel für die Richtung	alphanumerisch: • „1234“ • „newLinks“ • „B96“
capacity	setzt die Kapazität der Links auf den gegebenen Wert (Fzg. pro Zeitintervall)	numerischer Wert • „3000“
freespeed	setzt die Höchstgeschwindigkeit der Links auf den gegebenen Wert (Meter pro Sekunde)	numerischer Wert • „16.6666“
permlanes	setzt die Anzahl der ständigen Fahrstreifen der Links auf den gegebenen Wert	numerischer Wert (i.d.R. ganzzahlig) • „3“
modes	setzt die erlaubten Verkehrsmodi der Links auf die gegebenen Werte	Aufzählung vordefinierter Strings, mit Komma getrennt • „car,pt,bike“
length (optional)	setzt die Länge der Links auf den anteiligen Wert des gegebenen Werts; der jeweilige Anteil entspricht dem Anteil der Links an der dargestellten Gesamtlänge des Ways (in Metern)	numerischer Wert • „385,68“

Das Entscheidende ist nun, dass ein Way auch durch das Setzen dieser Tags konvertiert wird. Es müssen jedoch bis auf ‚*id*‘ und ‚*length*‘ alle Tags gesetzt sein. Wenn nur MATSim Tags genutzt werden, wird der Way immer nur in eine Richtung konvertiert und zwar in Zeichenrichtung. Besitzt der Way jedoch ein unterstütztes ‚*highway*‘ Tag und MATSim Tags, wird der Way je nach Highway Typ bzw. ‚*oneway*‘ Tag in ein oder zwei Richtungen konvertiert. Die MATSim Attribute müssen dann nicht vollständig sein und überschreiben die Standardwerte des Highways für diesen Way. Für die $Nodes_{OSM}$ kann auch ein optionales ‚*id*‘ Tag gesetzt werden, das bei der Konvertierung berücksichtigt wird.

Der eigentliche Ablauf soll nun mithilfe des Flussdiagramms aus Abbildung 6 für eine vollständige Konvertierung erläutert werden.

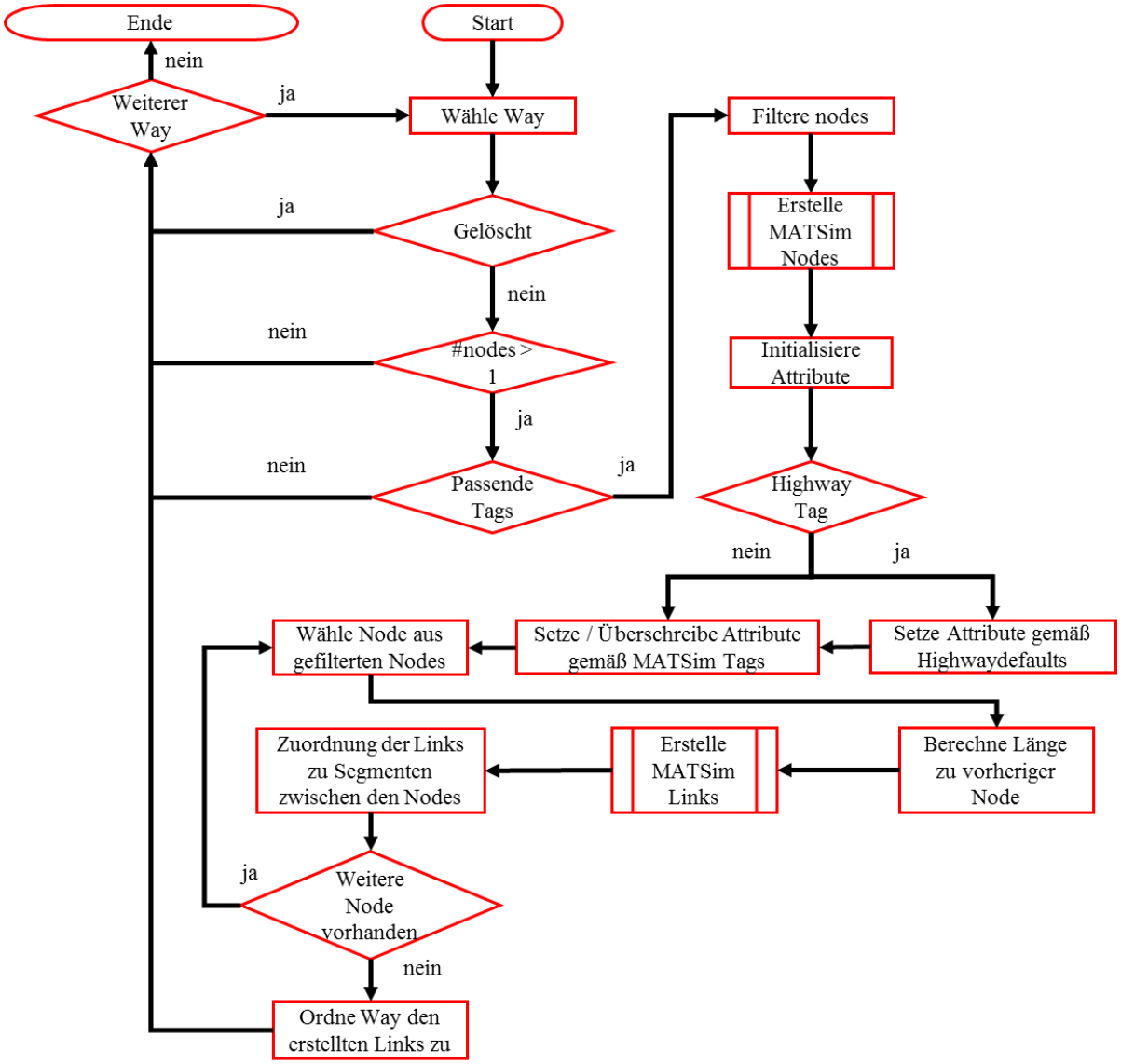

Abbildung 6: Flussdiagramm des Konvertierungsablaufs für Straßen
Quelle: eigene Darstellung

Bei der Konvertierung werden zuerst alle Ways aus dem Datensatz des Layers gelesen. Nacheinander werden diese dann überprüft. Um in MATSim Informationen umgewandelt werden zu können, muss ein Way

1. als nicht gelöscht markiert sein: JOSM behält manche Daten auch nach dem Löschen durch den Benutzer im Datensatz zurück und markiert sie als ‚*deleted*'.
2. mehr als einen Knoten besitzen: Auch wenn die grafischen Editierprogramme einen Way mit nur einem $Node_{OSM}$ nicht zulassen, wird dies nicht durch das Datenmodell an sich verboten und muss daher ausgeschlossen werden.
3. passende Tags besitzen: Nur Ways mit unterstütztem ‚*highway*' Tag oder vollständiger MATSim Tags werden konvertiert.

Erfüllt ein Way all diese Bedingungen, werden seine $Nodes_{OSM}$ gegebenenfalls gefiltert. Da ein Way eine Kantenfolge beschreibt, bei der einzelne Knoten innerhalb der Folge auch nur rein geometrische Informationen und keine Kreuzung darstellen können, werden für die Simulation unerhebliche Knoten aussortiert. Das Aussortieren ist in folgendem gekürzten Codeausschnitt dargestellt (aus der Klasse *NewConverter.java*):

```
for(int l=0; l < way.getNodesCount(); i++) {
       Node current = way.getNode(l);
       List<Node> nodeOrder = new ArrayList<Node>();
       for (int l = 0; l < way.getNodesCount(); l++) {
              if (l == 0 || l == way.getNodesCount() - 1 || keepPaths) {
                     nodeOrder.add(current);
              } else if (current.equals(way.getNode(way.getNodesCount() - 1))) {
                     nodeOrder.add(current);
              } else if (current.isConnectionNode()) {
                     for (OsmPrimitive prim : current.getReferrers()) {
                            if (prim instanceof Way && !prim.equals(way)) {
                                   if (((Way) prim).hasKey(TAG_HIGHWAY) ||
                                   meetsMatsimReq(prim.getKeys())) {
                                          nodeOrder.add(current);
                                          break;
                                   }
                            }
                     }
              }
       }
}
```

Zuerst wird eine Liste für die Reihenfolge der behaltenen $Nodes_{OSM}$ erstellt (`nodeOrder`). Danach wird jeder Knoten (`current`) der Reihe nach (die Reihenfolge entspricht der Zeichenrichtung des Ways) überprüft. Die nachfolgend genannten Knotennummern beziehen sich auf Abbildung 7.

1. Handelt es sich um den ersten oder letzten Knoten wird er auf jeden Fall aufgenommen (Knoten 1 und 6).
2. Falls der aktuelle Knoten dem letzten Knoten entspricht, aber bereits an früherer Stelle auftaucht, dann stellt dieser Way einen geschlossenen Kreis dar (z.B. Wendeschleife, Kreisverkehr). Der aktuelle Knoten wird dann zweimal aufgenommen, da der Way an dieser Stelle sich selbst kreuzt (Knoten 6).
3. Ist der aktuelle Knoten ein Verbindungsknoten (‚`isConnectionNode()`') bedeutet das, dass er Bestandteil von mindestens zwei verschiedenen Ways ist. Stellt einer der anderen Ways dann ebenfalls eine Straße dar (egal ob ‚*highway*' Tag oder MATSim Tags), wird der Knoten aufgenommen, da er eine verkehrliche Relevanz als Kreuzung besitzt (Knoten 4).

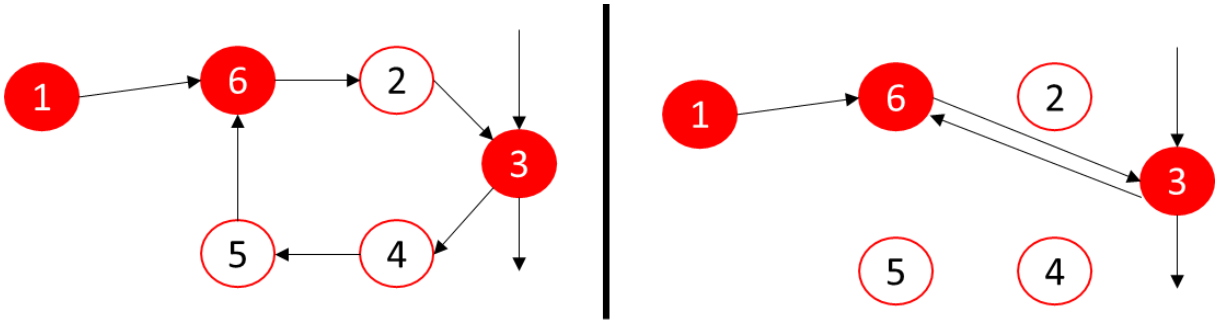

Abbildung 7: Filtern der Nodes – Verbindungen davor (links) und danach (rechts)
Quelle: eigene Darstellung

Die Knotenreihenfolge aus dem Beispiel wird dabei von ‚1-6-2-3-4-5-6' zu ‚1-6-3-6'.

Alle anderen Knoten sind damit aussortiert. Das Sortieren kann mittels der Option *keepPaths* (siehe Kapitel 3.8) unterbunden werden, dann werden alle Knoten ausnahmslos in das MATSim-Modell übertragen.

Nach dem Herausfiltern der $Nodes_{OSM}$ werden diese in $Nodes_{MATSim}$ umgewandelt. Dabei wird jedoch zuerst überprüft, ob der Knoten bereits im zugrundeliegenden Network vorhanden ist. Das geschieht, indem die ID der $Node_{OSM}$ innerhalb der $Nodes_{MATSim}$ des Networks gesucht wird (die ID der $Nodes_{MATSim}$ entspricht zur Laufzeit immer der ID der $Nodes_{OSM}$). Ist dies der Fall, wird lediglich überprüft, ob ein ‚*id*' Tag gesetzt wurde. Wenn ja, wird die OriginalId der $Node_{MATSim}$ auf den Inhalt dieses Tags gesetzt. Außerdem werden die Koordinaten der $Node_{MATsim}$ an das OSM Äquivalent angepasst. Ist ein Knoten noch nicht im Network vorhanden, so wird ein neuer $Node_{MATSim}$ mit den jeweiligen Koordinaten erstellt. Auch dieser erhält die OSM ID und auch hier wird das ID Tag auf die gleiche Weise gehandhabt.

Sobald alle Knoten als $Nodes_{MATSim}$ vorliegen, werden die Attribute der Links initialisiert. Anschließend wird überprüft, ob der Way durch ein ‚*highway*' Tag definiert wird. Wenn ja, dann werden die Attribute nach den jeweiligen Standardwerten für den Highway Typ gesetzt. Falls zusätzlich MATSim Tags vorliegen, überschreiben sie das jeweilige Attribut. Liegt kein ‚*highway*' Tag vor, werden die Attribute allein aus den MATSim Tags kreiert. Die Links werden dann nur in eine Richtung erstellt, da Kanten nach MATSim Definition immer gerichtet sind. Bei einem *ungerichtetem* Highway hingegen werden alle Links als entgegengesetzte, *parallele Kanten* erstellt.

Das Erzeugen der Links geschieht abschnittsweise für je zwei aufeinanderfolgende $Nodes_{OSM}$ aus der gefilterten Liste. Zuerst werden all Segmente des Ways erfasst, die zwischen den beiden $Nodes_{OSM}$ liegen, also die Kanten, die innerhalb der Kantenfolge

zwischen je zwei $Nodes_{OSM}$ verlaufen. Daraufhin wird die Länge zwischen den $Nodes_{OSM}$ berechnet. Dabei wird nicht die direkte Entfernung zwischen den $Nodes_{OSM}$ genommen, sondern die Strecke, die alle Way Segmente dazwischen aufsummieren. Falls jedoch ein ‚*length*' Tag gesetzt worden ist, wird die im Tag beschriebene Länge je nach Anteil der zuvor errechneten Länge an der Gesamtlänge des Ways aufgeteilt. Falls also ein ‚*length*' Tag mit dem Wert ‚300' gesetzt ist, und die Way Segmente zwischen den zwei $Nodes_{OSM}$ ein Drittel der Gesamtlänge des Way ausmachen, erhält die Strecke die Länge ‚100'. Nach der Längenberechnung erfolgt die eigentliche Linkerzeugung. Zwischen den zwei $Nodes_{OSM}$ werden entweder zwei parallele, entgegen gerichtete Links, ein Link in Richtung des Way (‚*oneway = true*') oder ein Link entgegen der Richtung des Way (‚*oneway = -1*') erzeugt. Die Links erhalten die ID des zugrundeliegenden Ways. Da die IDs eindeutig sein müssen, wird eine aufsteigende Zahl, getrennt durch einen Unterstrich, für jeden weiteren Link an die OSM ID angehangen. Für die jeweilige parallele Rückrichtung eines Links, erhält die ID des entgegengesetzten Links neben der Zahl auch noch ein ‚*r*', wie ‚*reverse*'. Ein ungerichteter Way mit zwei zu konvertierenden Segmenten und der ID ‚7' resultiert dann in Links mit den IDs ‚7_0', ‚7_0_r', ‚7_1' sowie ‚7_1_r'. Falls dem Way ein ID Tag zugewiesen wurde, wird die OriginalId der Links auf diesen Wert gesetzt. Auch dabei werden wieder aufsteigende Nummern sowie das ‚*r*' verwendet um die Eindeutigkeit sicherzustellen. Für jeden der Links wird dann ein Eintrag in der Referenzmap Link2Segments angelegt, mit den Way-Segmenten als zugehörige Werte. Somit können alle Segmente eines Ways, die ein bestimmter Link darstellt, referenziert werden.

Nachdem nacheinander alle Links entlang der $Nodes_{OSM}$ des Ways erzeugt wurden, wird wiederum die Referenzmap Way2Links mit dem jeweiligen Way als Schlüssel gefüllt, dem alle Links, die entlang des Ways konvertiert wurden, zugeordnet werden. Sobald alle Ways nach diesem Schema behandelt wurden, ist die Konvertierung abgeschlossen. Das Network enthält die erzeugten MATSim Elemente, die intern über die Referenzmaps mit den OSM Daten referenziert und konsistent gehalten wird.

Da JOSM zur Erleichterung des Setzen von Tags Presets verwendet, wurde für das MATSim Schema ein eigenes Preset erstellt (siehe Abbildung 8 und Anhang D). Dieses kann auf bestehende Ways angewendet werden und erleichtert das Eintragen der Werte. Falls Tags keinen Wert durch den Nutzer zugewiesen bekommen, werden sie mit Standardwerten versehen (z.B. ‚*capacity=0.00*').

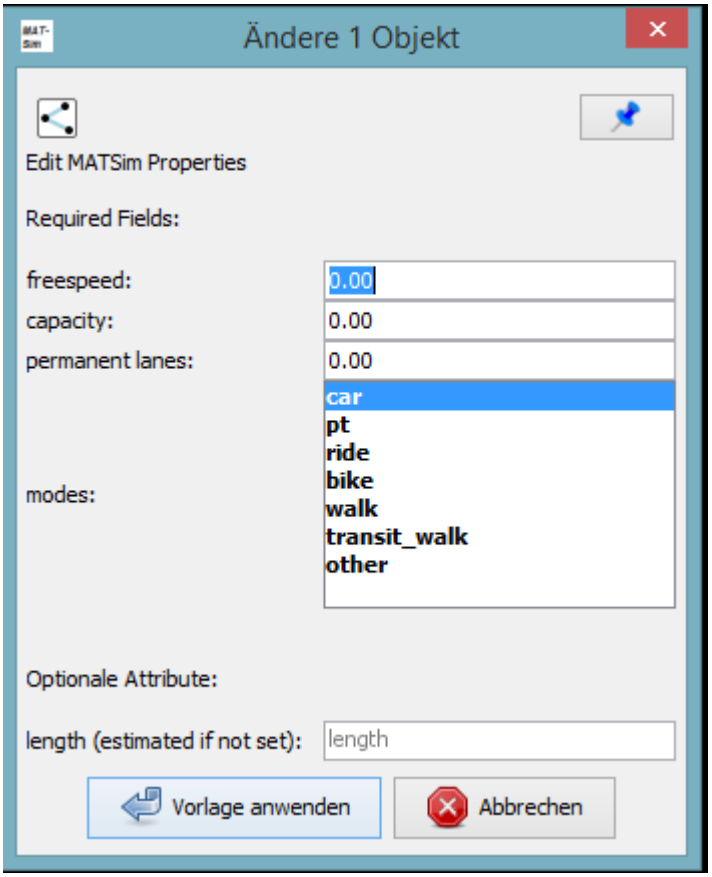

Abbildung 8: MATSim Tagging Preset für Ways
Quelle: eigene Darstellung, entnommen aus JOSM

Die Standardwerte für die Konvertierung der OSM Highways können in den Einstellungen angepasst werden. Mehr dazu in Kapitel 3.8.

3.2.2. ÖV / Routen

Die Konvertierung der Routen des ÖV bezieht sich vor allem auf die verwendeten Routenrelationen, die, wie bereits beschrieben, in großen Teilen den TRoutes einer TLine in MATSim entsprechen. Bevor der eigentliche Ablauf der Konvertierung beschrieben wird, werden zunächst einige Überlegungen und Probleme, die während der Entwicklung auftraten, angeführt.

Die Haltestellen in OSM sind aufgrund der unverbindlichen Tagging Schemata recht unterschiedlich eingepflegt (siehe auch Kapitel 2.2.3), was einige Probleme mit sich bringt. Grundsätzlich werden Haltestellen dort als $Nodes_{OSM}$ auf der Strecke oder ‚frei' neben der Strecke gesetzt. Während für viele Haltestellen jede Straßen-/Gleisrichtung einzeln markiert wird, gibt es auch einige Haltestellen, die von mehreren Richtungen genutzt werden und als

ein einziger $Node_{OSM}$ gesetzt sind. Abbildung 9 zeigt die U-Bahnhaltestelle Olympia-Stadion in Berlin. Obwohl es mehrere Gleise gibt und die dort verkehrende U2 in zwei unterschiedliche Richtungen fährt, wird die Haltestelle für beide Richtungen als ein einziger $Node_{OSM}$ mit dem Tag *‚railway=station‘* außerhalb der Ways dargestellt. Wie in Abschnitt 2.2.3 beschrieben, sollten Haltestellen eigentlich mit dem Tag *‚public_transport=stop_position‘* auf dem Way definiert werden. Es wird deutlich, dass das offene Datenformat ohne feste Richtlinien zu Inkonsistenz führen kann.

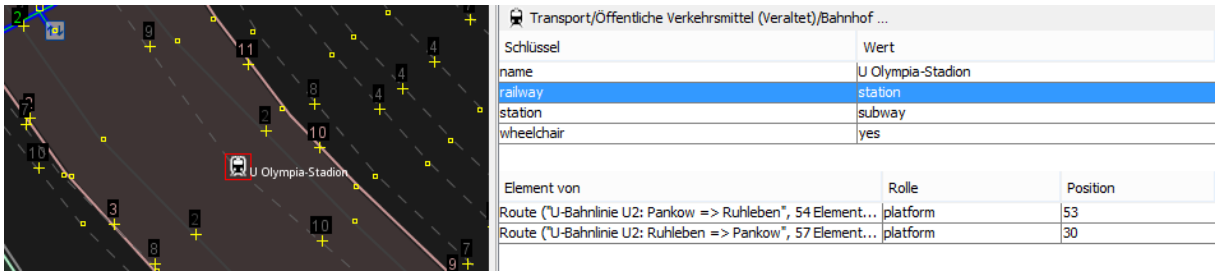

Abbildung 9: U-Bahnhaltestelle Olympia-Stadion ist Element entgegengesetzter Routen
Quelle: eigene Darstellung, entnommen aus JOSM

In MATSim hingegen sind die StopFacilities immer an *einen* Link der Route gekoppelt, über dessen Ende sie erreicht werden können. Das bedeutet auch, dass sie nur von einer bestimmten Richtung erreicht werden können. *Rieser* beschreibt zwei daraus resultierende Probleme (vgl. Rieser, 2010, S. 86f.):

1. auch die erste Haltestelle einer Route benötigt einen eingehenden Link
2. und es kann vorkommen, dass eine Haltestelle von mehreren Linien / Richtungen bedient wird und daher am Knotenpunkt, der die Haltestelle repräsentiert, mehr als ein eingehender Link existiert.

Es gibt nun mehrere Möglichkeiten, diese Probleme zu lösen. Eine erste Möglichkeit wäre es, an jedem $Node_{MATSim}$, der eine Haltestelle darstellen soll, eine kleine Schleife einzubauen, die den $Node_{MATSim}$ mit sich selbst verbindet und an deren Ende die StopFacility gekoppelt wird. Jeder eingehende Link, der dann mit dem $Node_{MATSim}$ verbunden ist, kann über die Schleife die StopFacility erreichen. Somit können auch von mehreren Seiten bediente Haltestellen abgebildet werden.

Eine andere Möglichkeit ist, die StopFacility für jede Route, die sie verwendet, zu duplizieren. Das löst das zweite Problem, da zwar jede StopFacility nur einen eingehenden Link haben darf, ein Link aber unbegrenzt viele StopFacilities anbinden kann. Die erste Haltestelle müsste hingegen weiterhin einen ‚dummy Link‘ als Schleife erhalten. Diese Methode hat den Vorteil, dass nicht so viele sichtbare Elemente erzeugt werden müssen.

Dummy Links sind im Network vorhanden und müssten daher visualisiert werden. Jedoch müssten dann automatisch Ways erzeugt werden, was schwierig und nicht konsistent ist. StopFacilities hingegen werden nicht visualisiert, weshalb die Duplikation unproblematisch ist. Für eine spätere Simulation des ÖV ist dies auch unkritisch, da die StopFacilities, wenn dupliziert, dieselben Koordinaten besitzen und somit vom Router der Simulation angemessen behandelt werden. Der Router sucht Haltestellen zum Umsteigen in der Nähe anderer Haltestellen innerhalb einer einstellbaren Distanz. Bei gleichen Koordinaten werden somit alle Darstellungen dieser Haltestelle gefunden oder gar keine (vgl. Rieser, 2010, S. 74). Der zweite Ansatz wurde daher gewählt und ähnelt dem Vorgehen, den *Rieser* für seine ÖV-Integration genutzt hat. Der Unterschied ist hierbei, dass er nur die erste gemeinsame StopFacility zweier Linien dupliziert (vgl. ebd., S. 87). Da die genaue Fahrtroute zwischen den Stopps in OSM nicht immer einwandfrei rekonstruiert werden kann, werden in einer ersten Implementation der Konvertierung alle mehrfach genutzten StopFacilities dupliziert.

Die nicht immer genaue Abbildung der Fahrtrouten ist ein weiteres Problem des offenen OSM Datenformats. Die Routen benötigen lediglich eine Relation mit den entsprechenden Tags, um als solche definiert zu sein. Ob die Elemente der Routenrelation vollständig oder plausibel sind, wird nicht überprüft. So gibt es viele Strecken, deren Routenverlauf falsch eingepflegt ist. Neben fehlenden Haltestellen, die einheitlich auf oder neben dem Fahrweg platziert werden, ist für den Routenverlauf vor allem die vollständige sowie lückenlose Aufnahme der Ways der Route in richtiger Abfolge entscheidend. Abbildung 10 zeigt einen Auszug aus der Relationsliste der Routenrelation des *RE5* von Lutherstadt Wittenberg nach Rostock. An den gelb markierten Punkten sind Unstetigkeiten vorhanden. Die Symbole zeigen an, dass der jeweilige Way nicht mit dem vorherigen und / oder nachfolgendem Way verbunden ist.

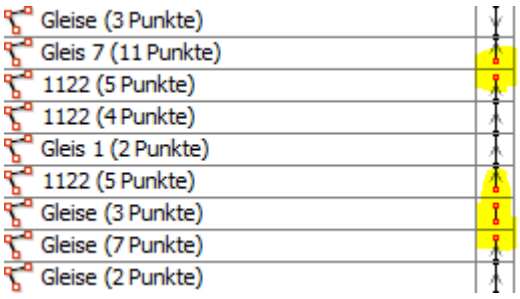

Abbildung 10: Unstetigkeiten in der Route des RE 5
Quelle: eigene Darstellung, entnommen aus JOSM

Um dieses Problem zu umgehen, wurde die Konvertierung vorerst so ausgelegt, dass die Haltestellen einer Route, unabhängig von der Route in der Relation, direkt über eine

Luftlinie verbunden werden. Die MATSim Routen werden daher abstrakt und beeinflussen einander nicht, da durch die doppelten StopFacilities und den direkten Luftverbindungen für jede Route kein Link von mehr als einer Route genutzt wird. Der ÖV ist somit vorerst auch vom Netz des IV abgeschnitten und existiert quasi eigenständig. Des Weiteren sind auch die konvertierten Werte der Links der Luftlinienroute falsch. Neben der wesentlich kürzeren Gesamtlänge der Route, können nur Standardattribute für Kapazität, Geschwindigkeit etc. angenommen werden.

Da OSM Routen bisher nur über Luftlinienverbindungen konvertiert werden, wurde ein für MATSim geeignetes, eigenes Tagging Schema erstellt, das eine genauere Routenkonvertierung erlaubt. Dabei werden die Relationen mit dem Tag *‚type=matsimRoute‘* und *‚routeId=id‘* gekennzeichnet. Näheres zu dieser Vorgehensweise erfolgt bei der Erläuterung der Konvertierung mittels dieser Tags.

Es folgt eine Beschreibung des implementierten Ablaufs der ÖV Konvertierung. Zur Veranschaulichung soll auch hier ein Flussdiagramm, das auf Abbildung 11 zu sehen ist, dienen.

Für die Konvertierung der ÖV-Routen wird jede Relation eines Datensatzes geprüft. Zuerst wird untersucht, ob die Relation als gelöscht markiert wurde. Ist das nicht der Fall, wird nun nach verwendeten Tags unterschieden. Weist die Relation die Tags einer OSM Route auf, erfolgt die Konvertierung auf anderem Wege, als wenn MATSim Tags verwendet werden. Die Konvertierung durch OSM-Tags wird als erstes beschrieben.

Zunächst werden die Elemente der Relation sortiert. Dabei werden die Elemente nach ihrer geometrischen Verbundenheit angeordnet, d.h. es wird versucht, eine zusammenhängende Kantenfolge herzustellen. Somit können falsch geordnete Relationen vorweg ‚repariert‘ werden.

Anschließend werden alle Knoten der Relation untersucht. Die Ways werden aufgrund der direkten Verbindung der Haltestellen nicht berücksichtigt. Da die Tags der Haltestellen bei vielen Haltestellen nicht den der beschriebenen Richtlinie entsprechen, aber die Haltestellen dennoch immer durch $Nodes_{OSM}$ dargestellt werden, wird in diesem Algorithmus lediglich geprüft, ob ein Relationsmitglied ein $Node_{OSM}$ ist. Es wird dann automatisch angenommen, dass dieser eine Haltestelle darstellt. Bevor nun eine StopFacility an diesen Koordinaten erstellt wird, wird zuerst ein $Node_{MATSim}$ an dieser Stelle erstellt (bzw. aktualisiert, falls der

Knoten bereits konvertiert wurde). Die Haltestelle wird zu einer Liste von StopFacilities hinzugefügt. Zu diesem Zeitpunkt wird noch kein eingehender Link spezifiziert.

Die ID der StopFacility setzt sich aus einer aufsteigenden Nummer für jedes Duplikat der Haltestelle, der OSM ID der Routenrelation sowie der OSM ID des Knotens zusammen (Beispiel: ‚2_123_456' für eine Haltestelle, die zum dritten Mal verwendet wird, der Relation ‚123' angehört und deren zugehöriger $Node_{OSM}$ die ID ‚456' besitzt). Außerdem wird der Name der Station, falls getaggt, ausgelesen und der StopFacility zugewiesen.

Nachdem alle Mitglieder der Relation durchlaufen und Haltestellen gefunden wurden, wird die *network route* zwischen den StopFacilities erstellt. Dies geschieht über eine Funktion, die für spätere Anpassungen modular eingebettet ist und die Links sowie deren Reihenfolge zurückgibt. Bei der momentanen Herangehensweise der direkten Luftlinienverbindungen werden in der Funktion nacheinander direkte Links zwischen den Knoten, die die StopFacility repräsentieren, erzeugt. Die Länge der Links entspricht der Distanz zwischen den Knoten und ist daher nicht geeignet, um die reale Strecke zwischen zwei Haltestellen abzubilden. Da die Routen immer genau einer Richtung entsprechen, kann die jeweilige StopFacility, zu der ein Link führt, an dessen Ende „gehangen" werden. Somit entspricht dieser dem eingehenden Link. Für die erste Haltestelle der Route wird ein Link als Schleife am Knoten erzeugt, der als eingehender Link gesetzt wird. Die ID der Links besteht aus einer aufsteigenden Nummer beginnend bei null sowie der OSM ID der zugehörigen Relation (zum Beispiel „123_4 für den Link, der die fünfte und sechste Halstestelle miteinander verbindet). Der Modus wird zunächst allgemein auf ‚*pt*' gesetzt.

Die einzelnen Links werden in der Reihenfolge ihrer Erstellung einer Liste hinzugefügt, die für die Erstellung der Route genutzt wird. Ist die Link Abfolge komplett, wird die Route entweder einer bestehenden TLine hinzugefügt, wenn bereits eine andere Route der Linie existiert oder eine neue TLine erstellt. Die TLine wird über den Namen der Linie, der über das Relations-Tag ‚*name=*name**' festgelegt wird, identifiziert. Schließlich wird die TLine inklusive der Route(n) in die TSchedule aufgenommen. Die Referenzierung der Relation zu der TRoute erfolgt über Relation2Route, wo die Werte nach der Konvertierung eingetragen werden. Diese Vorgehensweise wird für alle Relationen mit OSM Tag Schema wiederholt.

Die betrieblichen Informationen zu Abfahrts- und Ankunftszeiten können über das OSM-Format nicht ermittelt werden und werden daher auch nicht gesetzt.

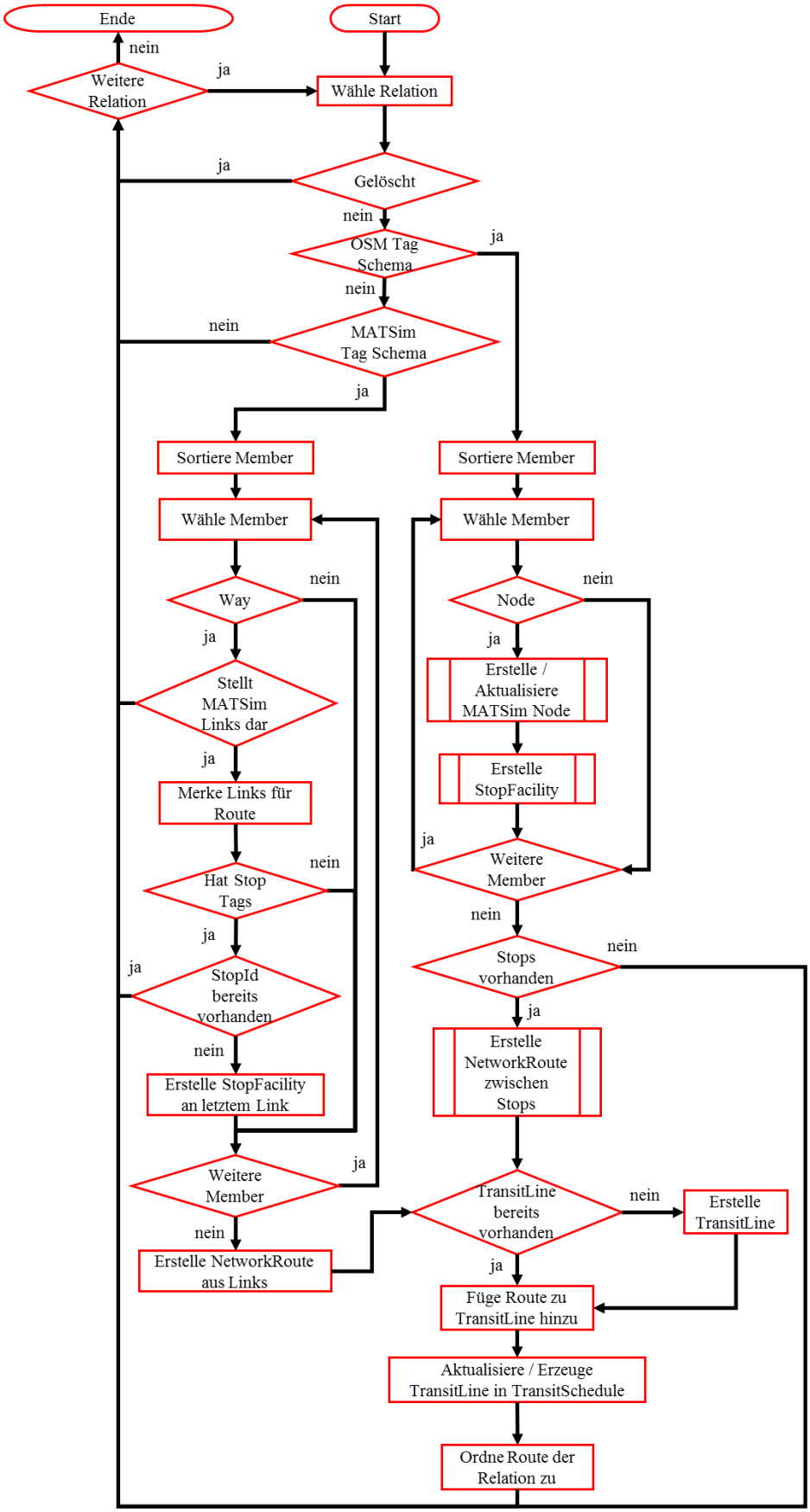

Abbildung 11: Flussdiagramm des Konvertierungsablaufs für ÖV-Routen
Quelle: eigene Darstellung

Wird die Route stattdessen über die MATSim Tags ‚*type=matsimRoute*' und ‚*routeId=id*' definiert, liegt der Fokus auf den Ways der Relation. $Nodes_{OSM}$ werden hier nicht mehr benötigt. Die Route setzt sich allein aus aufeinanderfolgenden Ways zusammen, die bereits Links darstellen müssen, d.h. sie sollten über die MATSim Tags, mit der auch Links für den IV konvertiert werden, verfügen (der Modus sollte auf ein öffentliches Transportmittel gesetzt werden). Die Haltestellen werden hier so definiert, dass sie am letzten Knoten eines Ways liegen. Diese Festlegung lässt sich leicht auf das Modell der StopFacilities übertragen, die einen eingehenden Link benötigen. Da Ways nun aber Kantenfolgen sind und die StopFacilities einfache Kanten benötigen, wird die StopFacility immer an den letzten, durch den jeweiligen Way dargestellten, Link ‚gehangen'. Der Way muss dafür außerdem ein Tag ‚*stopId=*stopId**' besitzen. Andernfalls stellt er lediglich einen Abschnitt des Routenverlaufs dar.

Für die Konvertierung einer solchen Relation wird folgendermaßen vorgegangen. Zuerst werden auch hier die Mitglieder der Relation sortiert. Danach werden die Ways der Relation untersucht. Stellen sie bereits Links im Network dar, was durch die Abfrage in Way2Links sichergestellt wird, werden die Links einer Liste für die spätere Route hinzugefügt. Stellt auch nur ein Way der Relation keine Links dar, wird die gesamte Relation nicht konvertiert, da dann eine lückenlose Route nicht möglich ist. Ist ein Way als Haltestelle getaggt, wird eine Haltestelle am Ende des letzten Links des Ways erzeugt und in die Route aufgenommen. Aus diesen Informationen wird dann die *network route* erzeugt. Das weitere Vorgehen nach dem Erstellen der *network route* gleicht dem der Konvertierung mit OSM Tags (siehe Abbildung 11).

Somit gibt es, ähnlich wie bei der Konvertierung des Straßennetzes für den IV, auch hier zwei Wege, ÖV-Routen zu definieren.

3.2.3. Networklistener / Online Prinzip

Die in Kapitel 2.6 geforderte interaktive Bearbeitung führt zu dem Anspruch, dass die Konvertierung in Echtzeit bzw. ‚online' geschehen soll. Änderungen an den zugrundeliegenden OSM-Daten sollen sofort ersichtliche Auswirkungen am MATSim Network hervorrufen. Zu diesem Zweck verfolgt das Plugin die vom Benutzer hervorgerufenen Manipulationen und sorgt für eine stete aktualisierte und konsistente Datenhaltung beider Formate. Dazu wird an den jeweils aktuellen Datensatz ein *Listener*

registriert, der die von JOSM erzeugten *Events*, die bei Änderungen an den Daten geworfen werden, verarbeitet. Der Listener reagiert auf jegliche Änderung, indem er die betroffenen OSM Elemente einzeln neu konvertiert. Die Konvertierung einzelner Elemente erfolgt dabei auf derselben Art und Weise wie bei einer vollständigen Konvertierung des Datensatzes, sodass die in Echtzeit verarbeiteten Daten in denselben Informationen resultieren. Der Nutzer kann somit alle Auswirkungen seiner Änderungen sofort einschätzen. Die erfassten *Events* der Änderungen sind in Tabelle 8 dargestellt.

Tabelle 8: erfasste events für die Echtzeitkonvertierung
Quelle: eigene Darstellung

Event	**Bedeutung**	**Reaktion durch Listener**
data changed	tritt auf, wenn sich eine große Menge von Daten auf einmal geändert hat	neu Laden des Layers und damit verbundene komplette Konvertierung
node moved	tritt auf, wenn sich die Koordinaten eines Knotens geändert haben	Konvertierung aller Ways, die diese verbinden
primitive added	tritt auf, wenn ein OSM-Element hinzugefügt wird	keine Reaktion bei Knoten; jeweilige Konvertierung des Ways oder der Relation
primitive removed	tritt auf, wenn ein OSM-Element entfernt wird	- Knoten: entsprechende Node wird aus Network gelöscht; Neukonvertierung aller Ways die den Knoten verbinden - Way: entsprechende Links werden aus Network und den HashMaps way2Links / link2Segments entfernt - Relation: Löschen der Links der Route, wenn sie nur Luftlinienverbindungen sind; Löschen der StopFacilities; Löschen der TransitRoute in der TransitLine; Löschen der TransitLine, wenn keine Route mehr vorhanden
relation members changed	tritt auf, wenn sich die Mitglieder einer Relation geändert haben	Neukonvertierung der Relation
tags changed	tritt auf, wenn sich die Tags eines OSM Elements geändert haben	Neukonvertierung der betroffenen Elemente und aller mit ihnen verbundenen Elemente

way nodes changed	tritt auf, wenn sich die Anzahl oder Reihenfolge der Knoten eines Ways geändert hat	Neukonvertierung des betroffenen Ways und der mit den geänderten Knoten verbundenen Ways
other dataset change	tritt bei allen übrigen Änderungen auf	keine Reaktion

Es sei angemerkt, dass die beschriebene Neukonvertierung und Entfernung der Elemente nur dann auftritt, wenn konvertierungsrelevante Tags vorhanden sind. Im Falle einer Neukonvertierung werden zuvor die entsprechenden MATSim Informationen aus dem Network / der TSchedule und den Referenzmaps gelöscht. Somit ist sichergestellt, dass jegliche MATSim Entsprechungen der OSM Element entfernt werden, falls sie nach den Änderungen keine MATSim Daten mehr darstellen.

Die Echtzeitkonvertierung sorgt also für ein ständig aktuelles Abbild der MATSim Daten aus den visualisierten OSM Daten. Neben der Reaktion auf Änderungen gehört dazu auch, dass OSM Layer, die keine MATSim Informationen enthalten, automatisch konvertiert werden, sobald sie als aktuell aktives Layer gesetzt werden. Hierfür wird auf das von JOSM geworfene ‚*Layer Change*' - *Event* reagiert. Der Network Listener existiert einzeln für jedes MATSim Layer und wird für OSM Layer bei jedem Layer Wechsel nach der Konvertierung neu initialisiert.

Die in diesem Abschnitt beschrieben Funktionen ermöglichen ein interaktives Bearbeiten der Netzdaten für MATSim. Das Editieren erfolgt hauptsächlich durch zwei von JOSM bereitgestellten Funktionen: dem Zeichnen der geographischen Verläufe und Abhängigkeiten (Knoten und Kanten) und dem Hinzufügen relevanter Informationen durch Tags (Gewichtung der Kanten). Hervorzuheben ist hierbei, dass nicht nur explizite MATSim Informationen bearbeitet werden können, sondern auch die zugrundeliegenden OSM Daten. Die Netzbearbeitung kann zum Großteil durch in OSM etablierte Tags erfolgen.

3.3. Visualisierung

Da neben den OSM Daten nun zusätzlich MATSim Daten zur Laufzeit von JOSM vorhanden sind, wurden Möglichkeiten zum Anzeigen dieser Informationen implementiert. Zum einen wurde ein eigener Dialog erstellt, der sich in die Dialogleiste am Bildschirmrand einpasst. Zum anderen wurde die visuelle Darstellung auf der Karte angepasst.

3.3.1. Toggle Dialog

Der MATSim Toggle Dialog wird neben anderen JOSM-eigenen Dialogen am Bildschirmrand angezeigt, sobald die Kartenansicht geöffnet wurde. Er gibt die zugrundeliegenden Network (bzw. TSchedule) Daten wieder und ist somit ein Bindeglied zwischen Benutzer und den zu bearbeitenden Daten. In ihm kann der Nutzer jederzeit Information zu bestimmten Elementen aufrufen. Dabei basiert das Prinzip auf der Verhaltensweise der JOSM-eigenen Dialoge. In diesen werden immer nur Informationen zu den momentan selektierten Elementen auf der Karte angezeigt. Ist die Selektion leer, sind auch die meisten Dialoge inhaltlos. Demnach werden im MATSim Toggle Dialog auch nur Daten zu selektierten Elementen angezeigt, wenn sie ein MATSim-Äquivalent verkörpern. Der Dialog registriert sich dafür für *‚selection changed‘ - Events*, die von JOSM geworfen werden, sobald sich die selektierten Elemente auf der Karte ändern. Wird ein solches Event empfangen, wird in den Referenzmaps des aktuellen Layers nach der MATSim Entsprechung des selektierten OSM Elements gesucht. Ist eine solche vorhanden, werden die Informationen angezeigt. Das setzt voraus, dass der Dialog die aktuellen Referenzmaps kennt. Daher werden diese dem Dialog bei jedem Wechsel des Layers mitgeteilt, indem sie einem MATSim Layer entnommen werden oder neu aus einem OSM Layer konvertiert werden. Wichtig ist hierbei, dass die hier angezeigten Werte nicht in diesem Dialog bearbeitet werden können. Die Bearbeitung erfolgt ausschließlich über die von JOSM mitgelieferten Funktionen. Die einzelnen Links der selektierten Ways können jedoch wiederum innerhalb des Dialogs ausgewählt werden. Daraufhin werden die jeweiligen Segmente, die die Strecke des Links verkörpern, hervorgehoben und die Ansicht springt zu diesem Bereich der Karte. Dies geschieht mithilfe der Link2Segments Referenzmap.

Der Dialog ist auf Abbildung 12 dargestellt. Er zeigt vier Links an, die durch einen selektierten Way dargestellt werden. Es ist gut zu sehen, wie die IDs vergeben werden. Die

ungerichtete Kantenfolge des Ways kann durch vier gerichtete MATSim Kanten dargestellt werden. Dabei sind je zwei Links für die beiden entgegengesetzten Richtungen vorhanden.

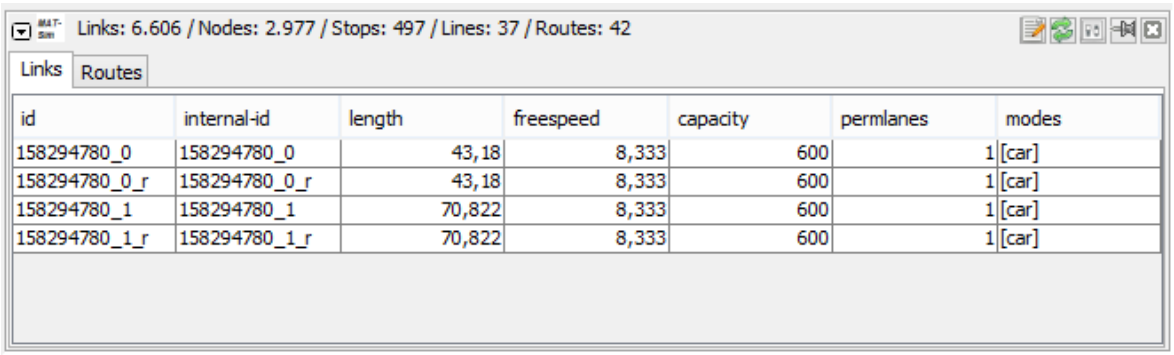

Abbildung 12: MATSim Toggle Dialog
Quelle: eigene Darstellung, entnommen aus JOSM

Der Dialog besteht aus einer Titelzeile und zwei Registerkarten. Eine der Registerkarten gibt die referenzierten Links des Networks wieder. Die andere zeigt Inhalte bezüglich den TRoutes der TSchedule. Die Informationen zu den Links werden mithilfe einer modifizierten JTable dargestellt. Sie enthalten die folgenden Attribute, die im Wesentlichen den Angaben aus dem MATSim Tagging Schema entsprechen:

- *id*
- *internal-id*
- *length*
- *freespeed*
- *capacity*
- *permlanes*
- *modes*

Die *id* entspricht hierbei der OriginalId der Links, also der ID, die auch durch das Setzen eines ‚*id*' Tags manipuliert werden kann und die für den Nutzer interessantere ID. Die *internal-id* entspricht der eigentlichen ID der Links zur Laufzeit und damit zeitgleich der OSM ID des Ways, der diesen Link repräsentiert. Die Angabe dieser Information ist eher für programmiertechnische und Debugging-Funktionen eingebaut und ist im Regelfall für den Nutzer bedeutungslos.

Angaben zu den TRoutes werden angezeigt, wenn ein selektiertes Element einer konvertierten Relation angehört, die in Relation2Route vermerkt ist. Die folgenden Informationen werden dargestellt:

- *route id*
- *mode*
- *#stops*
- *#links*

Die *route id* gibt die ID der TRoute wieder. Der Verkehrsmodus wird in *mode* angegeben. Die Anzahl der StopFacilities sowie der Links der Route werden in den Spalten *#stops* bzw. *#links* vermerkt.

In der Titelzeile gibt es außerdem, neben den Buttons zum Schließen und Loslösen des Dialogs vom Hauptfenster, noch drei weitere Buttons. Der Button mit dem schreibenden Stift auf dem Dokument öffnet ein kleines Fenster, in dem die beiden globalen Attribute des Networks durch Eingabe von numerischen Werten bearbeitet werden können: *effective lane width* und *capacity period*. Der Button mit den zwei grünen Pfeilen aktualisiert die kompletten MATSim Daten, indem es das komplette Layer neu konvertiert. Der dritte Button öffnet schließlich die Pluginspezifischen Einstellungen (siehe auch Abschnit 3.8).

3.3.2. MapRenderer

Neben dem Toggle Dialog wurde auch die Kartendarstellung der OSM Elemente verändert. Dazu wurde ein eigener Map Renderer geschrieben, der den bestehenden Renderer erweitert, indem er alle Ways, die MATSim Informationen darstellen, hervorhebt. Ist ein Way in der aktuellen Way2Links Referenzmap vorhanden, wird er in einem MATSim-typischen Blauton gezeichnet. Damit kann der Nutzer leicht erkennen, welche Ways zur Visualisierung von Links verwendet werden und Informationen im Toggle Dialog liefern. Zusätzlich zu der farblichen Hervorhebung kann der Renderer alle IDs der Links, die durch einen Way dargestellt werden, nacheinander über den jeweiligen Way zeichnen.

Eine weitere Funktion, die vor allem die Bearbeitung und Übersichtlichkeit erleichtert, ist das automatische Versehen zweier überlappender Links mit einem Offset, sodass diese nebeneinander statt übereinander gezeichnet werden. Das ist vor allem für die Anwendung von MATSim von Vorteil, da beim Nutzen der MATSim Tags Ways als gerichtete Kanten nur in eine Richtung konvertiert werden. Für zwei parallele, entgegengesetzte Kanten müssen demnach Ways übereinander gezeichnet werden. Wie sonst auch die mit ‚oneway' getaggten Highways werden die mit MATSim Tag Schema versehenen Ways mit

Richtungspfeilen versehen. Schließlich stellt der Renderer alle Ways, deren Links durch ein *mode=pt* Tag als ÖV-Strecken definiert sind, durch eine zweifarbige gestreifte Linie dar.

Abbildung 13 zeigt einen Kartenausschnitt auf dem die beschriebenen visuellen Funktionen des Renderers sichtbar sind.

Abbildung 13: Ways mit MATSim Informationen, vom neuen Renderer dargestellt
Quelle: eigene Darstellung, entnommen aus JOSM

Die hier vorgestellten Visualisierungsfunktionen dienen dazu, dem Benutzer die aktuell vorhandenen MATSim Informationen anzuzeigen. Sie verdeutlichen ihm die aktuelle Beschaffenheit des Networks bzw. der Transit Schedule.

3.4. Validierung

Eine weiterer Punkt, der in Abschnitt 2.6 angesprochen wurde, ist die Unterstützung bei Fehlern durch automatische Testverfahren. Wie in Kapitel 0 beschrieben, bietet JOSM eigene Validatoren an, die das Kartenmaterial prüfen. Im Rahmen des Plugins wurden eigene Tests in Form eines Validators geschrieben. Bisher gibt es zwei implementierte Tests.

Der erste Test überprüft die IDs der $Nodes_{MATSim}$ und Links. Dieser Test ist notwendig, da eine doppelte ID Vergabe beim Setzen der Tags nicht verboten ist und auch nicht zu einer Fehlermeldung führt, da die vom Nutzer getaggte ID zur Laufzeit des Programms als OriginalId der Links gespeichert wird und somit mehrfach vorhanden sein darf. Der Test

vergleicht nun jeweils die OriginalIds zwischen den $Nodes_{MATSim}$ und zwischen den Links. Ist eine ID doppelt vorhanden, wird dies als Fehler registriert.

Der zweite Test überprüft die Links des aktuellen Networks auf unglaubwürdige Attribute. Das bedeutet, dass er die Kapazität, Länge, freie Geschwindigkeit sowie Spuranzahl auf Nullwerte untersucht. Ist eines dieser Attribute auf Null gesetzt, resultiert der Test in einer Warnung.

Nach Ablauf des Tests können die Ergebnisse in einem Error Layer angezeigt werden. In diesem werden die Fehler und Warnungen angezeigt. Ein Dialog hilft außerdem bei der Fehlerbehebung, in dem er alle Elemente auflistet und die jeweiligen Fehler und Warnungen auflistet. Sollte eine mehrfache Vergabe derselben ID aufgetreten sein, werden alle Elemente, die dieselbe ID besitzen, im Error Layer angezeigt und hervorgehoben. Für diesen Fehler wurde eine automatische Reparaturmöglichkeit eingebaut. Wird diese aktiviert, werden die OriginalIds der Elemente mit einer in Klammern geführten aufsteigenden Zahl erweitert (zum Beispiel erhalten drei Elemente mit der ID ‚8' dann die IDs ‚8(1)', ‚8(2)' sowie ‚8(3)'). Auch die Ways, deren Links Nullwerte in den Attributen vorweisen, werden im Error Layer angezeigt. Für diesen Fehler gibt es jedoch keine automatische Reparatur.

Die bisherigen Tests liefern vor allem Ergebnisse zu Fehlern bei den Qualitätsangaben. Fehler in den Verbindungen des Netzes, also den geometrischen Gegebenheiten, werden bisher noch nicht durch eine geeignete Testlogik geprüft.

3.5. Speichern von Netzwerken und ÖV-Routen

Um die Verkehrsnetze in der Simulation benutzen zu können, werden sie in der Regel als Network sowie TSchedule Dateien benötigt. Daher muss es eine Funktion geben, die das bearbeitete MATSim Netz inklusive TSchedule in Dateien exportiert.

Damit die Dateien bei der Verwendung des Plugins herausgeschrieben werden können, muss das aktuelle Layer vom Typ MATSim Layer sein. Ein OSM Layer kann in ein MATSim Layer überführt werden, indem es explizit über die Funktion ‚Convert to MATSim Layer' in der Werkzeug-Menüleiste konvertiert wird. Dabei werden die bereits in Echtzeit konvertierten Network und TSchedule in ein neues MATSim Layer übernommen. Die $Nodes_{MATSim}$, Links werden dabei als OSM Elemente im MATSim Tagging Schema erstellt. Das heißt, dass für jeden einzelnen Link ein Way erstellt wird, der die Tags *capacity*,

freespeed, etc. erhält. Die *TRoutes* werden ebenfalls ausschließlich im MATSim Schema dargestellt.

Die Ausgabe der Dateien erfolgt nun über den konventionellen Speicherbefehl. Nachdem ein Dateipfad angegeben wurde, startet der im Abschnitt 3.4 vorgestellte Validator. Treten Fehler auf, wird der Vorgang abgebrochen. Bei Warnungen kann der Nutzer entscheiden, ob der Vorgang fortgesetzt werden soll. Fehler treten bisher nur bei mehrfacher Verwendung derselben ID auf, was auf alle Fälle verhindert werden muss. Nach der Validation werden die Objekte im Network in ein neues, leeres Network kopiert. Dabei werden alle Attribute übernommen, bis auf die ID. In diesem Schritt wird schlussendlich die eigentliche ID, die zur Laufzeit als OriginalId gespeichert wurde, als ‚richtige' ID der neu erstellten $Nodes_{MATSim}$ und Links verwendet. Die bisher verwendete OSM ID wird hingegen als neue OriginalId verwendet. Die beiden Datenfelder werden somit ausgetauscht. Danach kann der von MATSim bereitgestellte *NetworkCleaner* auf das Netzwerk angewendet werden, der lose Links und tote Enden aussortiert. Letztendlich wird ein *NetworkWriter* sowie ein *TransitScheduleWriter* des MATSim Codes genutzt, um Netzwerk sowie TSchedule in Dateien zu schreiben. Dafür wird zuerst ein neuer Ordner unter dem angegebenen Dateipfad erstellt. Dieser enthält dann jeweils eine Network sowie eine TSchedule Datei.

Zu beachten ist, dass MATSim Layer nicht in OSM Layer zurück konvertiert werden können. Somit tritt immer ein Datenverlust auf. Es bietet sich daher an, die ursprüngliche OSM Datei nach jeder Bearbeitung ebenfalls zu speichern und nur dann eine MATSim Formate als Momentaufnahme herauszuschreiben, wenn sie für eine Simulation benötigt werden. Derzeit können die Netzwerke nur in *WGS84* Koordinaten ausgegeben werden.

3.6. Importieren von Netzwerken

Bestehende Netzwerke können durch Importieren einer Network Datei eingelesen werden. Dafür muss die Funktion ‚Import MATSim Network' im Werkzeugmenü ausgewählt werden. Um das Importieren zu veranschaulichen, wurde das in Abbildung 14 abgebildete Flussdiagramm erstellt.

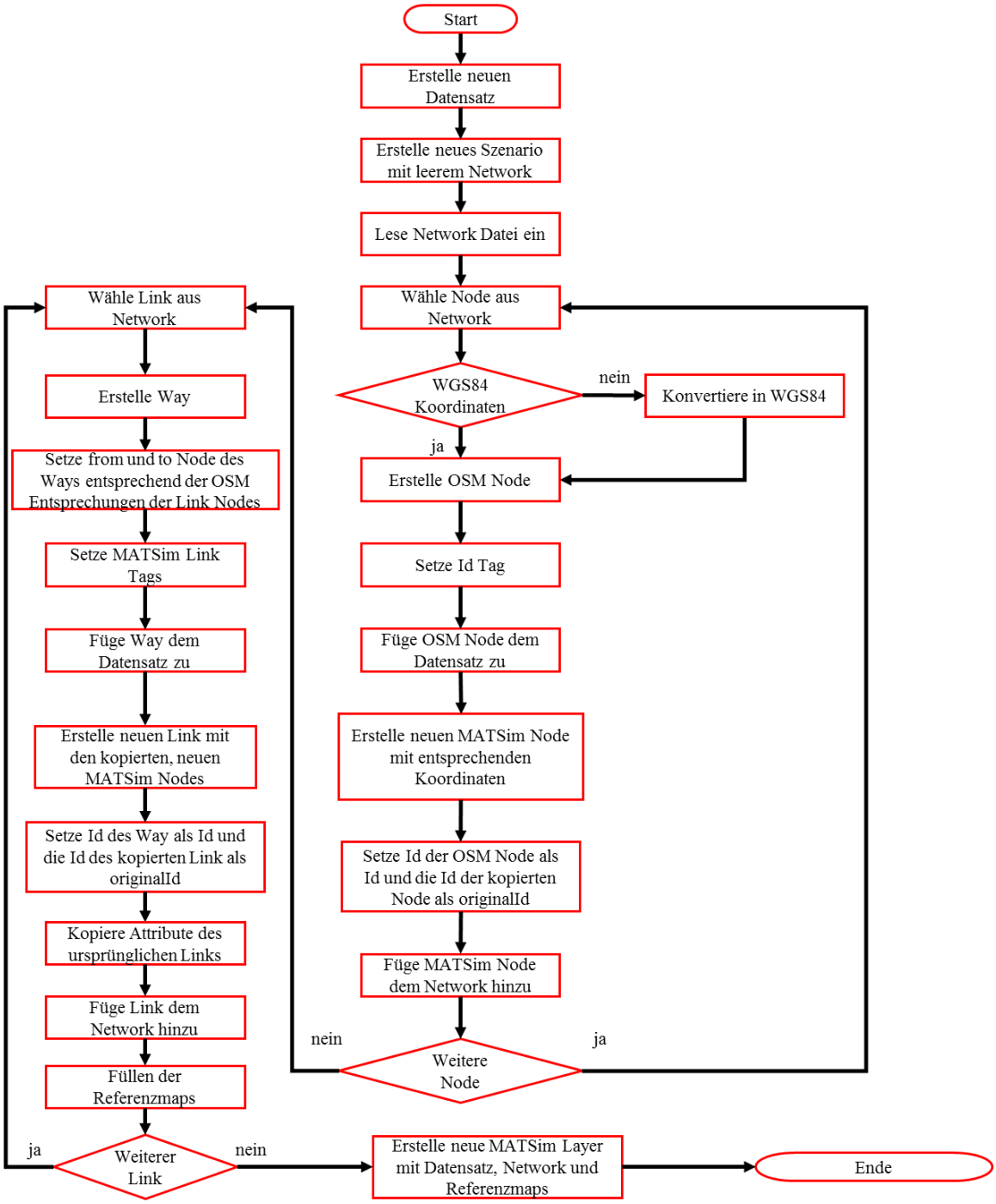

Abbildung 14: Flussdiagramm der Importfunktion
Quelle: eigene Darstellung

Zuerst werden ein neuer Datensatz und ein neues MATSim Szenario mit leerem Network erstellt. Die zu importierende Network Datei wird anschließend mithilfe des MATSim Network Readers eingelesen. Die Knoten eines Netzes bilden die Grundlage der Kanten, weshalb die $Nodes_{MATSim}$ zuerst verarbeitet werden. Für jeden dieser $Nodes_{MATSim}$ wird ein $Node_{OSM}$ an den jeweiligen Koordinaten (in *WGS84*) erstellt, der die ID der eingelesenen $Node_{MATSim}$ als Tag erhält. Daraufhin wird das OSM Element dem Datensatz hinzugefügt.

Dem leeren Network wird eine neu erstellte Kopie des $Node_{MATSim}$ hinzugefügt, mit den *WGS84* Koordinaten des $Nodes_{OSM}$. Die ID der Kopie entspricht der OSM ID des referenzierenden Knotens. Gleichzeitig wird die eigentliche ID des kopierten $Node_{MATSim}$ als OriginalId gesetzt. Sobald alle Knoten verarbeitet wurden, werden die Links aufbereitet.

Für jeden Link wird genau ein Way erstellt. Der Way wird von den OSM Äquivalenten der Start- und Endknoten des Links gezogen. Danach erhält der Way alle Tags nach dem MATSim Tagging Schema, die die Attribute des darzustellenden Links wiedergeben. Der Way wird dann dem Datensatz hinzugefügt. Nun wird auch der Link kopiert, wobei auch hier die OSM ID des entsprechenden Ways verwendet wird, während die ursprüngliche ID als OriginalId gesetzt wird. Der Link verbindet die beiden neu erstellten Kopien der ursprünglichen $Nodes_{MATSim}$ und erhält ansonsten alle Attribute des ursprünglichen Links. Die Referenzmaps Way2Links und Link2Segments werden für jede Link/Way Kombination gefüllt.

Nachdem alle MATSim-Elemente eingelesen und verarbeitet wurden, wird ein neuer MATSim Layer erstellt, in dem der OSM Datensatz, das Network sowie die Referenzmaps gespeichert wird. Mit dem erstellten Layer kann nun gearbeitet werden. Das Kopieren der ursprünglichen MATSim Elemente ist notwendig, um die IDs mit den neu erstellten OSM Elementen in Einklang zu bringen.

Transit Schedules können derzeit noch nicht importiert werden.

3.7. Leeres Netzwerk erstellen

Das Plugin bietet neben dem Konvertieren bestehender OSM Daten und dem Bearbeiten importierter Netzwerke auch die Möglichkeit, Netzwerke komplett neu zu erstellen. Dafür wird die Funktion ‚New MATSim Network' in der Werkzeugmenüleiste genutzt. Dies führt dazu, dass ein neues MATSim Layer erstellt wird, dessen Network sowie OSM Datensatz leer sind. Der Nutzer kann dann ein eigenes Netzwerk von Grund auf neu zeichnen.

3.8. Preferences / Einstellungsoptionen

Die bereits erwähnten pluginspezifischen Einstellungsoptionen bieten die Möglichkeit, die Visualisierung sowie die Konvertierung der Daten anzupassen. Sie befinden sich im ‚Einstellungen'-Dialog. Dort gibt es zwei Registerkarten.

Die erste Registerkarte ‚Visualization' passt die Funktionen des Map Renderers und des Toggle Dialogs an. Der Renderer kann ausgeschaltet werden, sodass keine MATSim spezifischen Darstellungen angezeigt werden. Außerdem kann das Darstellen der Link IDs über den Ways deaktiviert werden. Dies ist vor allem in komplexen Netzwerken von Vorteil, da es die Performanz erhöht. Der Offset zwischen den Ways kann mittels eines Reglers vergrößert oder verkleinert werden. Die eher für Debugging gedachte *internal-id* Spalte im Toggle Dialog kann je nach Bedarf an- oder ausgeschaltet werden.

In der zweiten Registerkarte ‚Converter Options' befinden sich Optionen und Parameter die sich auf die Konvertierung und die Datenhaltung auswirken. Zum einen kann der *Network Cleaner*, der vor dem Rausschreiben eines Networks angewendet wird, an- oder ausgeschaltet werden. Zum anderen lässt sich die ‚*keep paths*' Option (de-)aktivieren, die dafür sorgt, dass die Knoten beim Konvertieren der Ways nicht gefiltert sondern geometrisch exakt behalten werden. Über den Button ‚Converting Defaults' gelangt der Nutzer zu einem neuen Fenster, in dem er die Standardwerte der konvertierten Highway Typen anpassen und zurücksetzen kann. Das Fenster ist auf Abbildung 15 dargestellt.

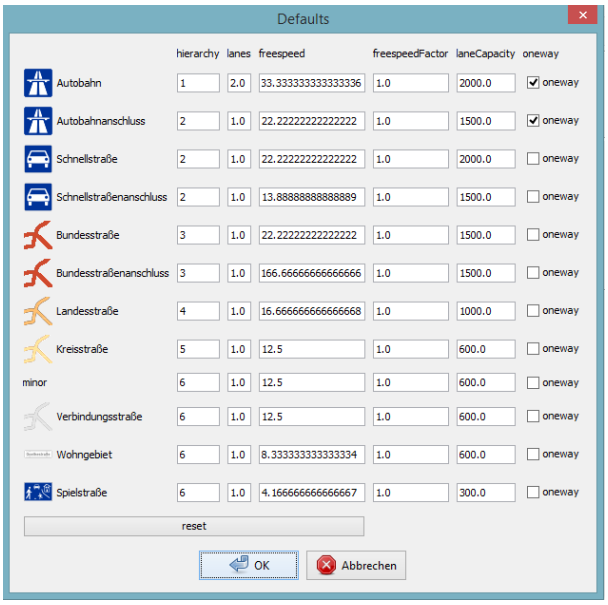

Abbildung 15: Anpassen der Converting Defaults
Quelle: eigene Darstellung, entnommen aus JOSM

Die Änderungen in den Visualisierungs- und Konvertierungsoptionen wirken sich größtenteils sofort auf die Daten bzw. deren Darstellung aus. Die Karte wird neu gezeichnet und die Elemente des aktuellen Layer werden nach den neuen Parametern konvertiert. Das wird ermöglicht, indem an den entsprechenden Stellen im Programm auf Änderungen in den Preferences geachtet wird, indem die von JOSM geworfenen *‚Preference changed‘ - Events* ausgewertet werden.

Damit die Einstellungen auch bei einem Neustart erhalten bleiben, werden sie in automatisch in der von JOSM verwalteten Preferences-Datei gespeichert.

4. Anwendung Beispielszenario

Um die Funktionen testen und evaluieren zu können, soll im Rahmen dieser Arbeit ein Verkehrsnetz der Stadt Ulm in Baden-Württemberg erstellt und bearbeitet werden. Grundlage dafür sind OSM Daten, die zuvor aufbereitet werden. Da die Anzahl der herunterladbaren Elemente innerhalb von JOSM seitens des Servers beschränkt ist, wird zunächst ein vorgefertigter Datensatz Baden Württembergs von der Webseite *GeoFabrik* heruntergeladen. Für das Programm ist der Umfang eines kompletten Bundeslandes zu umfangreich, um eine reibungslose Nutzung zu ermöglichen. Es soll lediglich das ungefähre Gebiet der Stadt Ulm konvertiert werden. Das Filtern des Datensatzes erfolgt mit dem Programm *Osmosis*. Es wurde eine rechteckige Fläche um das Stadtgebiet herausgefiltert (rötlich markiertes Gebiet auf Abbildung 16).

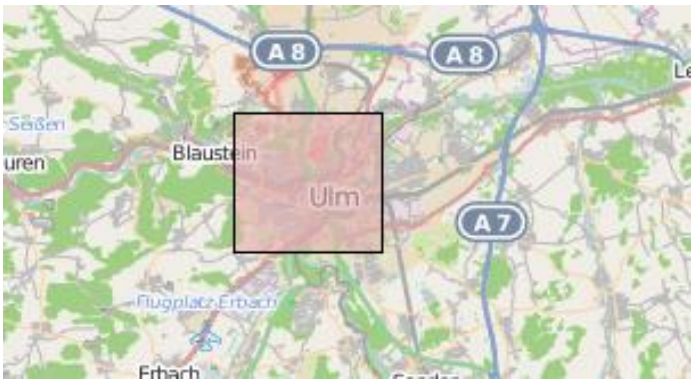

Abbildung 16: Verwendetes Stadtgebiet Ulms
Quelle: eigene Darstellung, entnommen aus JOSM

Um eine übersichtlichere Darstellung zu erreichen, wurden außerdem nur für die Konvertierung relevante Daten herausgefiltert. Es wurden Highways sowie in ÖV-Routen verwendete Railways übernommen. Die OSM Datei wird anschließend in JOSM geöffnet. Die Konvertierung erfolgt umgehend nach dem Öffnen der Ebene. Insgesamt wurden 2.775 $Nodes_{MATSim}$, 6.023 Links, 494 StopFacilities, 41 Routen und 36 Linien für das gewählte Gebiet erstellt (ohne *keepPaths*-Option). Die hohe Anzahl an StopFacilities ergibt sich aus der Tatsache, dass die Haltestellen für jede Route dupliziert wurden. Alle Straßen, die erfolgreich konvertiert wurden, werden in blauer Farbe gezeichnet und mit den IDs der Links versehen (siehe Abbildung 17). Da alle gefilterten Highways blau dargestellt werden, wurden sie erfolgreich konvertiert. Stichproben beim Untersuchen einzelner Ways mittels des Toggle Dialogs ergeben, dass die Werte sowie Richtungen der Links nach dem beschriebenen Mustern erzeugt wurden. Das Erstellen von Straßennetzmodellen mithilfe von OSM Daten ist demnach für die verwendeten Highway Typen funktionsfähig.

Alle ÖV-Systeme wie Bus, Bahn und Straßenbahn, wurden mit gleichen Parametern konvertiert. Die Bahnstrecken des Bahnhofs gehören zu längeren Linien (wie zum Beispiel „ICE 11 Berlin – Frankfurt – München“). Sie wurden jedoch an den Grenzen des Kartenausschnitts abgeschnitten. Auch andere ÖV-Routen, die über diese Grenzen hinweggehen, wurden verkürzt und sind unvollständig. Die unvollständigen Relationen wurden daher auch nur mit den in diesem Ausschnitt vorhandenen Halteposition konvertiert. Die TRoutes sind demnach ebenfalls zum Teil unvollständig.

Abbildung 17: Ulm mit OSM Tagging Schema
Quelle: eigene Darstellung, entnommen aus JOSM

In der Tabelle in Anhang E sind die Linien des ÖV in Ulm aufgelistet. Sie enthält die regulären Bus und Straßenbahnlinien des Stadtverkehrs sowie der des Verkehrs von und nach Ulm (vgl. Donau-Iller-Nahverkehrsbund GmbH, 2014). In der Tabelle wurde auch eingetragen, ob und mit wie vielen Routen die jeweilige Linie im OSM Datensatz vorhanden ist. Es lässt sich sagen, dass ein großer Teil der Linien (31 von 39), insbesondere der des innerstädtischen Verkehrs (12 von 13), vorhanden ist. Allerdings ist keine dieser Linien vollständig eingepflegt. Fast alle Linien werden durch lediglich eine Routenrelation dargestellt. In diesen Routen sind oftmals alle Haltestellen doppelt vorhanden, womit wahrscheinlich Hin- und Rückrichtung in nur einer Relation definiert werden sollen. Das entspricht nicht den in Kapitel 2.2.3 vorgestellten Richtlinien. Außerdem sind auch die Anzahl und Reihenfolge der Haltestellen in vielen Routen falsch, bzw. unvollständig. Die Streckenverläufe sind oftmals keine zusammenhängenden Kantenfolgen, was für die Minimallösung der Verbindung der Haltestellen per Luftlinie spricht. Abbildung 18 zeigt

beispielhaft die Relationsliste für die einzige Route der Linie 21. Zu sehen ist, dass es lediglich drei Haltestellen gibt, wovon eine doppelt vorhanden ist. Die Ways der Strecke sind an vielen Teilen nicht miteinander verbunden.

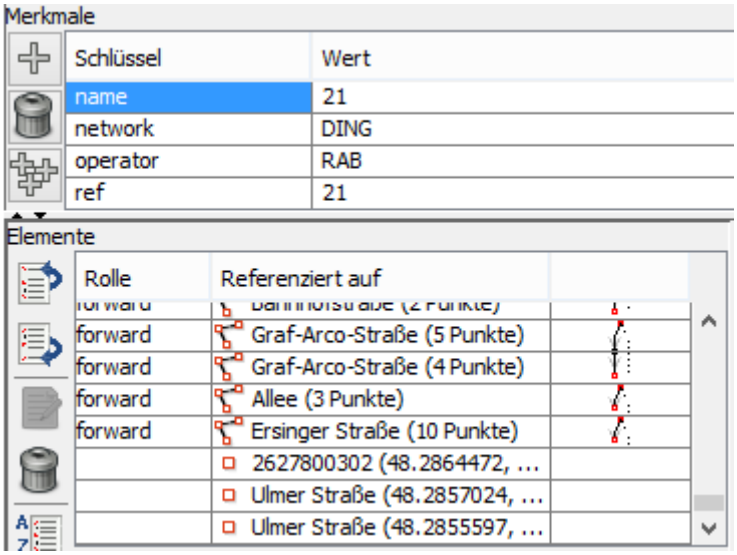

Abbildung 18: Auszug aus der Relationsliste der Linie 21 in Ulm
Quelle: eigene Darstellung, entnommen aus JOSM

Die Konvertierung mit der bisher implementierten Methode liefert für das ÖV-Netzmodell Ulms keine vollständigen Daten. Das liegt vor allem an den nicht richtlinienkonformen Routenrelationen im Stadtgebiet. Die mit dem Plugin erzeugten Daten können als Grundlage für ein realistisches ÖV-Netzmodell dienen. Dazu müssen jedoch Dopplungen von Haltestellen und exakte Routenverläufe manuell nachgebessert werden. In OSM nicht vorhandene Routen müssen komplett eigenhändig eingebaut werden. Dafür kann das Plugin mit dem vorgestellten MATSim-Tagging Schema verwendet werden. Das Ausbessern der ÖV-Routen durch nachträgliches Bearbeiten mithilfe des Plugins wird im Rahmen dieses Anwendungsbeispiels nicht erläutert.

Nun sollen Network und TSchedule des erstellten Netzes herausgeschrieben werden. Dazu wird die OSM-Datenebene in ein MATSim Layer umgewandelt. Durch die damit verbundene Darstellung der Daten im MATSim Tagging Schema werden die einzelnen Links durch eigene Ways für jede Richtung dargestellt. Außerdem werden die Luftlinienrouten der ÖV-Konvertierung sichtbar. Gerade im Stadtzentrum kann diese Ansicht aufgrund der Luftlinienverbindungen und der unvollständigen OSM Grundlagen unübersichtlich und verwirrend sein. Die Visualisierung Ulms mit dem konvertierten MATSim Tag Schema ist in Abbildung 19 dargestellt. Zum Exportieren der Dateien wird

nun der Speicherdialog verwendet. TSchedule sowie Network sind damit für die Simulation nutzbar.

Im Rahmen dieses Beispiels sei angedacht, eine neue Brücke über die durch Ulm verlaufende Donau zu erbauen. Die Auswirkungen sollen mit einer Simulation durch MATSim evaluiert werden können. Dafür muss das Netzwerk neben dem bestehenden Netzwerk um diese Maßnahme erweitert werden, damit eine Vergleichssimulation zwischen Null- und Planfall durchgeführt werden könnte.

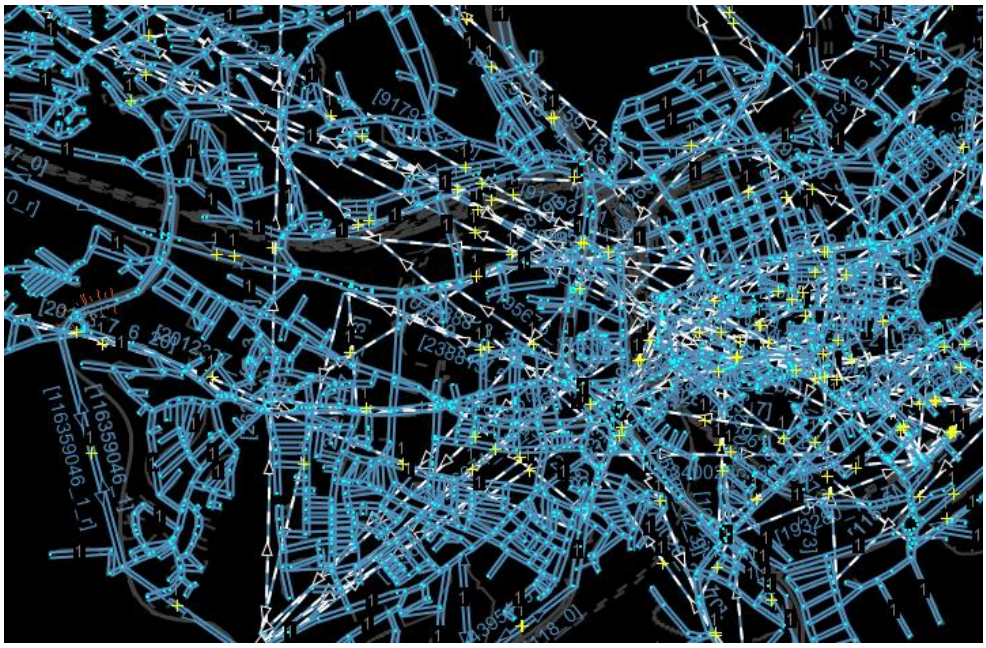

Abbildung 19: Ulm mit MATSim Tag Schema
Quelle: eigene Darstellung, entnommen aus JOSM

Die neue Brücke wird in der OSM Datenebene zwischen zwei bestehenden Ways am Ufer gezeichnet und mit einem Tag ‚*highway=tertiary*' versehen. Sie resultiert dann in je einem Link für jede Querungsrichtung. Am südlichen Ende ist die Brücke mit einem ‚*highway=secondary*' verbunden. Die Straße am nördlichen Ende wird durch einen ‚*highway=residential*' dargestellt. Da durch den Neubau der Brücke angenommen sei, dass sich das Verkehrsaufkommen an dieser Stelle erhöhe, soll die nördliche Anbindung ausgebaut werden. Dazu werden die beiden Ways am nördlichen Ende mit dem Tag ‚*highway=tertiary*' versehen. Nun kann die Ebene wieder in eine MATSim-Ebene überführt werden und eine Network- sowie TSchedule Datei herausgeschrieben werden. Die Lage der Brücke ist in Abbildung 20 verdeutlicht.

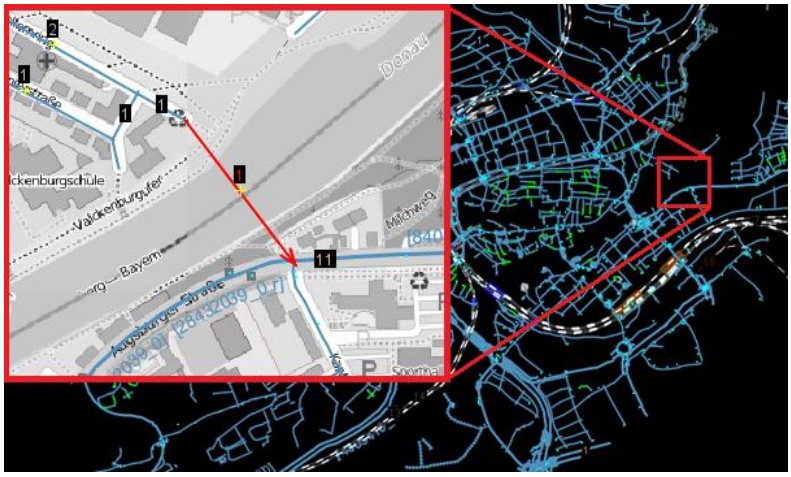

Abbildung 20: Lage der neuen Brücke
Quelle: eigene Darstellung, entnommen aus JOSM

Wie in dem vergrößerten Abschnitt auf Abbildung 20 zu sehen, ist es auch möglich, fertig gerenderte Karten oder auch Satellitenbilder als Hintergrund einzuschalten. Die abstrakten Kanten und Knoten können so leichter mit der Umgebung verglichen werden.

In dem beschriebenen Fall sollen nun auch die weiteren Einflüsse der wichtigen Verkehrsverbindungen Baden-Württembergs berücksichtigt werden. Die vorgenommenen Änderungen sollen jedoch erhalten bleiben. Der große Vorteil beim Nutzen des Plugins ist, dass die reinen OSM-Rohdaten zum Visualisieren und Bearbeiten von MATSim-Netzen genutzt werden können. Dadurch können alle Informationen auch immer ohne Datenverlust als *.osm*-Datei gespeichert werden. Der zuvor bearbeitete Ulm-Datensatz wird demnach als *.osm*-Datei gespeichert. Anschließend werden die Autobahnen, Kraftfahrstraßen und Bundesstraßen sowie alle Eisenbahnrouten mit der Hilfe von *Osmosis* aus dem Baden-Württemberg Datensatz herausgefiltert und in einem neuen OSM Datensatz gespeichert. Daraufhin wird auch dieser in JOSM als neue Ebene geöffnet. Nun kann mittels des Befehls ‚Ebene vereinigen' unter ‚Bearbeiten' in der Menüleiste ein Verschmelzen beider Ebenen bewirkt werden. Dadurch werden beide Ebenen zu einer neuen Ebene zusammengeführt, die die Elemente beider Ursprungsebenen enthält. Bei Elementen, die in beiden Ebenen existieren, wird die jeweils aktuellere Version genommen. Es kann außerdem auch nur die selektierte Auswahl mit dem entsprechenden Befehl in dem Menü vereint werden. Das Vereinigen ist nicht auf MATSim Layer anwendbar.

Die nun erstellte zusammengeführte Ebene enthält ein detailliertes Netz der Stadt Ulm sowie alle größeren Verbindungen innerhalb Baden-Württembergs. Nach dem Öffnen und

automatischen Konvertieren der neuen Ebene enthält das zugehörige Netzwerk 38.250 Knoten und 61.030 Links. Dieses Größenverhältnis entspricht in etwa dem des in Kapitel 2.6 erwähnten Planungsnetzes der Schweiz und ist in JOSM gut darstellbar. Lediglich in hohen Zoomstufen lies die Performanz auf dem verwendeten Computer nach. Die TSchedule enthält 141 TLines, die 251 TRoutes bedienen, die wiederum 3.056 StopFacilities verbinden. Viele der zuvor unvollständigen ÖV-Routen sind nun vollständig(er) als zuvor. Die verwendeten Änderungen, also die eingefügte Brücke sowie die Änderung der Tags der angebundenen Straßen wurden übernommen.

Auf demselben Weg können somit auch Datensätze über längere Zeit mit neuen Updates seitens der OSM-Community bei gleichzeitiger Beibehaltung eigener Änderungen gepflegt werden. Wichtig ist hierbei, dass der Datensatz als *.osm*-Datei beibehalten wird und nicht nur die herausgeschriebenen MATSim Dateien.

5. Zusammenfassung

Die für die Verkehrssimulation notwendige Infrastruktur kann durch Netzmodelle abstrahiert werden. Diese können durch Graphen mit Knoten und Kanten visualisiert werden. Neben den geometrischen Gegebenheiten können den Kanten und Knoten auch für die Simulation notwendige Charakteristika zugeordnet werden.

Aufgrund der Komplexität der Daten entsteht das Bedürfnis nach einer benutzerfreundlichen Anwendung, die das modellieren der Netze visuell unterstützt. Dabei entstehen mehrere Ansprüche, die ein optimales Arbeiten ermöglichen sollen. Dazu zählen vor allem die Interaktivität und Transparenz beim Editieren sowie eine simple Darstellung der Daten.

In dieser Arbeit wurde ein Programm beschrieben, das entwickelt wurde, um ein graphisches Darstellen und Bearbeiten der Netzmodelle für MATSim zu ermöglichen. Dafür nutzt es das OSM Datenmodell um die einzelnen Elemente innerhalb von JOSM darzustellen. Kern der Anwendung ist ein Konvertierungsalgorithmus, der OSM und MATSim Datenmodelle konsistent hält. Durch die Nutzung des Hauptprogramms JOSM stehen dem Plugin viele Editier- sowie Visualisierungsoptionen zur Verfügung.

Die entwickelten Funktionen wurden anhand eines Beispielszenarios in der Stadt Ulm getestet. Die Konvertierung und Bearbeitung von Straßennetzmodellen funktionierte dabei auf die erhoffte Weise. ÖV-Netzmodelle können zwar auf eine bisher simple Form bearbeitet und konvertiert werden. Die Datengrundlage der OSM war dafür jedoch unzureichend für ein zufriedenstellendes Ergebnis. Das Herausschreiben von MATSim Dateien erfolgte wie erhofft. Auch die Visualisierungsfunktionen sowie das Vereinen mehrerer OSM Datensätze zur Fortschreibung von Netzwerken bereiteten keine Probleme.

Das Programm ist vor allem in Hinsicht auf den ÖV noch mit Problemen konfrontiert. Diese entspringen vor allem dem offenen OSM Datenformat. Es gibt viele Sonder- und Einzelfälle und keine verbindlichen Richtlinien. Die Bearbeitung und Konvertierung von ÖV Routen ist noch nicht ausgereift genug für eine einfache aber vollständige Erstellung von MATSim Daten. Bisher werden noch keine Verkehrssysteme wie Bus, Bahn oder U-Bahn unterschieden. Außerdem werden alle OSM Routen unabhängig davon, ob sie mustergültig nach den Richtlinien eingepflegt sind, nicht mit dem realen Streckenverlauf konvertiert. Im Gegensatz zu den einstellbaren Konvertierungsparametern für Highways oder Tagging

Presets für Links, gibt es solche noch nicht für ÖV-Relationen bzw. TRoutes. Auch das Einlesen bestehender TSchedules ist noch nicht möglich.

Ein weiteres Problem ist, dass Network Dateien bisher nur im WGS84 Koordinatenformat exportiert werden können. Für MATSim ist das ungeeignet, da es Distanzen am besten in kartesischen Koordinaten berechnet (vgl. Rieser et al., 2014, S.24).

Die genannten Probleme können durch spätere Updates des Plugins behoben werden. Außerdem sind weitere Funktionen denkbar. So könnten mithilfe von geschlossenen Ways Gebiete innerhalb des geöffneten Layers definiert werden, sodass nur in diesen befindliche Elemente konvertiert werden. Darüber hinaus könnten diese Gebiete unterschiedliche Hierarchie Filter erhalten, die nur bestimmte Highway Typen in den jeweiligen Gebieten konvertieren lassen. Eine weitere Möglichkeit wäre, eine Bearbeitungsmöglichkeit für Count-Dateien, ähnlich dem bestehenden *networkEditor* MATSim-Modul, zu implementieren.

Zusammenfassend lässt sich sagen, dass das Plugin für Straßennetze gut funktioniert. ÖV-Netze werden jedoch bislang unzureichend unterstützt. Die Visualisierung der Elemente ermöglicht ein übersichtliches Verständnis der zugrundeliegenden Daten. Durch die Echtzeitkonvertierung kann der Nutzer Netze interaktiv bearbeiten. Ein Vorteil gegenüber früheren Herangehensweisen ist außerdem die Möglichkeit der Fortschreibung der Daten im OSM Format. Alle Anpassungen seitens des Nutzers können somit jederzeit mit aktuellen OSM Daten zusammengeführt werden, ohne dass sie verlorengehen. Zukünftig kann die JOSM Erweiterung eine geeignete Hilfe für das graphische Bearbeiten von Straßen- sowie ÖV-Netzmodellen werden. Für eine optimale Funktionalität bedarf es jedoch weiterer Verbesserungen - im Plugin selbst, aber auch in den OSM Daten.

V. Literaturverzeichnis

Balmer, M., 2011. 4.3.2 Network | MATSim [online]. Netw. MATSim. URL http://www.matsim.org/docs/tutorials/8lessons/input/creating/network (Zugriff: 16.09.2014).

Bekhor, S., Dobler, C., Axhausen, K.W., 2011. Integration of Activity-Based with Agent-Based Models: an Example from the Tel Aviv Model and MATSim. Presented at the 90th Annual Meeting of the Transportation Research Board, Washington, D.C. [online], URL http://e-collection.library.ethz.ch/eserv/eth:1575/eth-1575-01.pdf (Zugriff: 10.09.2014).

Bennett, J., 2010. OpenStreetMap - Be your own Cartographer. Packt Publishing, Birmingham.

Budelmann, H., 1977. Möglichkeiten rechnergestützen Entwerfens in der Verkehrsplanung - Entwicklung und Anwendung eines Programmsystems zur Unterstützung von Entwurf und Beurteilung von Verkehrsnetzen, Veröffentlichungen des Instituts für Stadtbauwesen. Braunschweig.

Clausen, A., 1981. Die Netzredundanz im Rahmen eines mehrstufigen Beschreibungsansatzes für Verkehrsnetze. Technische Universität Carolo-Wilhelmina zu Braunschweig, Braunschweig.

De:Key:lanes – OpenStreetMap Wiki [online], 2014. URL http://wiki.openstreetmap.org/wiki/DE:Key:lanes (Zugriff 07.10.2014).

De:Key:railway – OpenStreetMap Wiki [online], 2014. URL http://wiki.openstreetmap.org/wiki/DE:Key:railway (Zugriff 08.10.2014).

DE:OSM XML – OpenStreetMap Wiki [online], 2013. . DEOSM XML. URL http://wiki.openstreetmap.org/wiki/DE:OSM_XML (Zugriff 09.10.2014).

Desktop Help 10.0 - Was ist Network Analyst? [online], o.J. . ArcGIS Resour. Cent. URL http://help.arcgis.com/de/arcgisdesktop/10.0/help/index.html#/na/00470000000100 0000/ (Zugriff 07.10.2014).

De:TaggingPresets – JOSM [online], 2014. . Objektvorlagen. URL https://josm.openstreetmap.de/wiki/De%3ATaggingPresets (Zugriff 09.10.2014).

DevelopersGuide/DevelopingPlugins – JOSM [online], o.J. URL http://josm.openstreetmap.de/wiki/DevelopersGuide/DevelopingPlugins (Zugriff 15.09.2014).

Donau-Iller-Nahverkehrsbund GmbH, o.J. DING - Donau-Iller-Nahverkehrsverbund GmbH: Linienfahrplan [online]. DING Linienfahrplan. URL http://www.ding.eu/fahrplaene/linienfahrplan.html (Zugriff 08.10.2014).

gnu.org [online], o.J. . GNU Gen. Public License GPL. URL http://www.gnu.org/licenses/old-licenses/gpl-2.0.html (Zugriff 09.10.2014).

Hölsken, D., Ruske, W., 1977. Entwicklung und Anwendung operationaler Verfahren zur Aufnahme und Fortschreibung von Netzmodellen, Stadt Region Land. Selbstverlag am Institut für Stadtbauwesen RWTH Aachen, Aachen.

Institut für Verkehrsplanung und Transportsysteme, 2009. ETH - IVT - VS - Glossar [online]. Öffentl. Verk. URL http://www.ivt.ethz.ch/oev/glossar (Zugriff 08.10.2014).

ITWissen, 2014a. Java :: Java :: ITWissen.info [online]. Java. URL http://www.itwissen.info/definition/lexikon/Java.html (Zugriff 07.10.2014).

ITWissen, 2014b. Plug-In :: plug-in :: ITWissen.info [online]. Plug-In. URL http://www.itwissen.info/definition/lexikon/Plug-In-plug-in.html (Zugriff 07.10.2014).

ITWissen, 2014c. Repository :: repository :: ITWissen.info [online]. Repository. URL http://www.itwissen.info/definition/lexikon/Repository-repository.html (Zugriff 07.10.2014).

ITWissen, 2014d. XML :: extensible markup language :: ITWissen.info [online]. XML Extensible Markup Lang. URL http://www.itwissen.info/definition/lexikon/extended-markup-language-XML.html (Zugriff 07.10.2014).

Java Archive. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 1. Januar 2014, 05:34 UTC [online]. Java Archive. URL http://de.wikipedia.org/w/index.php?title=Java_Archive&oldid=125993218 (Zugriff 09.10.2014).

Kommandozeile. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 25. Juli 2014, 12:39 UTC [online]. Wikipedia. URL http://de.wikipedia.org/w/index.php?title=Kommandozeile&oldid=132484244 (Zugriff 09.10.2014).

Meister, K., Balmer, M., Ciari, F., Horni, A., Rieser, M., Waraich, R.A., Axhausen, K.W., 2010. LARGE-SCALE AGENT-BASED TRAVEL DEMAND OPTIMIZATION APPLIED TO SWITZERLAND, INCLUDING MODE CHOICE. Presented at the

12th World Conference on Transport Research Society, Lissabon [online]. URL http://www.ivt.ethz.ch/vpl/publications/reports/ab625.pdf (Zugriff 08.09.2014).

Müller, K., 1998. Verkehr in systemtechnischer Darstellung und ihre Anwendung auf ein multimodales Güterverkehrskonzept, Fortschritt-Berichte VDI. VDI Verlag, Düsseldorf.

Nagurney, A., 2007. Mathematical Models of Transportation and Networks. Encycl. Life Support Syst. EOLSS, Mathematical Models in Economics [online]. URL http://supernet.isenberg.umass.edu/articles/eolss.pdf (Zugriff: 15.09.2014).

Netzwerk. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 06.10.2014, 14:07 UTC [online]. URL http://de.wikipedia.org/w/index.php?title=Netzwerk&oldid=134651425 (Zugriff 05.09.2014).

OpenStreetMap Taginfo [online], 2014. . Datenbank-Stat. URL http://taginfo.openstreetmap.org/reports/database_statistics (Zugriff 26.09.2014).

Osmosis – OpenStreetMap Wiki [online], 2014. URL http://wiki.openstreetmap.org/wiki/Osmosis (Zugriff 04.10.2014).

Points of interest – OpenStreetMap Wiki [online], 2014. Points Interest. URL http://wiki.openstreetmap.org/wiki/Points_of_interest (Zugriff 26.09.2014).

Proposed features/Public Transport [online], 2014. . Propos. Featur. Transp. URL http://wiki.openstreetmap.org/wiki/Proposed_features/Public_Transport (Zugriff 14.09.2014).

PTV Visum [online], 2014. . PTV Visum. URL http://vision-traffic.ptvgroup.com/de/produkte/ptv-visum/ (Zugriff 03.10.2014).

Public transport -OpenStreetMap Wiki [online], 2014. URL http://wiki.openstreetmap.org/wiki/DE:Public_transport (Zugriff 12.09.2014).

Ramm, F., Topf, J., 2009. OpenStreetMap - Die freie Weltkarte nutzen und mitgestalten, 2nd ed. Lehmanns Media, Berlin.

Rieser, M., 2010. Adding Transit to an Agent Based Transportation Simulation - Concepts and Implementation. TU Berlin, Berlin [online]. URL http://www.matsim.org/uploads/phdthesis_mrieser.pdf (Zugriff 10.09.2014)

Rieser, M., 2014. networkEditor | MATSim [online]. networkEditor. URL http://matsim.org/docs/extensions/networkEditor (Zugriff 10.8.14).

Rieser, M., Dobler, C., Dubernet, T., Grether, D., Horni, A., Lämmel, G., Waraich, R.A., Zilske, M., Axhausen, K.W., Nagel, K., 2014. MATSim User Guide - Development

Version [online]. URL http://ci.matsim.org:8080/view/All/job/UserGuide/ws/trunk/docs/user-guide/user-guide.pdf (Zugriff 15.09.2014).

senozon AG, 2014. Via [online]. URL http://senozon.com/de/produkte/via (Zugriff 20.09.2014).

Springer Gabler Verlag (Herausgeber), o.J. Stichwort: Individualverkehr. Gabler Wirtsch [online]. URL http://wirtschaftslexikon.gabler.de/Definition/individualverkehr.html (Zugriff 07.10.2014).

utf-8 - Computer Lexikon - Fachbegriffe verständlich erklärt | PC, EDV Glossar [online], 2006. . Utf-8 - Comput. Lex. URL http://www.computerlexikon.com/begriff-utf-8 (Zugriff 08.10.2014).

What is black box? definition and meaning [online], 2014. . BusinessDictionary.com. URL http://www.businessdictionary.com/definition/black-box.html (Zugriff 08.10.2014).

VI. Glossar

Black-Box-Verfahren	„Device, process, or system, whose inputs and outputs […] are known, but whose internal structure or working is (1) not well, or at all, understood, (2) not necessary to be understood for the job or purpose at hand […]“ (“What is black box?,” 2014)
Individualverkehr	„Verkehrsart, bei der die Verkehrsmittel nur von einem einzelnen oder einem beschränkten Personenkreis eingesetzt werden und bei dem der oder die Benutzer völlig frei sind in der Bestimmung der Zeit, des Fahrweges und des Zieles der Fahrt.“ (Springer Gabler Verlag)
Java	„[…] objektorientierte und plattformunabhängige Programmiersprache, deren Grundlagen in der ersten Hälfte der 1990er Jahre von Sun Microsystems entwickelt worden sind. […]“ (ITWissen, 2014a)
Konsole (Kommandozeile)	„[…] ist ein Eingabebereich […] für die Steuerung einer Software, der typischerweise (aber nicht zwingend) im Textmodus abläuft. […]“ (“Kommandozeile,” 2014)
Layer	hier: In JOSM verwendete Datenebene, die einem Datensatz zugeordnet ist. Siehe auch Kapitel 3.1
Listener	hier: ein Objekt, dass auf bestimmte Ereignisse reagiert (nach ihnen ‚horcht‘) Siehe auch Kapitel 3.2.3
öffentlicher Verkehr	„[…] stellt ein Leistungsangebot zur Ortsveränderung von Personen und materiellen Gütern mit definierter örtlicher und zeitlicher Verfügbarkeit bereit, das von jedermann aufgrund vorgegebener Beförderungsbestimmungen beansprucht werden kann […]“ (Institut für Verkehrsplanung und Transportsysteme, 2009)
Plugin	„Ein Plug-In ist eine Software, die eigenständig eine Teil-Funktionalität abdeckt und damit die Funktionalität eines Gesamtsystems erweitert. Für die Ausführung eines Plug-Ins wird dabei eine übergeordnete Plattform benötigt, die wiederum die benötigten Schnittstellen zur Integration des Plug-Ins zur Verfügung stellt[…]“(ITWissen, 2014b)
Point of Interest	„A point of interest or POI is a feature on a map (or in a geodataset) that occupies a particular point, as opposed to linear features like roads or areas of landuse.“ (“Points of interest – OpenStreetMap Wiki,” 2014)
Repository	„Bei einem Repository handelt es sich um eine zentrale Ablage für Daten, Dokumente, Objekte und Programme mit

	ihren Metadaten. Neben den eigentlichen Daten werden im Repository auch die verschiedenen Versionen […] abgelegt[…].“ (ITWissen, 2014c)
UTF-8	„Unicode ist ein internationaler Standard, der […] definiert, wie Bits (bzw. Bytes) in ein Zeichen am Bildschirm umgesetzt werden.“ (“utf-8 - Computer Lexikon - Fachbegriffe verständlich erklärt | PC, EDV Glossar,” 2006)
WGS84	*World Geodetic System 1984 - Weltweites Geodätisches System 1984*. Beschreibt ein weltweit gültiges Koordinatensystem zum Bestimmen von Positionen.
XML	„Extensible Markup Language (XML) dient der Beschreibung sowie dem Austausch von komplexen Datenstrukturen.“ (ITWissen, 2014d)

VII. Anhang

Titel	**Seite**
A Beispiel .osm Datei	XIII
B Beispiel network.xml	XIV
C Beispiel schedule.xml	XV
D matsimPreset.xml	XVI
E In OSM eingepflegte ÖV-Linien Ulms	XVII

A Beispiel .osm Datei

aus (“DE:OSM XML – OpenStreetMap Wiki,” 2013)

```
<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.6" generator="OpenStreetMap server">
  <bounds minlat="51.5073601795557" minlon="-0.108157396316528" maxlat="51.5076406454029"
maxlon="-0.107599496841431"/>
  <node id="319408586" lat="51.5074089" lon="-0.1080108" version="1" changeset="440330"
user="smsm1" uid="6871" visible="true" timestamp="2008-12-17T01:18:42Z"/>
  <node id="319408587" lat="51.5074343" lon="-0.1081264" version="1" changeset="440330"
user="smsm1" uid="6871" visible="true" timestamp="2008-12-17T01:18:42Z"/>
  <node id="275452090" lat="51.5075933" lon="-0.1076186" version="3" changeset="2980587"
user="nickb" uid="1697" visible="true" timestamp="2009-10-29T12:14:35Z">
    <tag k="name" v="Jam's Sandwich Bar"/>
    <tag k="amenity" v="cafe"/>
  </node>
  <node id="304994980" lat="51.5074645" lon="-0.1075735" version="2" changeset="388960"
user="BiIbo" uid="3516" visible="true" timestamp="2009-02-13T00:41:47Z">
    <tag k="barrier" v="gate"/>
  </node>
  <node id="304994981" lat="51.5074723" lon="-0.1075014" version="1" changeset="374143"
user="Matt" uid="70" visible="true" timestamp="2008-10-16T16:35:57Z"/>
  <node id="304994979" lat="51.507406" lon="-0.1083348" version="4" changeset="2114003"
user="jamicu" uid="38244" visible="true" timestamp="2009-08-12T01:33:32Z"/>
  <way id="27776903" visible="true" timestamp="2009-05-31T13:39:15Z" version="3"
changeset="1368552" user="Matt" uid="70">
    <nd ref="304994979"/>
    <nd ref="319408587"/>
    <nd ref="319408586"/>
    <nd ref="304994980"/>
    <nd ref="304994981"/>
    <tag k="access" v="private"/>
    <tag k="highway" v="service"/>
  </way>
</osm>
```

B Beispiel network.xml

aus (Rieser et al., 2014, S.18)

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE network SYSTEM "http://www.matsim.org/files/dtd/network_v1.dtd">
<network name="example network">
	<nodes>
		<node id="1" x="0.0" y="0.0"/>
		<node id="2" x="1000.0" y="0.0"/>
		<node id="3" x="1000.0" y="1000.0"/>
	</nodes>
	<links>
		<link id="1" from="1" to="2" length="3000.00" capacity="3600"
			freespeed="27.78" permlanes="2" modes="car" />
		<link id="2" from="2" to="3" length="4000.00" capacity="1800"
			freespeed="27.78" permlanes="1" modes="car" />
		<link id="3" from="3" to="2" length="4000.00" capacity="1800"
			freespeed="27.78" permlanes="1" modes="car" />
		<link id="4" from="3" to="1" length="6000.00" capacity="3600"
			freespeed="27.78" permlanes="2" modes="car" />
	</links>
</network>
```

C Beispiel schedule.xml

aus (Rieser et al., 2014, S.22)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE transitSchedule SYSTEM "http://www.matsim.org/files/dtd/transitSchedule_v1.dtd">
<transitSchedule>
	<transitStops>
		<stopFacility id="1" x="990.0" y="0.0" name="Adorf"
							linkRefId="1" isBlocking="false"/>
		<stopFacility id="2" x="1100.0" y="980.0" name="Beweiler"
							linkRefId="2" isBlocking="true"/>
		<stopFacility id="3" x="0.0" y="10.0" name="Cestadt"
							linkRefId="3" isBlocking="false"/>
	</transitStops>
	<transitLine id="Blue Line">
		<transitRoute id="1">
			<description>Just a comment.</description>
			<transportMode>bus</transportMode>
			<routeProfile>
				<stop refId="1" departureOffset="00:00:00"/>
				<stop refId="2" arrivalOffset="00:02:30" departureOffset="00:03:00"
								awaitDeparture="true"/>
				<stop refId="3" arrivalOffset="00:05:00" awaitDeparture="true"/>
			</routeProfile>
			<route>
				<link refId="1"/>
				<link refId="2"/>
				<link refId="3"/>
			</route>
			<departures>
				<departure id="1" departureTime="07:00:00" vehicleRefId="12"/>
				<departure id="2" departureTime="07:05:00" vehicleRefId="23"/>
				<departure id="3" departureTime="07:10:00" vehicleRefId="34"/>
			</departures>
		</transitRoute>
	</transitLine>
</transitSchedule>
```

D matsimPreset.xml

nach den Richtlinien aus (“De:TaggingPresets – JOSM,” 2014)

```
<?xml version="1.0" encoding="UTF-8"?>
<presets xmlns="http://josm.openstreetmap.de/tagging-preset-1.0"
	author="nkuehnel" version="0.1_2013-11-13" description="MATSim Preset"
	shortdescription="MATSim Preset" de.description="MATSim Vorlage"
	de.shortdescription="MATSim Vorlage" fr.description="Modèle de MATSim"
	fr.shortdescription="Modèle de MATSim">

	<!-- Please recheck translation. -->

	<group icon="images/dialogs/logo.png" name="MATSim" de.name="MATSim"
		fr.name="MATSim">
		<item name="Link" de.name="Kante" icon="images/dialogs/logo.png"
			type="way">
			<label text="Edit MATSim Properties" />
			<space />
			<label text="Required Fields:" />
			<space />
			<text key="freespeed" text="freespeed" default="0.00" delete_if_empty="false" />
			<text key="capacity" text="capacity" default="0.00" delete_if_empty="false" />
			<text key="permlanes" text="permanent lanes" default="0.00"
	delete_if_empty="false" />
			<multiselect key="modes" text="modes"
values="car;pt;ride;bike;walk;transit_walk;other" default="car"/>
			<space />
			<optional>
				<text key="length" text="length (estimated if not set)" default=""
delete_if_empty="true" />
			</optional>
		</item>
	</group>
</presets>
```

E In OSM eingepflegte ÖV-Linien Ulms

eigene Darstellung, (vgl. Donau-Iller-Nahverkehrsbund GmbH, n.d.)

	Linie	Verbindung	vorhanden	Anzahl Routen
Stadtverkehr	1	Söflingen – Böfingen	ja, unvollständig	2
	3	Wiblingen (Alte Siedlung) - Wissenschaftsstadt	ja, unvollständig	1
	4	Grimmelfingen – Kuhberg – Böfingen Süd	ja, unvollständig	1
	5	Ludwigsfeld / Wiley – Wissenschaftsstadt	ja, unvollständig	1
	6	Donaustadion – Eselsberg – Universität Süd	ja, unvollständig	1
	7	Willy–Brandt–Platz - Jungingen (Rathaus)	ja, unvollständig	1
	9	Wiblingen (Reutlinger Straße) – Rosengasse	ja, unvollständig	1
	10	Donautal (Hans-Lorenser-Straße) – Blautal-Center	ja, unvollständig	1
	11	Roter Berg – Söflingen - Gleißselstetten	ja, unvollständig	1
	12	Eichenplatz - Eichenhang	nein	-
	13	Am Hochsträß - Kuhberg - Eselsberg - Universität Süd	ja, unvollständig	1
	14	Wiblingen (Pranger) - Donautal - Kuhberg	ja, unvollständig	1
	15	Willy-Brandt-Platz - Wissenschaftsstadt	ja, unvollständig	1
Von / Nach Ulm	20	RINGVERKEHR Ulm - Einsingen - Ermingen - Ulm	ja, unvollständig	1
	21	Ulm - Erbach - Ehingen / Laupheim	ja, unvollständig	1
	22	Ulm - Dellmensingen - Laupheim	ja, unvollständig	1
	23	Ulm - Unterweiler - Schnürpflingen - Beuren	ja, unvollständig	1
	24	Ulm - Hüttisheim - Burgrieden - Laupheim	ja, unvollständig	1
	30	Ulm - Laichingen - Westerheim - Bad Urach	ja, unvollständig	1
	36/37/38	Ulm - Blaustein - Wippingen / Bermaringen / Markbronn	ja	3
	44	Mähringen - Lehr - Jungingen - Böfingen	nein	-
	45	Jungingen - Lehr - Wissenschaftsstadt	ja	1
	46	Ulm - Amstetten - Geislingen (Steige)	nein	-
	47	Jungingen - Lehrer Feld - Jungingen	nein	-
	48	GESAMTVERKEHR Ulm - Lehr - Mähringen	ja, unvollständig	1

	49	Ulm - Lehr - Dornstadt - Nellingen	ja, unvollständig	1
	58	Ulm - Göttingen - Bernstadt	nein	-
	59	Ulm - Langenau - Niederstotzingen - Heidenheim	nein	-
	70	Ulm - Dietenheim (- Illertissen)	ja, unvollständig	1
	71	Ulm - Neu-Ulm Industriegebiet - Ludwigsfeld	ja, unvollständig	1
	73	Ulm - Senden - Vöhringen	ja, unvollständig	1
	73	Neu-Ulm ZUP - Wiblinger Steig	nein	-
	74	Ulm - Illerkirchberg - Staig - Bihlafingen	ja, unvollständig	1
	77	Ulm - Holzschwang - Hittistetten	ja, unvollständig	1
	77	Neu-Ulm ZUP - Starkfeld - Schwaighofen - Reutti - Jedelhausen - Holzschwang	nein	-
	78	Ulm - Finningen - Weißenhorn	ja, unvollständig	1
	79	Ulm - Neu-Ulm Pfuhler Ried	ja, unvollständig	1
	84/85/88	GESAMTVERKEHR Ulm - Pfuhl - Burlafingen - Steinheim	ja, unvollständig	3
	89	Neu-Ulm ZUP - Pfuhl Steinhäule	ja, unvollständig	1