Technische Universität München
Ingenieurfakultät Bau Geo Umwelt
Lehrstuhl für Geoinformation
Univ.-Prof. Dr. rer. nat. Thomas H. Kolbe

Kopplung von Verkehrssimulation und semantischen 3D-Stadtmodellen in CityGML

Roland Ruhdorfer, B.Sc.

Master's Thesis

Bearbeitung:	01.02.2017 - 01.08.2017
Studiengang:	Geodäsie und Geoinformation (Master)
Betreuer:	Univ.-Prof. Dr. rer. nat. Thomas H. Kolbe
	Dipl.-Geogr. Maximilian Sindram
	M.Sc. Bruno Willenborg

2017

Selbstständigkeitserklärung

Erklärung gemäß §18 Absatz 9 APSO der Technischen Universität München (TUM):

”Ich versichere, dass ich diese Master’s Thesis selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe."

Roland Ruhdorfer

München, den 19. Juli 2017

Kurzfassung

Im Hinblick auf ein stetig wachsendes Verkehrsaufkommen gilt es, die Nutzung bestehender Verkehrswege zu optimieren. Dabei stellen Verkehrssimulationen ein wichtiges Werkzeug dar. Diese benötigen als Eingangsdaten insbesondere detaillierte Informationen zum Straßennetz, deren Einpflege aktuell mit großem manuellen Aufwand verbunden ist. Zunehmende Bedeutung als Datengrundlage für Analysen und Simulationen verschiedenster Anwendungen gewinnen semantische 3D-Stadtmodelle. Sie liefern neben Informationen zur Semantik die Geometrie und das Aussehen sowie topologische Beziehungen von Stadtobjekten. In dieser Masterarbeit wird zunächst auf theoretischer Ebene untersucht, welche der für eine Verkehrssimulation erforderlichen Daten ein semantisches 3D-Stadtmodell liefern kann. Eine Kopplung von Verkehrssimulation und 3D-Stadtmodell wird auf Modellebene vollzogen und Potentiale wie Herausforderungen werden diskutiert. Im Anschluss an die theoretischen Betrachtungen wird ein Konzept zur automatischen Ableitung eines Simulationsgraphen für die Verkehrssimulation Vissim aus Stadtmodellen im CityGML Format entwickelt. Ein weiteres Konzept behandelt die Visualisierung dynamischer Ergebnisse einer Verkehrssimulation in 3D-Stadtmodellen. Beide Konzepte erfahren eine praktische Umsetzung als prototypische Implementierung in der Umgebung der 3DCityDB. Eine Evaluierung der Ableitung von Simulationsgraphen wird anhand zweier Testdatensätze durchgeführt. Die Visualisierung von Simulationsergebnissen erfolgt in Google Earth und im auf Cesium basierenden 3D-Web-Map-Client der 3DCityDB.

Abstract

Due to steadily growing volume of traffic, using existing transport routes must be optimized. In doing so traffic simulations are a vital tool. As input data detailed information about the transport system is primarily required. Generating and adding such kind of data is currently a time-consuming act. As database for various types of analysis and simulations semantic 3D city models recently gained importance. Apart from semantic data they provide geometry, appearance and topologic informations on city objects. This Master's Thesis examines on theoretical basis, which information can be supplied by city models for traffic simulation applications. A coupling of traffic simulation with 3D city models is done at model level and resultant potentials as challenges are discussed. Following theoretical considerations a concept for an automatic derivation of road networks for the traffic simulation Vissim from city models in CityGML format is developed. Another concept deals with the visualization of dynamic traffic simulation results in 3D city models. Both are prototypical implemented in the enviroment of the 3DCityDB. An evaluation is carried out on basis of two example datasets. The visualization of simulation results utilizes Google Earth and the 3D Web Map Client, a web-based viewer for 3DCityDB based on Cesium.

Inhaltsverzeichnis

Abkürzungsverzeichnis

3D	3-Dimensional
ADE	Application Domain Extension
API	Application Programming Interface
COM	Component Object Model
CRS	Coordinate Reference System
CityGML	City Geography Markup Language
CRS	Coordinate Reference System
CSV	Comma Separated Values
DGM	Digitales Gelände Modell
EPSG	European Petroleum Survey Group Geodesy
GIS	Geoinformationssystem
GML	Geography Markup Language
ID	Identifikation
ISO	International Organization for Standardization
KML	Keyhole Markup Language
LoD	Level of Detail
MDE	Model Driven Engineering
MTS	Microscopic Traffic Simulation
OGC	Open Geospatial Consortium
OSM	Open Street Map
SRDBMS	Spatially-Enhanced Relational Database Management System
SST	Schnittstelle
UML	Unified Modeling Language

URI Uniform Resource Identifier

WebGL Web Graphics Library

WGS World Geodetic System

XML Extensible Markup Language

1 Einleitung

Das Bundesministerium für Verkehr und digitale Infrastruktur nennt Mobilität als „zentrale Voraussetzung für wirtschaftliches Wachstum, Beschäftigung und Teilhabe des Einzelnen am gesellschaftlichen Leben“ [BMVI, 2016]. Mobilität im Sinne räumlicher Bewegungen von Personen und Gütern ist ein komplexes System. Quantitativ lässt es sich definieren als Verkehrsaufkommen (Anzahl an Wegen, Fahrten oder Tonnen je Zeiteinheit) bzw. als Verkehrsleistung (Wege-, Fahrten- oder Tonnenkilometer je Zeiteinheit) [Steierwald et al., 1994]. Das zunehmende Verkehrsaufkommen insbesondere in dichtbesiedelten Gebieten stellt eine Herausforderung dar. Um dieser Belastung gerecht zu werden, gilt es die bestehenden Verkehrswege optimal zu nutzen. Eine wichtige Rolle fällt dabei Politik und Wissenschaft zu. Sowohl in der Verkehrsplanung als integralem Bestandteil der Stadtplanung als auch mit Hilfe von Verkehrsflussmodellierung können Maßnahmen gefunden werden, um bestimmte Verkehrswege zu entlasten oder das Verkehrsaufkommen zu verringern. Die Durchführung von Verkehrssimulationen dient der Erprobung solcher Maßnahmen. Dabei wird der Verkehrsablauf durch Modelle beschrieben [Stegemann & Ameling, 1982]. Verkehrsmodelle helfen auf Basis bekannter Abhängigkeiten und Zusammenhänge verkehrliche und andere Folgen unter definierten künftigen Rahmenbedingungen abzuschätzen [Steierwald et al., 1994]. Allgemein lassen sich mit Hilfe von Simulationen Erkenntnisse erlangen, die durch Nachbildung eines dynamischen Prozesses in einem Modell zustande kommen und auf die Wirklichkeit übertragbar sind [Dallmeyer, 2014]. Die Vorteile von Simulationen sind vielfältig. Neben Einsparung von Kosten und Zeit ist die Datenauswertung einfach und detailliert. Ergebnisse von Simulationen sind reproduzierbar und Veränderungen an der Testumgebung lassen sich auf einfache Art und Weise vollziehen [Margreiter, 2016].

1.1 Thematischer Hintergrund und Motivation

Für Verkehrssimulationen werden Datenquellen benötigt, die reale Straßenverläufe abbilden [Dallmeyer, 2014]. Neben detaillierten Informationen zum Straßennetz sind verkehrsspezifische Informationen für den Aufbau der Modelle wichtige Eingangsgrößen. Das Straßennetz wird in erster Linie mit Hilfe von Straßenkarten, aber auch anderen Geodaten beschrieben. Zu verkehrsbeschreibenden Daten zählt beispielsweise das Verkehrsaufkommen, das heißt die Anzahl an Fahrzeugen auf einer bestimmten Strecke. Quellen hierfür können Trajektoriendaten aus kamerabasierten Messmethoden sowie Floating-Car-Daten sein [Treiber & Kesting, 2010]. Bei ersteren werden aus Video-Bildern oder

Foto-Serien mittels automatischer Algorithmen die Positionen von Fahrzeugen über die Zeit extrahiert. Floating-Car-Daten werden dagegen mit Sensoren an diskreten Messfahrzeugen erfasst, die im Verkehrsstrom treiben (engl.: “float“).
Als Datengrundlage für verschiedenste Analysen und Simulationen, unter anderem in den Bereichen Stadt- und Raumplanung, Umwelt und Energie, wächst die Bedeutung von Stadtmodellen. Semantische 3D-Stadtmodelle können die Geometrie und das Aussehen von Objekten abbilden, sowie topologische Beziehungen und semantische Daten liefern. Damit sind sie mehr als eine Möglichkeit zur reinen Visualisierung des urbanen oder ländlichen Raums und für eine Vielzahl von Anwendungen geeignet, die Lösungen raumbezogener Aufgabenstellungen erfordern.
Aktuell ist die Eingabe der Ausgangsdaten einer Simulation mit großem Aufwand verbunden und bedarf besonderer Kompetenz [Liang et al., 2008]. Häufig müssen die Simulationsnetze manuell aus Straßenkarten und anderen Informationsquellen abgeleitet werden. Die Kombination von 3D-Stadtmodellen mit Verkehrs- und Fahrsimulationen verspricht dahingehend großes Potenzial. Durch die detaillierte Modellierung des Verkehrsraumes eröffnen sich Aussichten, den Import von Verkehrsnetzen in Verkehrssimulationen stark zu vereinfachen. Für Simulationen bietet das virtuelle Stadtmodell darüber hinaus neben Möglichkeiten zur Ergebnis-Visualisierung eine Vielzahl weiterer wertvoller Informationen und Analysemethoden. So ermöglicht eine Kopplung von Verkehrssimulationen mit Stadtmodell-Datenbanken eine verbesserte Abbildung der Verkehrsnachfrage beispielsweise durch Daten zur Anwohnerzahl oder zu Attraktoren wie Arbeitsstätten und Einkaufsmöglichkeiten. Im urbanen Kontext können Verkehrssimulationen Grundlage für weitere Anwendungen und Analysen sein. Durch die Simulation der Bewegungen von Bewohnern können beispielsweise Untersuchungen zum Kraftstoffverbrauch und zur Emissionsausbreitung in Städten durchgeführt werden [Dallmeyer, 2014]. Neben Analysen zur CO2- oder Lärmemission sind in Kombination mit digitalen 3D-Stadtmodellen weitere Untersuchungen wie z.B. Sichtbarkeitsanalysen denkbar. Auch Kommunikationssysteme wie Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I) oder Mobilfunk können simuliert werden [Margreiter, 2016].

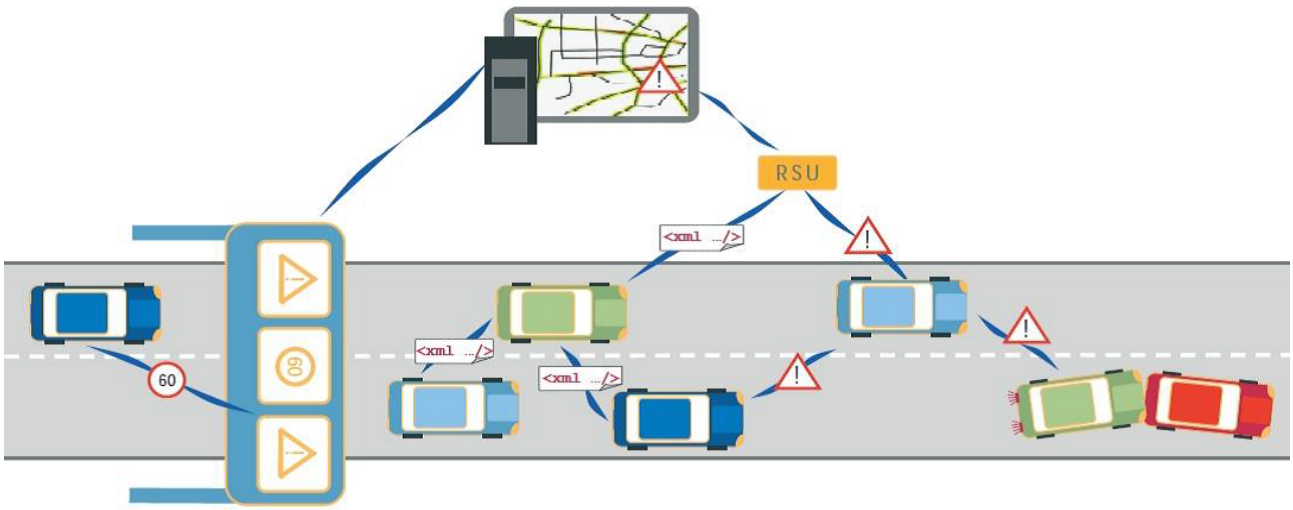


Abbildung 1.1 – Vehicle-2-X Szenario [Margreiter, 2016]

Wachsende Funktionalität und Verknüpfungen zwischen mehreren Fahrzeugen (vehicle-2-X Kommunikation, Abbildung 1.1) führen zu immer komplexeren Systemen. Als virtuelle Testumgebungen für Fahrerassistenzsysteme und im Bereich Autonomes Fahren erlangen Verkehrs- und Fahrsimulationen eine immer größere Bedeutung [Richter et al., 2016]. Zum einen können mit Hilfe derartiger Simulationen Entwicklungszeit und -kosten reduziert werden [Colditz et al., 2007], [Richter et al., 2016]. Zum anderen spielen sie eine Rolle bei der Zulassung autonomer Autos. Realtests sollen in Zukunft nicht ersetzt werden, aber der Anteil an virtuellen Tests mit Hilfe von Simulationen steigt gegenüber Tests im Realverkehr Autoherstellern zufolge stetig [Greis, 2016].
Die Verknüpfung von Verkehrssimulation und virtuellem 3D-Stadtmodell können zu verbesserten, da realitätsnäheren, Fahrsimulatoren führen und damit ein den Anforderungen entsprechendes virtuelles Testfeld liefern [Margreiter, 2016]. Mit Hilfe von digitalen Modellen realer Städte lässt sich die Kluft zwischen simulierter und echter Welt schließen. Voraussetzungen sind ein detailliertes und hochgenaues Straßennetzwerk sowie ein realistisches Umgebungsmodell [Richter et al., 2016]. Abbildung 1.2 zeigt ein Beispiel für die Modellierung einer komplexen Straßenkreuzung im urbanen Bereich. Links ist der Graph eines Vekehrsnetzwerkes dargestellt, rechts die resultierende 3D-Szene.

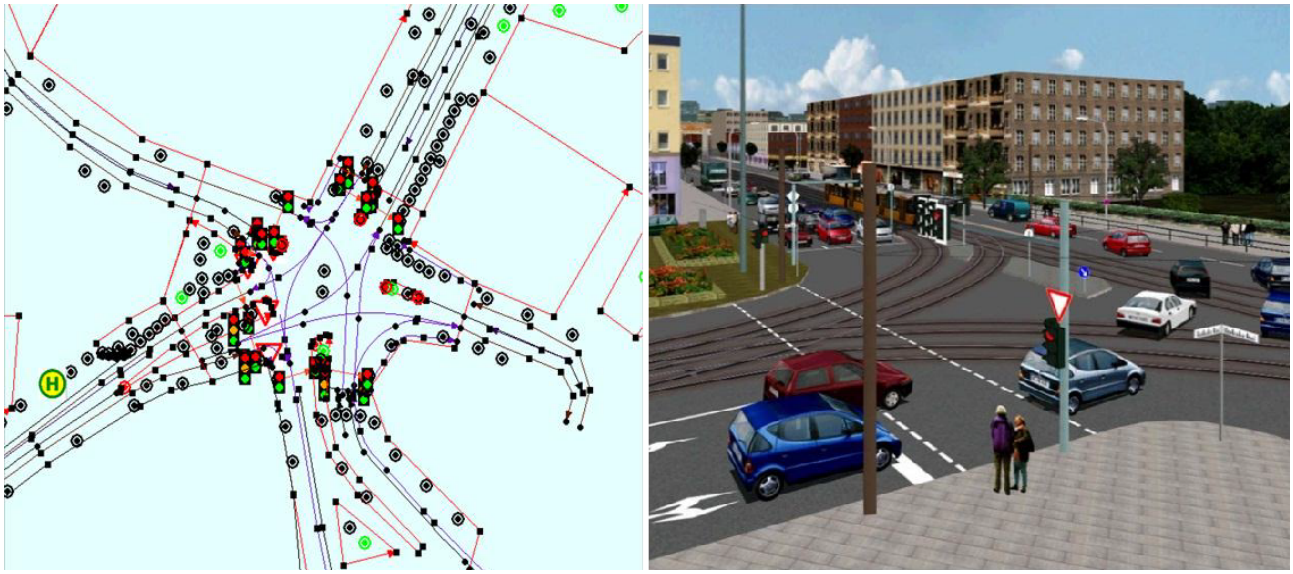


Abbildung 1.2 – Modellierung einer komplexen Kreuzung im urbanen Bereich (links: Simulationsgraph des Verkehrsnetzes, rechts: resultierende 3D-Szene) [Gröger et al., 2012, S.127]
Bildquelle: Rheinmetall Defence Electronics

1.2 Problemstellungen und Forschungsfragen

Die vorliegende Masterarbeit beschäftigt sich mit zwei der genannten Potenziale einer Verknüpfung von Verkehrssimulationen mit Stadtmodellen in CityGML. Hinsichtlich einer automatischen Ableitung von Simulationsdaten werden folgende Fragestellungen untersucht:

- Welcher visueller, geometrischer, topologischer und semantischer Informationen über den Verkehrsraum bedarf es in Verkehrs- und Fahrsimulationen?
- Welche Formen der Verkehrsraummodellierung bestehen und welche Anforderungen stellen sich an ein Verkehrsraummodell, um die benötigten Informationen zu liefern?
- Welche der erforderlichen Daten kann ein CityGML-Stadtmodell als Datenquelle liefern?
- Wie kann die Kopplung von Verkehrssimulation mit 3D-Stadtmodellen auf Modellebene beschrieben werden und wie lassen sich konkret Simulationsdaten aus CityGML ableiten und der Simulation zur Verfügung stellen?

Im Rahmen einer Visualisierung von Simulationsergebnissen in 3D-Stadtmodellen sollen weitere Fragen bearbeitet werden:

- Wie lassen sich Verkehrssimulationsdaten, konkret am Beispiel der Verkehrssimulation Vissim, formal in UML beschreiben, um Transformationen auf Basis der Modellebene zwischen Simulation (Vissim) und Stadtmodell (CityGML) durchführen zu können?
- Wie lassen sich zeitabhängige Simulationsdaten in CityGML integrieren?
- Wie können dynamische Ergebnisse einer Verkehrssimulation in einem Stadtmodell visualisiert werden?

1.3 Aufbau und Inhalt der Arbeit

Der Aufbau der Arbeit setzt sich zusammen aus einem theoretischen Teil mit anschließender Konzepterarbeitung sowie einer praktischen Umsetzung mit Evaluierung. Zu Beginn werden für die Arbeit wichtige Grundlagen (Kapitel 2) und relevante Forschungsarbeiten (Kapitel 3) dargelegt. In Kapitel 4 wird die Modellierung des Verkehrsraumes auf allgemeiner Ebene und hinsichtlich der Anwendungen Verkehrssimulation bzw. Stadtmodell betrachtet. Kapitel 5 widmet sich der Kopplung der Verkehrssimulation Vissim (Verkehr in Städten Simulation) mit Stadtmodellen im CityGML-Format auf Modellebene. Darauf aufbauend werden in den Kapiteln 6 und 7 Konzepte entwickelt, wie eine automatische Ableitung eines Simulationsgraphen und die Visualisierung von Simulationsergebnissen in einem Stadtmodell umgesetzt werden können. In Kapitel 8 wird eine Realisierung der Konzepte als prototypische Implementierung vorgestellt. Kapitel 9 präsentiert Ergebnisse zur automatischen Ableitung von Simulationsgraphen aus CityGML sowie zur Visualisierung von dynamischen Fahrzeugdaten aus Verkehrssimulationen in 3D-Stadtmodellen. In Kapitel 10 wird ein zusammenfassendes Fazit gezogen und ein Ausblick auf weiterführende Möglichkeiten hinsichtlich einer Verknüpfung von Verkehrssimulationen mit 3D-Stadtmodellen gegeben.

2 Theoretische Grundlagen

Inhalt dieses Kapitels sind die theoretischen Grundlagen, auf denen die Arbeit aufbaut. Einer Beschreibung von Fahr- und Verkehrssimulationen folgt ein Überblick über den CityGML-Standard, einem offenen Datenmodell für virtuelle, semantische 3D-Stadtmodelle. Darüber hinaus wird die 3DCityDB vorgestellt, eine Datenbanklösung zur Speicherung und zum Austausch von Stadtmodellen im CityGML-Format. Weiter umfasst das Kapitel eine Darstellung des zugehörigen Softwaretools, dem 3D City Database Importer/Exporter. Abschließend findet sich eine Zusammenstellung, welche Relationen der Komponenten untereinander und in Bezug zum vorliegenden Thema aufzeigt.

2.1 Fahr- und Verkehrssimulationen

Eine Komponente der thematisierten Kopplung stellen Verkehrssimulationen dar. Dieser Abschnitt dient der detaillierten Beschreibung von Verkehrssimulationen sowie Verkehrssimulationssystemen und insbesondere auch der Abgrenzung gegenüber Fahrsimulationen. Beide Simulationsarten sind eng mit dem Straßenraum verbunden, weshalb sich mehrere Elemente überschneiden. Ein Unterschied liegt im Anwendungsbereich. So sind Fahrsimulationen Bestandteil des Virtual Prototyping, wobei das Zusammenspiel zwischen Fahrer und virtuellem Prototyp eines Fahrzeugs bzw. einer Fahrzeugkomponente untersucht werden [Colditz et al., 2007], [Kreft, 2012]. Demgegenüber dienen Verkehrssimulationen als wichtiges Werkzeug in der Verkehrsplanung und zur Optimierung des Verkehrsmanagements [Treiber & Kesting, 2010].

2.1.1 Modelle zur Fahrsimulation

Fahrsimulationen werden für verschiedene Tests im Bereich der Automobil-Entwicklung verwendet, so unter anderem zur Beurteilung der Fahrdynamik und Fahreigenschaften sowie zur Evaluierung von Fahrassistenzsystemen. Dazu ist es erforderlich, Verkehrssituationen realitätsnah nachzubilden [Colditz et al., 2007]. Eine weitere Anwendung von Fahrsimulationen ist im Bereich des Trainings von Auto-, LKW- oder Busfahrern sowie Lokführern. Ebenso sind derartige Trainingssimulatoren für Fahrer von Einsatzfahrzeugen verfügbar. Neben der Simulation des umliegenden Verkehrs ist dabei die realistische Modellierung der virtuellen Umgebung von hoher Bedeutung [Randt et al., 2007].

Zu den Systemkomponenten einer Fahrsimulation zählen die Simulationsmodelle sowie Software- und Hardwarekomponenten. Unter letztere fallen ein Simulationskern zur Kontrolle und Synchronisation der Datenflüsse, ein Bildgenerator zur Darstellung der virtuellen Szene, ein Bewegungssystem, eine Fahrzeugattrappe mit Bedienelementen und nicht zuletzt ein leistungsfähiger Rechner. Simulationsmodelle beinhalten die Modellierung der Fahrdynamik, von Assistenzsystemen, des Verhaltens von Fremdverkehr, der Akustik und der Umgebung [Kreft, 2012]. In der Folge soll lediglich auf das für das vorliegende Thema relevante Umgebungsmodell eingegangen werden. Für Details zu den sonstigen Komponenten einer Fahrsimulation sei auf die Literatur verwiesen (vgl. u.a. [Colditz et al., 2007], [Negele, 2007], [Kreft, 2012]).
Das Umgebungsmodell lässt sich in das Logik- und das Graphikmodell unterteilen. Das Logikmodell als Grundlage der eigentlichen Simulation beinhaltet topologische, geometrische, logische und physikalische Aspekte [Kreft, 2012]. Zur Spezifikation des Logikmodels gibt es derzeit zwei de-facto-Standards: OpenDrive und RoadXML, die beide XML-basierte Formate sind und damit anwenderspezifisch erweiterbar. Ein Straßennetzwerk im RoadXML-Format ist eine Zusammensetzung aus Teilnetzen, die jeweils eine Sammlung von Kreuzungen und Pfaden darstellt. Diese sind wiederum angereichert mit verschiedenen Datenlayern, zum Beispiel Straßenprofilen, Verkehrsschildern und Verkehr [Ducloux et al., 2016]. Das OpenDrive-Format beschreibt auf analytische Weise die Geometrie des Straßennetzes sowie Elemente, die die Logik des Verkehrssystems beeinflussen. Möglich sind damit detaillierte Beschreibungen von Straßen- und Kreuzungssystemen sowie Definitionen von Verkehrssignalen und Schildern [Dupuis, 2015]. In Kapitel 4 werden im Rahmen einer Analyse verschiedener Verkehrsraummodelle beide Formate ausführlicher betrachtet. Das Graphikmodell ist Voraussetzung für die computergraphische Darstellung der Umgebung. Die Rolle des Graphikmodells im Gesamtsystem ist damit in erster Linie, eine realitätsnahe Szene zu schaffen. Zum Graphikmodell gehören Straßen und Kreuzungen inklusive Verkehrsmobiliar, Gebäuden, Gelände sowie Dekorationsobjekte [Kreft, 2012].
Für die Simulation des Gegenverkehres in Fahrsimulatoren finden Verkehrsflussmodelle Eingang, welche im nachfolgenden Abschnitt beschrieben werden.

2.1.2 Modelle zur Verkehrssimulation

Folgt man einer inhaltlichen Abstraktion, können Verkehrsflussmodelle im Wesentlichen in zwei Arten unterschieden werden:
In makroskopischen Modellen (Aggregatsmodelle) wird der Verkehrsfluss analog zu strömenden Flüssigkeiten oder Gasen abgebildet. Zu den Kenngrößen zählen Verkehrsdichte, Verkehrsfluss, mittlere Geschwindigkeit oder Geschwindigkeitsvarianz, die lokal aggregiert werden und damit räumlich und zeitlich veränderliche Größen darstellen. Somit können mit Makromodellen Staus und Ausbreitungsgeschwindigkeiten von Störungen untersucht werden [Treiber & Kesting, 2010]. Generell bieten sich makroskopische Modelle bei der Betrachtung ausreichend großer Zeiträume und Streckenabschnitte an, um anhand bestimmter Parameter über Zeit und Raum gemittelte Kenngrößen vorherzusagen [Dallmeyer, 2014].

Mikroskopische Modelle beschreiben den Verkehrsstrom ausgehend von einzelnen „Fahrer-Fahrzeug-Teilchen“. Damit werden Reaktionen wie Beschleunigen, Bremsen oder Spurwechseln eines jeden Fahrers in Abhängigkeit von den anderen Verkehrsteilnehmern modelliert. Hierbei sind relevante Größen die Positionen, Geschwindigkeiten und Beschleunigungen der einzelnen Verkehrsteilnehmer. Im Gegensatz zu makroskopischen Modellen ist bei mikroskopischen Modellen die Heterogenität des Verkehrs von zentraler Bedeutung. Ausgangspunkt sind einzelne Verkehrsteilnehmer und deren Verhalten. Simuliert werden können damit Auswirkungen von Tempolimits oder Überholverboten [Treiber & Kesting, 2010]. Mikroskopische Modelle können zusätzlich nach dem Detailierungsgrad unterschieden werden. Um das Verhaltensmodell von Fahrer und Fahrzeug bis auf Details wie z.B. den Einfluss des Motors, der Reifen oder der Stellung von Gaspedal und Lenkung zu spezifizieren, werden submikroskopische Modelle verwendet [Hoffmann, 2013]. Modelle, die hoch komplexe Wirkungszusammenhänge abbilden, nennt man High-Fidelity-Modelle [Dallmeyer, 2014].
Als Kombination beider Ansätze sind mesoskopische Modelle anzusehen. Beispielsweise kann der Verkehr innerhalb eines Teilgebiets mikroskopisch und der Ein- und Ausgangsverkehr aus umliegenden Bereichen makroskopisch simuliert werden [Dallmeyer, 2014].
Stegemann & Ameling [1982] beschreiben schematisch den Aufbau einer Verkehrssimulation (Abbildung 2.1). Unter den Eingabeparametern werden die notwendigen Größen zusammengefasst, um das zu untersuchende Verkehrsnetz nachzubilden. Dazu zählen Knotenpunkte und Verbindungsstrecken (Kanten) zur Beschreibung der Netzstruktur, die Verkehrssituation und verkehrslenkende Maßnahmen. Ausgabeparameter sind Verkehrsgrößen wie Verkehrsstärke, Verkehrsdichte oder Geschwindigkeit, welche die Verkehrsqualität erfassen. Eine Optimierung des Verkehrsablaufs wird durch systematische Änderung der Eingabeparameter erzielt.

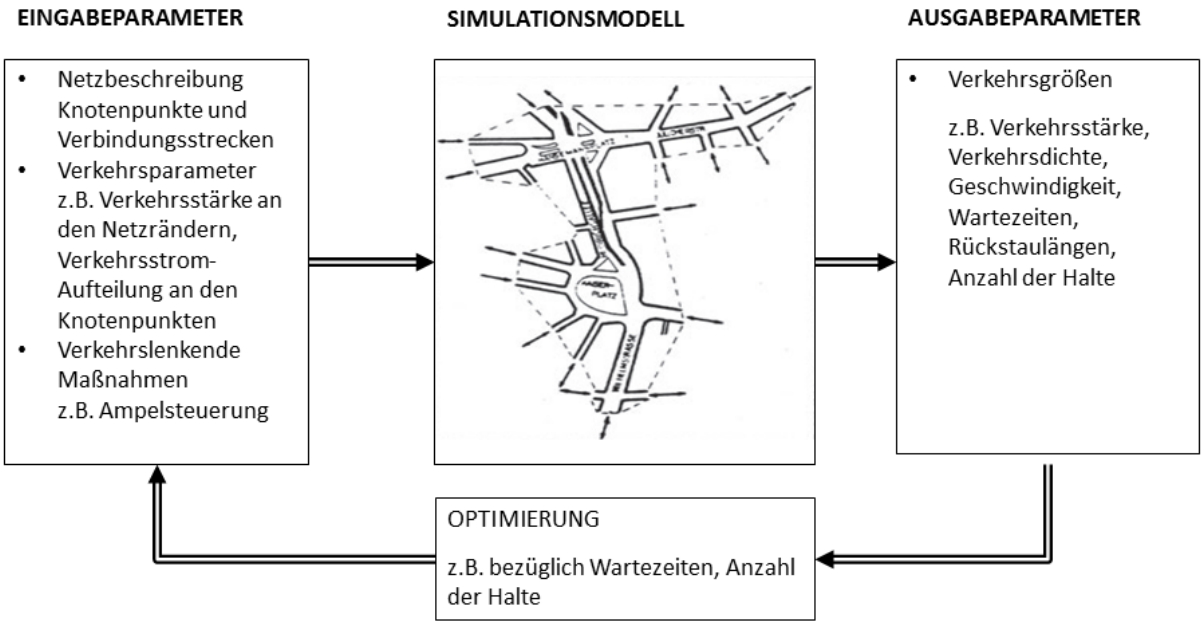


Abbildung 2.1 – Schematischer Aufbau einer Verkehrssimulation [Stegemann & Ameling, 1982, S.476]

2.1.3 Verkehrssimulationssysteme

Die Basis einer Verkehrssimulation ist das Straßennetzwerk in Form eines Simulationsgraphen. Die Modellierung des Verkehrsraumes ist nicht standardisiert, sodass verschiedene Formate existieren. Insbesondere unterschiedliche Verkehrssimulationssysteme besitzen eigene, spezifische Formate. Ein Simulationssystem verwendet (austauschbare) Simulationsmodelle [Dallmeyer, 2014]. Ein Verkehrssimulationssystem ist damit die Umsetzung eines Verkehrsmodells in einer Software. Das Angebot an Systemen zur Durchführung von Verkehrssimulationen ist vielfältig [Hoffmann, 2013]. Einige Simulationssysteme sind frei verfügbar und quelloffen, so zum Beispiel:

- MATSIM (Multi-Agent Transport Simulation)
- SUMO (Simulation of Urban Mobility)

Daneben gibt es auch kommerzielle Software, unter anderem:

- AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks)
- CORSIM (Coridor Simulation)
- PARAMICS (Parallel Microscopic Simulation)
- VISSIM (Verkehr in Städten Simulation)

Verschiedene Systeme sind in erster Linie für bestimmte Teilbereiche der Verkehrsforschung entwickelt [Dallmeyer, 2014]. Je nach Anwendungsbereich und Forschungsfrage weisen Sie demnach Stärken und Schwächen auf. Tabelle 2.1 stellt einige Merkmale der genannten sechs Verkehrssimulationssysteme gegenüber. Neben einer Unterscheidung nach Lizenztyp sind die Modellart, inkludierte Verkehrsteilnehmer-Typen und Schnittstellen (SST) aufgeführt. Alle genannten Systeme unterliegen einem mikroskopischen oder mesoskopischen Modell. AIMSUN und Vissim unterstützen eine Kombination beider Modelle. Die Betrachtung der abgebildeten Verkehrsteilnehmer und implementierter Schnittstellen wirft Unterschiede zwischen kommerziellen und quelloffenen Systemen auf. Erstere zeichnen sich durch eine höhere Komplexität aus, was die Abbildung verschiedener Verkehrsteilnehmer und deren Interaktion angeht (Multimodalität). So beinhalten alle kommerziellen Systeme eine Fußgänger-Modellierung. Vissim differenziert noch weiter in insgesamt vier Verkehrsteilnehmer-Typen Pkw, Lkw, Radfahrer und Fußgänger. Die OpenSource-Systeme sind weniger komplex aufgebaut, punkten dafür mit zahlreichen Schnittstellen (Interoperabilität). Insgesamt bietet jeweils die Hälfte der genannten Simulationen OSM- und/oder GIS-Unterstützung. Mit einer Ausnahme ist der Import bzw. Export von anderen Formaten (anderer Simulationen) möglich. Einer Sonderstellung kommt dabei dem vom Deutschen Zentrum für Luft- und Raumfahrt (DLR) für das System SUMO entwickelten Tool “Netconvert“ zu. Mit dem Importwerkzeug lassen sich Straßennetzwerke in verschiedenen Formaten, unter anderem SUMO, Vissim, MATSIM, OpenDrive, OSM oder Shape, in SUMO importieren. Eine Export-Funktion erlaubt ebenfalls die Ausgabe in verschiedenen Formaten [Behrisch et al., 2011].

	Lizenz		Modell			Verkehrsteilnehmer					SST		
	proprietär	OpenSource	mikro	meso	makro	Pkw	Lkw	Fahrrad	Fußgänger	Interaktion	OSM	GIS	andere Sim.
AIMSUN	✓		✓	✓		✓			✓	✓	✓		
CORSIM	✓		✓			✓	✓		✓	✓			✓
MATSIM		✓		✓		✓					✓	✓	✓
PARAMICS	✓		✓			✓			✓			✓	✓
SUMO		✓	✓			✓	✓				✓	✓	✓
VISSIM	✓		✓	✓		✓	✓	✓	✓	✓			✓

Tabelle 2.1 – Vergleich verschiedener Verkehrssimulationssysteme

Vissim gehört zu den kommerziellen Systemen und wird von der PTV AG in Karlsruhe entwickelt. Mit der Simulation lassen sich sowohl Standardknoten als auch komplexe Kreuzungen modellieren und visualisieren. In Abbildung 2.2 (links) ist ein komplexes Beispiel zu sehen. Knotenpunkte lassen sich mit Vorfahrtsregelungen und Signalanlagen sowie deren Steuerung abbilden und optimieren. Durch das Strecken-Verbinder-Konzept lässt sich ein Maximum an Detailtreue erzielen. Es besteht die Möglichkeit multimodalen Verkehr zu berücksichtigen, das heißt neben motorisiertem Individualverkehr, öffentlichen Verkehr, Güterverkehr, Fußgänger und Radverkehr. Für letztere kann ein nichtspurgebundenes Verhalten modelliert werden. Damit ist Vissim ein umfangreiches Werkzeug für das Verkehrsmanagement mit dem Ziel die Verkehrsqualität zu steigern. Maßnahmen wie verkehrsabhängige Geschwindigkeitsbeschränkungen, Lkw-Überholverbote oder Standstreifenfreigabe können getestet werden. Abbildung 2.2 (rechts) zeigt die genannten Beispiele für Verkehrsmanagementmaßnahmen. Auch für Forschungsbereiche wie Car-2-X Applikationen bietet die Simulation eine Testumgebung [PTV, sa].

Abbildung 2.2 – Modellierung eines komplexen Kreuzungssystems (links) und Visualisierung eines Autobahnabschnitts mit verkehrsabhängiger Geschwindigkeitsregelung (rechts) in Vissim [PTV, sa]

Über verschiedene Schnittstellen können externe Algorithmen an Vissim angebunden werden. Das Eingreifen von externen Programmen in die Simulation wird über die COM-Schnittstelle ermöglicht [Hoffmann, 2013]. Diese bietet Zugriff auf das Straßennetzwerk mit all seinen Attributen, die Signalsteuerung, die Evaluierung, auf Fahrzeuge im Netzwerk und auf die Simulation [Fellendorf & Vortisch, 2010]. Auf innere Komponenten kann jedoch nicht zugegriffen werden und es ist auch nicht möglich, Erweiterungen oder Veränderungen zu implementieren, da Vissim nicht quelloffen ist. Die Ansteuerung erfolgt über Eingabe- und Ausgabedateien [Dallmeyer, 2014]. Vissim verfügt nicht über die Möglichkeit, Graphen für mikroskopische Verkehrssimulationen automatisch aus Eingabedaten abzuleiten. Die Möglichkeit Straßennetze sehr detailliert abzubilden, ist folglich auch mit hohem Modellierungsaufwand verbunden [Dallmeyer, 2014]. Import-Funktionen existieren lediglich für abstrahierte Netzmodelle aus Synchro7, einem Editor zur Erstellung, Modellierung und Optimierung von Signalanlagen der Trafficware Group und aus VISUM (Verkehr in Städten - Umlegungsmodel), dem Verkehrsplanungstool von PTV [PTV, 2016].
Eine Möglichkeit dennoch automatisiert Simulationsgraphen für Vissim zu erzeugen ist, anderweitig bestehende Netzwerkdaten zum Beispiel aus CityGML zu manipulieren und in das Vissim eigene Netz-Format zu überführen. Durch das beschriebene Tool “Netconvert“ könnte ein derart automatisch erzeugtes Straßennetz auch für andere Verkehrssimulationssysteme Anwendung finden.

2.2 CityGML - Internationaler Standard für 3D-Stadtmodelle

Virtuelle 3D-Stadtmodelle werden zunehmend zur Lösung raumbezogener Aufgabenstellungen in verschiedensten Anwendungsbereichen herangezogen. Grundstein für diese Entwicklung bilden Fortschritte in der (semi-)automatischen Datenerfassung, die eine dreidimensionale Aufnahme urbaner oder ländlicher Gebiete ermöglicht. Zur Erzeugung von 3D-Stadtmodellen haben sich verschiedene Verfahren etabliert, mit denen entweder durch Konstruktion oder automatisierte Ableitung Daten in Form eines Stadtmodelles bereitgestellt werden können [InGeoForum, sa].
CityGML ist ein offenes Datenmodel zur Speicherung und zum Austausch virtueller 3D-Stadtmodelle, herausgegeben vom Open Geospatial Consortium (OGC). Die Version 2.0.0 aus dem Jahre 2012 ist ein Anwendungsschema für die Geography Markup Language 3.1.1 (GML3, [Portele, 2007]). Ziel der Entwickler ist, aufbauend auf den Standards der ISO 19100er Gruppe, eine einheitliche Definition der grundlegenden Einheiten, Attribute und Relationen von 3D-Stadtmodellen zu schaffen [Gröger et al., 2012]. Definiert werden die dreidimensionale Geometrie, Topologie, Semantik und das Erscheinungsbild der wichtigsten Elemente im urbanen und ländlichen Raum. Der Fokus liegt auf der Struktur, der Taxonomie und den Beziehungen von semantischen Aspekten eines Stadtmodells [Gröger & Plümer, 2012].

2.2.1 Räumliches Modell

Das räumliche Modell umfasst Bestimmungen zur Geometrie und Topologie. Dabei besteht eine enge Beziehung zum GML3-Standard. So werden räumliche Eigenschaften in CityGML durch eine Untermenge des GML3-Geometriemodells repräsentiert, welches auf der ISO 19107 (“Geographic Information – Spatial Schema“) basiert. Ebenfalls aus GML3 werden Regelungen zur Handhabung von Koordinatenreferenzsystemen (engl.: Coordinate Reference System, kurz: CRS) übernommen. So sollte ein CityGML-Instanzdokument ein CRS für alle enthaltenen Geometrieelemente spezifizieren [Gröger et al., 2012].
Die Basis des GML3 Geometriemodells sind Primitive, aus denen aggregierte (“GM_Aggregates“), komplexe (“GM_Complexes“) und zusammengesetzte (“GM_Composites“) Geometrien gebildet werden können [Portele, 2007]. Primitive können Punkte (0D), Kurven (1D), Oberflächen (2D) und 3D-Quader sein. 3D-Geometrien werden durch die begrenzenden Flächen dargestellt, was unter dem Begriff “Boundary Representation“ (B-Rep) bekannt ist. So ist ein Quader begrenzt durch Flächen, welche wiederum durch Linien beschrieben werden können. In CityGML sind Kurven auf die Geometrieklasse “LineString“ begrenzt, wobei es sich ausschließlich um Geraden handelt. Oberflächen werden durch Polygone repräsentiert [Gröger et al., 2012].
Die Unterschiede zwischen den kombinierten Geometrien zeigt Abbildung 2.3 für den 2D-Fall. Aggregate (Abbildung 2.3 - links) unterliegen keinen topologischen Restriktionen. So können die Flächen, die ein “MultiSurface“ bilden, disjunkt sein, sich überlappen, sich berühren oder unzusammenhängend sein. Ein GM_Complex (Abbildung 2.3 - mittig) ist dagegen topologisch strukturiert. Seine Teile müssen disjunkt sein und dürfen sich nicht überlappen. Erlaubt sind gemeinsame Begrenzungen sowie Bestandteile unterschiedlicher Dimension. Als Spezialfall des GM_Complex gilt das GM_Composite (Abbildung 2.3 - rechts), welches nur Elemente einer Dimension enthalten kann. Die Flächen eines “CompositeSurface“ müssen topologisch an ihren Grenzen verknüpft und ebenfalls disjunkt sein [Gröger et al., 2012]. Bestehende Klassen für Aggregate sind “MultiPoint“, “MultiCurve“, “MultiSurface“ und “MultiSolid“ sowie für Composites “CompositeCurve“, “CompositeSurface“ und “CompositeSolid“ (vgl. [Portele, 2007]).

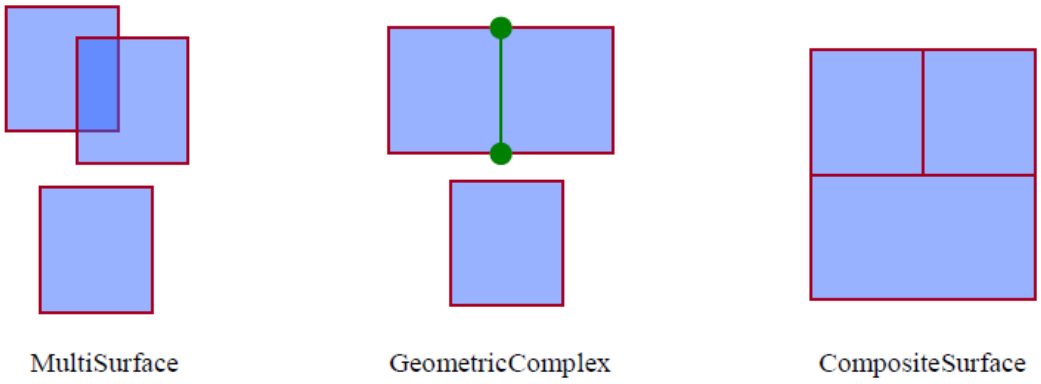


Abbildung 2.3 – GM_Aggregate, GM_Complex und GM_Composite am Beispiel 2D [Gröger et al., 2012, S.26]

Neben der Geometrie umfasst CityGML eine Modellierung der Topologie. CityGML verwendet dabei nicht das Topologiemodell von GML3, welche eine separate Modellierung von Topologie- und Geometrieobjekten vorsieht. Die Umsetzung in CityGML erfolgt über das Konzept der XLinks. Jedes Geometrieobjekt erhält eine eindeutige ID, wodurch es von einer GML-Geometrieklasse referenziert werden kann. Vorteil dieser Implementierung ist eine höhere Flexibilität und Vermeidung von Redundanzen. Nachteilig ist, dass die topologischen Assoziationen nicht zweiseitig gerichtet sind, sondern nur vom Ganzen zu seinen Teilen.

2.2.2 Thematisches Modell

CityGML ist ein Standard für semantische 3D-Stadtmodelle. Neben Geometrie und Topologie stellt somit die Semantik einen zentralen Bestandteil des Standards dar. Auf abstrakter Ebene wird eine thematische Differenzierung von Stadtobjekten vorgenommen. Dazu führt CityGML eine hierarchische Strukturierung ein, die insgesamt dreizehn thematischen Module, unter anderem Gebäude, Tunnel, Brücken, Relief, Wasserkörper, Vegetation und **Verkehr**, unterscheidet. Auf Objektebene wird die Semantik eines Stadtobjektes explizit über eine attributive Beschreibung modelliert. Damit gibt CityGML eine umfangreiche Beschreibung des Raumes und insbesondere auch des Verkehrsraumes [Kolbe, 2009]. Abbildung 2.4 gibt einen Überblick über die hierarchische Struktur in CityGML.

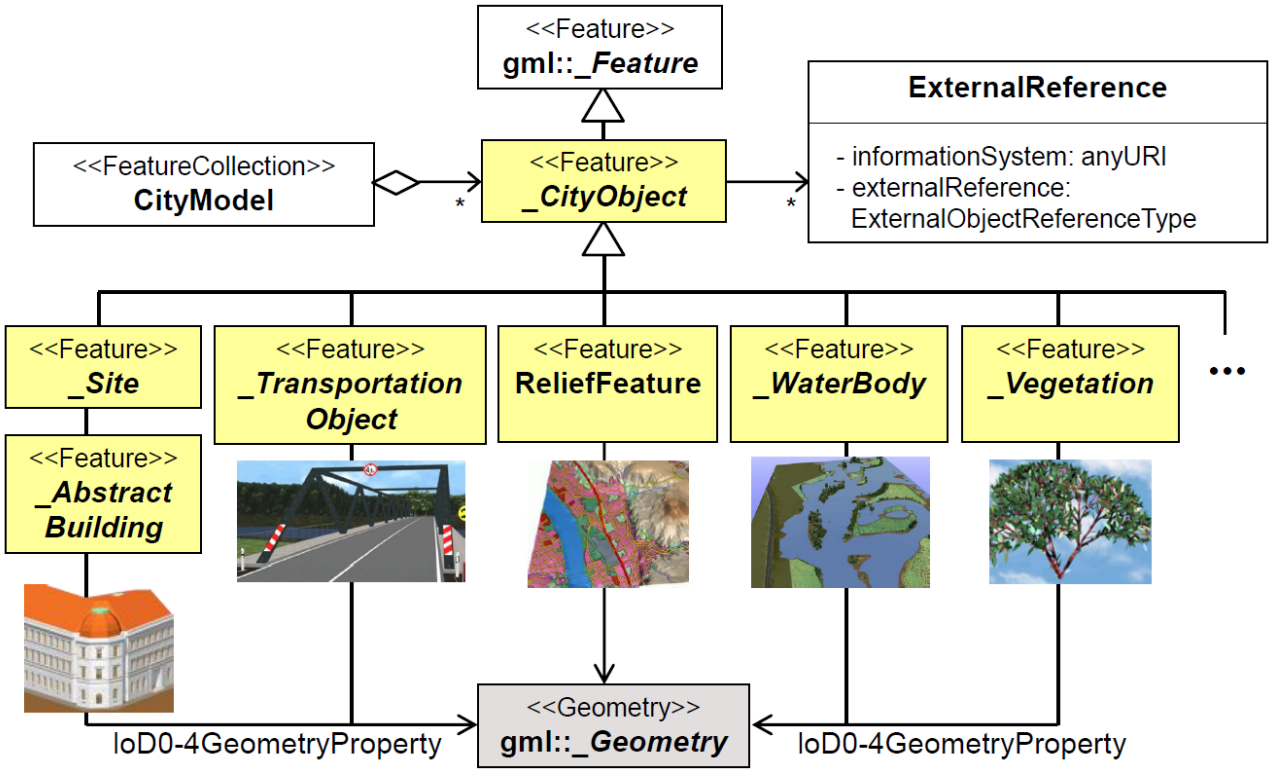


Abbildung 2.4 – UML-Diagramm der Klassenhierarchie in CityGML [Kolbe, 2009, S.19]

Nachfolgend soll für die thematischen Module, die für diese Arbeit von besonderer Relevanz sind, eine genauere Beschreibung gegeben werden.

CityGML Core

Die Basis aller Klassen stellt die abstrakte Klasse “CityObject“ dar, welche eine Unterklasse zur GML Klasse “Feature“ ist und damit die Attribute “name“, “description“ und “gml:id“ erbt. Weiter beinhaltet sie die Attribute “creationDate“ und “terminationDate“ [Gröger et al., 2012]. Über “External-Reference[s]“ kann jedes Stadtobjekt mit Objekten und Daten aus anderen, externen Datensätzen verknüpft werden. Aus Aggregation mehrerer Stadtobjekte entsteht ein “CityModel“ [Kolbe, 2009]. Die Klasse “CityModel“ ist dabei eine Unterklasse der GML Klasse “FeatureCollection“. Zusammengefasst bilden die abstrakten Basisklassen das CityGML “Core“ Modul. Neben den abstrakten Klassendefinitionen beinhaltet es modulübergreifende Kassen und Datentypen [Löwner et al., 2012]. Die verschiedenen thematischen Erweiterungsmodule, wie das Verkehrsmodul "Transportation Objects", sind alle durch ein eigenes XML Schema spezifiziert. Als Unterklasse von “CityObject“ importieren sie das Schema des CityGML “Core“ Moduls.

Transportation Objects

Für die vorliegende Arbeit von besonderem Interesse ist das Verkehrsraummodell von CityGML, dessen Struktur deshalb überblicksweise zusammengefasst werden soll. Eine ausführliche Beschreibung findet sich in Kapitel 4.3.2.
Das Transportation Modell von CityGML als eines der thematischen Erweiterungsmodule beinhaltet thematische und funktionale Aspekte sowie Informationen zur Geometrie und Topologie. Abbildung 2.5 zeigt die Visualisierung einer Szene im städtischen Raum und bezeichnet korrespondierende Objekte des CityGML Transportation Moduls. Wie in allen CityGML-Modulen sind Repräsentationen in verschiedenen Detailierungsstufen LoD 0-4 möglich. Während in LoD 0 der Verkehrsraum als rein lineares Netzwerk dargestellt wird, beginnt mit LoD 1 die geometrische Beschreibung von Verkehrsraumelementen durch Flächengeometrien. Die zentrale Klasse im Verkehrsmodul ist “TransportationComplex“, welche verschiedene thematische Klassen wie “Road“, “Track“, “Railway“ und “Square“ umfasst. Ab LoD 2 wird unterschieden in die Bestandteile “TrafficArea“ (Verkehrsflächen) und “AuxiliaryTrafficArea“ (Nebenverkehrsflächen) [Gröger et al., 2012]. Abbildung 2.5 veranschaulicht die Zusammenhänge beispielhaft für LoD 2-4. Der Verkehrsraum repräsentiert durch ein “Road“-Objekt setzt sich zusammen aus Verkehrsflächen (Straße, Gehwege, Parkplätze, etc.) und Nebenflächen (Randstein, Grünfläche, Verkehrsinsel, etc.).

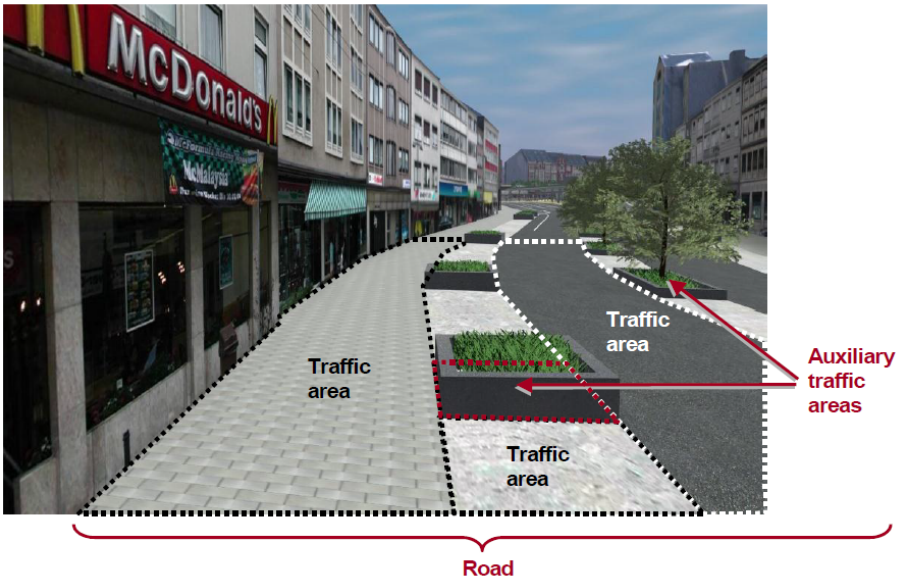


Abbildung 2.5 – LoD2-Modell eines “TransportationComplex“ mit den Bestandteilen “TraficArea“ und “AuxiliaryTrafficArea“ [Gröger et al., 2012, S.124]

Generic City Objects and Attributes

Mit dem Modul “Generics“ stellt CityGML ein Werkzeug zur Verfügung, auch solche Objekte oder Attribute zu speichern oder auszutauschen, die von den bestehenden thematischen Klassen in CityGML nicht abgedeckt werden. Die Realisierung dieser generischen Erweiterungen erfolgt mittels der Klassen “GenericAttribute“ und “GenericCityObject“.
Generische Attribute sind assoziiert mit einem Stadtobjekt. Verfügbare Datentypen sind ’String’, ’Integer’, ’Double’, ’URI’, ’Date’ und ’gml:MeasureType’.
Generische Stadtobjekte können über die Attribute “class“, “function“ und “usage“ verfügen, welche mit den üblichen Codelisten definiert werden. Mit der Funktion wird dabei das thematische Feld (z.B. Transportation) angegeben, in dem das generische Objekt auftritt. Eine Geometrierepräsentation ist sowohl durch explizite als auch durch implizite Geometrien möglich.

Application Domain Extensions

Darüber hinaus erlaubt das Konzept Application Domain Extensions (ADE) anwendungsspezifische Erweiterungen zu erstellen. Im Unterschied zu generischen Objekten und Attributen wird eine ADE in einem extra XML-Schema mit eigenem Namensraum definiert. Dies birgt den Vorteil, dass Erweiterungen formal spezifiziert sind und sich Instanzdokumente gegen die Schemata validieren lassen. Dieser Umstand hilft semantische und syntaktische Interoperabilität zwischen verschiedenen Systemen sicherzustellen. Auf diese Weise lassen sich entweder neue Objekttypen basierend auf bestehenden CityGML Klassen definieren oder existierende Klassen um anwendungsspezifische Eigenschaften erweitern.

2.2.3 Appearance Model

Für Anwendungen, bei denen die Visualisierung im Vordergrund steht, ist das optische Erscheinungsbild von großer Bedeutung. Mit dem Appearance Model werden die sichtbaren Eigenschaften von Objekten wiedergegeben. In CityGML besteht die Möglichkeit, jeder Oberfläche mehrere optische Erscheinungsbilder zuzuweisen. Dies können beispielsweise RGB-Texturen von Häuserfassaden oder auch thematische Informationen wie Messwerte sein [Kolbe, 2009]. Die Möglichkeit Objekten Messwerte zuweisen zu können führt dazu, dass das Appearance Model neben einer reinen Visualisierung auch zu Analysezwecken verwendet werden kann. Für die vorliegende Masterarbeit spielt das Erscheinungsbild hinsichtlich der optischen, möglichst realitätsnahen Darstellung eines Stadtmodells im Kontext der Visualisierung der Simulationsergebnisse eine Rolle.

2.2.4 Anwendungen für CityGML

Die Anwendungsmöglichkeiten von CityGML sind vielfältig. Es kann als Datengrundlage, als Integrationsplattform oder als Ausgangspunkt für fachspezifische Erweiterungen eingesetzt werden [Löwner et al., 2013]. Im Rahmen von baulichen Maßnahmen der Stadtplanung werden 3D-Informationen genutzt. Mit CityGML können Planungen aus unterschiedlichen Quellen in einem Modell integriert und visuell präsentiert werden [Löwner et al., 2013]. In der Umwelt- und Energieplanung können Stadtmodelle als Datengrundlage für Analysen und Simulationen dienen. Beispielsweise werden im Energieatlas Berlin die geometrischen und semantischen Informationen von CityGML genutzt, um energetische Gebäudeeigenschaften und -sanierungspotenziale aufzuzeigen [Löwner et al., 2013]. 3D-Modelle spielen ebenso eine Rolle im Katastrophenschutz, sei es zum Test von Evakuierungsplänen, zur Simulation von Einsatzszenarien in Großstädten oder von Überschwemmungsszenarien [Kolbe, 2009]. Hinzukommen Anwendungen im Bereich Indoor-Navigation, für die es topologischer und geometrischer Informationen ebenso wie einer semantischen Charakterisierung von Objekten bedarf [Gröger & Plümer, 2012]. Auch für Simulationen und die Kartierung von Umgebungslärm können geometrische und semantische Informationen von CityGML genutzt werden. Weitere Anwendungen im Bereich Echtzeit-Simulationen stellen unter anderem Flug- und Fahrsimulatoren [Gröger & Plümer, 2012] sowie Trainingssimulatoren für Einsatzfahrzeuge [Randt et al., 2007] dar. Für viele der genannten Anwendungsfälle wurden fachspezifische Erweiterungen (ADE) von CityGML entwickelt. Über die genannten Anwendungen hinaus sind eine Reihe weiterer Einsatzmöglichkeiten für CityGML denkbar. So sind auch Anwendungen im Bereich Verkehrssimulation basierend auf der Modellierung des Verkehrsraumes vielversprechend. Generell ist eine Stärke von CityGML gegenüber anderen Modellen, dass neben topologischen und geometrischen auch semantische Informationen inbegriffen sind. Nach Löwner et al. [2013] werden die Vorteile von CityGML in Bezug auf semantische Analysen noch nicht komplett ausgeschöpft.

2.3 3D City Database - die CityGML Datenbank

Die 3D City Database (3DCityDB) ist ein Open-Source Programmsystem zur effizienten Speicherung, Verwaltung und Visualisierung von 3D-Stadtmodellen basierend auf dem CityGML-Standard. Die Version 3.3.0 wurde in Zusammenarbeit des Lehrstuhls für Geoinformatik der Technischen Universität München, der virtualcity SYSTEMS GmbH und der M.O.S.S. Computer Grafik System GmbH entwickelt [Kolbe et al., 2016]. Das Datenbankschema resultiert aus einer Abbildung des objektorientierten Datenmodels von CityGML auf die relationale Struktur eines räumlich erweiterten relationalen Datenbankmanagementsystems (engl.: Spatially-Enhanced Relational Database Management System, kurz: SRDBMS). Als SRDBMS werden das kommerzielle System von Oracle und die Open-Source Lösung PostgreSQL in Verbindung mit der räumlichen, ebenfalls kostenfreien Erweiterung PostGIS unterstützt [Kolbe et al., 2016].

Im folgenden Abschnitt wird das Datenmodell und Datenbankdesign der 3DCityDB mit besonderen Augenmerk auf die für diese Arbeit wichtigen Abschnitte, das geometrisch-topologische Modell und die thematischen Module “CityGML Core“, “Transportation Objects“ und “Generic Objects and Attributes“, näher vorgestellt. Im Anschluss wird das in dieser Arbeit verwendete Datenbankmanagementsystem PostgreSQL und dessen räumliche Erweiterung PostGIS beschrieben.

2.3.1 3DCityDB-Datenmodell und Datenbankdesign

Das Datenmodel der 3DCityDB weist einige Vereinfachungen gegenüber dem CityGML-Standard auf. Darunter fallen Änderungen bezüglich der Multiplizitäten von Attributen, Kardinalitäten und Beziehungen zwischen Tabellen und eine vereinfachte Behandlung von rekursiven Strukturen. Zudem werden CityGML spezifische Datentypen angepasst, um eine effizientere Repräsentation in der Datenbank zu erzielen. Insbesondere werden GML-Geometrietypen durch datenbankkonforme Geometrietypen ersetzt. Vorteile eines vereinfachten Schemas sind ein optimierter Workflow sowie eine höhere Effizienz, was die Verarbeitungszeit betrifft [Kolbe et al., 2016].

Datenmodell

Das Geometrisch-topologische Datenmodell wird einer wesentlichen Änderung unterworfen. CityGML nutzt verschiedene GML-Geometrieklassen wie “Surface“ oder “Solid“ in 2D und 3D. In der Datenbank wird die Geometrierepräsentation vereinfacht, indem alle Oberflächen im Sinne der Boundary-Representation durch Polygone gespeichert werden, die sich zu 2D- bzw. 3D-Körpern aggregieren lassen. Abbildung 2.6 zeigt das vereinfachte UML-Diagramm für 2D- und 3D-Geometrien. In der Datenbank werden die Geometrien in einer Tabelle “SURFACE_GEOMETRY“ gespeichert.

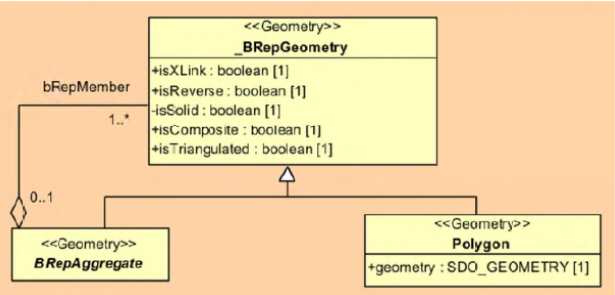


Abbildung 2.6 – Ausschnitt aus dem Geometrisch-Topologisches Modell - Vereinfachungen für 2D- und 3D-Geometrien [Kolbe et al., 2016]

Für den 0D- und 1D-Fall bleibt die Geometriepräsentation gegenüber CityGML unverändert. Auch die Repräsentation der Topologie mit Hilfe des Konzepts der XLinks wird beibehalten.

Das thematische Modell umfasst alle CityGML Module: CityGML Core, Building Model, Bridge Model, City Furniture, Digital Terrain Model, Generic Objects and Attributes, Land Use, Transportation Objects, Tunnel Model, Water Bodies und Vegetation Objects.

Datenbankschema

Nach Vereinfachung des Datenmodells wird selbiges in ein relationales Datenbankschema überführt. Dazu werden eine oder mehrere Klassen eines UML-Diagramms auf eine entsprechende Tabelle abgebildet. Vereinigungen mehrerer Tabellen ergeben sich aus den Klassenrelationen. Für bestimmte n:m Beziehungen müssen zusätzliche Tabellen eingeführt werden. Weiter werden die Datentypen der Attribute entsprechend der Datentypen des Datenbanksystems (Oracle bzw. PostgreSQL) angepasst.

- Core Model [Kolbe et al., 2016, S.57-59]
  Alle Stadtobjekte liegen in der Tabelle “CITYOBJECT“ repräsentiert durch Tupel. Die Spalten der Tabelle entsprechen den Attributen der UML-Klasse “_CityObject“. Zusätzliche Spalten beinhalten Metadaten. Zur eindeutigen Identifikation eines Stadtobjektes steht eine Spalte mit der gml:id sowie zusätzlich eine Spalte “GMLID_CODESPACE“ mit dem vollen Pfad eines importieren Objektes zur Verfügung. Weiter werden der oder die Namen gespeichert. Eine Mehrfachbelegung des Attributs gml:name wird mit Hilfe eines Trenn-Strings der Form ’–/\–’ beschrieben. In der Spalte gml:Envelope ist die Bounding Box - definiert durch fünf Punkte - abgelegt. Die “OBJECTCLASS_ID“ liefert Information über die Objektklasse eines Stadtobjekts. Dies hilft die Tabellen der jeweiligen Subklasse zu bestimmen.

- Geometrie und Topologie [Kolbe et al., 2016, S.60-66]
  Die Tabelle “SURFACE_GEOMETRIE“ enthält drei Attribute zur Speicherung von Geometrien: “GEOMETRY“ umfasst alle oberflächenbasierten Geometrien, “IMPLICIT_GEOMETRY“ implizite Geometrien und “SOLID_GEOMETRY“ Volumenkörper. Die Generalisierung aus Abbildung 2.6 ist über die Attribute “PARENT_ID“ und “ROOT_ID“ realisiert. Aggregierten Geometrien wird demnach keine explizite Geometrie zugewiesen. Stattdessen sind sie mit ihren Bestandteilen über die “PARENT_ID“ und “ROOT_ID“ referenziert.
  Das Attribut “IS_XLINK“ ist die Umsetzung des Konzepts XLinks mit dessen Hilfe in CityGML die Topologie modelliert wird. Die Referenzierung erfolgt über die “gml:id“.

- Generic Objects and Attributes [Kolbe et al., 2016, S.81-82]
  3D-Objekte, die nicht von den thematischen Modulen in CityGML abgedeckt werden, können in der Tabelle “GENERIC_CITYOBJECT“ beschrieben werden. Möglich ist eine explizite oder implizite Geometrierepräsentation.
  Generische Attribute werden in der Tabelle “CITYOBJECT_GENERICATTRIB“ gehalten. Differenziert nach Datentyp werden die Attributwerte in den Spalten “STRVAL“, “INTVAL“, “REALVAL“, “URIVAL“ und “DATEVAL“ gespeichert. Über den Fremdschlüssel “CITYOBJECT_ID“ erfolgt die Verknüpfung mit dem zugehörigen Stadtobjekt.

- Transportation Model [Kolbe et al., 2016, S.83-84]
  Das Transportation Modell setzt sich aus den Tabellen “TRAFFIC_AREA“ und “TRANSPORTATION_COMPLEX“ zusammen. Die Abbildung der UML-Attribute erfolgt nahezu 1:1. Die Aggregationsbeziehungen zwischen “TransportationComplex“ und seinen Bestandteilen wird im Fremdschlüssel “TRANSPORTATION_COMPLEX_ID“ gespeichert. Der Fremdschlüssel “OBJECTCLASS_ID“ gibt an, ob es sich um eine “TrafficArea“ oder eine “AuxillaryTrafficArea“ handelt. Das Attribut “LOD0_NETWORK“ steht für die linienhafte Repräsentation eines Verkehrskomplexes zur Verfügung. Höhere LoD-Stufen werden über Fremdschlüssel mit der Bezeichnung “LODx_MULTI_SURFACE_ID“ referenziert. Das Datenbankschema für das Verkehrsmodell findet sich im Anhang A.9.

2.3.2 PostgreSQL

PostgreSQL ist neben Oracle eines der Datenbankmanagementsysteme (DBMS), das die 3DCityDB unterstützt. Allgemein ist ein DBMS ein Programmsystem, das für die “Verwaltung der zugreifenden Menschen und Prozesse, für deren Zugriffsrechte, für die Interpretation von deren Anweisungen und die Bereitstellung der Daten“ [Fischer & Hofer, 2008, S.199] verantwortlich ist. Man unterscheidet hierarchische, relationale und objektorientierte DBMS. PostgreSQL ist ein objekt-relationales Datenbankmanagementsystem (ORDBMS). Ein ORDBMS hat den gleichen Funktionsumfang wie ein relationales DBMS mit dem Unterschied, dass es auf einem objekt-orientiertem Modell basiert.

Ein Merkmal davon ist, dass objektorientierte Konzepte wie die Vererbung eingeschlossen sind. PostgreSQL unterstützt Einfach- und Mehrfachvererbung [PostgreSQL, sa].

Im Sinne des Client-Server-Modells besteht PostgreSQL aus den beiden Prozessen:

- Server:
  Der Server verwaltet Daten, ermöglicht Clients eine Verbindung zur Datenbank aufzubauen und führt Aktionen auf der Datenbank aus. Im Falle von PostgreSQL nennt sich das Server Programm POSTGRES (Version 4.2), welches an der University of Carolina entwickelt wurde.
- Client:
  Die Nutzer-Komponente, der sogenannte Client, ist eine Anwendung zur Durchführung von Datenbankanfragen. Client-Tools können textbasiert, Web-Browser oder grafische Benutzeroberflächen sein. Im Falle der 3DCityDB stellt der "Importer/Exporter" eine grafische Oberfläche zur Verfügung, um ausgewählte Anfragen zum Import und Export von 3D-Stadtmodellen an die Datenbank zu stellen.

Unterstützt werden große Teile des SQL-Standards. Zudem ist es dank der freien Lizenz jedermann möglich, PostgreSQL um Datentypen, Funktionen, Operatoren und andere Aspekte zu erweitern. Weiter verfügt PostgreSQL über Schnittstellen zu diversen Programmiersprachen, so u.a. Java (JDBC), Python, C, C++ oder PHP [PostgreSQL, 2016].

2.3.3 PostGIS

PostGIS ist eine Erweiterung um räumliche Aspekte für das Datenbankmanagementsystem PostgreSQL. Neben der Möglichkeit GIS-Objekte direkt in der Datenbank zu speichern, werden räumliche Indexe sowie Funktionen für die Analyse und Verarbeitung räumlicher Daten bereitgestellt. Wie PostgreSQL ist PostGIS Open Source [PostGIS Project Steering Committee, 2016].
PostGIS stellt in verschiedenen Bibliotheken Geometrie-Funktionen zur Verfügung. Einen Überblick inklusive kurzer Beschreibung zu den in der vorliegenden Arbeit verwendeten PostGIS Funktionen gibt Anhang C.1. Eine Funktion ("ST_ApproximateMedialAxis") ist Teil der SFCGAL C++ Wrapper Library, die erweiterte 2D und 3D-Funktionen umfasst. Vor Verwendung der zusätzlichen Funktion ist die Bibliothek zu laden.

2.4 3D City Database Importer/Exporter

Der 3DCityDB Importer/Exporter wird gemeinsam durch die drei Kooperationspartner der 3D City Database entwickelt. Die kostenfreie Software unterliegt der Apache License (Version 2.0). Die grafische Benutzeroberfläche ermöglicht Nutzern, auf einfache Weise 3D-Stadtmodelle in die 3DCityDB zu laden oder Daten aus der Datenbank zu exportieren.

Der Importer/Exporter ist das zentrale Werkzeug der vorliegenden Arbeit. Zum einen werden im Laufe des Workflows bestehende Funktionen des Tools wie die Import-Funktion von CityGML-Stadtmodellen genutzt. Zum anderen stellt es die Umgebung für neue Funktionen bezüglich der Kopplung von Verkehrssimulationen mit 3D-Stadtmodellen dar, welche im Kontext dieser Arbeit entwickelt wurden. Die folgenden Abschnitte geben einen Überblick über bestehende Funktionen des Tools und Möglichkeiten zur Erweiterung in Form von Plugins.

2.4.1 Funktionen und Werkzeuge

Im Einzelnen umfasst das Java-basierte Tool Operationen zur Verwaltung und zum Aufbau einer Datenbankverbindung, zum Import und Export von Modellen im CityGML-Format sowie zum Export von Daten im KML, COLLADA oder glTF Format.
Um eine Datenbankverbindung herzustellen, ist die Eingabe der Datenbankverbindungsdetails erforderlich. Grundsätzlich werden Verbindungen zu Oracle und PostgreSQL Datenbanken unterstützt. Über ein Dropdown-Menü lassen sich mehrere Datenbankverbindungen verwalten.
Neben der Verwaltung von Verbindungen stehen bestimmte Datenbankoperationen zur Verfügung. Die Liste aller beinhalteten Tabellen einer Datenbank lässt sich als Datenbankbericht im Konsolenfenster des Importer/Exporter ausgeben. Eine 2D Bounding Box von Stadtobjekten kann gespeichert oder aktualisiert werden. Die Bounding Box enthält die Koordinaten der linken unteren sowie der rechten oberen Ecke der Ausdehnung eines oder mehrerer Stadtobjekte. Diese Informationen sind beispielsweise für räumliche Filter während eines Import- oder Exportprozesses hilfreich.
Die Verwendung von räumlichen und normalen Datenbankindexen kann manuell selektiert werden. Allgemein sind Indexe eine separate Hilfsdatenstruktur in sortierter Form mit dem Zweck, eine Datenabfrage zu beschleunigen [Fischer & Hofer, 2008]. Räumliche Indexe liegen der Geometrie-Spalte zugrunde, andere Indexe können für Spalten mit beliebigem Datentyp aktiviert werden.
Die Version 3.3.1 des Importer/Exporter unterstützt den Import von CityGML-Modellen der Versionen 2.0.0, 1.0.0 und 0.4.0 des CityGML-Standards. Für den Importprozess stehen ein einfacher Filter über das Attribut “gml:id“ sowie verschiedene komplexe Filter zur Verfügung. Zu den komplexen Filtern zählen ein Filter nach dem Attribut “gml:name“, ein Zählerfilter, der eine Untermenge der Features importiert, ein Bounding Box Filter sowie ein Feature-Klassen Filter. Mit letzterem lassen sich nur Objekte einer oder mehrere Klassen wie Gebäude, Transportation Objects, etc. importieren. Die zum Import ausgewählte CityGML-Datei kann vor dem Import auf Validität überprüft werden. Diese Option ist sinnvoll, um Fehlern während späterer Datenbankoperationen vorzubeugen.
3D-Stadtmodelle können komplett oder in Teilen aus der Datenbank im CityGML-Format exportiert werden. Für den Export bestimmter Untermengen des Stadtmodells stehen die gleichen thematischen und räumlichen Filter wie im Import-Dialog zur Auswahl. Als weitere Export-Formate sind KML, COLLADA und glTF verfügbar. Damit können die 3D-Stadtmodelle zur Visualisierung in Anwendungen wie Google Earth, ArcGIS Explorer oder Cesium verwendet werden.

2.4.2 Plugins

Der Importer/Exporter kann um Plugins erweitert werden. Mit der Standardsoftware lässt sich gegenwärtig ein Plugin, der Spreadsheet Generator, installieren, der den Importer/Exporter um die Möglichkeit erweitert, Daten aus der Datenbank in CSV oder Microsoft Excel Dateien zu exportieren. Diese lassen sich in einer Tabellenkalkulationssoftware öffnen, in der der gesamte Funktionsumfang einer Tabellenkalkulation genutzt werden kann.

Für die Entwicklung eigener Plugins ist eine java-basierte Plugin API verfügbar. Die Plugin API stellt eine Service-Definition dar, neu erstellte Plugins sind dementsprechend Service-Provider. Ein Plugin kann eine oder mehrere der folgenden Erweiterungen beinhalten [Nagel, 2011]:

- View Extension
- Preference Extension
- Menu Bar Extension
- Config File Extension

Mit der “View Extension“ kann ein neuer Reiter dem Importer/Exporter hinzugefügt werden. Das Menü zur Speicherung von Voreinstellungen kann über eine “Preference Extension“ erweitert werden. Mit der "Menu Bar Extension“ lassen sich Plugin-Funktionen in die Menüleiste integrieren. Möchte man spezifische Einstellungen in der Hauptkonfigurationsdatei speichern, lässt sich dies mit der “Config File Extension“ erreichen. Mindestens eine Erweiterung sowie das “Main Service Interface“ müssen zwingend von einem neuen Plugin implementiert werden.
Die API stellt Entwicklern eine Reihe von Features zur Verfügung, so unter anderem multi-thread Programmierung, Event Notification mittels “Global Message Bus“ oder Kontroller mit Zugang zu Kernfunktionalitäten bezüglich GUI, Datenbank oder Protokollierung.

2.5 Virtual Globes - Google Earth und Cesium

Virtuelle Globen (engl.: Virtual Globes) sind dreidimensionale Software-Modelle, welche Planeten abbilden. Sie zeichnen sich unter anderem durch die Fähigkeit aus, große Datenmengen wie Gelände-, Bild- oder Vektordaten wiederzugeben und ermöglichen dem Nutzer derlei Daten interaktiv zu erkunden. So lässt sich beispielsweise die ganze Erde als virtueller Globus oder Teilbereiche aus Flugperspektive sowie auf Straßenebene in beliebigen Zoom-Stufen betrachten.
Der wohl bekannteste und am weitesten verbreitete Virtual Globe ist Google Earth. Eine der Gründe für den Erfolg von Google Earth ist die Möglichkeit, eigene Daten in Google Earth anzeigen zu lassen. Mit Hilfe der Keyhole Markup Language (KML), welche 2008 zu einem OGC Standard wurde, können Nutzer raumbezogene Daten visualisieren und lokalisieren sowie durch Objekte navigieren

[Bailey & Chen, 2011]. Google stellte über sechs Jahre eine JavaScript basierte Schnittstelle zur Verfügung, welche es Entwicklern erlaubte, umfangreiche 3D-Kartenanwendungen für Webbrowser zu entwickeln. Im Jahre 2015 wurde die Unterstützung der Google Earth API aus sicherheitstechnologischen Gründen jedoch eingestellt (vgl. Google Maps APIs Blog[1]).
Eine Alternative für webbasierte Virtual Globe Anwendungen ist das Open Source Software-Paket Cesium. Die Cesium API stellt eine JavaScript Bibliothek zur browserbasierten Visualisierung von 3D-Inhalten in Form digitaler Globen oder als 2D-Kartenprojektion zur Verfügung. Die Architektur von Cesium setzt sich aus mehreren Abstraktionsebenen zusammen, welche aufeinander aufsetzen. Die jeweils nächst höhere Ebene fügt weitere Funktionalitäten hinzu und hebt dadurch gleichzeitig das Abstraktionslevel. Die niedrigste Ebene ist der “Core Layer“, welcher vor allem mathematische Funktionen wie Matrizen, Vektoren und Quaternionen sowie Koordinatentransformationen oder Projektionen enthält [Analytical Graphics Inc., 2015b]. Im Zentrum der graphischen Darstellung steht der “Renderer“, eine Cesium spezifische WebGL Engine [Analytical Graphics Inc., 2015a]. Web Graphics Library (WebGL) ist ein lizenzfreier Standard zum Rendern interaktiver 2D und 3D Grafiken im Web-Browser ohne den Einsatz zusätzlicher Plugins. Entwickelt von der Khronos Group und Mozilla wird WebGL von aktuellen Webbrowsern wie Firefox oder Google Chrome unterstützt [Mozilla Developer Network, 2017]. Die Bündelung von WebGL Aufrufen in Cesium mittels des “Renderer“ Moduls hat verschiedene Vorteile wie beispielsweise eine Optimierung der Performance oder eine weniger fehleranfällige Ausführung aller Cesium Inhalte [Analytical Graphics Inc., 2015a]. Der “Renderer“ wird von der nächst höheren Abstraktionsebene, dem “Scene“ Layer genutzt, um WebGL Ressourcen von Cesium Primitiven wie dem Globus oder 3D-Objekten zu erzeugen und um damit einhergehende Befehle zum Aufbau eines Frames auszuführen. Die “Scene“ repräsentiert alle graphischen Objekte und beschreibt damit den Zustand der Zeichenfläche (“Canvas“). In Cesium kann eine Szene in 3D, 2D oder in Form des Columbus View, einer 2.5D-Ansicht, vorliegen. Eine Szene setzt sich zusammen aus Cesium Primitiven (Geometrien, 3D-Objekten, Bildlayer, ...), deren graphischem Erscheinungsbild, Animationen sowie aus Kamera-Optionen zur Einstellung der Ansicht. Auf höchster Abstraktionsebene steht die "Dynamic Szene", welche durch ein JSON Schema zur datengetriebenen Visualisierung beschrieben wird. Dieses Schema, bekannt als Cesium Language (CZML), beschreibt in präziser Form Daten im Laufe der Zeit. CZML ist streambar und optimiert für Clients. Das Format lässt sich einfach lesen und schreiben und ist erweiterbar. Neben CZML unterstützt Cesium andere Datenformate. So lassen sich JSON, ESRI Shapefiles und KML-Dateien prozessieren [Analytical Graphics Inc., 2015b]. Auch die Visualisierungsoberfläche der 3DCityDB, der 3DCityDB-Web-Map-Client, basiert auf dem Cesium Virtual Globe. Zahlreiche Erweiterungen hinsichtlich des Kontextes 3D-Stadtmodelle wurden vorgenommen. So können beispielsweise Stadtmodelle, die mit dem Importer/Exporter im KML respektive glTF Format exportiert wurden, direkt im Webbrowser visualisiert werden. Auf diese Option wird im Rahmen dieser Arbeit bei der Visualisierung von Simulationsergebnissen im 3D-Stadtmodell zurückgegriffen. Eine Visualisierung in Google Earth ist ebenfalls möglich.

[1] https://maps-apis.googleblog.com/2014/12/announcing-deprecation-of-google-earth.html (Zugriff: 01.06.2017)

2.6 Zusammenstellung und Kontext

Die vorangegangenen Abschnitte bilden das theoretische Gerüst, auf dem die Umsetzung der Ziele der Masterarbeit basiert, einen Simulationsgraphen aus CityGML-Daten zu generieren und die Ergebnisse an die Simulationsdurchführung anschließend in einem Stadtmodell zu visualisieren. Abbildung 2.7 gibt die Zusammenhänge zwischen den einzelnen Komponenten im Kontext des Realisierungsablaufs wieder. Im Mittelpunkt steht die Kopplung von 3D-Stadtmodell (links) und Verkehrssimulation (rechts). Die linke Seite stützt sich auf CityGML, einem internationalen Standard für 3D-Stadtmodelle. Als Verkehrssimulation findet die kommerzielle Software PTV Vissim Anwendung. Die Kopplung erfolgt in der Umgebung der 3DCityDB, der Datenbank-Lösung für 3D-Stadtmodelle im CityGML Format.

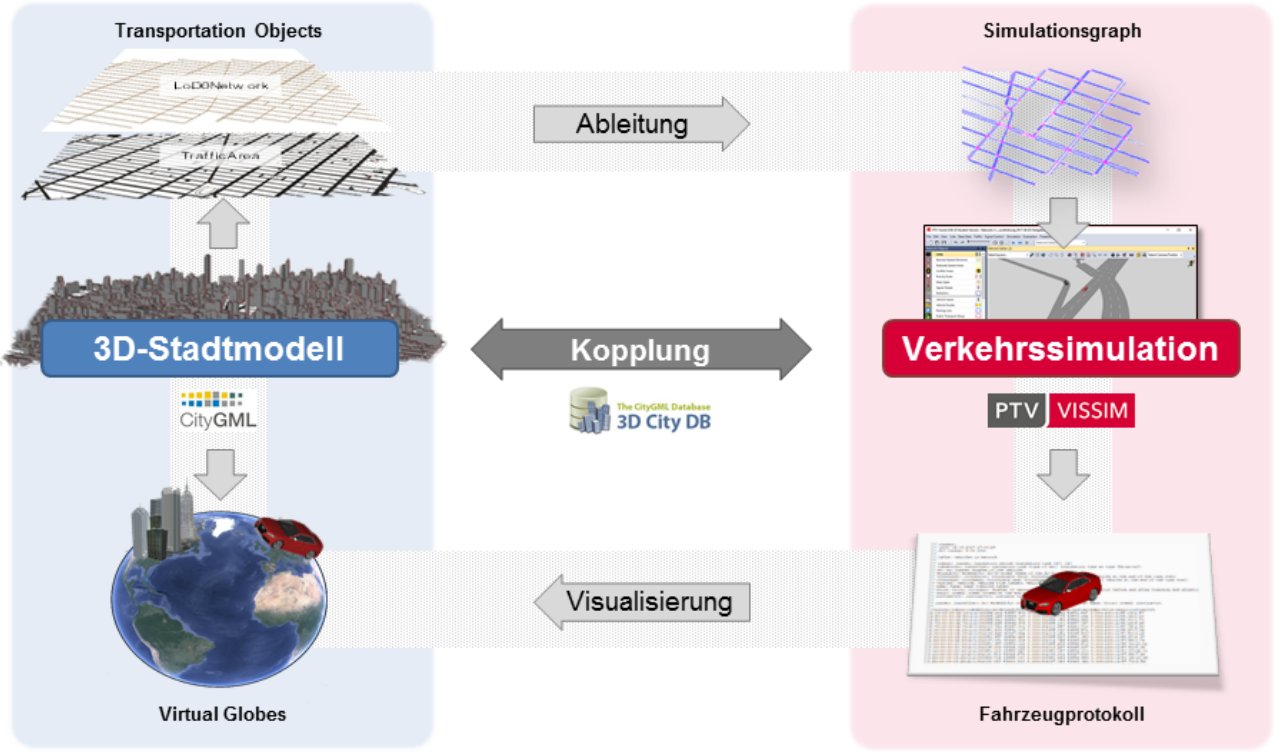


Abbildung 2.7 – Zusammenstellung der Grundsteine der Arbeit und ihrer Assoziationen zueinander unter Bezug zum Workflow

Der Ablauf, auf dessen Umsetzung in den nachfolgenden Kapiteln näher eingegangen wird, lässt sich nach Abbildung 2.7 folgendermaßen zusammenfassen: Ausgangspunkt ist ein 3D-Stadtmodell im CityGML Format. Für die Ableitung eines Simulationsgraphen für Verkehrssimulationen ist das Verkehrsmodul “Transportation Objects“ (oben links) des CityGML Standards relevant. Die Ableitung erfolgt entweder aus linienhaften oder flächenhaften Ausgangsdaten und beinhaltet eine Formatüberführung des Verkehrsnetzes von CityGML auf das Format der Verkehrssimulation Vissim. Dazu werden Funktionen und Werkzeuge der Datenbank verwendet, um die notwendigen Daten abzurufen und auf gewünschte Weise zu manipulieren. Resultat der Ableitung ist ein Simulationsgraph (oben

rechts), auf dessen Grundlage in Vissim eine Verkehrssimulation durchgeführt werden kann. Die Ergebnisse der Verkehrssimulation können als Fahrzeugprotokoll ausgegeben werden (unten rechts). Darin enthalten sind die Fahrzeugbewegungen über den Simulationszeitraum, welche die Grundlage für eine grafische Darstellung bilden. Als Plattform für eine Visualisierung der Simulationsergebnisse im 3D-Stadtmodell dienen die virtuellen Globen Cesium (bzw. der darauf basierende 3DCityDB-Web-Map-Client) oder Google Earth.

3 Relevante Forschungsarbeiten

Klassischer Weise ist der Modellierungsaufwand zur Erstellung von Simulationsgraphen sowie korrespondierender Umgebungsmodelle groß. Zur Reduzierung von selbigem gibt es in der Literatur verschiedene Ansätze. Eine Variante ist bestehende Geodaten oder GIS heranzuziehen, um daraus Netzgraphen abzuleiten und Umgebungsmodelle aufzubauen. Im ersten Teil des Kapitels werden Ansätze vorgestellt, die automatisch oder semi-automatisch Straßennetzwerke aus Geodaten oder anderen Quelldaten und Bedingungen generieren. Der zweite Teil widmet sich verschiedenen Konzepten zur Verknüpfung von Verkehrs- und Fahrsimulationen mit Geodaten und GIS. Der letzte Abschnitt der Literaturrecherche befasst sich mit der Integration dynamischer Prozesse in GIS, was für die vorliegende Arbeit in Hinblick auf eine Visualisierung dynamischer Ergebnisse einer Verkehrssimulation von Relevanz ist.

3.1 Konzepte zur (semi-)automatischen Erzeugung von Straßennetzwerken

Haubrich et al. [2013] und Haubrich et al. [2014] stellen einen strukturierten Workflow zur Erstellung von 3D-Szenarien für Verkehrssimulatoren vor. Dieser sieht vor aus OSM-Daten eine Beschreibung des Straßennetzwerkes im OpenDrive-Format zu generieren, um daraus automatisch einen Simulationsgraphen zu erzeugen [Haubrich et al., 2013]. In Haubrich et al. [2014] wird neben existierenden Formaten ein neues Modell zur Beschreibung von semantischen Straßenmodellen diskutiert. Das Modell ist angelehnt an die Straßenraummodellierung in der Verkehrssimulation Vissim und besteht im Wesentlichen aus einer Menge von Fahrspuren, Konnektoren, Pfaden, Informationen und Informationsknoten sowie aus einer Menge an Kreuzungen. Der Prozess zur automatischen Erstellung eines Straßennetzwerkes setzt sich aus vier Schritten zusammen. Nach Erstellung der Netzwerkelemente erfolgt die Verknüpfung einzelner Komponenten. Auf Basis dieses verknüpften Netzwerkes werden Wegpunkte entlang der Netzwerkpfade und Kreuzungen einschließlich der zugehörigen Konnektoren generiert. Im letzten Schritt werden zusätzliche, verkehrsrelevante Informationen jeglicher Art in das Netz integriert. Das Konzept wird an drei verschiedenen Szenarien evaluiert, um anschließend Vorteile und bestehende Einschränkungen zu diskutieren. Als größte Limitation werden Inkonsistenzen in den Ausgangsdaten benannt [Haubrich et al., 2014].

Applegate [2013] beschäftigt sich in seiner Doktorarbeit mit dem Design und der Simulation von Straßennetzwerken in virtueller Umgebung. Ein wichtiger Bestandteil ist die (semi-)automatische Erzeugung, Gestaltung und Visualisierung von Straßennetzwerken. Betrachtet werden zwei Varianten, die eine basierend auf digitalen Straßenkarten, die andere auf Skizzen in Verbindung mit bestimmten Randbedingungen wie der Krümmung oder Steigung. Aus einem Graphen bestehend aus Knoten und Kanten werden zunächst Kreuzungen und Straßensegmente identifiziert, um das Netzwerk anschließend in Zellen aufzuteilen, welche während der Simulation als Wegpunkte dienen. Kreuzungen werden als Knotenpunkte mit mindestens drei ein- bzw. ausgehenden Kanten definiert. Straßen bestehen aus einer Folge an Kanten. Ob eine Straße an einem Knoten endet oder weiterführt, wird anhand des Skalarproduktes zweier normalisierter Kantenvektoren an einer Kreuzung entschieden. Nach Applegate [2013] benötigen Ansätze zur Generierung von Straßennetzwerken, denen eine prozeduale Modellierung zugrunde liegt, ein hohes Maß an Verfeinerung. Skizzen-basierte Systeme verfügen demnach über mehr Einflussmöglichkeiten bereits während des Designprozesses. Der vorgestellte Algorithmus zur Skizzen-basierten Netzwerkgenerierung besteht aus insgesamt sieben Schritten, u.a einer Abtastung der Skizze, der Identifizierung von Randbedingungen und der Konstruktion der Straßengeometrie.
Campos et al. [2015] zeigen im Kontext der automatischen Erzeugung von Straßennetzwerken verschiedene Alternativen zur Beschreibung des Straßenverlaufs auf. Einfluss auf die Erstellung haben Landschaft bzw. Umgebung sowie geotechnische Faktoren und Verkehrsaspekte. Hohe Bedeutung wird der Interaktion während des Erzeugungsprozesses beigemessen. Ausgehend von Knoteninformationen wird basierend auf festgelegten Randbedingungen und in Bezug zum Terrain der optimale Pfad gesucht. Weitere Aspekte des Konzeptes sind die Erzeugung einer korrespondierenden 3D-Straßenumgebung sowie einer semantischen Beschreibung des Straßennetzwerkes und seiner Bestandteile. Implementiert wurde das Konzept für ein Beispiel in Portugal anhand einer Landstraße sowie an einer Autobahn.

3.2 Verknüpfung von Verkehrs- bzw. Fahrsimulation mit Geodaten

In der Verknüpfung von Verkehrs- bzw. Fahrsimulationen mit Geodaten wird großes Potenzial gesehen. Zum einen kann durch automatische Generierung von Simulationsgraphen aus Karten oder GIS der Modellierungsaufwand reduziert werden. Zum anderen helfen Geodaten ein detailliertes Umgebungsmodell aufzubauen, welches Grundlage einer realitätsnahen, visuellen Wiedergabe ist. Jedoch sind GIS-Daten im Allgemeinen nicht dafür gedacht für Simulationen verwendet zu werden [Wilkie et al., 2012], sodass es notwendig ist die Daten entsprechend anzupassen.

3.2.1 Digitale Karten und GIS als Datenquelle für Simulationen

[Dallmeyer, 2014] verwendet OpenStreetMap-Daten (OSM-Daten) zur Generierung eines Simulationsgraphen. Als Vorteil dieses Ansatzes wird die kostengünstige Verfügbarkeit bei meist hoher Quali-

tät genannt. Dennoch wird der Unterschied zu simulationsoptimierten Karten betont. Für High-Fidelity Modelle ist OSM als Datenquelle demnach nicht geeignet, da Details fehlen, die manuell ergänzt werden müssen. Die Vorgehensweise lässt sich wie folgt zusammenfassen: Die OSM-Daten werden nach den enthaltenen Informationen gefiltert und in Layer aufgeteilt. Aus Linien wird ein Graph erstellt, der Knoten und Kanten enthält. Dieser wird anschließend mit Informationen angereichert und zu einem “ExtendedGraph“ weiter verarbeitet, der für Simulationen geeignet ist.

Nach Richter et al. [2016] ist es notwendig Geodaten aus verschiedenen Quellen zu nutzen, um ein präzises Straßenmodell erzeugen zu können. OpenStreetMap Daten stellen die beliebteste Quelle dar, weil die Daten kostenlos und praktisch weltweit verfügbar sind. Allerdings ist OSM oftmals fehlerhaft und die Informationen daraus unzureichend. Als Grundlage-Daten sehen Richter et al. [2016] vielmehr Katasterdaten. Das Konzept sieht vor - aufbauend auf dem Kataster - eine geometrische Aufbereitung durchzuführen und anschließend Zusatzinformationen aus OSM, Navigationsdaten oder anderen Quellen wie Luftbildern oder Laserscanning zu integrieren. Als Referenz zur Datenfusion dienen die Straßenmittelachsen. Geometrisch ungeeignete Kreuzungsmodellierungen werden durch realistische Kreisbogen-, Spline- oder Klothoidenverbindungen ersetzt. Anzahl und Breite der Fahrspuren können durch regelbasierte Interpolation bestimmt werden. Logische Verknüpfungen der Fahrspuren an Kreuzungen werden generisch eingefügt und können mit OSM oder anderen Metadaten überprüft und korrigiert werden.

Haunert et al. [2005] beschreiben wie Geoinformationssysteme genutzt werden können, um Fahrsimulatorszenen zu definieren, zu visualisieren und zu verwalten. Das Umgebungsmodell wird dabei in der GIS-Datenbank abgelegt. Eine Grundlage können bestehende Straßennetze sein, die über eine Georeferenzierung verfügen. Objekte wie Gebäude und Verkehrsmobiliar können gleichermaßen hinzugefügt werden. Aus dem GIS-Datenbestand wird eine Simulationsszene abgeleitet. Dabei erfolgt eine automatische Konvertierung der Straßenpolylinien in eine Routenbeschreibung bestehend aus Geraden, Klothoiden und Kreisbögen wie sie für einen Fahrsimulator benötigt werden. Objekte entlang der Route werden an der korrekten Position eingesetzt. Die resultierende Szene kann direkt in die Fahrsimulation geladen werden.

Wilkie et al. [2012] präsentieren eine Methode zur Transformation von GIS-Daten in eine topologische und geometrische Repräsentation, die für den Einsatz in Verkehrssimulationen geeignet ist. Das System verwendet ein Straßennetzwerk aus einem GIS als Eingabe. In einem ersten Schritt, der topologischen Phase, wird aus den Netzwerkinformationen ein Graph generiert. Häufig auftretende Fehler wie übereinstimmende Knoten werden durch Filter entfernt. Straßen werden als Kante mit je einem Knoten an beiden Enden definiert. Es wird sichergestellt, dass die Schnittstellen zwischen den einzelnen Fahrspuren korrekt wiedergegeben werden. Dazu zählen einfache Kreuzungen, Einfädelzonen, genauso wie Auffahrten, sowie Über- und Unterführungen. In der zweiten, geometrischen, Phase werden die Fahrspuren und Kreuzungen durch optisch geglättete, bandartige Geometrien modelliert. Jeder Fahrspur werden Begrenzungslinien zugewiesen.

Kreft [2012] gibt über eine Zusammenstellung verschiedener Ansätze hinaus eine neue systematische Methode zur effizienten Bildung geospezifischer Umgebungsmodelle für Fahrsimulationen. Die Bildung des Umgebungsmodells wird in sechs Phasen gegliedert, wobei als erstes das Logik-

modell aufgebaut wird. Mit der Festlegung der Achsverläufe einzelner Strecken wird die Topologie des Verkehrsnetzes definiert. Die Geometrie der Fahrbahnen wird durch Generierung der einzelnen Fahrstreifen beschrieben. Die Integration logischer Eigenschaften wie Abbiegerelationen oder Höchstgeschwindigkeiten vervollständigt das Logikmodell. Das Grafikmodell entsteht mit Bildung von 3D-Modellen des Verkehrsnetzes und der Landschaft. In der letzten Phase wird durch manuelle Modellierung das Grafikmodell um Details erweitert, um es für den Benutzer möglichst realistisch erscheinen zu lassen. Aufbauend auf dieser Systematik wird ein Konzept für ein Software-Werkzeug beschrieben sowie dessen prototypische Implementierung. Abschließend wird die vorgestellte Systematik an zwei Beispielen validiert.

3.2.2 Verknüpfungsansätze basierend auf Ontologie

Schüle et al. [2004] beschreiben verschiedene Ansätze, um GIS-Daten mit Simulationen zu verknüpfen. Die Lösung aller Ansätze basiert auf der Verwendung einer gemeinsamen Ontologie. Darunter versteht man eine formale Beschreibung, womit Klassen, Attribute und deren Beziehungen zueinander definiert werden. Eine Möglichkeit ist, Simulation und GIS unabhängig voneinander zu betreiben und Daten über ein gemeinsames Format auszutauschen. Effizienter werden Ansätze gesehen, bei denen das eine System in das andere eingebettet wird. Das bedeutet entweder, dass GIS in ein Simulationssystem integriert wird oder umgekehrt. Um ein derart komplexes Gesamtsystem zu erschaffen, bedarf es eines hohen Entwicklungsaufwands. Des Weiteren unterscheiden Schüle et al. [2004] das sogenannte “direct cooperative coupling“ und das “indirect cooperative coupling“. Simulation und GIS nehmen dabei die Rolle als Server bzw. Client ein. Der Unterschied liegt darin, ob eine gemeinsame integrierte Oberfläche (direkt) oder eine zusätzliche Software (indirekt) besteht, die als Kontrollkomponente fungiert. Implementiert wird eine lockere Kopplung der Simulation SeSAm über ESRI Shape Files mit ATKIS Daten.
Liang et al. [2008] und Feng et al. [2010] beschreiben ein Rahmenkonzept für den Datenaustausch zwischen Mikroskopischen Verkehrssimulationen (engl. Microscopic Traffic Simulation, kurz: MTS) und GIS basierend auf Ontologie. Vorgeschlagen wird ein standardisiertes Datenmodell, welches die Merkmale beider Datenmodelle vereinigt. Als Vorteil einer ontologischen Methode wird die dadurch resultierende Unabhängigkeit in der Datenbeschreibung von einem bestimmten System genannt, wodurch sich die Flexibilität hinsichtlich des Datenformates einer Informationsquelle erhöht. Liang et al. [2008] erarbeiten dafür ausgehend von den GIS-Datenmodellen GDF und UNETRANS sowie einem MTS-Datenmodell (FLOWSIM) ein standardisiertes Datenmodell, welches in den vier Ebenen “Road Network Layer“, “Route Layer“, “Event Layer“ und “Nonspatial Object Layer“ organisiert ist, wobei das Straßennetzwerk im Mittelpunkt steht. Die Validität und Machbarkeit des Modells zum Datenaustausch zwischen GIS und MTS wird anhand einer ESRI-Geodatenbank und der mikroskopischen Verkehrssimulation FLOWSIM verifiziert. Feng et al. [2010] spezifizieren das Rahmenkonzept, in dessen Mittelpunkt das standardisierte Datenmodell steht. Gegliedert ist das Konzept in fünf Ebenen: Datenquelle, semantische Anpassung, semantische Datenschicht, semantische Schnittstelle und Anwendungsebene. Der Aufbau einer semantischen Datenschicht umfasst sowohl die Verknüp-

fung zwischen dem Inhalt verschiedener Informationsquellen sowie deren Objekten, Attributen und Relationen als auch Bedingungen zur Einschränkung von Datenquellen, logische Schlussfolgerungen und hierarchische Beziehungen zwischen Objekten. MTSON (“Microscopic Traffic Simulation Ontology“) ist die Umsetzung einer semantischen Datenschicht und bildet den Kern des vorgeschlagenen Datenaustauschkonzeptes zwischen GIS und MTS. Es definiert das konzeptuelle Modell einer Verkehrssimulation und fügt Daten eine semantische Bedeutung an. Implementiert und validiert ist MTSON mit Hilfe der Ontolgy Web Language (OWL).

3.3 Dynamische Prozesse in GIS

Mit Hilfe Geographischer Informationssysteme lassen sich räumliche Informationen verwalten und verarbeiten. Allerdings werden keine dynamischen Prozesse unterstützt [Schüle et al., 2004]. In vielen Bereichen spielt die Zeit als vierte Dimension jedoch eine wichtige Rolle. In der Stadtentwicklung sind langsame Veränderungen von Interesse, in Simulationen zum Energiebedarf oder bewegter Objekte ist die Abbildung dynamischer Prozesse notwendig.
Fan [2010] stellt ein objektorientiertes und Ereignis-Zustand-basiertes raumzeitliches Datenmodell zur Speicherung und Verwaltung semantischer und geometrischer Veränderungen der digitalen Gebäude eines 3D-Stadtmodells vor. Die Methode basiert auf drei Erweiterungen herkömmlich statischer Modelle. So erhalten Attribute einen Lebenszeitraum, der angibt, wann diese erscheinen und verschwinden. Um den Prozess von Veränderungen darzustellen, wird das Objektmodell mit einem Eventmodell kombiniert. Als dritte Erweiterung werden der „Spatiotemporal City Object“-Klasse bestimmte Verhalten zugewiesen. Darunter sind sowohl räumliche Veränderungen wie Erscheinen oder Verschwinden als auch Höhen-, Flächen- oder Volumenänderungen von Objekten zu verstehen. Der Lebenszyklus von Objekten wird als zeitliche Sequenz von Zuständen und Events modelliert. Ein Event löst eine Zustandsänderung aus, die als Ergebnis einen neuen Zustand zur Folge hat. Umgesetzt wird die Methode als Application Domain Extension (ADE) in CityGML. Dynamische Veränderungen von Geometrien können noch nicht visualisiert werden. Auch bewegte Objekte sind nicht Teil des modellierten Systems.
Chaturvedi & Kolbe [2015] präsentieren ein neues Konzept namens “Dynamizers“, welches statische 3D-Stadtmodelle um eine Unterstützung von zeitlichen Veränderungen erweitert. Der Grundgedanke ist dynamische Daten in Form einer Zeitreihe zu repräsentieren und in ein Stadtmodell zu integrieren. Während der Ansatz nach Chaturvedi & Kolbe [2015] auf tabellierte Werte beschränkt ist, bietet das neue Dynamizer Konzept nach Chaturvedi & Kolbe [2016] weitere Möglichkeiten. Die dynamischen Daten können demnach aus externen Dateien (z.B. CSV-Dateien), aus externen Datenbanken oder direkt von Sensoren (z.B. Echtzeitbeobachtungen) stammen. Die Grundlage stellt die ISO 19123 dar, welche ein Schema für räumliche, zeitliche oder raumzeitliche Coverages definiert. Der Dynamizer nutzt die dynamischen Daten, repräsentiert in Form von Zeit-Werte-Paaren, als Domain Range mit Abbildungsfunktion oder als Inline-Repräsentation von Sensordaten, um den statischen Wert von einem Attribut eines Stadtobjektes zu überschreiben. Das Konzept wird als Erweiterung zum CityGML

Standard implementiert. Dynamizer wird als Datentyp (Feature Type) mit drei Attributen definiert. Das Attribut “attributeRef“ stellt über einen XPath die Verknüpfung zum statischen Attribut eines Stadtobjektes her. Start- und Endzeit geben die Zeitspanne an, für die der Dynamizer zeitabhängige Werte liefert [Chaturvedi & Kolbe, 2016]. Mit Hilfe des Konzeptes können abstrakte und räumliche Werte modelliert werden, die Veränderungen von thematischen und räumlichen Attributen darstellen. Das bedeutet, Änderungen in Lage und Gestalt sind realisierbar, womit insbesondere auch bewegte Objekte (wie Fahrzeuge) modellierbar werden [Chaturvedi & Kolbe, 2015].

3.4 Zusammenfassung und Folgerungen

Als grundlegende Problematik im Bereich Verkehrs- und Fahrsimulationen wird der hohe Modellierungsaufwand für Simulationsgraphen und Umgebungsmodelle genannt. Die Ansätze, diesen zu reduzieren, sind vielfältig und reichen von einer Ableitung eines Simulationsgraphen - zum Teil bereits weitestgehend automatisch - bis hin zu komplexen Kopplungen von Simulationsanwendungen mit GIS. Zu Visualisierungszwecken werden verschiedene Konzepte zur Erstellung eines Umgebungsmodells vorgeschlagen. Ansätze zur Integration dynamischer Prozesse in GIS könnten in Zukunft Visualisierungen von Simulationsergebnissen innerhalb eines Geoinformationssystems ermöglichen.

Unter den recherchierten Gesichtspunkten erscheint CityGML einmal mehr als vielversprechendes System zur Verknüpfung mit Verkehrssimulationen. Stadtmodelle im CityGML Format verfügen über einen hohen Informationsgehalt. Im Laufe der Arbeit gilt es zu klären, inwieweit diese Informationen genutzt werden können und welche Prozessschritte zur Ableitung eines Simulationsgraphen sowie zur anschließenden Visualisierung notwendig sind.

4 Modellierung des Verkehrsraumes

Eine zentrale Stellung in der Bearbeitung des vorliegenden Themas nehmen Fragestellungen rund um die Modellierung des Verkehrsraumes ein. Die Repräsentation des Verkehrsraumes stellt die essentielle Grundlage für Verkehrssimulationen dar. Ziel der Arbeit ist die Nutzung bestehender Verkehrsraumdaten aus 3D-Stadtmodellen zur Erzeugung eines Simulationsgraphen sowie die Visualisierung von Simulationsergebnissen in einer virtuellen, dreidimensionalen Repräsentation des Verkehrsraumes.
Für beide Problemstellungen ist von Bedeutung, wie sich der Verkehrsraum modellieren lässt. Diese Frage soll ausgehend von der allgemeinen Modelltheorie beantwortet werden, ehe eine Übertragung auf die Anwendung Verkehrs- und Fahrsimulationen erfolgt. In diesem Zusammenhang wird explizit die Struktur eines für derartige Anwendungen geeigneten Verkehrsraummodelles analysiert, um anschließend bestehende Modelle hinsichtlich der gefundenen Anforderungen zu untersuchen und untereinander zu vergleichen. Die letzten beiden Abschnitte widmen sich im Detail den Verkehrsraummodellen der dieser Arbeit zugrunde liegenden Komponenten, der Verkehrssimulation Vissim sowie dem Standard für 3D-Stadtmodelle CityGML.

4.1 Übertragung der allgemeinen Modelltheorie auf den Verkehrsraum

Stachowiak [1973] führt in seinem Werk “Allgemeine Modelltheorie“ drei Hauptmerkmale des allgemeinen Modellbegriffs auf:

- **Abbildungsmerkmal:**
  “Modelle sind stets Abbildungen von etwas, nämlich Abbildungen, Repräsentationen natürlicher oder künstlicher Originale, die selbst wieder Modelle sein können“ [Stachowiak, 1973, S. 131]

- **Verkürzungsmerkmal:**
  “Modelle erfassen im allgemeinen nicht alle Attribute des durch sie repräsentierten Originals, sondern nur solche, die den jeweiligen Modellerschaffern und/ oder Modellbenutzern relevant erscheinen.“ [Stachowiak, 1973, S. 132]

- **Pragmatisches Merkmal:**
  “Modelle sind ihren Originalen nicht per se eindeutig zugeordnet. Sie erfüllen ihre Ersetzungs-

funktion a) für bestimmte [...] Subjekte, b) innerhalb bestimmter Zeitintervalle und c) unter Einschränkung auf bestimmte [...] Operationen.“ [Stachowiak, 1973, S. 132-133]

Übertragen auf die Modellierung des Verkehrsraumes stellt ein Verkehrsraummodell eine Abbildung des realen Verkehrsraumes dar. Inhalt sollten sämtliche verkehrsrelevante Objekte sein, die in der realen Welt auftreten können. Um diesen Bedingungen bei der Erstellung eines abstrakten Verkehrsraummodells nachzukommen, soll als Orientierung ein Foto dienen, das einen Verkehrsraum am Beispiel der Dachauer Straße in München zeigt (Abbildung 4.1).

Abbildung 4.1 – Beispiel für einen Verkehrsraum anhand der Dachauer Straße in München (Quelle: Google Streetview, ©Google 2017)

Das Bild zeigt auf der rechten Seite die Tram-Bahnstation “Hauptbahnhof Nord“ mit auf dem verbreiterten Gehweg angebrachten Sitzgelegenheiten und Wartemöglichkeiten, die von Passanten genutzt werden. Das Wartehaus wird von Vegetation gesäumt, die jenseits des Fußgängerraumes einen kleinen Grünstreifen bildet. Die Straße selbst, die vom Fußgängerbereich durch Randsteine getrennt wird, ist von Autos und der Trambahn befahren. Für letztere sind Schienen vorhanden. Auf der linken Seite des Bildes stehen parkende Autos, bevor wiederum ein Gehweg den Verkehrsraum zu den angrenzenden Häusern abschließt. Am oberen Rand kreuzt die Dachauer Straße die von rechts kommende Elisenstraße, die nach links in die Marsstraße übergeht. Die Vorfahrt ist an dieser Kreuzung durch eine Ampelanlage geregelt.

Möchte man ein dieser Situation entsprechendes abstraktes Verkehrsraummodell bilden, gilt es die aufgeführten Objekte als Klassen abzubilden. Dies ist im UML-Diagramm in Abbildung 4.2 umgesetzt. Die Farbgebung der Klassen gibt eine thematische Einteilung in Verkehrsraumobjekte (gelb),

Verkehrsteilnehmer (blau), Straßenausstattung (rot) und Vegetation (braun) an. Darüber hinaus liegt eine Geometrieklasse (grün) vor. Ausgehend von der Oberklasse "_Transportation" wird der Verkehrsraum in mehreren Ebenen durch Spezialisierungen seiner selbst repräsentiert. Der abstrakte Verkehrsraum "_TrafficSpace" lässt sich unterteilen in Bereiche für Fahrzeuge und einen für Fußgänger. Eine dritte Klasse "_MiscStreetSpace" repräsentiert sonstige Flächen, die weder von Fahrzeugen befahren noch von Fußgängern begangen werden können. Dies können beispielsweise Verkehrsinseln oder Grünflächen sein. Die Assoziation zur Klasse Vegetation illustriert, dass in derartigen Flächen Vegetation auftreten kann.
In allen drei Unterklassen des "_TrafficSpace" treten verschiedene Ausstattungsobjekte ("_StreetFurniture") auf. Dazu zählen unter anderem Gullydeckel, Bänke, Briefkästen oder Telefonzellen. Straßenmarkierungen, Straßenschilder, Ampelanlagen und Straßenlaternen werden ebenfalls der Klasse "_StreetFurniture" zugeteilt.
Nutzer des Verkehrsraumes sind die Verkehrsteilnehmer, welche sich weiter unterteilen lassen in Fahrzeuge, die den Fahrzeugraum nutzen, und Fußgänger als Benutzer des Fußgängerraumes. Auch für dieses thematische Feld ist eine weitere Spezifizierung möglich, welche im vorliegenden Diagramm lediglich für Fahrzeuge mit der Unterscheidung in Straßen- und Schienenfahrzeuge sowie Fahrräder vorgenommen wurde.
Der Fahrzeugraum ist durch die Klassen Kreuzung, Straße, Schiene, Fahrradweg und Einfahrt untergliedert. Die Assoziationen "isAccesibleBy" stehen für die Verknüpfung mit den nicht genauer spezifizierten Klassen Parkmöglichkeiten und Haltestellen öffentlicher Verkehrsmittel. Haltestellen sollten ebenfalls über den Fußgängerraum zugänglich sein. Der abstrakte "_PedestrianSpace" wird durch die Klassen Fußweg, Fußgängerübergang, Fußgängerzone und Platz spezialisiert. Die vier Klassen verfügen über eine geometrische Beschreibung. Der Fußweg ist weiter spezifiziert in die Unterklasse "Sidewalk" mit dem Element Randstein ("Kerbstone"). Ein Gehsteig zeichnet sich dadurch aus, dass er entlang einer Straße verläuft. Gekreuzt werden Gehwege von Grundstückseinfahrten.
Ebenfalls auf oder entlang einer Straße können Fahrradwege und Schienen verlaufen. Die Straße wird in einzelne Straßensegmente unterteilt. Die Straßenabschnitte liegen in geordneter Reihenfolge vor und ergeben als Ganzes eine Straße gekennzeichnet durch einen Namen, z.B. "Dachauer Straße". Diesem Verständnis einer Straße als Aggregat mehrerer Segmente folgend kann eine Kreuzung durch zwei unterschiedliche Assoziationen charakterisiert werden. Zwei oder mehr Straßen schneiden sich an einer Kreuzung. Für eine Beschreibung über Straßenabschnitte sind für eine Kreuzung mindestens drei Segmente notwendig, die sich am Kreuzungspunkt treffen. Einen Sonderfall stellen Kreisverkehre dar, die deshalb als Unterklasse einer normalen Kreuzung geführt werden.
Als kleinster Bestandteil einer Straße lassen sich die Fahrspuren modellieren, die existenziell vom Vorhandensein eines Straßensegments abhängig sind. Fahrspuren weisen eine Fahrtrichtung auf und verfügen über eine Geometrie. Die Kompositionsmenge aller Fahrspuren ergibt die Geometrie eines Straßensegments. Die Geometrie einer Straße ist dann wiederum als Aggregat seiner Straßensegmente darstellbar. Kreuzungsflächen, Schienen, Fahrradwege und Einfahrten lassen sich direkt expliziten Geometrien zuordnen.

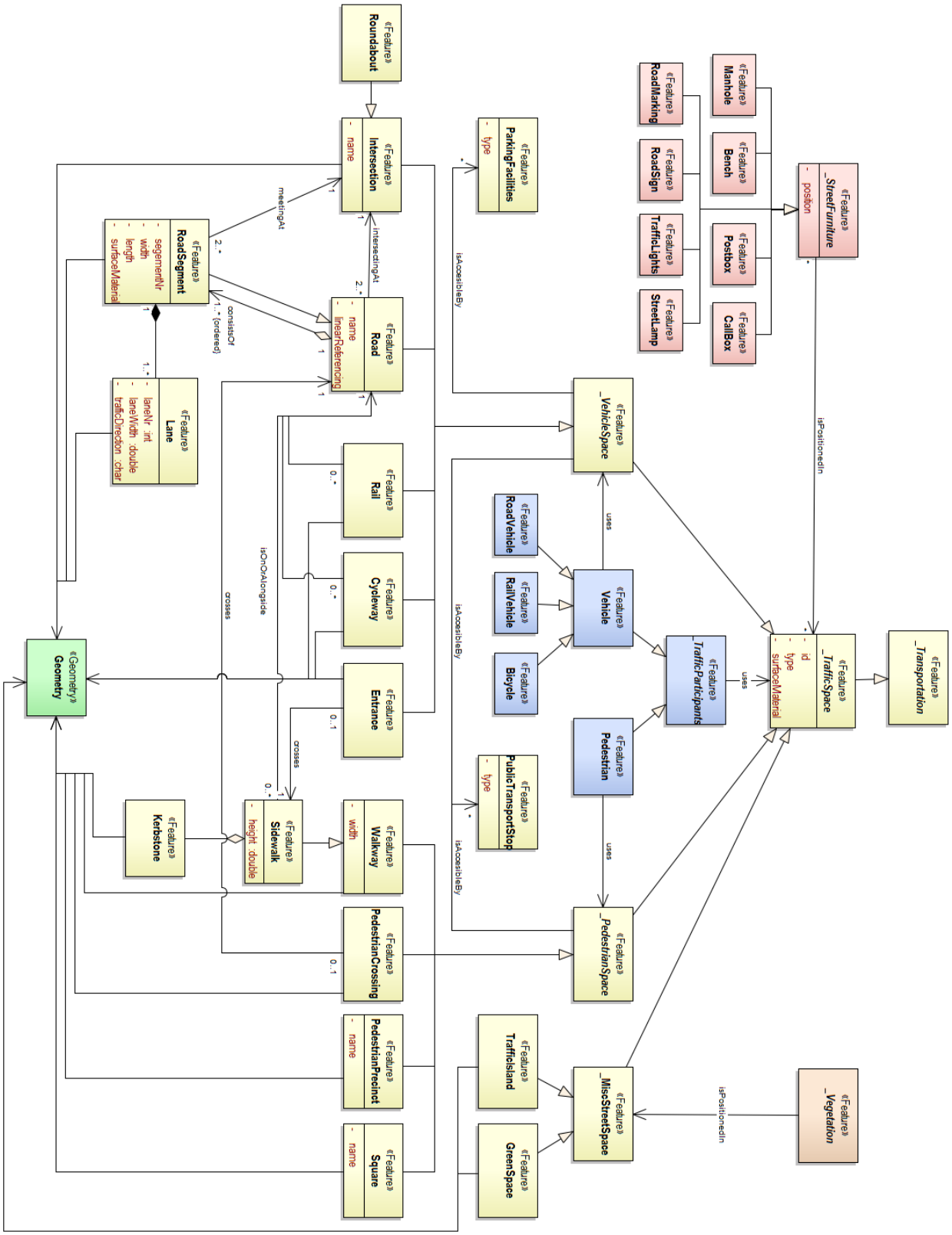


Abbildung 4.2 – UML-Diagramm eines abstrakten Verkehrsraummodells

Im Sinne des Verkürzungsmerkmals wurden lediglich Objekte der realen Welt modelliert, die im Kontext Verkehr bzw. Verkehrsraum stehen. Dieser Kontext ist allerdings ein weitläufiger Bereich mit zahlreichen Anwendungsmöglichkeiten, für die das obige Verkehrsraummodell lediglich eine Grundlage darstellen kann. Je nach Anwendung kann es notwendig sein, dem abstrakten Modell weitere Objekte oder Relationen hinzuzufügen. Bei der Entwicklung eines konkreten Modelles ist nicht nur zu berücksichtigen, wovon etwas Modell ist (im vorliegenden Fall ein Modell vom Verkehrsraum), sondern zusätzlich "für wen, wann und wozu bezüglich seiner je spezifischen Funktionen es Modell ist" [Stachowiak, 1973, S. 133]. Die gesamte Art der Modellbildung ist folglich abhängig vom Modellentwickler, den Nutzern und damit auch vom jeweiligen Anwendungsfeld. Dies erklärt, weshalb es eine Vielzahl unterschiedlicher, bestehender Modelle im Bereich Verkehr gibt.

4.2 Anwendungsspezifische Verkehrsraummodelle

Beil [2017] nennt allein 16 verschiedene Anwendungen, für die Informationen über den Verkehrsraum von Relevanz sind. Die Art der Informationen unterscheidet sich je nach Anwendung. Verdeutlichen lässt sich dies am Beispiel der Anwendungen Stadt- und Raumplanung im Vergleich zum Bereich der Navigation und Routenplanung. Während im ersten Fall neben thematischen Daten insbesondere Flächeninformationen der verschiedenen Verkehrsraumobjekte Eingang finden, wird im Bereich der Navigation bzw. Routenplanung eine Modellierung des Straßenraumes in Form eines linienhaften Netzwerkes vorgezogen, welches die Topologie des Straßennetzes wiedergibt.
Neben der Art der Informationen unterscheiden sich auch die Anforderungen an die Daten je nach Anwendungsgebiet. Beil [2017] analysiert die unterschiedlichen Anforderungen von 16 potenziellen Nutzern von Verkehrsraummodellen anhand der Lagegenauigkeit, der thematischen Genauigkeit sowie der Aktualität der Daten. Weitere Analysekriterien sind zusätzliche, verkehrsfremde Informationen, die Rolle der Zeit als 4. Dimension sowie die Bedeutung der Visualisierung. Essenz der Untersuchungen ist, dass die Anforderungen je nach Anwendungsfall stark variieren. So hat beispielsweise für Fahrtrainingssimulationen die Visualisierung größte Bedeutung, wohingegen die absolute Genauigkeit weniger relevant ist. In anderen Anwendungsfällen wie beim Autonomen Fahren spielt die Visualisierung eine untergeordnete Rolle, während die absolute Genauigkeit von besonderer Wichtigkeit ist.
Die Art der Informationen und die Anforderungen an die Daten haben also ebenfalls Auswirkungen auf die Modellbildung. Augenmerk der vorliegenden Arbeit liegt auf dem Anwendungsfall der Verkehrssimulation. Der vorausgegangene Abschnitt hat die Modellbildung ausgehend von der realen Welt thematisiert. Auf gleiche Weise soll im nachfolgenden Abschnitt ausgehend von einer Recherche notwendiger Verkehrsrauminformationen für Verkehrs- und Fahrsimulationen die Kriterien einer anwendungsspezifischen Modellbildung analysiert werden. Anhand der ergründeten Charakteristiken eines Verkehrsraummodelles sollen dann bestehende Standards und Modelle aus dem Bereich Simulation, aber auch anderen Bereichen (Stadt- und Landschaftsmodelle) betrachtet werden.

4.2.1 Strukturanalyse - Verkehrsraummodell für Verkehrs- und Fahrsimulationen

Die Modellierung des Verkehrsraumes und seiner direkten Umgebung ist Grundvoraussetzungen für Verkehrs- und Fahrsimulationen. Grundsätzlich hängen die Art und der Detailierungsgrad der benötigten Informationen von der jeweiligen Simulationsanwendung ab. Nach Schüle et al. [2004] stehen die Anforderungen an den Detaillierungsgrad der Umgebungsmodellierung im direkten Zusammenhang mit dem Simulationsmodell. Werden menschliche Verhaltensweisen modelliert, ist es unabdingbar die Umgebung so genau wie möglich darzustellen. Das Umgebungsmodel ist in diesem Fall von großer Bedeutung, da es das Systemverhalten der Simulation stark beeinflusst.
Für einfache Routing-Anwendungen genügen Graphen mit zugehörigen Metadaten. Für Verkehrssimulationen oder Fahrsimulationen ist dies nicht ausreichend. Verkehrssimulationen sind im Allgemeinen spurbasiert angelegt. Darauf aufbauend können entweder agenten-basierte (mikroskopische) Simulationen oder dichte-basierte (makroskopische) Simulationen durchgeführt werden. Je nachdem welcher Ansatz verfolgt wird, muss der Simulationsgraph in der Lage sein bestimmte Anfragen rechnereffizient zu unterstützen [Wilkie et al., 2012]. Zusätzliche Anforderungen entstehen bei der Berücksichtigung von multimodalem Verkehr. In diesem Fall ist es beispielsweise notwendig, dass Straßentypen unterschieden werden können [Dallmeyer, 2014].
Eine mögliche Gliederung der benötigten Ausgangsdaten lässt sich durch Unterscheidung in logische und graphische Daten aufstellen. Kreft [2012] unterteilt Umgebungsmodelle für Fahrsimulationen in das Logik- und das Grafikmodell. Während ersteres die „nicht sichtbaren Elemente“ umfasst, sind im Grafikmodell die „sichtbaren Bestandteile“ der Umgebung abgebildet. Zur Logik zählen demnach die Geometrie von Streckenabschnitten, logische Verknüpfungen der Fahrspuren in den Verkehrsknoten, physikalische Eigenschaften und Objekte entlang der Strecken. Graphisch wird korrelierend zum Logikmodell das Aussehen der einzelnen Objekte näher beschrieben. Wilkie et al. [2012] nennen Informationen zur Topologie des Straßennetzwerkes, geometrische Details zu den Fahrspuren und andere Informationen wie Aufbau von Kreuzungen, Ampelregelungen oder Straßenbeläge als notwendige Voraussetzungen für Verkehrs- und Fahrsimulationen. Chaplier et al. [2010] strukturieren die zur Beschreibung des Verkehrsraum notwendigen Informationen in vier Layern: einem topologischen, einem logischen, einem physikalischen und einem visuellen Layer.
Die **topologische Darstellung** eines Verkehrsnetzes entspricht einem gerichteten Graphen (siehe Abbildung 4.3 - “Topological Layer“). Ein solcher Graph besteht aus einer Menge von Knoten und gerichteter Kanten, wobei die Ordnung der Knotenpaare die Richtung des Verkehrsflusses vorgibt [Stegemann & Ameling, 1982]. Bei den Kanten muss weiter differenziert werden, über wie viele Fahrspuren sie verfügen. Esser & Schreckenberg [1997] unterscheiden drei Typen von Kanten: Single-Lanes mit optionalen Abbiegezonen, Multi-Lane Kanten mit der Möglichkeit zum Spurwechsel und Transfer Kanten, das heißt zum Beispiel Autobahnauffahrten mit Beschleunigungsstreifen. Wichtiger Bestandteil des Netzwerks sind die topologischen Beziehungen zwischen den einzelnen Spuren [Wilkie et al., 2012], das bedeutet Spur-zu-Spur Verknüpfungen [Krajzewicz et al., 2002]. Diese können auf verschiedene Arten umgesetzt werden. Stegemann & Ameling [1982] schlagen eine Verknüpfung aufeinanderfolgender Spurnummern vor. Im OpenDrive-Format gibt es zwei Möglichkeiten:

eine Nachfolger-Vorgänger Beschreibung oder, wenn eine derartige Verknüpfung mehrdeutig ist, die Aufstellung einer Verknüpfungsmatrix für den betroffenen Knoten [Dupuis, 2015].
Eng mit der Topologie verknüpft sind logische Informationen zu Netzwerkelementen (Abbildung 4.3 - “Semantical/ Logical Layer“). **Semantische** Attribute definieren die Bedeutung bestimmter Objekte des Verkehrsraumes. Der Straßentyp steht in Verbindung mit einer Vielzahl weiterer Informationen, u.a. mit der Anzahl und Breite von Fahrspuren, Zufahrtsbeschränkungen und Geschwindigkeitsbegrenzungen. Insbesondere für Simulationen von multimodalem Verkehr sind derartige Informationen wichtig [Kreft, 2012]. Zufahrtsbeschränkungen ermöglichen eine Differenzierung, welche Kanten nur für bestimmte Verkehrsteilnehmer wie Radfahrer oder Fußgänger zugänglich sind. Neben einfachen Abbiegerelationen sind für Kreuzungen Vorfahrtsregelungen zu modellieren [Krajzewicz et al., 2002]. Ein realistischer Verkehrsfluss an Kreuzungen ist von hoher Bedeutung für Simulationen, was die Modellierung komplexer Vorfahrtsregelungen und realitätsnaher Ampelschaltungen erfordert [Esser & Schreckenberg, 1997]. Hinweise zur Vorfahrtsregelung und zu Abbiegerelationen enthalten auch Fahrbahnmarkierungen und Verkehrsschilder. Auch Zusatzinformationen zu Verkehrsbauwerken wie Brücken oder Tunnel sind für eine korrekte topologische Darstellung erforderlich. Weitere semantische Informationen enthalten Straßennamen und Attribute beispielsweise zur Oberflächenbeschaffenheit wie Material, Rauheit oder Reibung [Dupuis, 2015]. Sind Spurausweitungen für Abbieger vorhanden, muss klar sein, ob genügend Platz für abbiegende Fahrzeuge vorhanden ist oder ob die Spuren durch die Abbieger blockiert werden [Richter et al., 2016]. Hierbei spielt neben der Topologie und Logik die **Geometrie** der Fahrspuren eine entscheidende Rolle (siehe Abbildung 4.3 - “Geometrical/ Physical Layer“). Nach Richter et al. [2016] stellt die topographische Beschreibung der Straßengeometrie den Hauptbestandteil von Fahrsimulationen dar. Eine physikalische Repräsentation wird hauptsächlich in geometrischer Form realisiert. Die vollständige Geometrie wird durch Fahrbahnachse und Querschnittsprofil definiert [Kreft, 2012]. Der Achsverlauf wird durch Trassierungselemente wie Klothoiden, Kreissegmente und Geraden beschrieben [Negele, 2007]. Hinzukommt für eine dreidimensionale Darstellung der Gradient des Straßenabschnitts [Richter et al., 2016]. Stegemann & Ameling [1982] zählen als Angaben zur Fahrbahnbeschreibung die Länge des Straßenabschnittes, die Breite, den Kurvenradius und die Steigung auf. Der Modellierung des Querprofils werden Attribute wie Kurvenüberhöhung und Querneigung zugeordnet [Dupuis, 2015].
Nicht nur das Straßennetzwerk selbst, sondern auch die direkte Umgebung erfordert eine Beschreibung. Zur Modellierung der Umgebung finden 3D-Modelle zur Beschreibung der Landschaftsoberfläche (DGM), von Vegetation, Gebäuden, Objekten wie Tunneln oder Brücken und Verkehrsmobiliar Eingang [Kreft, 2012]. Texturen aus Bilddaten sorgen für ein realistisches Aussehen der Objekte [Negele, 2007]. All diese Formen der Visualisierung fallen unter das äußere, grafische **Erscheinungsbild** von Verkehrs- und Fahrsimulationen (Abbildung 4.3 - “Appearance/ Visual Layer“).

Abbildung 4.3 fasst die Bildung eines Verkehrsraummodelles für Verkehrs- und Fahrsimulationen am Beispiel einer Kreuzung zusammen. Das Verkehrsraummodell (“Traffic Space Model“) setzt sich wie von Chaplier et al. [2010] vorgeschlagen zusammen aus vier Layern, die unterschiedliche Informationen beinhalten.

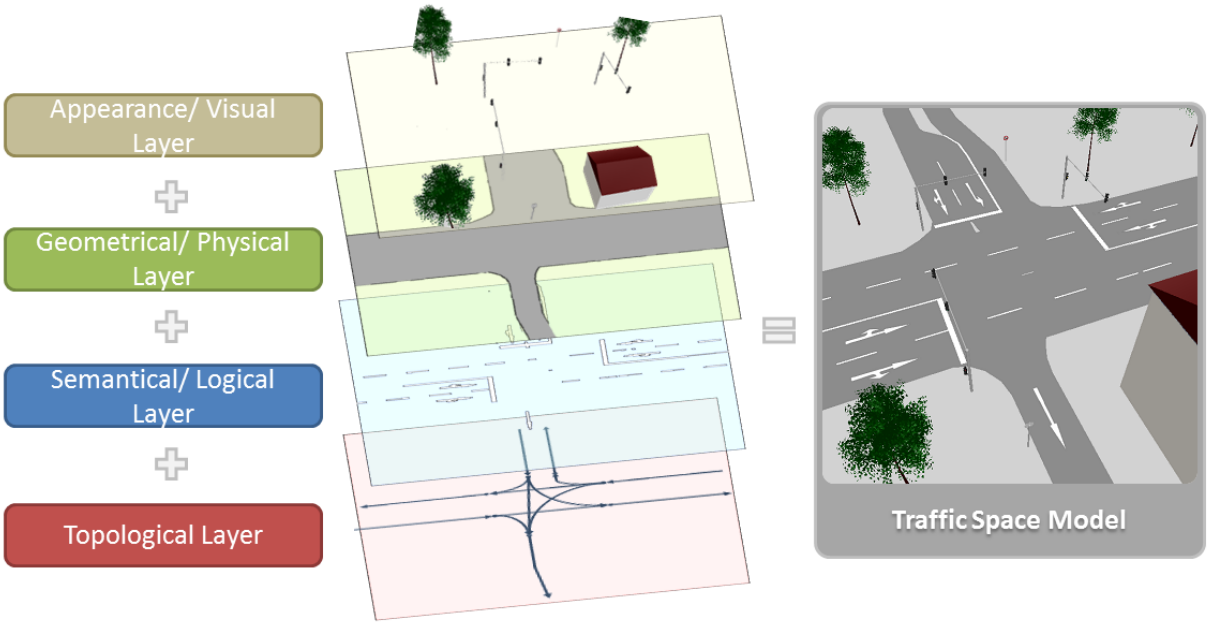


Abbildung 4.3 – Aufbau eines Verkehrsraummodells für Verkehrs- und Fahrsimulationen bestehend aus vier Layern

Ein den Anforderungen einer Verkehrs- oder Fahrsimulation entsprechendes Modell beinhaltet demnach eine Datenschicht mit topologischen Informationen und eine semantisch-logische Ebene. Hinzukommen Informationen, die die geometrisch physikalische Eigenschaften beschreiben und eine Datenebene, die das visuelle, grafische Erscheinungsbild des Verkehrsraumes prägt. Abbildung 4.4 stellt die zusammengetragenen Informationen zur Repräsentation des Straßenraums ohne Anspruch auf Vollständigkeit in Form eines Flussdiagrammes dar. Ausgehend von den beiden Elementen Logik und Grafik werden die Objekte und Attribute eines Verkehrsnetzes zueinander in Beziehung gesetzt. Die farbige Kennzeichnung der Elemente differenziert nach den vier Ebenen Topologie, Semantik, Geometrie und Visualisierung.

Vergleicht man das Flussdiagramm aus Abbildung 4.4 mit dem abstrakten Verkehrsraummodell, fallen verschiedene Komponenten auf, die das abstrakte Modell nicht abdeckt. Der Hauptaugenmerk des abstrakten Verkehrsraummodells liegt darin Realwelt-Objekte abzubilden. Die Objekte des Straßenraumes bilden die Grundlage eines Modells für Verkehrs- und Fahrsimulationen. Sie reichen aber alleine nicht aus, um eine Simulation durchführen zu können. Tamminga et al. [2013] sehen insgesamt vier Einflüsse auf die Modellbildung für Verkehrs- und Transportmodelle: Die berücksichtigten Verkehrsmittel legen die notwendigen Objekte fest. Wird lediglich der Automobilverkehr einbezogen, reicht eine Modellierung der Straßen aus. Fußgänger erfordern zusätzlich Gehwege, Busse und andere öffentliche Verkehrsmittel bedürfen Haltestellen. Weiteres Kriterium ist die Repräsentation der Bewegung von Verkehrsteilnehmern, welche die Art der geometrischen Modellierung beeinflusst. Tamminga et al. [2013] unterscheiden dabei die Varianten “Network Space[s]“ und “Scene Space[s]“.

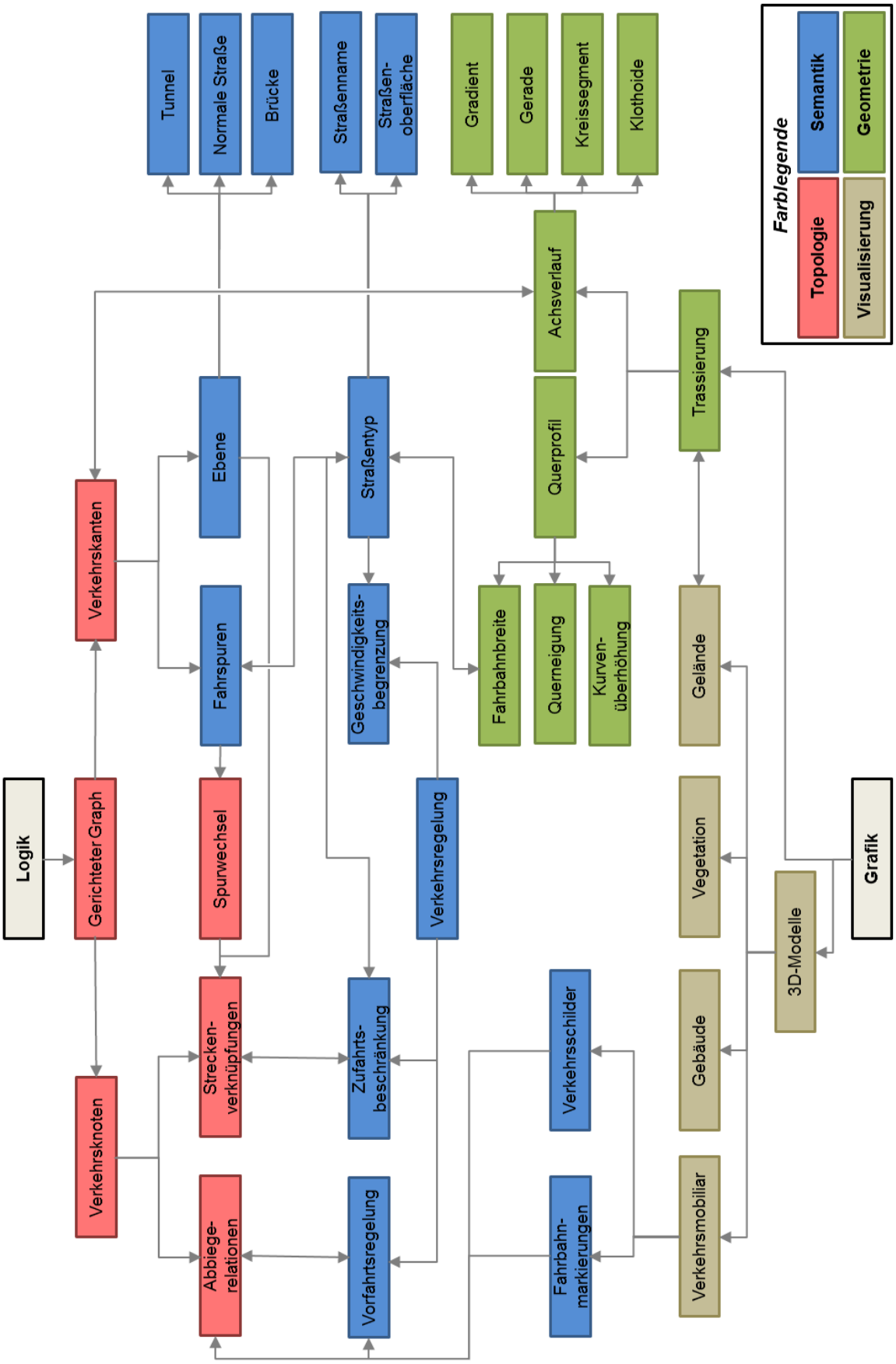


Abbildung 4.4 – Datenanalyse als Flussdiagramm: für Verkehrs- und Fahrsimulationen benötigte Informationen über den Verkehrsraum und deren Relationen

Im ersten Fall sind Routen durch konkrete Pfade zwischen Knoten definiert. Eine netzwerkbasierte Repräsentation des Straßenraumes ist die klassische Art der Modellierung für Routing-Anwendungen und Verkehrssimulationen. Kanten geben die spurgebundene Bewegungsrichtung vor, Knoten beinhalten Abbiege- bzw. Verknüpfungsinformationen. Die Alternative stellen flächenbasierte Modellierungen des Verkehrsraumes dar. Der “Scene Space“ kommt zur Anwendung, falls ein Bewegungsstrom auf einer Fläche auftritt, deren physikalische Charakteristiken und Begrenzungen die Bewegung beeinflussen. Dies kommt typischerweise bei Infrastruktur für Fußgänger zum Tragen. Plätze werden nicht durch einen einzelnen Knoten, sondern als Fläche mit Zu- und Abgängen sowie möglicherweise Hindernissen repräsentiert. Im sonstigen Verkehr spielen flächenhafte Darstellungen eine Rolle, wenn laterale Bewegungen, d.h. Bewegungen quer zur Fahrbahnachse, modelliert werden sollen. In der Verkehrssimulation Vissim ist dies für Fußgänger und Fahrradfahrer möglich. Der Automobilverkehr ist spurgebunden und basiert damit auf der Netzwerkmodellierung. Die letzten beiden Einflüsse, die Tamminga et al. [2013] nennen sind Detaillierungsgrad und Ziele des Modells. Diese bestimmen die Anforderungen an die Daten.

4.2.2 Analyse bestehender Modelle und Standards

Es gibt eine Reihe bestehender Standards und Modelle, die eine Repräsentation des Verkehrsraumes spezifizieren. Der folgende Abschnitt widmet sich einer Auswahl an Verkehrsraummodellen aus den Bereichen Verkehr, Infrastruktur, Stadtmodellierung und Simulation. Nach einer kurzen Beschreibung des Anwendungsbereichs der Modelle werden diese auf Basis der ergründeten Anforderungen hinsichtlich einer Eignung als Grundlage für Verkehrssimulationen analysiert.

OGC® Land and Infrastructure Conceptual Model Standard (LandInfra)

Das “Land and Infrastructure Conceptual Model“ (LandInfra) beinhaltet die Beschreibung von Land sowohl als administrative Einheit (Grund, Parzellen, etc.) als auch als Oberfläche (Topographie). Der Standard umfasst eine konzeptionelle Modellierung von Infrastruktureinrichtungen (Straße, Schiene und Wasserwege). Ziel von LandInfra ist die Dokumentation von gemeinsamen Konzepten, die Anwendungen im Bereich Infrastruktur unterstützen. Bestehende Anwendungsentwürfe sollen nicht ersetzt werden. Der Standard bietet in Form verschiedener “Requirement Classes“ eine gemeinsame konzeptionelle Ebene, von welcher und auf welche anwendungsspezifische Modelle abgebildet werden können. Die “Road Requirements Class“ soll Entwickler beim Austausch eines Straßendesigns mit anderen Nutzern unterstützen [Gruler et al., 2016]. Das UML-Diagramm der LandInfra Road Requirements Class findet sich im Anhang A.1, die UML-Struktur eines Road Elements in Anhang A.2.
Unter Betrachtung der erörterten Struktur eines Verkehrsraummodelles für Verkehrssimulationen deckt LandInfra lediglich Teile der gestellten Anforderungen ab. Schwerpunkt des LandInfra-Modells liegt auf der geometrischen Beschreibung von Land und darauf errichteten Infrastruktureinrichtungen als physikalische Einheit. Neben einer umfassenden geometrischen Repräsentation sind semantische Informationen wie die Funktion oder Beschaffenheit eines Objektes Teil des Konzepts.

Topologische und visuelle Ebene sind dagegen nicht enthalten, was eine Eignung als Grundlage für Verkehrssimulationen stark limitiert.

OGC City Geography Markup Language (CityGML 2.0) Encoding Standard

Der internationale Standard zur Speicherung und zum Austausch von 3D-Stadtmodellen CityGML beinhaltet dreizehn verschiedene thematische Module, darunter eines zum Verkehrsraum. Das Transportation Modul zergliedert den Verkehrsraum nach logischen Kriterien in Teile, die sich aus in der Realität vorgegebenen oder beobachtbaren Strukturen ergeben [Kolbe, 2009]. Als Standard für semantische 3D-Stadtmodelle liefert CityGML über die reine Visualisierung hinaus zusätzliche Informationen zu Objekten der Stadt oder Landschaft.
Im Sinne der Datenanalyse (vgl. Abbildung 4.4) ist das Logikmodell in CityGML stark unterrepräsentiert. Ein Graph ist in Form des CityGML LoD0-Network zwar vorhanden, allerdings nicht genau spezifiziert. Die Ebene, Fahrspuren, Spurwechsel, Streckenverknüpfungen sowie Abbiegerelationen oder Vorfahrtsregelungen sind nicht Teil des CityGML Transportation Moduls. Die Stärken des Stadtmodells in Hinsicht auf die Anwendung Verkehrssimulation liegen im Bereich der geometrischen und semantischen Komponenten. Hier stellt CityGML eine Vielzahl an Daten u.a. zur Funktion, zur Oberfläche oder zu Zufahrtsbeschränkungen bereit. Zum Grafikmodell einer Verkehrssimulation kann CityGML nicht nur mit dem Transportation Modul, sondern mit weiteren thematischen Modulen wie dem Appearance-Modul oder Relief(DTM), Gebäuden, Tunnel, Brücken, Vegetation und Stadtmobiliar beitragen. Im Kontext dieser Arbeit wird jedoch lediglich auf das Verkehrsraummodell “Transportation Objects“ eingegangen werden.

INSPIRE Data Specification on Transport Networks – Technical Guidelines

INSPIRE verfolgt das Ziel europaweit eine einheitliche Infrastruktur für räumliche Informationen zu schaffen, um Probleme rund um die Verfügbarkeit, Qualität, Organisation und den Austausch von räumlichen Daten zu lösen. Im Bereich Verkehr ist die Intention einen Rahmen für ein über Grenzen hinweg nahtloses Netzwerk (Straße, Schiene, Wasser- und Luftwege) zu schaffen, welches Anforderungen intelligenter Transportsysteme wie Location Based Services (LBS), Telematik und Navigation gerecht wird. Zweck der Technischen Richtlinien ist es, eine praktische Umsetzungshilfe zu geben, die den rechtlich bindenden Anforderungen bezüglich Interoperabilität räumlicher Daten und Diensten genügt. Datentypen zum Austausch und zur Klassifikation räumlicher Objekte werden in sechs Anwendungsschemas definiert: “Common Transport Elements“, “Air Transport Network“, “Cable Transport Network“, “Railway Transport Network“, “Road Transport Network“ und “Water Transport Network“ (vgl. [INSPIRE, 2014]). Einen Überblick über die Klasse “Road Transport Network“ gibt Anhang A.3.
Unter den Gesichtspunkten der Anforderungen an ein Verkehrsraummodell für Verkehrs- und Fahrsimulationen ist das INSPIRE Straßennetzwerk wie folgt zu beurteilen: Grundlage der Schemata bilden Netzwerkgraphen in Form von Knoten und Kanten. Besondere Bedeutung wird der Verknüpfung zwischen Netzwerkteilen (länderübergreifend) und zwischen verschiedenen Teilbereichen des Verkehrswesens (intermodale Vernetzung) beigemessen. Während die Topologie überregional bestens modelliert werden kann, ist auf lokaler Ebene eine spurgenaue Abbildung von Abbiegerelatio-

nen nicht inbegriffen. Für die semantische Beschreibung stehen eine Reihe von Klassen, u.a. zur Funktion, Oberfläche, Fahrspurenanzahl oder zur Geschwindigkeitsbegrenzung zur Verfügung. Eine geometrische Repräsentation ist in Form von Punkt- und Linienobjekten sowie Flächen vorgesehen. Eine parametrische Geometriedarstellung ist zumindest für die Straßenbreite vorgesehen. Für Objekte entlang des Netzwerkes, aber auch für Attribute wie Geschwindigkeitsrestriktionen, wird das Konzept der linearen Referenzierug unterstützt. Ein Konzept zur Visualisierung ist in der Implementierungsrichtlinie nicht enthalten.

Intelligente Transportsysteme – Geographische Dateien – GDF5.0 (ISO 14825:2011)

Geographische Dateien (GDF) sind als Resultat intensiven Normierungs- und Harmonisierungsbestrebens seitens der Hersteller von digitalen Straßenkarten anzusehen, welchen Ende der 1980er Jahre klar wurde, dass das Fehlen eines internationalen Standards ein Entwicklungshindernis darstellt. Ursprünglich entwickelt als Grundlage zur Speicherung und zum Austausch geographischer Inhalte, sah das Design ein umfangreiches, anwendungsunabhängiges Datenmodell vor, welches insbesondere den Anforderungen einer navigierbaren Kartendatenbank gerecht wurde. Im Laufe seiner Entwicklung kamen weitere Aspekte hinzu, so dass GDF gegenwärtig eine Reihe von Anwendungsbereichen abdeckt. Im Mittelpunkt stehen nach wie vor Anwendungen und Dienste im Kontext von Intelligent Transport Systems (ITS) wie der Navigation oder dem Verkehrsmanagement, weshalb der Fokus auf Straßen und Informationen liegt, die im Zusammenhang mit Straßen stehen [ISO 14825, 2011]. In den Anhängen A.4 bis A.6 ist das konzeptuelle Datenmodell in Form von UML-Diagrammen wiedergegeben.
Die GDF Norm deckt alle vier Layer der Strukturanalyse ab. Der Fokus liegt auf der topologischen und semantischen Ebene, geometrische und visuelle Aspekte sind dagegen weniger explizit repräsentiert. Das Konzept gliedert sich in drei Ebenen, welche eine primitive (Level-0), einfache (Level-1) bzw. komplexe (Level-2) Repräsentation von Objekten beschreibt. Die Topologie kann implizit oder explizit modelliert werden. Explizit modelliert werden können Abbiegespuren durch Zuweisung einer semantischen Bedeutung (z.B. 'Ahead', 'Right' oder 'Merge into right lane'). Sowohl Straßen als übergeordnete Einheit als auch einzelnen Fahrspuren kann eine Funktion zugeteilt werden. Texte auf Straßenschildern und Straßenmarkierungen repräsentieren logische Elemente, wodurch Fahrtrichtungen, Vorfahrtsregelungen oder Zufahrtsbeschränkungen ebenfalls modelliert sind. Für die Positionierung von Objekten wie Straßenschildern steht eine lineare Referenzierung zur Verfügung. Geometrisch erfolgt die Darstellung einfacher Objekte in Form von Punkten, Linien oder Flächen. Komplexe Objekte entstehen aus Aggregation mehrerer Level-1-Objekte.

OpenDRIVE® Format Specification, Rev. 1.4

Open Drive ist ein Standard zur Beschreibung spurbasierter Straßennetzwerke. Auf analytische Weise werden die Geometrie einer Straße und weitere Objekte wie Fahrspuren, Straßenschilder oder Signalanlagen beschrieben, welche die Logik des Netzwerkes beeinflussen. Entwickelt wurde der Standard von VIRES Simulationstechnologie GmbH in Kooperation mit Daimler [Dupuis, 2015].
Entwickelt für Fahrsimulationen überrascht es nicht, dass Open Drive sämtliche Anforderungen an ein Verkehrsraummodell erfüllt. Lediglich die Integration von 3D-Modellen und Texturen fehlt, was auf

eine Kopplung mit einer auf Grafik fokussierten Komponente schließen lässt. Die Topologie geht über eine einfache Knoten-Kanten-Repräsentation hinaus. Kreuzungen werden über sogenannte “Connection Roads“ modelliert, welche die spurgenauen Verknüpfungen zwischen eingehenden respektive ausgehenden Straßen abbilden. Die logische Ebene wird durch verschiedene Eigenschaften beschrieben, die einer Fahrspur zugeordnet sind. Dazu zählen Materialeigenschaften, Zufahrtsberechtigung oder Spurwechselverhalten. Vorfahrtsregelung an Kreuzungen lässt sich mit Signalanlagen abbilden. Die Trajektorie fungiert als Referenz zur Positionierung von Objekten mittels linearer Referenzierung und zur geometrischen Repräsentation einer Straße. Letztere lässt sich mit lateralen Querschnitten oder parametrisch über die Fahrbahnbreite beschreiben. Für eine flächenhafte Darstellung besteht eine Schnittstelle zu Curved Regular Grids (CRG), was eine hoch detaillierte Wiedergabe einer Straßenoberfläche ermöglicht. Empfohlen wird dabei die Verwendung des OpenCRG Formates.

RoadXML 2.4.1 - Road Network Description

Wie Open Drive ist RoadXML ein Format ursprünglich entwickelt für Fahrsimulationen. Die Daten zur Beschreibung eines Straßennetzwerkes sind in fünf Layern gegliedert, welche Verkehrsdaten, Oberflächendaten, topologische Daten, Geräuschdaten und Nutzerdaten umfassen [Ducloux et al., 2016]. Der Aufbau eines RoadXML Netzwerkgraphen ist in Anhang A.7 abgebildet. Ein Straßennetzwerk wird weiter in Teilbereiche - sogenannte “Subnetworks“ - unterteilt. Ähnlich zu Open Drive werden Abbiegerelationen spurgenau abgebildet. Fahrspurwechsel sind über die Semantik der Spurmarkierungen definiert. Explizit modellieren lassen sich Vorfahrts- und Verkehrsregelungen sowie Zufahrtsbeschränkungen mit Hilfe von Straßenschildern und Signalanlagen. Die Straßengeometrie quer zur Fahrtrichtung wird als Profil beschrieben, welches sich aus den Elementen Fahrspur und Fahrspurrand zusammensetzt. Die Trajektorie ist in 3D durch eine “XYCurve“ und eine “SZCurve“ definiert, welche durch einzelne Geradensegmente, Kreisbögen oder Klothoiden beschrieben werden. Objekte werden über lineare Referenzierung entlang der Trajektorie positioniert. Zur Visualisierung besteht die Möglichkeit Texturen und 3D-Modelle einzubinden.

Verkehr in Städten Simulation - PTV Vissim

Vissim ist ein mikroskopisches Simulationsprogramm zur Nachbildung von Stadtverkehr und Außerortsverkehr sowie von Fußgängerströmen [PTV, 2016]. Die Möglichkeiten, die die Software bietet, gehen zum Teil über die im vorhergehenden Abschnitt gestellten Anforderungen hinaus. Für eine mesoskopische Simulation genügt ein einfaches Knoten-Kanten-Modell. Jede Kreuzung kann somit als ein Knoten abgebildet werden. Für mikroskopische Simulationen ist die genaue Verkehrsführung inklusive Fahrspurverknüpfungen zu modellieren. In Vissim bestehen beide Möglichkeiten der topologischen Netzmodellierung. Für ein Straßensegment lässt sich unter anderem die Darstellungsart (Oberfläche), Funktion und die Anzahl an Fahrspuren einstellen. Explizit können Zufahrtsbeschränkungen, Verkehrsregeln, Vorfahrt und Spurwechsel modelliert werden. Die Straßengeometrie wird parametrisch definiert. Die exakte Trajektorie kann über Spline-Interpolation angenähert werden. Ausgehend von der Achsrepräsentation führen Angaben zur Spurbreite zu einer flächenhaften Darstellung. Ein laterales Profil der Straßen ist nicht modellierbar. Recht umfangreich sind auch die Mög-

lichkeiten zur Visualisierung. Einige Funktionen wie Fahrbahnmarkierungen oder 3D-Signalanlagen sind Bestandteil der Software, weitere 3D-Modelle und externe Texturen und Grafiklayer lassen sich integrieren.

4.2.3 Vergleich und Schlussfolgerungen

Tabelle 4.1 fasst die Analyse der ausgewählten Modelle und Standards zusammen. Gegliedert nach den vier Ebenen sind die wichtigsten Bestandteile eines Verkehrsraummodelles für Verkehrssimulationen aufgeführt. Verfügt das jeweilige Modell über ein Element ist dies durch ein '+' gekennzeichnet und grün hinterlegt. Gelb '(+)' entspricht einem bedingten Vorhandensein, ein rotes '-' kennzeichnet das Fehlen des entsprechenden Kriteriums im jeweiligen Modell. Aus der Gegenüberstellung lassen sich die diskutierten Stärken und Schwächen der einzelnen Modelle ableiten. Allgemein wenig verwunderlich ist, dass die Modelle und Standards, die im Bereich Simulation angesiedelt sind, den Anforderungen in hohem Maße entsprechen, während die Modelle der linken Tabellenseite Einschränkungen aufweisen. Insbesondere auf der für Simulationen entscheidenden topologischen Ebene werden Unterschiede deutlich. LandInfra stellt gar keine Netzwerk-Repräsentation, CityGML und INSPIRE verfügen über ein Netzwerk im Sinne eines einfachen topologischen Graphen. GDF, OpenDrive, RoadXML und die Verkehrssimulation Vissim auf der anderen Seite beinhalten allesamt Konzepte, die über ein einfaches Knoten-Kanten-Modell hinausgehen. In RoadXML wird der Begriff "Subnetwork" eingeführt, der die Idee der Topologie-Modellierung gut auf den Punkt bringt. Ein Knotenpunkt als Repräsentant für eine Kreuzung lässt sich demnach weiter spezifizieren durch ein Teilnetz, welches sämtliche Abbiegerelationen der Kreuzung spurgenau abbildet. Der Knoten im Sinne eines topologischen Graphen wird ersetzt durch zusätzliche Kanten (Konnektoren), die die eingehenden und ausgehenden Straßen(kanten) verknüpfen. Abbildung 4.5 stellt ein einfaches Knoten-Kanten-Modell dem topologischen Modell von Routing- und Simulationsanwendungen am Beispiel der Umsetzung in Open Drive (vgl. [Dupuis, 2015, S.26]) gegenüber.

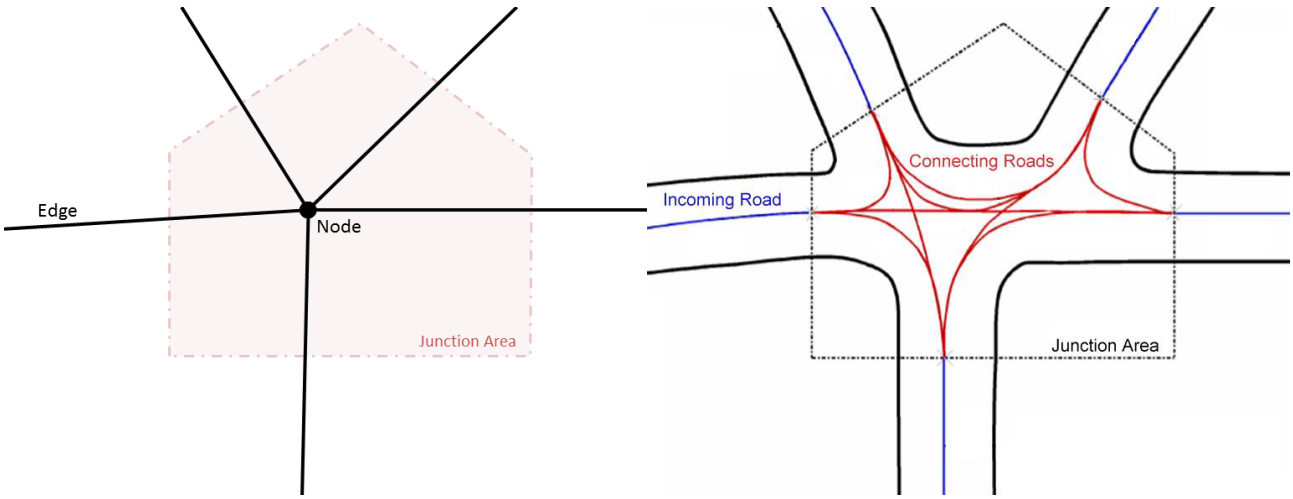


Abbildung 4.5 – Vergleich der Netzwerk-Topologie: Einfaches Knoten-Kanten Modell vs. detaillierte Kreuzungsmodellierung am Beispiel Open Drive (rechte Grafik aus [Dupuis, 2015, S.26])

Auch die semantische Beschreibung einer Verkehrskreuzung ist in den Modellen der rechten Tabellenseite umfangreicher. Während sich LandInfra, CityGML und INSPIRE weitestgehend auf eine Charakterisierung der Teilobjekte einer Kreuzung beschränken, sind in den anderen Modellen den Verkehrsfluss betreffende Informationen wie beispielsweise Geschwindigkeitsbegrenzungen oder Vorfahrtsregelungen inbegriffen. In Sachen Geometrie und visueller Ebene lässt sich kein klarer Trend zwischen linker und rechter Tabellenseite erkennen.
Im Gegensatz zu LandInfra und INSPIRE sticht CityGML unter den Modellen der linken Tabellenseite heraus, da es als einziger Standard alle vier Ebenen abdeckt. Dies und der Umstand, dass CityGML als Speicherformat für 3D-Stadtmodelle etabliert ist, macht eine Kopplung mit Verkehrssimulationen interessant. Sind für ein Simulationsgebiet keine simulationsspezifischen Verkehrsraumdaten vorhanden, kann durch eine Kopplung mit CityGML der aufwändige Schritt der Netzwerkdatengenerierung erleichtert werden. Für den Fall, dass ein Simulationsnetz bereits existiert, kann CityGML ebenso hinzugezogen werden, um beispielsweise die Visualisierung einer Verkehrssimulation zu unterstützen.

	LandInfra	CityGML	INSPIRE	GDF	OpenDrive	RoadXML	Vissim
Appearance/ Visual Layer							
Texture	-	+	-	(+)	-	+	+
3D-Models	-	+	-	-	-	+	+
Street Furniture	-	+	-	+	+	+	+
Road Marking	-	+	-	+	+	+	+
Geometrical/ Physical Layer							
Point-Objects	+	+	+	+	+	+	+
Line-Rep	+	+	+	+	+	+	+
Surface-Rep	+	+	+	(+)	(+)	-	+
Parametric-Rep	+	-	+	+	+	+	+
Trajectory	-	-	-	-	+	+	(+)
Linear Referencing	+	-	+	+	+	+	+
Lateral Profile	+	-	-	-	+	+	-
Semantical/ Logical Layer							
Surface Material	+	+	+	+	+	+	+
Function	+	+	+	+	+	+	+
Access	-	+	+	+	+	+	+
Driving Lanes	-	+	+	+	+	+	+
Driving Direction	-	-	+	+	+	+	+
Traffic Rules	-	-	+	+	+	+	+
Priority	-	-	-	+	+	+	+
Lane Change	-	-	-	+	+	+	+
Topological Layer							
Network							
Nodes	-	(+)	+	+	+	+	+
Edges	-	+	+	+	+	+	+
Subnetwork							
Road-Linkage	-	-	-	+	+	+	+
Lane-Linkage	-	-	-	+	+	+	+
Neighbours	-	-	-	+	+	-	+
Additional Information							
Markup Language	GML	GML	GML	-	XML	XML	XML
Developer	OGC	OGC	TWG	ISO/TC	VIRES	OKTAL	PTV

Tabelle 4.1 – Vergleich verschiedener Verkehrsraummodelle - rot: fehlend, gelb: bedingt vorhanden, grün: vorhanden

4.3 Exploration der Arbeit zugrunde liegender Verkehrsraummodelle

Im Laufe der Arbeit wurde bereits mehrfach auf die Verkehrsraummodelle der Verkehrssimulation Vissim und des OGC Standards CityGML eingegangen. Als Grundlage für die nachfolgende Konzeptentwicklung einer Ableitung von Simulationsgraphen für Vissim aus CityGML sollen an dieser Stelle beide Formate detailliert betrachtet werden. Ausgehend vom konzeptionellen Aufbau beider Verkehrsraummodelle werden ihre Elemente und die jeweilige Formatstruktur der zentralen Bestandteile vorgestellt.

4.3.1 Das VISSIM-Verkehrsmodell

Das Kernelement der Verkehrssimulation Vissim bildet das “Network“, welches aus einer Vielzahl von Komponenten aufgebaut wird. Für Vissim als kommerzielle Software ist im Gegensatz zu anderen Standards oder Modellen kein Diagramm verfügbar. Abbildung 4.6 versucht die wichtigsten Elemente in Form eines UML-Diagramms wiederzugeben. Die Grafik beinhaltet keinesfalls alle Komponenten und Möglichkeiten, die Vissim zur Verfügung stellt, und soll lediglich einen Überblick verschaffen. Gliedern lassen sich die aufgeführten Elemente nach den Bereichen Simulation, Grafik, Verkehrsteilnehmer, Fahrzeugmodellierung und Fußgängermodellierung.
Einfluss auf die Simulationsauswertung lässt sich mit Hilfe der Komponenten “Sections“ und “Nodes“ nehmen. Ausschnitte (“Sections“) ermöglichen es Teile des Simulationsnetzes auszuwerten. Dies spielt insbesondere bei hybriden Simulationsläufen eine Rolle. So können bestimmte Bereiche mikroskopisch simuliert werden, das restliche Netz mesoskopisch. In diesem Zusammenhang steht auch die Definition von Knotenpunkten (“Nodes“). Dadurch kann die Komplexität des Netzmodelles auf ein einfaches Knoten-Kanten-Modell heruntergebrochen werden und so Ausgang für dynamische Umlegung und mesoskopische Simulationen sein. Die dynamische Umlegung ist eine weitere Simulationsform, die an Stelle von Fahrzeugflüssen und Verkehrsstärke Quell-Ziel-Matrizen einführt.
Zu den Grafik-Elementen gehören statische 3D-Modelle (Sketch-Up-Dateien oder Autodesk-Dateien), Hintergründe (Karten, Satellitenbilder, Grafiken), 3D-Modelle von Ampelanlagen sowie Fahrbahnmarkierungen (Zebrastreifen, Abbiegepfeile). Die genannten Elemente nehmen keinen Einfluss auf das Simulationsverhalten, sondern dienen ausschließlich der Visualisierung.
Vissim ermöglicht die Simulation von multimodalem Verkehr. Dazu ist eine Unterscheidung verschiedener Verkehrsteilnehmer notwendig. Für Fußgänger stehen vier verschiedene Typen zur Verfügung. Die Fahrzeugmodellierung schließt die Kategorien Autos, Lastkraftwägen, Busse, Trambahnen, Fußgänger und Fahrradfahrer ein.
Das eigentliche Straßennetz wird durch das Element “Links“ beschrieben. Straßensegmente werden als “Link“ mit Attributen wie Länge, Anzahl an Fahrspuren oder Fahrspurbreite beschrieben. Konnektoren sind Verbindungsstrecken. Sie dienen der Verknüpfung zweier Strecken sowie der Abbildung von Abbiegerelationen an Kreuzungspunkten. Eine detaillierte Formatbeschreibung u.a. der Geometrierepräsentation von Strecken und Verbindungsstrecken findet sich im Anschluss an die allgemeine Aufführung der Bestandteile von Vissim.

Der Straßenraum wird durch zahlreiche weitere Komponenten wiedergegeben. Mit Wunschgeschwindigkeitsentscheidungen (“DesiredSpeedDecisions“) und Langsamfahrbereichen (“ReducedSpeedAreas“) lässt sich die Fahrgeschwindigkeit beeinflussen. Konfliktflächen (“ConflictAreas“) dienen der Modellierung der Vorfahrtsregelung an Kreuzungen. Vorfahrtsregelungen (“PriorityRules“) lassen sich auch über Querverkehrsstörungen modellieren. Vorteil der Konfliktflächen ist, dass das Fahrverhalten besser abgebildet wird, da Fahrer in Konfliktflächen planen, wie sie diese durchqueren. Stoppschilder (“StopSigns“) und Lichtsignalanlagen (“SignalHeads“) vervollständigen die umfangreichen Möglichkeiten zur Vorfahrtsmodellierung. Mit Hilfe von Detektoren lassen sich verkehrsabhängige Signalsteuerungsverfahren, wie z.B. Induktionsschleifen, umsetzen. Vergleichbare Sensoren können auch zur Verkehrsmengenerfassung eingesetzt werden. Dazu gibt es in Vissim Messquerschnitte (“DataCollectionPoints“). Weitere Werkzeuge der Simulationsdatenerfassung stellen Fahrzeugreisezeitmessungen (“VehicleTravelTimes“) und Stauzähler (“QueueCounter“) dar. Erstere erfassen die Zeit, die ein Fahrzeug von einem Start- zu einem Zielquerschnitt benötigt. Letztere können an Haltelinien von Knotenpunkten definiert werden und sammeln Informationen zur Anzahl und Länge von Staus. Die Anzahl an Fahrzeugen im Netz wird über Fahrzeugzuflüsse (“VehicleInputs“) geregelt, welche in Fahrzeugen pro Stunde definiert werden. Fahrzeugzuflüsse können nur auf Strecken platziert werden. Fahrzeugrouten (“VehicleRoutes“) dienen der Steuerung von Fahrzeugen. Sie können für Abbiegerelationen an einzelnen Kreuzungen oder über mehrere Knotenpunkte hinweg eingesetzt werden. Auch das Parkverhalten und öffentliche Verkehrslinien werden über Fahrzeugrouten geregelt. Parkmöglichkeiten (“ParkingLots“) können am Straßenrand in Form einer Parkspur oder als Parkplätze modelliert werden. Der öffentliche Verkehr wird mit Hilfe der Komponenten ÖV-Linien (“PublicTransportLines“) und Haltestellen (“PublicTransportStops“) modelliert.

Für die Fußgängermodellierung bietet PTV eine eigenständige Software namens VisWalk. Jedoch ist auch in Vissim standardmäßig eine Fußgängermodellierung möglich. Fußgänger und Fahrzeuge gleichzeitig lassen sich jedoch nur mit dem Zusatzmodul VisWalk simulieren. Fußgänger können sich auf Strecken (“Links“) bewegen, falls diese als Fußgängerfläche deklariert sind. Ansonsten stehen Konstruktionselemente zur Verfügung, um Fußgängerbereiche zu gestalten. Zu den begehbaren Objekten zählen Flächen (“Areas“), Rampen (“Ramps“), Treppen (“Stairs“) und Aufzüge (“Elevators“). Hindernisse (“Obstacles“) stellen nicht begehbare Flächen dar.

Für Konstruktionselemente bestehen verschiedene Importfunktionen. Linien und Polygone lassen sich nach VisWalk aus AutoCAD importieren. Eine weitere Schnittstelle besteht zum IFC-Standard Building Information Modeling (BIM). Dies ermöglicht für Fußgänger relevante Teile eines Gebäudes automatisch in Konstruktionselemente zu überführen und für eine Fußgängersimulation zu nutzen.

Die Komponenten Zuflüsse (“PedestrianInputs“), Routenentscheidungen und Routen (“PedestrianRoutes“) steuern Fußgänger entsprechend ihrer Pendants bei der Fahrzeugmodellierung. Eine Fußgängerreisezeitmessung (“PedestrianTravelTimes“) erfasst die benötigte Zeit eines Fußgängers von einem Start- zu einem Endpunkt. Weitergehende Informationen zu einzelnen Komponenten können dem Vissim-Handbuch entnommen werden (vgl. [PTV, 2016]).

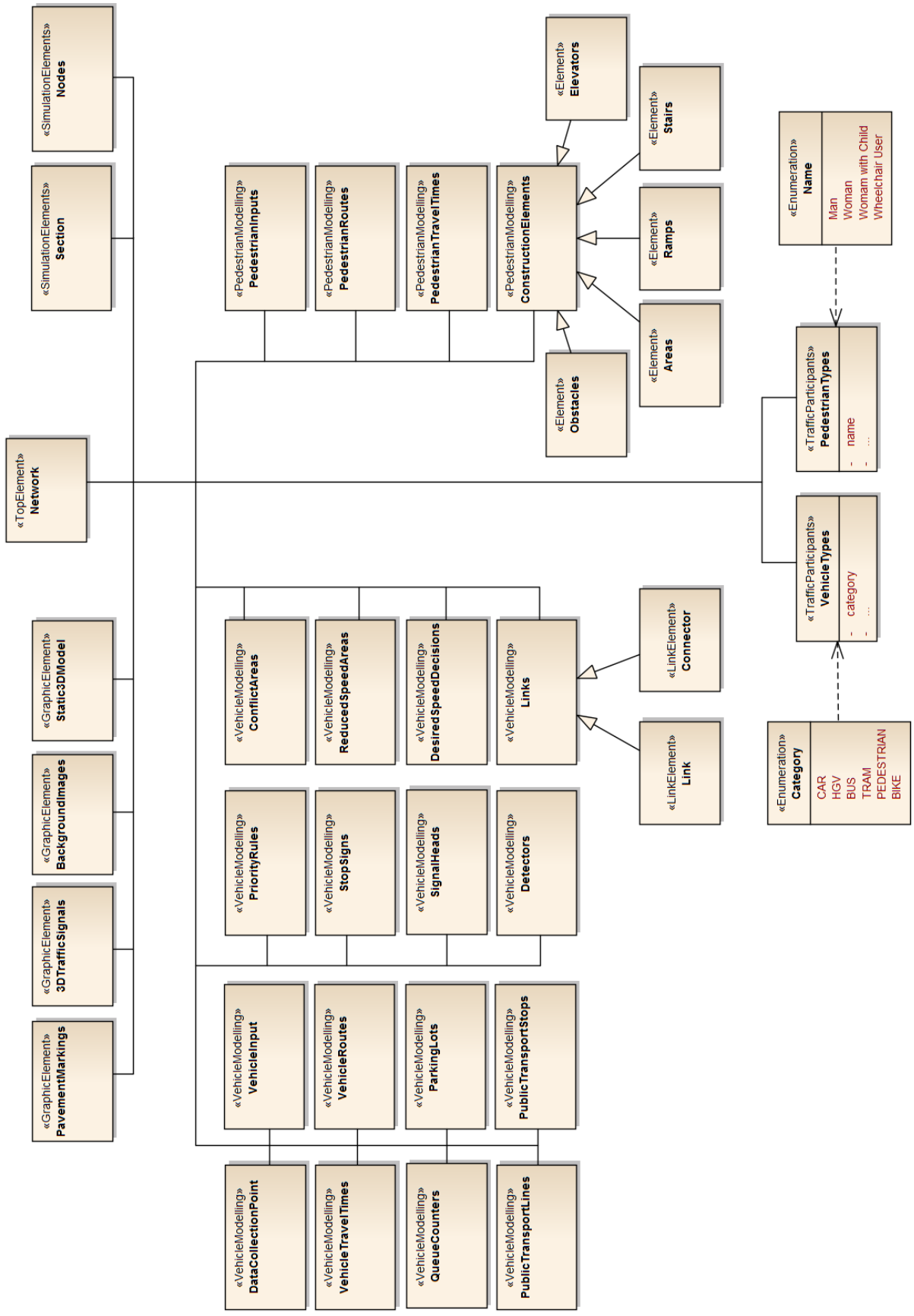


Abbildung 4.6 – UML-Diagramm der wichtigsten Komponenten der Verkehrssimulation Vissim

Alle Daten eines Simulationsprojektes werden in einem PTV-eigenen, xml-basierten Format mit der Dateiendung ".inpx" abgespeichert. Wie beschrieben sind die wesentlichen Bestandteile eines Vissim-Straßennetzes Strecken ("Link") und Verbindungsstrecken ("Konnektor") [PTV, 2016]. Jedes dieser Teilstücke ist unter einem Markup "link" explizit durch beschreibende Attribute und seine Geometrie gespeichert. Eine Strecke (Abbildung 4.7) ist definiert durch Anfangs- und Endpunkt sowie eventuelle Zwischenpunkte. Die Koordinaten dieser Punkte werden im Element "Points3D" als einzelne Punkte mit ihren x, y und z-Koordinaten gespeichert. Das zugrundeliegende Koordinatensystem ist ein kartesisches und basiert auf der Sphere-Merkator Projektion. Die Einheit ist metrisch [PTV, 2016]. Neben dem Achsverlauf ist die Anzahl an Fahrstreifen und deren Breite festzulegen. Zu den Attributen, die im Start-Tag "link" aufgeführt werden, zählen unter anderem eine Identifikationsnummer ("no"), der Straßenname ("name"), die Vorausschauweite für Überholvorgänge ("lookAheadDistOvt") oder die Steigung ("gradient"). Insgesamt werden 30 zum Teil optionale Attribute aufgeführt. Der Übersicht halber wurden in Abbildung 4.7 auf eine vollständige Auflistung aller Attribute verzichtet. Eine Übersicht über alle Attribute findet sich im Anhang in Tabelle B.1. Das in Abbildung 4.7 aufgeführte Beispiel beschreibt eine Strecke der Länge 25,0 m mit drei Fahrspuren der Breite 3,5 m in einem lokalen Koordinatensystem.

```
<link assumSpeedOncom="60" costPerKm="0" direction="ALL" displayType="1" . . . vehRecAct="true"/>
        <geometry>
                <points3D>
                        <point3D x="-20" y="0" zOffset="0"/>
                        <point3D x="5" y="0" zOffset="0"/>
                </points3D>
        </geometry>
        <lanes>
                <lane width="3.5"/>
                <lane width="3.5"/>
                <lane width="3.5"/>
        </lanes>
</link>
```

Abbildung 4.7 – XML-Struktur einer Strecke (Link) in Vissim

Mit Hilfe von Verbindungsstrecken werden Abbiegemöglichkeiten sowie Zusammenführen bzw. Aufweitungen von Fahrstreifen modelliert [PTV, 2016]. In den Tags "fromLinkEndPt" bzw. "toLinkEndPt" werden die Relationen zwischen den zu verbindenden Fahrspuren und die Anfangs- und Endposition des Konnektors fixiert (vgl. Abbildung 4.8). Das Attribut "Lane" besteht aus zwei Nummern, die durch ein Leerzeichen getrennt werden. Die erste Zahl identifiziert die Nummer (Attribut "no") der zu verbindenden Strecken, die zweite Zahl die Fahrspur ("Lane"). Das Attribut "pos" beschreibt die Position entlang der jeweiligen Strecken. Die Geometrie der Konnektoren wird in entsprechender Weise zu den Strecken definiert. Verbindungsstrecken enthalten mit Ausnahme des Attributes "Level" die selben Attribute wie Strecken. Das Beispiel in Abbildung 4.8 zeigt einen Konnektor, der die Fahrspuren "1, 2 und 3" einer Strecke "1" mit den Spuren "2, 3 und 4" einer Strecke "2" verknüpft. Angegeben wird die jeweils kleinste Spurnummer der Relation und die Anzahl an Fahrspuren, die der Konnektor besitzt. Der erste Punkt der Geometrie eines Konnektors muss innerhalb der referenzierten Strecke "fromLinkEndPt" liegen, der letzte Punkt innerhalb der Strecke "toLinkEndPt".

Es ist jedoch nicht zwingend notwendig, dass erster bzw. letzter Punkt dem Start- bzw. Endpunkt der jeweiligen Strecke entsprechen.

```
<link assumSpeedOncom="60" costPerKm="0" direction="ALL" displayType="1" . . . vehRecAct="true"/>
       <fromLinkEndPt lane="1␣1" pos="22"/>
       <geometry>
              <points3D>
                     <point3D x="3" y="0" zOffset="0"/>
                     <point3D x="20" y="0" zOffset="0"/>
              </points3D>
       </geometry>
       <lanes>
              <lane/>
              <lane/>
              <lane/>
       </lanes>
       <toLinkEndPt lane="2␣2" pos="4"/>
</link>
```

Abbildung 4.8 – XML-Struktur einer Verbindungsstrecke (Connector) in Vissim

Abbildung 4.9 zeigt die Geometrie und Topologie eines möglichen Szenarios in Vissim. Der Link aus Abbildung 4.7 ist orange hervorgehoben. Es folgt der Konnektor aus Abbildung 4.8. Im unteren Teil ist die Topologie des Netzes wiedergeben. Strecken sind dunkelblau, Verbindungsstrecken magenta dargestellt.

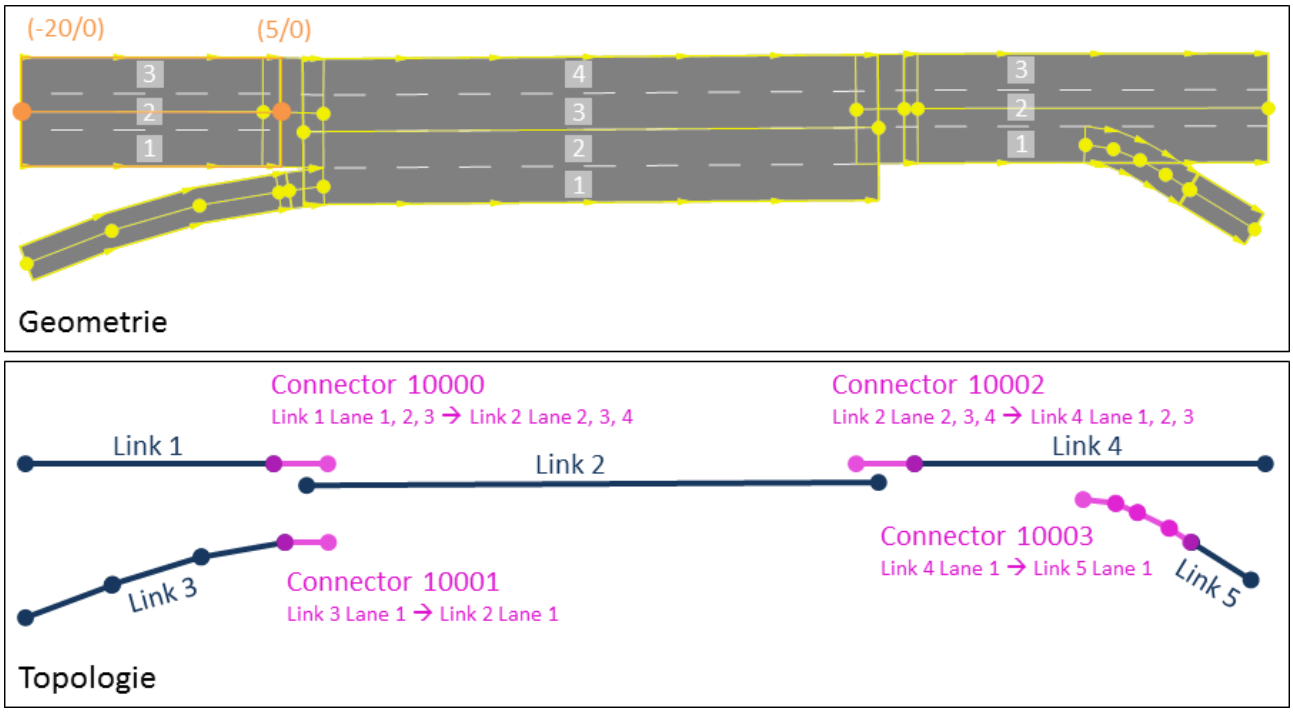


Abbildung 4.9 – Beispiel für Repräsentation der Geometrie (oben) und Topologie (unten) eines Szenarios in Vissim mit Links und Konnektoren

Einen Überblick über alle auftretenden Attribute und deren Bedeutung gibt Tabelle B.1 (Anhang). Die erste Spalte beinhaltet das XML-Element, in dem das jeweilige Attribut auftritt. Zu den 30 Attributen aus dem Start-Tag “Link“ kommen weitere 8 Attribute aus den Tags “Point3D“, “lanes“, “fromLinkEndPt“ und “toLinkEndPt“. Die dritte Spalte beinhaltet eine Beschreibung der Attribute. Daneben wird eine Einteilung vorgenommen, ob es sich beim entsprechenden Attribut um einen Einstellparameter der Simulation (SIM) oder einen Netzwerkparameter (NETZ) handelt. 20 der insgesamt 38 Attribute werden der Kategorie “SIM“ zugeteilt. Darunter fallen beispielsweise die angenommene Geschwindigkeit des Gegenverkehrs, das Fahrverhalten für eine Strecke oder verschiedene Optionen, die die Auswertung der Simulation betreffen. Die übrigen 18 Attribute beschreiben das Verkehrsnetz. Unter anderem sind hierbei die Nummer und der Name genauso wie die Lage, Steigung oder Segmentlänge einer Strecke bzw. Verbindungsstrecke enthalten.

4.3.2 Das CityGML-Verkehrsmodell (Transportation Objects)

Abbildung 4.10 zeigt das UML-Diagramm des CityGML Transportation Moduls. Die zentrale Klasse “_TransportationObject“ wird durch drei Unterklassen genauer spezifiziert. Ein “TransportationComplex“ steht für einen Verkehrskomplex, der sich bei einer detailreichen Beschreibung in LoD 2-4 aus der Verkehrsfläche (“TrafficArea“) und Nebenflächen (“AuxiliaryTrafficArea“) aggregiert. Weitere Spezialisierungen stellen die Klassen “Track“, “Road“, “Square“ und “Railway“ dar.

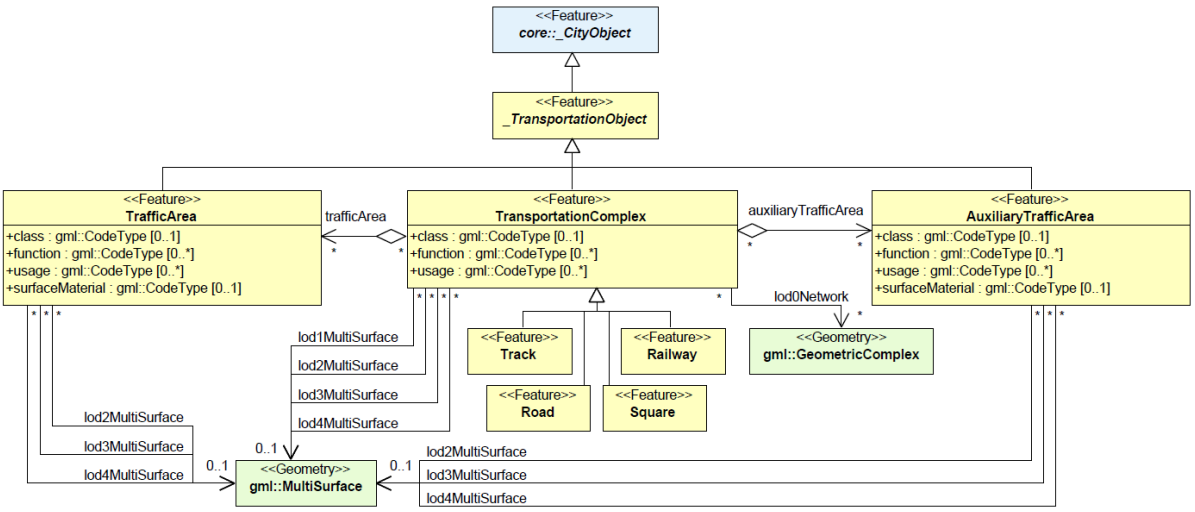


Abbildung 4.10 – UML-Diagramm des Transportation Moduls in CityGML [Gröger et al., 2012, S.125]

Jedes Objekt der Klasse “TransportationComplex“ besitzt als Unterklasse der GML-Klasse “Feature“ die Attribute “class“, “function“ und “usage“. Während “function“ den ausgewiesenen Zweck des Verkehrsraumelements wie zum Beispiel Fußweg, Radweg, Parkplatz oder Autobahn beschreibt, gibt “usage“ die tatsächliche Nutzung an, sofern diese abweicht. Objekte der Klassen “TrafficArea“ und “AuxiliaryTrafficArea“ (LoD 2-4) haben ebenfalls die Attribute “class“, “function“ und “usage“, wobei sich die Codeliste von der des “TransportationComplex“ unterscheidet. Die Funktion gibt den Zweck

der Nutzung (z.B. Gehweg, Radweg, Fahrspur, etc.) an, wohingegen hier das Attribut “usage“ die befugten Nutzer (z.B. Fußgänger, Fahrrad, Auto, etc.) spezifiziert. Zusätzlich kann die Oberflächenbeschaffenheit (z.B. Asphalt, Beton, Schotter, etc.) im Attribut “surfaceMaterial“ beschrieben werden. Straßennamen können im Attribut “gml:name“ abgespeichert werden, welches “TransportationComplex“ als Unterklasse von “CityObject“ erbt. Weitere funktionale Aspekte wie zum Beispiel Geschwindigkeitsbegrenzungen sind nicht Teil von CityGML. Allerdings besteht die Möglichkeit, zusätzliche Metadaten durch generische Attribute nach den vordefinierten Katalogen hinzuzufügen. Zudem können Daten aus anderen Informationensystemen über externe Referenzen verlinkt werden. Jedes “CityObject“ erlaubt eine beliebige Anzahl von Links über Uniform Resource Identifier (URI) auf korrespondierende Objekte in externen Informationssystemen [Gröger et al., 2012].
Die geometrische Darstellung hängt von der vorhandenen LoD-Stufe ab. Abbildung 4.11 stellt die verschiedenen Detaillierungsgrade an einem Autobahnabschnitt gegenüber. In der einfachsten Repräsentation (LoD 0) besteht der “TransportationComplex“ aus Linienobjekten, die aggregiert ein Lineares Netzwerk ergeben. Dies reicht aus, um Routing Algorithmen oder vergleichbare Analysen ausführen zu können. Ab LoD 1 erfolgt eine flächenhafte Darstellung als “gml::MultiSurface[s]“. LoD 2-4 modellieren explizit die thematischen Teilelemente wie Standstreifen, Fahrbahnen oder Grünflächen [Gröger et al., 2012].

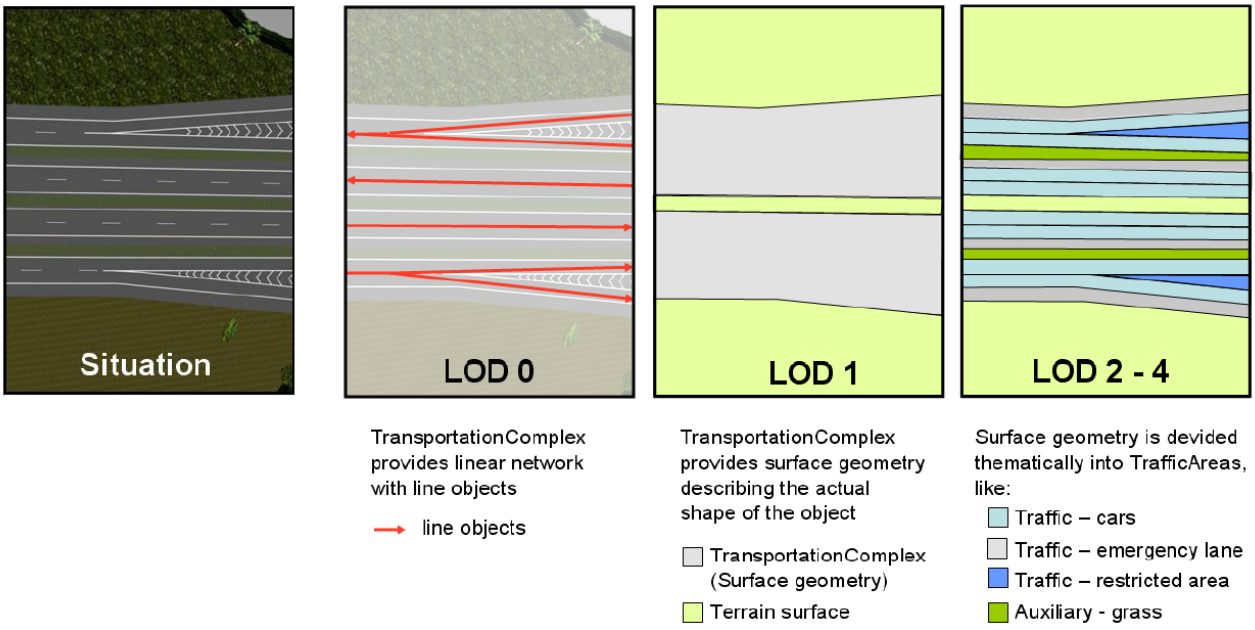


Abbildung 4.11 – Geometrische Repräsentation eines “TransportationComplex“ in verschiedenen Level of Details [Gröger et al., 2012, S.126]

Ein konkretes Beispiel, vergleichbar zu den vorangegangenen Beispielen in Vissim, zeigt Abbildung 4.12. Der Ausschnitt aus einem CityGML-Code beschreibt eine 10 m lange, gerade Straße im LoD 0. Verwendung finden die Namensräume “citygml“ (Core-Modul), “tran“ (Transportation Objects) und “gml“ (GML), die am Anfang eines CityGML-Dokumentes definiert werden. Eine Straße ist wie alle Elemente eines CityGML-Modells ein “cityObjectMember“. Das Beispiel ist als “Road“ und damit als

Spezialisierung eines “TransportationComplex“ modelliert. Da die Modellierung in LoD 0 erfolgt, gibt es keine Unterteilung in “TrafficArea“ und “AuxiliaryTrafficArea“ (vgl. Abbildung 4.10). Beschrieben wird die Straße durch die GML-Attribute “Name“ (Testroad 1), “class“ (1040 = *road traffic*), “function“ (1000 = *road*) und “usage“ (1000 = *road*). Die Geometrie ist im LoD 0 als lineares Netzwerk wiedergegeben. Definiert wird sie durch die GML-Geometrie-Klasse “LineString“, angegeben als Koordinaten-Liste (“posList“).

```
<citygml:cityObjectMember>
	<tran:Road>
		<gml:name>Testroad 1</gml:name>
		<tran:class>1040</tran:class>
		<tran:function>1000</tran:function>
		<tran:usage>1000</tran:usage>
		<tran:lod0Network>
			<gml:LineString srsDimension="3">
				<gml:posList>-5 0 0 5 0 0</gml:posList>
			</gml:LineString>
		</tran:lod0Network>
	</tran:Road>
</citygml:cityObjectMember>
```

Abbildung 4.12 – CityGML-Struktur einer Straße in LoD 0

4.4 Vorschläge zur Verbesserung des CityGML Transportation Moduls

CityGML als Standard zur Speicherung und zum Austausch von 3D-Stadtmodellen bietet Potenziale für eine Vielzahl an Anwendungen, so auch für den Anwendungsfall Verkehrssimulation. Eine Ableitung von Simulationsgraphen für Vissim aus CityGML scheint vielversprechend. Jedoch limitieren einige Mängel des aktuellen Verkehrsraummodell (“Transportation Objects“) die Aussichten auf einen hochwertigen Simulationsgraphen. Eine signifikante Verbesserung der Ergebnisse wird durch Erweiterung des Transportation Moduls erwartet. Dieser Abschnitt befasst sich mit der Frage, wie das CityGML Verkehrsraummodell verbessert werden könnte, um für Anwendungen wie Verkehrssimulationen bereits standardmäßig noch mehr Möglichkeiten zu bieten.
Vergleicht man CityGML mit dem abstrakten Modell aus Abbildung 4.2, lässt sich eine hohe Korrelation in der mittleren Ebene rund um die Gliederung des Verkehrsraumes in Verkehrsfläche und Nebenflächen erzielen. Große Unterschiede treten in den Ebenen darunter auf. Das CityGML Modell ist auf vier semantische Klassen beschränkt. Das abstrakte Modell differenziert in elf unterschiedliche Objektklassen. In CityGML können Elemente der untersten Ebene ebenfalls modelliert werden, als Objekte der Klasse “TransportationComplex“ versehen mit dem entsprechenden “function“-Attribut. Eine Fahrspur, modelliert in CityGML, ist demnach ein Objekt der Klasse “TransportationComplex“ mit der Funktion Fahrspur. Die Codeliste “TransportationComplex function and usage“ stellt folgende Codes für diesen Zweck bereit: 1400 = ’lane’, 1410 = ’lane, one direction’, 1420 = ’lane, both direction’. Die Geometrie dieser Fahrspur ließe sich flächenhaft als “TrafficArea“ mit der Funktion ’driving_lane’ (Codeliste “TrafficArea - function“ = 1) modellieren. Ungeklärt ist, in welchem Zusammenhang ein “Road“-Objekt charakterisiert mit der Funktion ’road’ zu seinen Fahrspuren steht. Nachvollziehbar

wäre, dass die Verkehrsfläche 'driving_lane' im Sinne der Aggregation Teil eines Road-Objektes in LoD 2-4 darstellt. Ein "TransportationComplex" mit der Funktion 'lane' könnte ein Netzwerk in LoD 0 spezifizieren. Derartige Restriktionen oder Regelungen finden sich im Standard jedoch nicht.
Vorteil zusätzlicher Objektklassen wäre neben einer Reduzierung der derzeitig zur semantischen Beschreibung von Objekten notwendigen Codes, dass Relationen zwischen den Objekten explizit modelliert werden. Im UML-Diagramm des abstrakten Verkehrsraummodells wie auch in den UML-Diagrammen anderer Standards (u.a. GDF - vgl. Anhang A.5) werden auf diese Weise topologische Informationen repräsentiert. Die Aggregation von Straßensegmenten zu einer Straße enthält beispielsweise eine geordnete Sequenz aufeinanderfolgender Straßensegmente. Die Assoziationen "intersectingAt" bzw. "meetingAt" im abstrakten Modell beschreiben die Interaktion von Straßen bzw. Straßensegmenten mit Kreuzungen.
Für die Anwendung Verkehrssimulation unumgänglich ist eine Erweiterung des Network-Konzeptes zur Stärkung des bisher im CityGML Verkehrsmodell unterrepräsentierten Logikmodells. Im Rahmen des Themas "Detaillierte Repräsentation des Straßenraums in 3D-Stadtmodellen" diskutiert Beil [2017] Defizite des Transportmoduls in CityGML 2.0 und präsentiert Vorschläge zur Weiterentwicklung des Standards. Ein Kritikpunkt stellt die Modellierung des Straßenraumes als Netzwerk dar. So ist im Standard weder spezifiziert, ob Bestandsachsen (Mittelachse des gesamten Fahrbahnbereichs) oder Fahrbahnachsen (Mittelachse einer einzelnen Fahrspur) modelliert werden, noch, ob in die linienhafte Darstellung auch Knotenpunkte im Sinne einer Knoten-Kanten-Repräsentation Eingang finden können. Weiter ist nicht klar vorgegeben, ob Liniengeometrien mit der Fahrtrichtung korrelieren. Beil [2017] schlägt hinsichtlich der Netzwerkmodellierung vor, linienhafte Repräsentationen nicht auf LoD 0 zu beschränken, sondern auch für höhere LoDs neben einer flächenhaften Repräsentation zuzulassen. Dabei ist in LoD 0 und LoD 1 eine Modellierung der Bestandsachsen vorgesehen, in höheren LoDs werden die Fahrbahnachsen modelliert. Abbildung 4.13 nimmt diesen Vorschlag auf. Nach dem beschriebenen Netzwerkkonzept würden in LoD 0 die Bestandsachsen modelliert. In LoD 1 erfolgt der Übergang auf eine Achse pro Fahrtrichtung. Dabei sind die Punkte der Liniengeometrie in der Form anzugeben, dass die Anordnung mit der Fahrtrichtung korrespondiert. Ab LoD 1 wird ein Kreuzungspunkt nicht mehr durch einen einzelnen Knoten, sondern durch eine Untermenge an Knoten und Kanten (hellblau) modelliert. Diese Verbindung zwischen LoD0-Knoten und der Kreuzungsrepräsentation ab LoD 1 könnte durch Aufnahme der LoD0-Knoten-ID in den Elementen der Untermenge hervorgehoben werden. Höhere Level of Details ermöglichen die fahrspurspezifische Abbildung eines Netzwerkgraphen. Bestandteil des LoD 2-4 Konzeptes mit Fahrspurachsen sollte ein Schema zur Nummerierung der Fahrspuren eines Straßensegmentes sein. Eine Möglichkeit, wie dies umzusetzen wäre, liefert der OpenDrive-Standard (vgl. [Dupuis, 2015, S.20]). In besagtem Standard werden auch Vorgänger-Nachfolger-Relationen explizit modelliert. Einfache Straßen verfügen über genau einen Vorgänger und Nachfolger, welche als Attribute zur Verfügung gestellt werden. Kreuzungen sind aus sogenannten "Connecting Roads" aufgebaut und werden durch eine Vorgänger-Nachfolger-Matrix beschrieben [Dupuis, 2015, S.25f] . Die Übertragung eines derartigen Modells auf das CityGML-Konzept der XLinks könnte zu einem Logikmodell führen, welches dem Anforderungsprofil von Anwendungen wie Verkehrs- und Fahrsimulationen entspricht.

Eine parametrische Geometrie-Repräsentation wie in Vissim lässt sich basierend auf der Achsdarstellung bereits jetzt mit Hilfe von generischen Attributen realisieren. Eine Einführung von zusätzlichen Attributen zur parametrischen Beschreibung der Straßengeometrie ist somit nicht zwingend erforderlich. Ein wichtiges Konzept, was Bestandteil vergleichbarer Verkehrsraummodelle ist (vgl. Tabelle 4.1), stellt die Lineare Referenzierung dar. Durch Einführung in CityGML wird eine Positionierung von Objekten entlang der Achsen möglich, was in vielen Anwendungen (z.B. auch Verkehrssimulationen) eine Rolle spielt.

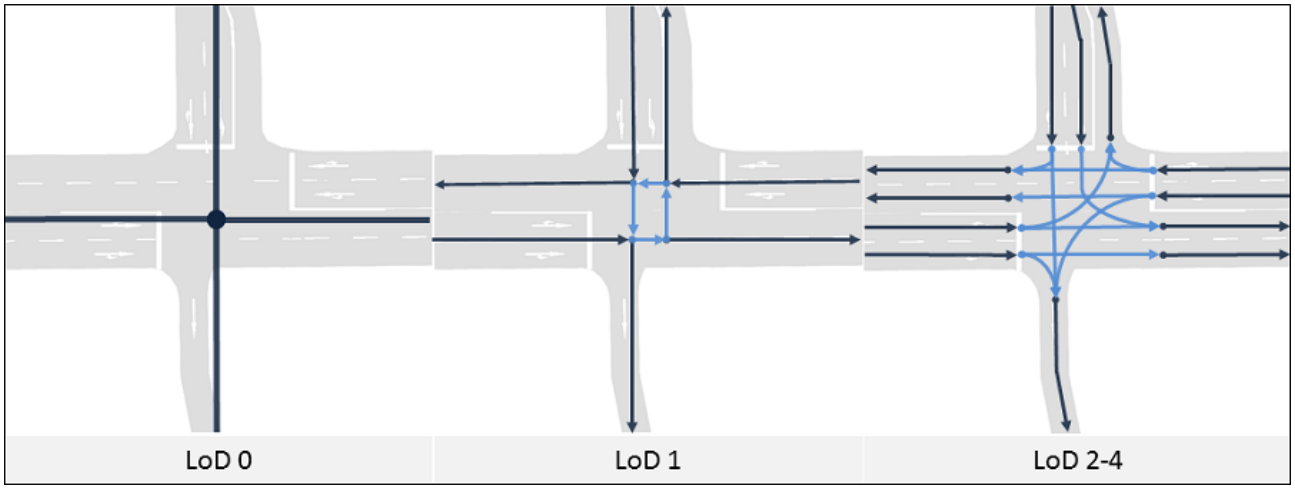


Abbildung 4.13 – Netzwerkrepräsentation in unterschiedlichen LoDs: Bestandsachsen in LoD 0, Achse pro Fahrtrichtung mit Verbindungskanten (hellblau) an Kreuzungen in LoD 1 und Fahrbahnachsen mit fahrspurspezifischen Verbindungskanten (hellblau) in LoD 2-4

5 Kopplung von Verkehrssimulation und Stadtmodellen auf Modellebene

Bei der Kopplung von Verkehrssimulation und semantischen 3D-Stadtmodellen soll das Stadtmodell zwei Funktionen einnehmen: als Informationsquelle und als Visualisierungsumgebung. Für die Umsetzung einer Kopplung ist eine Überführung von Konzepten des einen in das andere Modell notwendig. Für die Ableitung eines Simulationsgraphen gilt es die korrespondierenden Netzwerkelemente der beiden Modelle zu finden, um sie anschließend ineinander überführen zu können. Für die Visualisierung von Simulationsergebnissen im virtuellen 3D-Stadtmodell muss ein Konzept gefunden werden, um die dynamischen Verkehrsströme, die aus der Verkehrssimulation resultieren, im Modell des CityGML Standards zu integrieren.
In diesem Kapitel wird die Fragestellung beantwortet, wie sich Verkehrssimulation und Stadtmodell auf Modellebene verknüpfen lassen. Anschließend werden Potenziale und Herausforderungen diskutiert, die sich aus der erarbeiteten Kopplung ergeben.

5.1 Model Weaving

“Model Driven Engineering“ (MDE) ist ein Konzept zur automatisierten Ableitung von Programmen aus formalen Modellen. Im Gegensatz zur Objektorientierung, die dem Prinzip “Alles ist ein Objekt“ unterliegt, folgt MDE dem Leitsatz “Alles ist ein Modell“ [Bézivin, 2005, S.2]. Das grundsätzliche Verständnis des Modellbegriffes bleibt dabei das gleiche wie zu Anfang von Kapitel 4 definiert. Hinzukommt bei modernen MDE-Ansätzen, dass ein Modell nicht nur ein einfaches Abbild eines Systems ist, sondern zusätzlich eine formale Beschreibung liefert, die es Computerprogrammen ermöglicht bestimmte Operationen auszuführen. Dazu wird neben dem Modell ein Metamodell eingeführt, welches die Struktur eines Modells definiert. Der Zusammenhang zwischen System (Instanz), Modell und Metamodell ist in Abbildung 5.1 auf der rechten Seite dargestellt. Ein System im vorliegenden Fall am Beispiel eines Straßennetzes wird durch ein Modell repräsentiert (“representedBy“). Das Modell entspricht (“conformsTo“) der vorgegebenen Struktur eines Metamodells (vgl. [Bézivin, 2005, S.2ff]).
Übertragen auf die drei diskutierten Modelle folgt CityGML als Anwendungsschema von GML3 dem General Feature Model (GFM) der ISO 19109. Instanzen des Transportation Moduls können einzelne Road-Objekte sein. Vissim Netzwerke folgen ebenfalls einer vorgegebenen Struktur, welche allerdings nicht als Schema definiert oder zumindest nicht frei verfügbar ist. Instanzen stellen hier

einzelne Link-Objekte oder Konnektoren dar. Für das abstrakte Verkehrsraummodell existiert kein Metamodell, da das Modell lediglich einer grundlegenden Betrachtung des Themas Verkehrsraummodellierung dient. Wollte man ein Metamodell für das abstrakte Verkehrsraummodell einführen, müsste dieses die Struktur der realen Welt enthalten bzw. der Welt, wie sie im Modell abgebildet wurde. Der Vollständigkeit halber wird ein “Abstract Traffic Space Metamodell“ in die Abbildung aufgenommen und grau hinterlegt, wie das ebenfalls nicht explizit definierte Vissim Metamodell. Instanzen im Falle des abstrakten Modells stellen einzelne Straßensegmente und Kreuzungen dar.

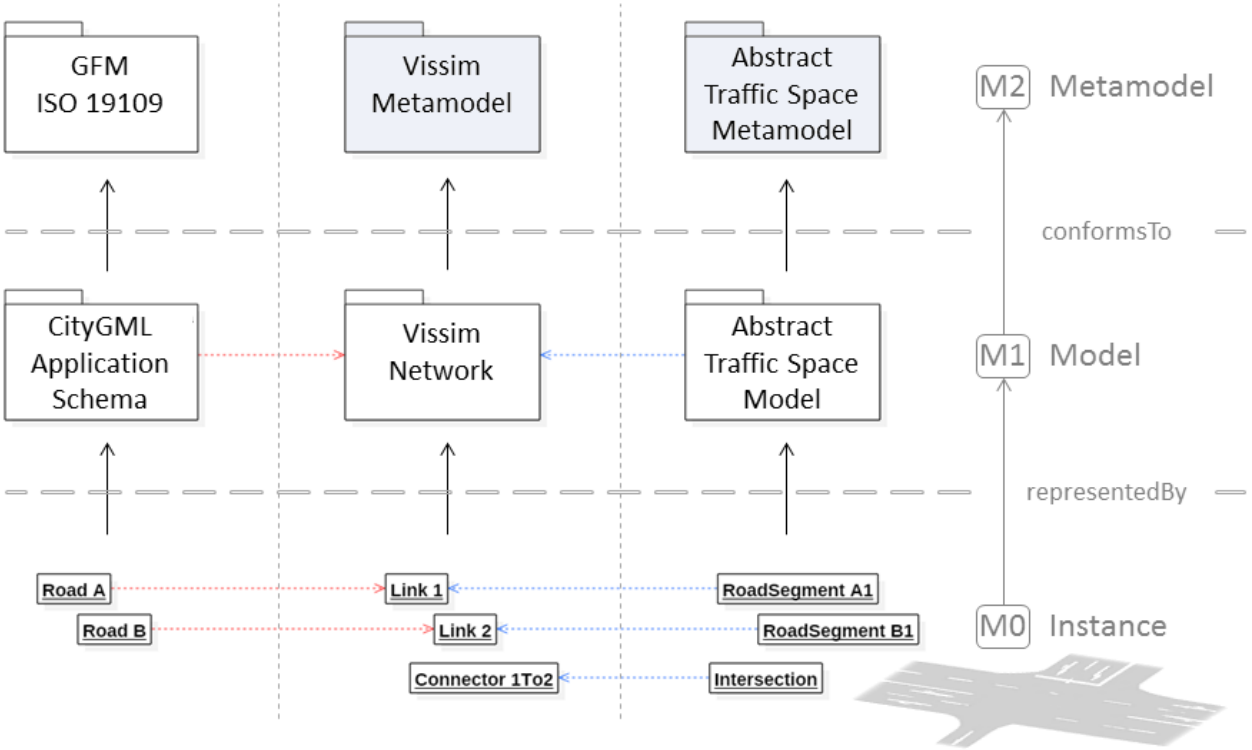


Abbildung 5.1 – Modelle aus der MDE-Perspektive

Dem Grundsatz “Alles ist ein Modell“ zu Folge können auch Transformationen zwischen verschiedenen Modellen als Modell beschrieben werden. “Model Weaving“ [Del Fabro et al., 2005] ist ein Konzept, um Korrespondenzen zwischen unterschiedlichen Modellen herzustellen. Ein “Weaving Model“ beinhaltet demnach die Zuordnung der jeweiligen Modellelemente zweier komplexer Modelle. Das Konzept liefert eine Möglichkeit heterogene Daten verschiedener Systeme ineinander zu überführen und dabei Konflikte zwischen allgemeiner Anwendbarkeit, Ausdrucksfähigkeit und Effizienz der Abbildung zu kontrollieren.

5.1.1 Korrespondenzen in der Netzwerk-Repräsentation

Abbildung 5.1 stellt Verknüpfungen auf Instanz- und auf Modellebene her. Korrespondenzen zwischen abstraktem Verkehrsmodel und Vissim sind blau, zwischen CityGML und Vissim rot abgebildet. Straßensegmente des abstrakten Modells korrelieren mit Vissim-Links. Ebenso lässt sich eine Verknüpfung zwischen Kreuzungsobjekten und Konnektoren herstellen. Auf der anderen Seite können CityGML-Straßenobjekte mit Link-Objekten aus Vissim verbunden werden. Eine umfassende Verknüpfung auf Modellebene wird in Abbildung 5.2 vorgenommen. Dazu sind die UML-Diagramme der drei Modelle in abstrahierter Form aufgeführt. Der Übersichtlichkeit halber wurden nur die für eine Transformation wesentlichen Klassen ohne Attribute aufgenommen.
Die Korrespondenzen geben an, welche Klassen der einzelnen Modelle semantisch verknüpft werden und damit auf eine bestimmte Art und Weise ineinander überführt werden können. Im Mittelpunkt des Interesses steht dabei die Überführung der CityGML-Klassen nach Vissim. Zunächst sollen jedoch die Korrespondenzen zwischen dem abstrakten Verkehrsraummodell und Vissim (blau) betrachtet werden. Dieser Schritt dient einer weiteren Evaluierung des Vissim-Modells vergleichbar mit der bereits in Kapitel 4.4 durchgeführten Diskussion des CityGML-Verkehrsraummodells.
Hinsichtlich des Modells einer Verkehrssimulation wurde bereits festgehalten, dass Vissim sowohl Logik als auch Grafik-Komponenten nahezu vollständig abdeckt. Die Korrespondenzen geben an, wie es sich mit der Abbildung der realen Welt verhält. Das abstrakte Verkehrsraummodell wurde in Anlehnung an das Beispiel Dachauer Straße entworfen. Abbildung 5.2 zeigt, dass sich alle Objektklassen in Vissim abbilden lassen. Die einzige Klasse, die keine Korrespondenz aufweist, ist die Klasse “Entrance“, welche über keinen expliziten Gegenpart verfügt, sich jedoch ebenfalls implizit modellieren lässt. Während im abstrakten Modell jedes einzelne Verkehrsraumobjekt in einer Klasse abgebildet ist, werden in Vissim einige Objekte in einer Klasse zusammengefasst. Daraus lässt sich keinesfalls ein geringerer Informationsgehalt des Vissim-Modells folgern. Der Unterschied liegt in der Art der Modellierung. Die semantische Differenzierung erfolgt einerseits über unterschiedliche Klassen, andererseits über eine attributive Modellierung. Verdeutlichen lässt sich dieser Umstand am Beispiel der Klasse “Link“. Ein Link kann Straßensegmente, Schienen, Fahrradwege und Fußwege repräsentieren. Die Information, worum es sich handelt, wird in den Attributen “BehaviourType“, “DisplayType“ und “BlockedVehicle“ festgehalten. Erstere lassen eine Unterscheidung zwischen Straße, Schiene, Fahrradweg und Fußweg zu, letzteres Attribut definiert, welche Verkehrsteilnehmer berechtigt sind den Verkehrsweg zu nutzen.Im Bereich der Fußgängermodellierung geht das Vissim-Modell mit den Klassen Fahrstühle, Treppen und Rampen über den im abstrakten Verkehrsraummodell abgebildeten Bereich hinaus. An dieser Stelle liegt der Übergang vom Außen- in den Innenraum und damit auch die Schnittstelle zu Innenraummodellen.
Im Vergleich zur Verknüpfung des abstrakten Modells mit Vissim liegen für die Kopplung von CityGML mit Vissim weniger Korrespondenzen vor. Das CityGML Transportation Modul verfügt ähnlich wie Vissim im Vergleich zum abstrakten Modell über eine geringe Anzahl an Klassen. Die semantischen Informationen zu Straßenraumobjekten werden weitestgehend in Attributen mit Hilfe von vordefinierten Codelisten gespeichert.

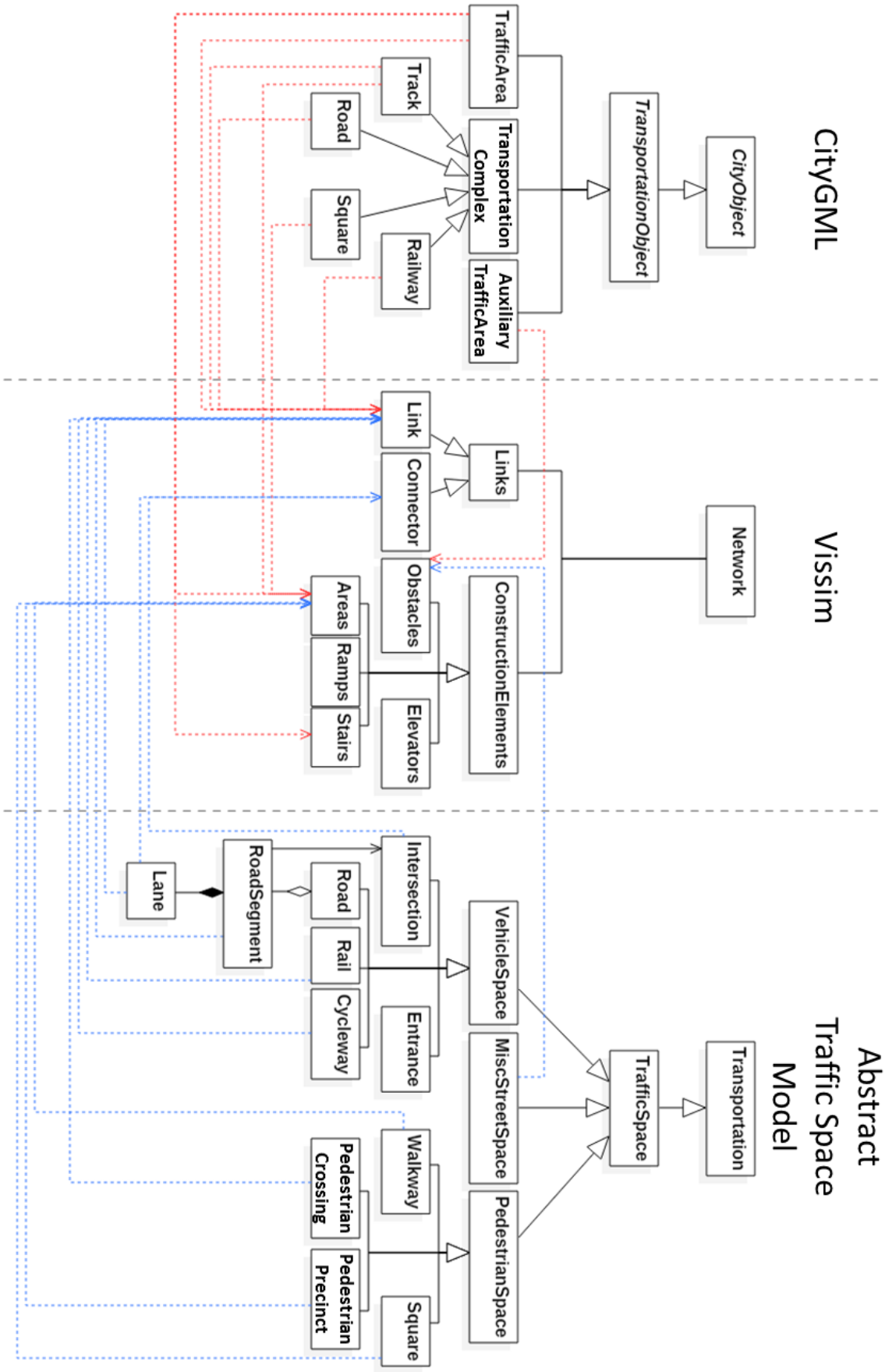


Abbildung 5.2 – Model Weaving: Korrespondenzen in der Netzwerk-Repräsentation

Ein CityGML Verkehrsobjekt kann als “Track“, “Road“, “Square“ oder “Railway“ vorliegen. Mit Ausnahme von Plätzen können diese Klassen in Vissim als Link-Objekte abgebildet werden. Die Klassennamen implizieren bereits semantische Details, für eine exakte Zuordnung ist allerdings eine Überprüfung der Attributwerte gemäß der Codeliste eines “TransportationComplex“ notwendig. In der Codelist zu den Attributen “function“ und “usage“ sind weitreichende Möglichkeiten zur Differenzierung von Wegen (’footpath’, ’hiking trail’, ’bikeway’, ...) über Straßen (’road’, ’highway’, ’agricultural road’, ...) bis hin zu Informationen über Gleisnutzung (’city railway’, ’tram’, ’subway’, ...) vorhanden [Gröger et al., 2012, S.240ff]. Auch aus Straßenteilflächen der Klasse “TrafficArea“ können Vissim-Links abgeleitet werden. Zu beachten ist, dass das Attribut “function“ der Klasse “TrafficArea“ über eine andere Codelist verfügt als das “function“-Attribut der Klasse “TransportationComplex“. Über die Attribute “class“, “function“ und “usage“ lassen sich aus CityGML Verkehrsflächen Vissim-Fußgängerflächen (“ConstructionElements“) bzw. deren Spezialisierungen ableiten. Potenziale zur Ableitung zusätzlicher Objekte enthalten auch andere thematische Module des CityGML Standards, wie das Modul “Building“ oder “CityFurniture“, auf die an dieser Stelle allerdings nicht näher eingegangen werden soll. Mit dem Fokus auf dem Transportation Modul besteht eine weitere Korrespondenz zwischen den Klassen “Track“ und “Square“ und der Vissim-Klasse “Areas“. Aufschluss über die Nutzung als Fußgängerfläche geben wiederum die Attribute “class“, “function“ und “usage“. Aus den CityGML “AuxiliaryTrafficAreas“, welche beispielsweise Grünflächen, Böschungen und Gräben beinhaltet, lassen sich Hindernisflächen herleiten. Weitere Hindernisse können wieder andere thematische Module liefern.
Geometrisch betrachtet bestehen zwei Varianten Link-Objekte aus CityGML abzuleiten (vergleiche dazu auch das vollständige UML-Diagramm des CityGML Transportation Moduls in Abbildung 4.10): entweder aus einem “gml:GeometricComplex“ in Form linienhafter Geometrie eines “lod0Networks“ oder aus “gml:MultiSurface[s]“. Aufgrund der achsbasierten Repräsentation von Link-Objekten ist eine Ableitung aus einem “lod0Network“ einfacher zu realisieren. Wie die Konzeptumsetzung im folgenden Kapitel zeigen wird, bringt dieser Ansatz weitere Vorteile mit sich. Für eine Ableitung von Fußgängerflächen ist dagegen klar eine Ableitung aus “MultiSurface[s]“ zu favorisieren.
Für die Durchführung einer Verkehrssimulation sind neben den Straßendaten Informationen über Abbiegerelationen von entscheidender Bedeutung. In Vissim werden diese in der Klasse “Connector“ modelliert. Wie aus Abbildung 5.2 hervorgeht, besteht keine Korrespondenz zu einer CityGML-Klasse. Das hat den Grund, dass in CityGML Straßensegmente keine expliziten Informationen über Vorgänger und Nachfolger enthalten. Hier zeigt sich erneut, dass CityGML wesentliche Teile des Logikmodells eines Verkehrsraummodelles für Verkehrssimulationen fehlen.

5.1.2 Visualisierung von dynamischen Fahrzeugdaten

Nach dem schematischen Aufbau einer Verkehrssimulation aus Abbildung 2.1 liefern Verkehrssimulationen Verkehrsgrößen als Ergebnis. Einem bestimmten Modell folgend werden für jeden Simulationszeitpunkt die Positionen der Verkehrsteilnehmer simuliert. Ein zentrales Ziel dieser Masterarbeit ist es, diese Informationen grafisch in einem Stadtmodell darzustellen.

Für eine Visualisierung relevante Modellelemente sind demnach die Verkehrsteilnehmer und deren Bewegungsdaten über den Simulationszeitraum. Die unterschiedliche Modellierung dieser Informationen stellt Abbildung 5.3 nach dem Konzept des Model Weaving einander gegenüber. Im abstrakten Modell sind Verkehrsteilnehmer als Nutzer des Verkehrsraumes modelliert (Abbildung 5.3 - rechts). Die Unterklassen “Vehicle“ respektive “Pedestrian“ entsprechen den Klassen “VehicleTypes“ und “PedestrianTypes“ aus dem Vissim Netzwerk (Abbildung 5.3 - mittig). Die sich aus der Simulation ergebenden, dynamischen Bewegungsinformationen der Fahrzeuge werden in einem Fahrzeugprotokoll (“VehicleRecord“) gespeichert. Im Wesentlichen ist darin ein Zeitstempel in Verbindung mit einer Fahrzeugidentifikation und der Position dieses Fahrzeugs zum entsprechenden Zeitpunkt enthalten.

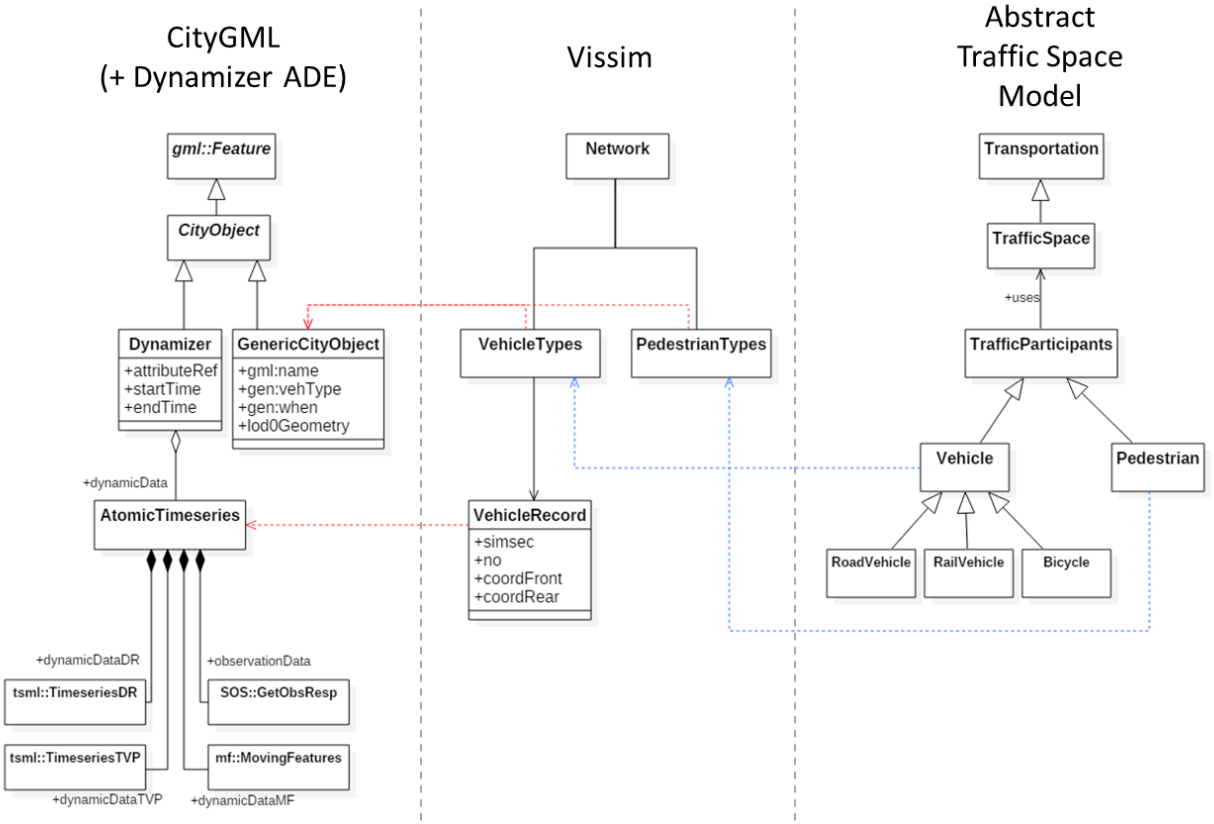


Abbildung 5.3 – Model Weaving: Korrespondenzen bei der Visualisierung dynamischer Ergebnisse einer Verkehrssimulation auf Modellebene

CityGML verfügt standardmäßig weder über Objekte vom Typ Verkehrsteilnehmer noch über die Möglichkeit dynamische Prozesse abzubilden. Kapitel 2 nennt im Rahmen der Vorstellung des CityGML Standards zwei Möglichkeiten für nutzerspezifische Erweiterungen: generische Objekte oder Application Domain Extension (ADE).
Verkehrsteilnehmer lassen sich in CityGML am einfachsten als generische Stadtobjekte modellieren. Einem generischen Objekt können (generische) Attribute und eine Geometrie, beispielsweise eine Punktrepräsentation in Form einer “lod0Geometry“ zugewiesen werden. Für die Einbeziehung dynamischer Daten wurde in Kapitel 3.3 ein Konzept nach Chaturvedi & Kolbe [2016] namens “Dynamizer“

vorgestellt. Abbildung 5.3 (links) zeigt die relevanten Klassen dieses Konzeptes als ADE für CityGML (Version 2.0) zusammen mit einem generischen Objekt zur Modellierung von Verkehrsteilnehmern. Das vollständige UML-Diagramm zum Dynamizer Konzept findet sich in Anhang A.10.
Zum derart erweiterten CityGML Model lassen sich für jedes Fahrzeug - beschrieben durch eine ID und ein Bewegungsprofil - Korrespondenzen herstellen. Die dynamischen (Fahrzeug-)Daten werden als Zeitreihe ("AtomicTimeseries") repräsentiert. Der Dynamizer sorgt für die Bereitstellung der dynamischen Werte im 3D-Stadtmodell, im vorliegenden Fall der aktuellen Fahrzeugposition. Dies wird erreicht, indem ein referenziertes, statisches Attribut eines Stadtobjektes - hier eines generischen Fahrzeugobjektes - mit den dynamischen Werten überschrieben wird.
Chaturvedi & Kolbe [2016] nennen verschiedene Möglichkeiten und Standards zur Repräsentation dynamischer Daten. In der Folge sollen in Abbildung 5.3 aufgeführte Varianten bzw. deren zugrunde liegende Standards vorgestellt werden.
"Timeseries Profile of Observations and Measurements" [Tomkins & Lowe, 2016] ist ein OGC Standard zur Strukturierung und zum Austausch von Beobachtungsdaten als Zeitreihen. TimeseriesML (TSML) ist die zugehörige XML Implementierung. Der Standard definiert Zeitreihenbeobachtungen durch zwei Untertypen: als "Time-Value-Pair" (TVP) oder als "Domain Range" (DR). Im ersten Fall der Zeit-Werte-Paare werden die Zeit, Werte und Metadaten als gekoppelte Einheit gespeichert. Die zweite Struktur repräsentiert Zeit und Werte getrennt von einer Beschreibung der Metadaten, was in den meisten Fällen eine kompaktere und damit effizientere Darstellung der Daten als Zeitreihe erlaubt. Die Metadaten definieren in beiden Fällen die Zeit und die Struktur der Werte [Tomkins & Lowe, 2016].
Eine weitere Möglichkeit stellt der OGC Standard "Moving Features" [Asahara et al., 2015] dar. Dieser wird in der Veröffentlichung von Chaturvedi & Kolbe [2016] nicht namentlich erwähnt, ist jedoch explizit zur Beschreibung dynamischer Fahrzeugdaten entwickelt. Es handelt sich dabei um eine XML (bzw. GML) Implementierung der ISO 19141:2008 "Geographic Information - Schema for Moving Features", welche ein Format zur geometrischen Beschreibung der Bewegung starrer Körper darstellt. Modelliert werden dabei Zustandsänderungen wie z.B. Geschwindigkeits- oder Richtungsänderungen. Die Bewegungen werden als Trajektorien beschrieben.
Neben verschiedenen Zeitreihen-Repräsentationen unterstützt das Dynamizer Konzept die direkte Einbeziehung von Sensoren bzw. Sensordaten. Dies kann zum einen durch Verknüpfung eines Sensors mit dem Dynamizer erfolgen. In diesem Zusammenhang liefert der OGC Standard "Sensor Observation Service" (SOS) [Bröring et al., 2012] einen Rahmen für standardisierte Zugriffe auf Sensorbeobachtungen. Mit den SOS-Operationen "DescribeSensor" und "GetObservation" können Beobachtungs- und Metadaten abgerufen und direkt mit eine CityGML Stadtobjekt verlinkt werden. Die andere Möglichkeit ist, Sensorbeobachtungen innerhalb eines Dynamizers darzustellen. Dynamizer unterstützen eine Inline-Repräsentation von Sensorbeobachtungen im O&M-Format. Die ISO 19156 - "Observations and Measurements" (O&M) ist ein allgemeines Informationsmodell zur Beschreibung von Beobachtungen, welche für verschiedene Standards als Grundlage dient. Enthalten sind Modelle zum Austausch der Beobachtungserfassung und -ergebnisse [Cox, 2013].

In der Folge soll eine beispielhafte Integration von Fahrzeuginformationen in ein 3D-Stadtmodell skizziert werden. Ein Fahrzeug lässt sich gemäß Abbildung 5.4 als generisches Objekt modellieren. Neben einem Namen können zusätzliche (generische) Attribute wie z.B. der Fahrzeugtyp definiert werden. Die Position eines Fahrzeuges kann über die Koordinaten einer Punktrepräsentation beschrieben werden (Zeile 11). Möchte man den zur Position zugehörigen Zeitpunkt ebenfalls abbilden, kann dazu ein Attribut vom Typ “Date“ definiert werden (Zeile 7). Auf diese Weise ließe sich eine Zeile eines Fahrzeugprotokolls statisch in das Stadtmodell integrieren. Das eigentliche Ziel ist jedoch die Integration der dynamischen Fahrzeugbewegungen in das Stadtmodell und nicht nur einer statischen Position zu einem spezifischen Zeitpunkt.

```
1  <cityObjectMember>
2     <genericCityObject gml:id="car1">
3        <gml:name>Auto 1</gml:name>
4        <gen:stringAttribute name="vehicle_type">typeX</gen:stringAttribute>
5        ...
6        <!-- static point in time -->
7        <gen:dateAttribute name="when">t</gen:dateAttribute>
8        <!-- (static) position, value may be dynamized -->
9        <lod0Geometry>
10          <gml:point gml:id="posCar1">
11             <gml:pos srsDimension="3">X Y Z</gml:pos>
12          </gml:point>
13       <lod0Geometry>
14    </genericCityObject>
15 </cityObjectMember>
```

Abbildung 5.4 – XML-Code Beispiel für die Modellierung von Verkehrsteilnehmern (Fahrzeugen) in CityGML als generisches Stadtobjekt mit Attributen und einer Punktgeometrie

Für die Integration der dynamischen Daten sorgt ein Dynamizer, zu sehen in Abbildung 5.5. Im Element “attributeRef“ ist der XPath zum zu überschreibenden Positionsattribut des generischen Stadtobjektes mit der ID ’car1’ angegben (Zeile 4). Danach wird die zeitliche Ausdehnung der Zeitreihe angegeben. Es folgen die dynamischen Daten, welche im vorliegenden Beispiel als Domain Range nach der TimeseriesML spezifiziert sind. Der Block unterteilt sich in eine Definition der Metadaten (Zeile 18-33) und die Angabe der Werte (Zeile 34-54). In den Metadaten wird zunächst die Zeit definiert. Neben der Angabe des Zeitraums (Zeile 22-26) werden die Basiszeit, d.h. der Ausgangszeitpunkt, (Zeile 27) und ein Intervall (Zeile 28) festgesetzt. Das Intervall (“tsml:spacing“) beträgt entsprechend der Berechnungsfrequenz der Verkehrssimulation eine Zehntelsekunde. Für die Werte ließen sich ebenfalls Metadaten definieren. Nach Tomkins & Lowe [2016] können dies u.a. Angaben zur Qualität oder zur Datenquelle sein. Dann werden die Werte als “gml:rangeSet“ angegeben. Dieser beinhaltet einen Werte-Array, worin die einzelnen Punkte als Komponenten gespeichert sind. Vorteil dieses Ansatzes ist, dass die Punkte unter Verwendung von GML und damit in exakt gleicher Weise wie im referenzierten Attribut des Stadtmodells abgelegt werden können. Für große Datenmengen und hochfrequente Zeitintervalle ist der Ansatz dagegen eher ineffizient. Darunter fällt auch eine Visualisierung von Ergebnissen einer Verkehrssimulation mit vielen Fahrzeugen. Effizienter für derartige Anwendungen sind Ansätze, die direkt auf externe CSV-Dateien verweisen, ohne die Daten redundant als Attributwerte innerhalb des Stadtmodells zu speichern. Eine derar-

tige Implementierung ist für das Dynamizer Konzept vorgesehen und könnte zukünftig Anwendung finden.

```
<cityObjectMember>
    <dyn:dynamizer>
        <!-- attribute reference using XPATH -->
        <dyn:attributeRef>//genericCityObject[@gml:id="car1"]/lod0Geometry/gml:Point[@gml:id="posCar1"]/gml:pos
        </dyn:attributeRef>
        <!-- time frame and interval -->
        <dyn:startTime>2017-07-01</dyn:startTime>
        <dyn:endTime>2017-07-01</dyn:endTime>
        <!-- dynamic data (vehicle positions) -->
        <dyn:dynamicData>
            <dyn:AtomicTimeseries>
                <dyn:dynamicDataDR>
                    <tsml:TimeseriesDR>
                        <tsml:TimeseriesDomainRange gml:id="fzp_timeseries1">
                            <!-- Metadata -->
                            <gmlcov:metadata>
                                <gmlcov:Extension>
                                    <tsml:TimeseriesMetadataExtension>
                                        <tsml:metadata>
                                            <tsml:TimeseriesMetadata>
                                                <tsml:temporalExtent>
                                                    <gml:TimePeriod>
                                                        <gml:beginPosition>2017-07-01T00:00:00.00Z</gml:beginPosition>
                                                        <gml:endPosition>2017-07-01T00:00:00.00Z</gml:endPosition>
                                                    </gml:TimePeriod>
                                                </tsml:temporalExtent>
                                                <tsml:baseTime>2017-07-01T00:00:00.00Z</tsml:baseTime>
                                                <tsml:spacing>0.1S<tsml:spacing>
                                            </tsml:TimeseriesMetadata>
                                        </tsml:metadata>
                                    </tsml:TimeseriesMetadataExtension>
                                </gmlcov:Extension>
                            </gmlcov:metadata>
                            <!-- dynamic values -->
                            <gml:rangeSet>
                                <gml:valueArray>
                                    <gml:valueComponent>
                                        <gml:Point>
                                            <gml:pos>X1 Y1 Z1</gml:pos>
                                        </gml:Point>
                                    </gml:valueComponent>
                                    <gml:valueComponent>
                                        <gml:Point>
                                            <gml:pos>X2 Y2 Z2</gml:pos>
                                        </gml:Point>
                                    </gml:valueComponent>
                                    ...
                                    <gml:valueComponent>
                                        <gml:Point>
                                            <gml:pos>Xn Yn Zn</gml:pos>
                                        </gml:Point>
                                    </gml:valueComponent>
                                </gml:valueArray>
                            </gml:rangeSet>
            </atomicTimeseries>
        </dynamicData>
    </dynamizer>
</cityObjectMember>
```

Abbildung 5.5 – XML-Code Beispiel für die Integration dynamischer Fahrzeugdaten in CityGML mit Hilfe des Dynamizer Konzeptes nach Chaturvedi & Kolbe [2016] und dem OGC Standard “TimeseriesML“ zur Beschreibung der dynamischen Fahrzeugdaten als Zeitreihe

5.2 Potenziale und Herausforderungen

In Kapitel 4 wurden allgemeine Anforderungen an die Modellierung des Verkehrsraumes diskutiert. Mit dem Konzept Model Weaving wurden Korrespondenzen zwischen CityGML und Vissim hergestellt. Darauf aufbauend soll in der Folge näher auf Potenziale und Schwierigkeiten eingegangen werden, die bei der Kopplung zu berücksichtigen sind.

5.2.1 Ableitung eines Simulationsgraphen

Die Potenziale einer Kopplung von CityGML und einer Verkehrssimulation wie Vissim liegen auf der Hand. Durch Nutzung des Verkehrsraummodelles eines bestehenden Stadtmodelles würde die aufwendige, manuelle Generierung eines Simulationsgraphen in der Verkehrssimulation entfallen oder zumindest stark vereinfacht werden. Tamminga et al. [2013] betrachten das CityGML Transportation Modul unter dem Aspekt einer GIS-konformen Datenstruktur für Verkehrs- und Transportmodelle. Einer Analyse notwendiger Daten, ähnlich der zu Beginn von Kapitel 4 durchgeführten Analyse, für eine netzwerkbasierte sowie für eine flächenhafte Modellierung des Verkehrsraumes folgt eine Abbildung der gefundenen Anforderungen auf das CityGML Transportation Modul. Als Ergebnis halten Tamminga et al. [2013] eine hohe semantische Übereinstimmung zwischen den Anforderungen von Verkehrs- und Transportmodellen und CityGML fest. Gleichzeitig werden Unterschiede in der Modellierung benannt. Während CityGML in LoD 0 eine Netzwerkrepräsentation zur Verfügung stellt, beinhalten nur höhere LoD-Stufen wichtige Detailinformationen beispielsweise zur Ausdehnung einer Straße. Eine rein parametrische Geometrierepräsentation ist aktuell nicht Teil von CityGML. Die in vorhergehenden Abschnitten angestellten Untersuchungen lassen ebenso wie die Ergebnisse von Tamminga et al. [2013] den Schluss zu, dass das Transportation Modul in der standardmäßigen Form des CityGML 2.0 Standards zwar ausreicht, um einen Großteil der notwendigen Anforderungen zu erfüllen, aber nicht alle Bereiche der Verkehrsraummodellierung abdeckt, die für die Anwendung Verkehrssimulation notwendig sind.
Auch die Ergebnisse des Model Weaving und ein Vergleich der Datenstruktur eines Road-Objektes aus CityGML (Abbildung 4.12) mit der Struktur eines Link-Objektes in Vissim (Abbildung 4.7) legen die gleiche Schlussfolgerung nahe. Eine große Übereinstimmung ist im Bereich der einfachen Geometrierepräsentation zu erkennen. Das LoD0-Network liefert genau die Informationen, die zur Modellierung der Start- und Endpunkte sowie etwaiger Zwischenpunkte eines Streckenelements in Vissim notwendig sind. Nur implizit in CityGML enthalten sind dagegen Informationen über die Ausdehnung von Straßen. Geht man weg von einzelnen Straßenstücken hin zur Art der Repräsentation des gesamten Straßennetzwerkes, treten weitere Probleme auf. In Vissim ist eine explizite Modellierung jeder Fahrtrichtung inklusive der Anzahl an Fahrspuren und deren jeweilige Breite erforderlich (siehe Abbildung 5.6 - rechts). Diese Detailfülle bietet CityGML in seiner standardmäßigen Ausprägung nicht. Das LoD0-Network beschreibt die Mittelachse pro Straßensegment (Abbildung 5.6 - mittig). Notwendige Informationen zur Topologie eines Straßennetzwerkes sind lediglich in impliziter Form vorhanden. CityGML Straßensegmente besitzen keinerlei Angaben über vorhergehende oder

nachfolgende Segmente, wodurch sich Abbiegerelationen (Vissim Verbindungsstrecken: magenta in Abbildung 5.6 - rechts) nicht direkt nachbilden lassen. Damit sind die Möglichkeiten limitiert einen umfassenden Simulationsgraphen für Vissim abzuleiten.

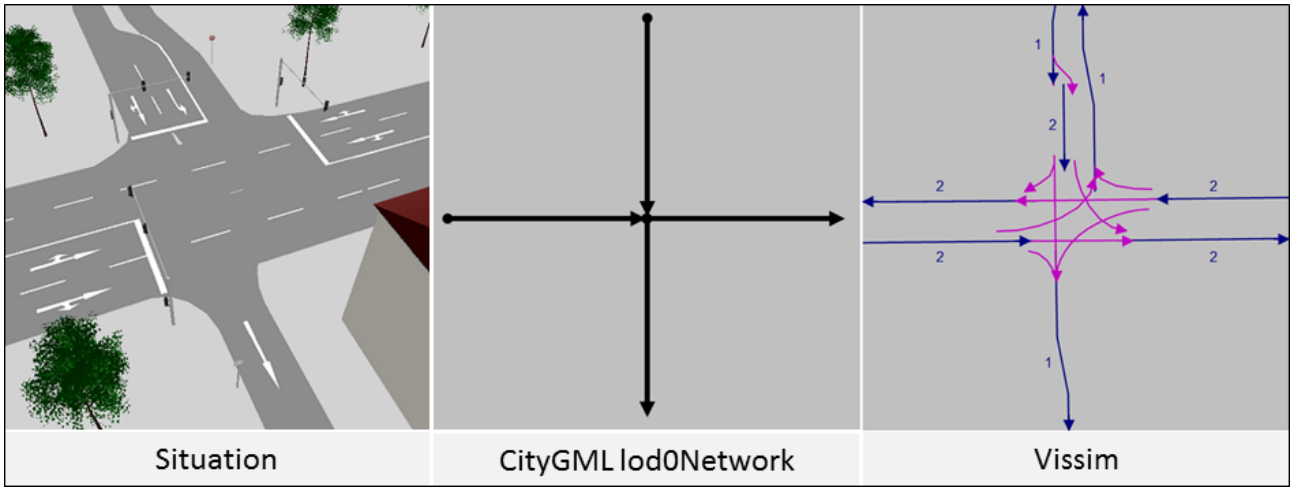


Abbildung 5.6 – Vergleich der Repräsentation einer Kreuzung in CityGML und Vissim

Mehr Potenzial verspricht eine Erweiterung des CityGML Transportation Moduls. Erweiterungen sind möglich durch generische Attribute oder sogenannte Application Domain Extensions (ADE) [Kolbe, 2009]. Tamminga et al. [2013] unterbreiten einen Vorschlag für eine ADE, um notwendige Attribute, die im CityGML 2.0 Standard nicht enthalten sind, ebenfalls abbilden zu können. Die vorgestellte ADE-Klasse stellt eine Unterklasse von “TrafficArea“ dar und ist somit ab LoD 2 verfügbar. Sie umfasst die zusätzlichen Attribute “length“, “maximum speed“, “number of lanes“, “capacity“, “width“, “height“, “maximum height“ und “gradient“.
Beil [2017] nutzt bei der Modellierung des Straßenraumes in 3D-Stadtmodellen eine Reihe von generischen Attributen, um zusätzlich Informationen zu speichern. Am Beispiel eines semantischen 3D-Stadtmodells von New York City sind auf diese Weise zusätzliche Attribute u.a. zum Zustand, zur Anzahl an Fahrspuren, zur Ausdehnung oder zur Fahrtrichtung enthalten.

5.2.2 Visualisierung von Simulationsergebnissen im 3D-Stadtmodell

Das Dynamizer Konzept bietet die Möglichkeit, dynamische Daten in ein Stadtmodel im CityGML Format zu integrieren. In Hinblick auf die praktische Umsetzung, beschrieben in den nachfolgenden Kapiteln, gilt es zu beachten, dass die 3DCityDB zum derzeitigen Zeitpunkt keine ADEs unterstützt und damit auch nicht das Dynamizer Konzept. Für die praktische Umsetzung der Visualisierung dynamischer Fahrzeugdaten in einem Stadtmodell im CityGML Format muss demzufolge ein alternativer Lösungsansatz gefunden werden.

6 Ableitung von Simulationsgraphen für Verkehrssimulationen aus CityGML

Auf Modellebene wurden Verknüpfungsmöglichkeiten zwischen den Verkehrsraummodellen aus CityGML und der Verkehrssimulation Vissim diskutiert. In diesem Kapitel folgt die praktische Umsetzung der Ableitung eines Simulationsgraphen aus CityGML für Vissim, welche als Export-Funktion in einem Plugin für den 3DCityDB Importer/Exporter realisiert wird (siehe auch Kapitel 8). Das Konzept der Formatüberführung setzt sich aus den Abschnitten Datenhaltung und -aufbereitung sowie der Erzeugung eines Vissim-Straßennetzwerkes zusammen.

6.1 Datenhaltung und -aufbereitung

Die Ausgangsdaten im CityGML-Format werden in der Datenbank (3DCityDB) gehalten. Die zugehörigen Tabellen sind im Schema “citydb“ zusammengefasst (vgl. Kapitel 2). Für das Vissim-Format wird ein neues Schema “vissim“ mit den Tabellen “Link“ und “Connector“ angelegt. In den Tabellen werden alle relevanten Informationen zur Geometrie, Topologie und Semantik abgespeichert. Sie dienen als Grundlage der späteren Netzwerk-Erstellung. Bevor ein Simulationsgraph im Vissim-eigenen Format geschrieben werden kann, ist eine Vorverarbeitung der Daten notwendig. Die Aufbereitung erfolgt in der Datenbank mit Hilfe der von PostgreSQL und der räumlichen Erweiterung PostGIS zur Verfügung gestellten Werkzeuge und Funktionen. Nachfolgend wird schrittweise das Konzept beschrieben, wie aus den vorliegenden CityGML Daten die für die Erstellung von Link- und Konnektor-Objekten notwendigen Informationen abgeleitet werden. Abbildung 6.1 gibt einen Überblick über die notwendigen Schritte der Datenaufbereitung und -ableitung. Rot hinterlegte Formen basieren auf Zusatzinformationen und sind optional.

Abbildung 6.1 – Übersicht über die Schritte der Datenaufbereitung und -ableitung

Datenüberführung in Tabelle Link

Das neu angelegte Schema gilt es mit Daten aus dem Stadtmodell zu füllen. Abbildung 6.2 gibt Tabelle Link mit zwölf Spalten und den zugehörigen Datentypen an.

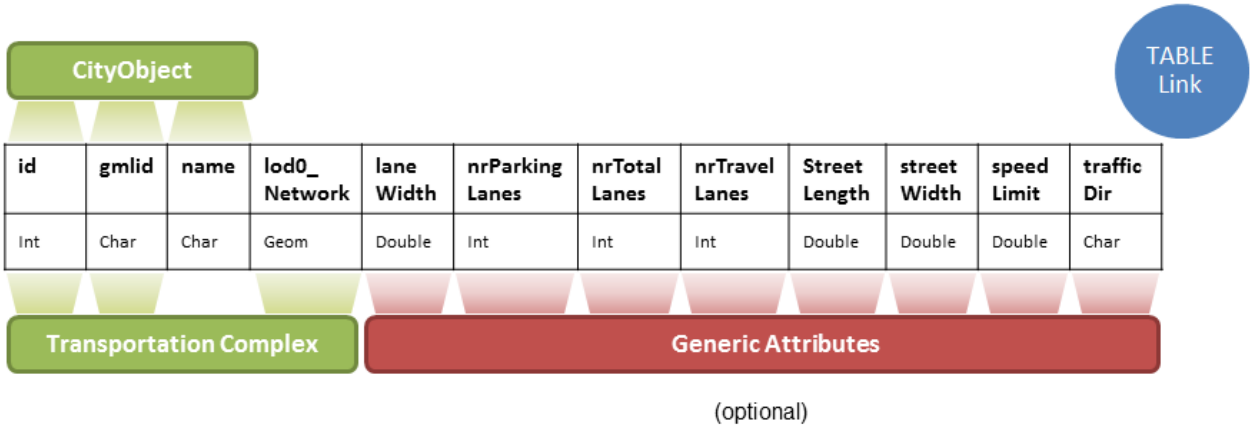


Abbildung 6.2 – Füllen der Tabelle Link mit Daten aus einem Stadtmodell im CityGML Format

Farbig aufgeführt sind die Tabellen des 3DCityDB-Schemas, welche als Datenquelle dienen. Für die Spalten “id“, “gmlid“, “name“ und “lod0_Network“ sind dies die Tabellen “CityObjekt“ respektive “TransportationComplex“. Die übrigen Spalten enthalten Daten, die nicht standardmäßig in CityGML enthalten sind. Zur Speicherung derartiger Daten stehen in CityGML generische Attribute zur Verfügung. Die zugehörige Tabelle des 3DCityDB-Schemas läuft unter der Bezeichnung “CityObject_genericAttrib“. Für die Ableitung eines Simulationsgraphen sind diese Informationen nicht zwingend notwendig, wenngleich sie zu einer signifikanten Verbesserung des Ergebnisses beitragen. Für den Fall, dass keine generischen Attribute vorliegen, werden die Spalten mit Standardwerten gefüllt. Welche Standardwerte anstelle der Werte aus generischen Attributen eingeführt werden, zeigt Tabelle 6.1. Eine Straße beschrieben durch Standardwerte verfügt demnach über eine Fahrspur der Breite 3,5 m. Die Länge wird aus der Geometriespalte “lod0_Network“ mit der PostGIS Funktion “ST_Length“ errechnet. Als Geschwindigkeitsbegrenzung werden 50 km/h angenommen. Die Verkehrsrichtung wird mit dem Wert ’W’ belegt, was einer Einbahnstraße in Richtung der vorliegenden Geometrie entspricht.

Spalte	**Standardwert**
laneWidth	3,5 m
nrParkingLanes	0
nrTotalLanes	1
nrTravelLanes	1
streetLength	ST_Length(LINESTRING)
streetWidth	laneWidth · nrTotalLanes = 3,5 m
speedLimit	50 km/h
trafficDir	W (’With’)

Tabelle 6.1 – Für Link-Objekte verwendete, generische Attribute und zugehörige Standardwerte, falls keine Werte verfügbar

Netzwerk-Geometrie

Die Geometrie eines Link-Tupels ist in der Spalte “lod0_network“ enthalten. In Kapitel 5 wurde im Rahmen des Model Weaving diskutiert, dass eine Ableitung der Link-Geometrie entweder aus LoD 0, d.h. aus linienhaften Geometrien, oder aus flächenhaften Geometrien höherer LoD-Ebenen möglich ist. Vorteil der LoD0-Geometrien ist, dass diese ohne größere Anpassung direkt für die parametrische Geometrie-Repräsentation in Vissim verwendet werden können. Mit der Fahrtrichtung lässt sich eine weitere wichtige Information aus einem CityGML Netzwerk in LoD 0 ableiten. Dazu muss sichergestellt sein, dass die Anordnung der Punkte einer Linie mit der Fahrtrichtung korrespondiert. Die Ableitung der notwendigen Parameter zur Geometrie-Repräsentation aus “MultiSurfaces“ (LoD 1-4) ist wesentlich aufwendiger. Die umgesetzte Lösung sieht vor die Mittelachse der Verkehrsfläche abzuleiten, um aus den resultierenden Liniengeometrien in gleicher Weise mit der Ableitung des Vissim-Netzwerkes fortfahren zu können wie im LoD0-Fall. Die dazu notwendigen Schritte zeigt Abbildung 6.3.

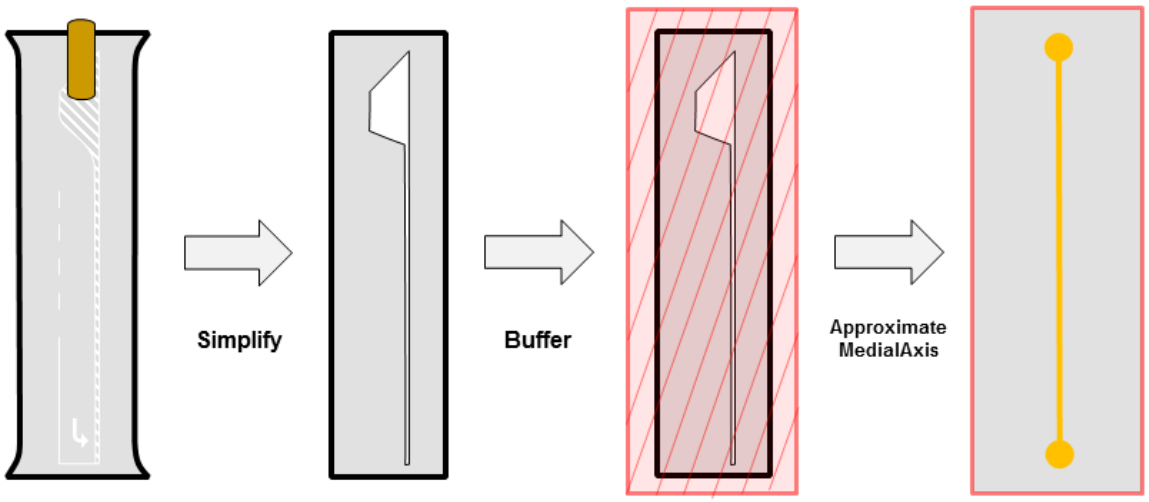


Abbildung 6.3 – Schritte zur Ableitung der Bestandsachse aus LoD 1-4 MultiSurfaces: Vereinfachung der Straßengeometrie, Puffern zum Schließen von Lücken und Berechnung der genäherten Mittelachse

Zunächst werden die als Polygone vorliegenden Verkehrsflächen nach der Methode des Douglas-Peucker-Algorithmus unter Anwendung der PostGIS Funktion “ST_Simplify“ vereinfacht. Eine mit der Vereinfachung der Geometrie einhergehende Reduktion der Punktanzahl einer Straßenfläche wirkt sich signifikant auf die Rechenzeit aus. Letztendlich ist es das Ziel, die Bestandsachse abzuleiten, d.h. die Mittelachse des gesamten Fahrbahnbereiches, wozu der punktgenaue Geometrieverlauf nicht entscheidend ist. Um Abweichungen von der Bestandsachse auszuschließen, die durch etwaige Lücken in der Verkehrsfläche (z.B. Mittelinseln, markierte Sperrflächen, etc.) verursacht werden können, werden die Polygone vor Berechnung der Mittelachse gepuffert. Die PostGIS Funktion “ST_ApproximateMedialAxis“ gibt als Ergebnis die Mittelachsen der gepufferten Verkehrsflächen zurück (Abbildung 6.3 - rechts in Gelb). Diese entsprechen nahezu den Daten eines Netzwerkes in

LoD 0, jedoch mit einem entscheidenden Unterschied: aus den “MultiSurfaces“ lässt sich keinerlei Information über die Fahrtrichtung ableiten. Die Geometrien der berechneten Mittelachsen besitzen demzufolge eine willkürliche Richtung, die sich im Gegensatz zur Linienrichtung aus einem LoD0-Network für nachfolgende Herleitungen nicht weiter nutzen lässt. Theoretisch wäre eine Kombination der berechneten Bestandsachsen mit Richtungsinformationen aus LoD 0 möglich, was aber keinen Vorteil gegenüber einer reinen Verwendung von LoD 0 zur Folge hätte. Im Gegenteil, der Rechenaufwand und damit die Verarbeitungszeit würden steigen. Aussichtsreicher wären Ansätze zur Ableitung von Fahrbahnachsen aus Straßenflächen. Dahingehende Überlegungen werden abschließend in Kapitel 10 angestellt.

Mit der achsbasierten Geometrie sowie Informationen zur Anzahl und Breite vorliegender Fahrspuren (Realwerte oder Standardwerte) ist die Überführung der Geometrie in eine parametrische Form der Straßenrepräsentation abgeschlossen. Die nun vorliegenden Informationen reichen aus, um ein Vissim Link-Objekt erzeugen zu können. Nachfolgende Schritte sind daher als optional anzusehen. Voraussetzung für deren Durchführung sind weitergehende Informationen, beispielsweise zur Fahrtrichtung.

Überprüfung der Fahrtrichtung

Das Attribut Verkehrsrichtung (“trafficDir“) steht in Bezug zur Geometrie einer Achse. Es kann folgende Werte annehmen[1]:

- ’W’ (’with’) → Fahrtrichtung korrespondiert mit Geometrie
- ’A’ (’against’) → Fahrtrichtung entgegengesetzt zur Geometrie
- ’T’ (’two-way’) → beidseitiger Verkehr

Sind die Achsgeometrien ermittelt, kann, sofern vorhanden, das Attribut zur Verkehrsrichtung genutzt werden, um die Richtigkeit der Punktanordnung der Liniengeometrien zu überprüfen. Zugleich kann für Bestandsachsen mit beidseitiger Fahrtrichtung eine zusätzliche Achse für die Gegenrichtung eingeführt werden. Illustriert ist dieser Schritt in Abbildung 6.4. Die Achsen mit den IDs 1, 2 und 3 verfügen über den Attributwert ’T’ und somit über eine beidseitige Befahrung. Auf der rechten Seite sind entsprechende Achsen mit entgegengesetzter Orientierung eingefügt. Die neuen Achsen erhalten eine eigene ID, die sich aus der alten ID plus einer Konstante von 50.000 ergibt. Achsen mit den Attributwerten ’A’ erfahren eine Umkehrung der Geometrie, Achsen mit dem Attributwert ’W’ (im Beispiel Achse mit ID 4) bleiben von diesem Schritt unberührt.

[1] Diese Wertebeschränkung (“Constraint“) wurde dem Beispiel Datensatz des Stadtmodells NYC (siehe auch Kapitel 9) entnommen und an dieser Stelle als allgemeine Definition festgelegt.

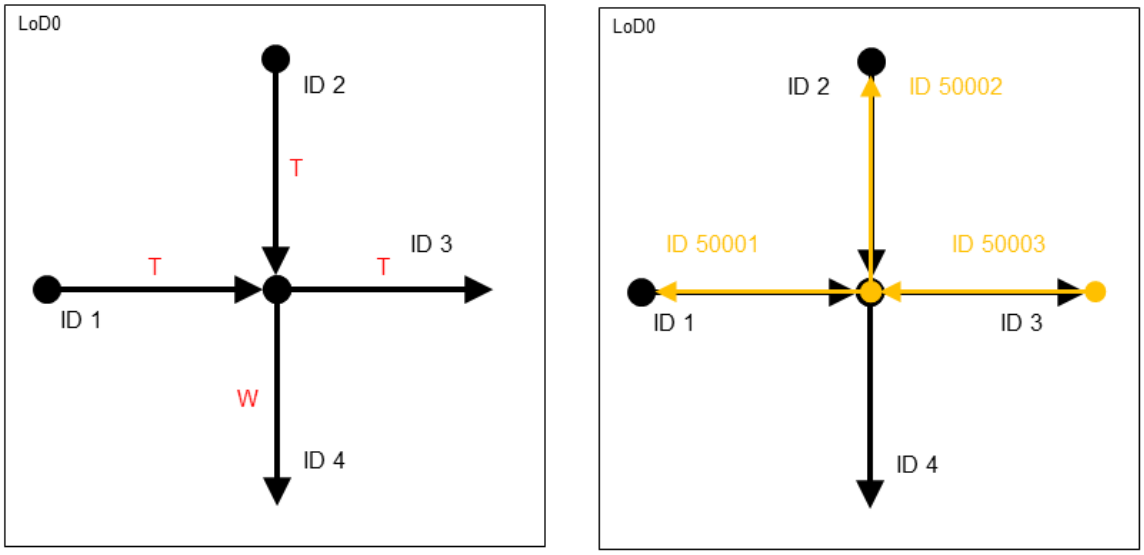


Abbildung 6.4 – Hinzufügen der Gegenfahrtrichtung (gelb) für Bestandsachsen mit dem Attributwert “trafficDir“ = ’T’]

Netzwerk-Topologie: Vorgänger und Nachfolger

Im Optimalfall liegt nun ein gerichteter Graph mit einer Achse pro Fahrtrichtung eines Straßensegments vor. Dieser Graph kann genutzt werden, um Informationen über Vorgänger und Nachfolger herzuleiten, welche für die Erstellung der Vissim Konnektoren notwendig sind. Dazu wird der Endpunkt einer Liniengeometrie mit sämtlichen Anfangspunkten im Liniennetz verglichen. PostGIS stellt für Koordinatenabgleiche die Funktion “ST_Equals“ zur Verfügung. Für das obige Kreuzungsbeispiel lässt sich die Vorgehensweise wie folgt übertragen. Abbildung 6.5 gibt die Ableitung der Konnektorinformationen am Beispiel des Links mit der ID ’1’ an. Anfangspunkte sind durch einen ausgefüllten Kreis, Endpunkte durch einen leeren Kreis gekennzeichnet. Die Suche nach Anfangspunkten, die mit dem Endpunkt des Links ’1’ deckungsgleich sind, ergibt das Ergebnis zur rechten Seite der Abbildung. In die Tabelle “Connector“ können für den Vorgängerlink ’1’ die Nachfolger ’3’, ’4’ sowie ’50002’ eingetragen werden. Nach dem gleichen Prinzip lassen sich sämtliche Vorgänger-Nachfolger Beziehungen in einem gerichteten Graphen bestimmen.
Für Mittelachsen, die aus “MultiSurfaces“ der LoDs 1-4 abgeleitet wurden, ist eine zuverlässige Ableitung von Konnektoren nicht möglich. Das Problem ist, dass Flächen keine Richtungsinformation liefern. Bei Verwendung der PostGIS Funktion “ST_ApproximateMedialAxis“ besitzt die errechnete Mittelachse eine zufällige Richtung. Der Umstand, dass sich die innerhalb des “MultiSurface“ liegenden Anfangs- und Endpunkte von Achsen nicht in einem Knotenpunkt treffen, ließe sich durch eine Bereichssuche umgehen. Anstelle der PostGIS Funktion “ST_Equals“ könnten mit der Funktion“ ST_Within“ alle Anfangspunkte gesucht werden, die in einem bestimmten Pufferbereich um einen Endpunkt liegen (Abbildung 6.6 - links). Für Fälle, in denen kein Kreuzungsbereich modelliert ist, wäre eine Suche nach Anfangs- und Endpunkten denkbar, die im gepufferten Umfeld eines Link-Objektes liegen (Abbildung 6.6 - rechts). Voraussetzung für eine sinnvolle Anwendung dieses Verfahrens wäre jedoch eine Überprüfung der Richtungen der abgeleiteten Mittelachsen.

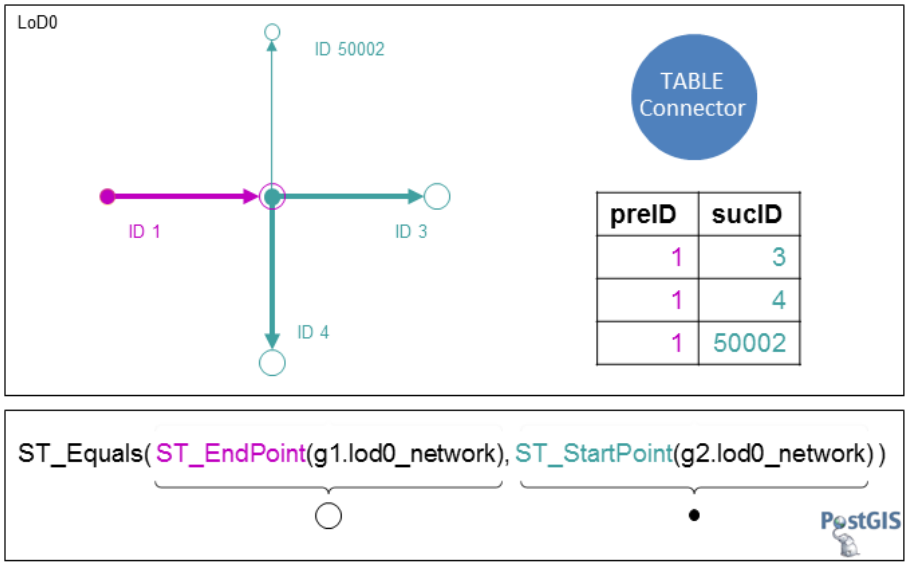


Abbildung 6.5 – Finden der Vorgänger/ Nachfolger aus Geometrieabgleich von Anfangs- und Endpunkten eines gerichteten Liniennetzwerkes

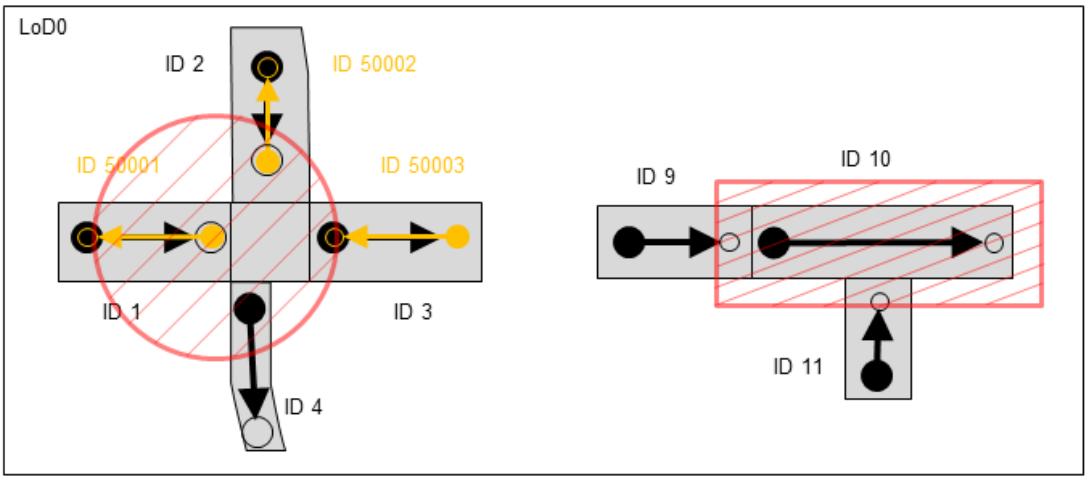


Abbildung 6.6 – Finden der Vorgänger/ Nachfolger mit Bereichssuche für den Fall LoD1-4

Geometrie-Update für Link-Objekte

Im Anschluss an die Bestimmung der topologischen Beziehungen zwischen Links des Straßennetzwerkes können zwei weitere die Geometrie betreffende Schritte vorgenommen werden. Strecken, die aus einem LoD0-Network stammen, werden um einen Faktor verkürzt, um Platz für die Verbindungsstrecken zu schaffen, welche über eine eigene Geometrie verfügen. Korrekterweise müsste die Länge eines Link-Objektes um exakt die Breite der kreuzenden Straße verkürzt werden. Dazu müsste an beiden Enden eines Links überprüft werden, wie breit die jeweils kreuzende Straße ist. Eine weniger aufwendige Näherung stellt die Kürzung einer Strecke um die Hälfte seiner eigenen Breite dar. Die Näherung ist umso fehlerhafter, je größer der Unterschied zwischen zu kürzendem Link und kreuzender Straße ist.
Für Link-Objekte mit dem Attributwert “trafficDir“ = ’T’ wird zudem eine Verschiebung der Achsen senkrecht zur Bestandsachse und eine Aufteilung gemeinsamer Eigenschaften vorgenommen. Abbildung 6.7 veranschaulicht beide Schritte. Auf der linken Seite ist die Ausgangssituation noch ohne Achsen der Gegenfahrtrichtung zu sehen. Nach Hinzufügen der Gegenfahrtrichtung werden beide Achsen gekürzt und um einen Faktor verschoben. Zugleich werden Eigenschaften wie die Anzahl an Fahrspuren (“nrParkingLanes“, “nrTotalLanes“ und “nrTravelLanes“) sowie die Straßenbreite auf die zwei Link-Objekte aufgeteilt. Das Attribut “trafficDir“ wird geändert in ’TA’ (für ’two-way against’) bzw. ’TW’ (für ’two-way with’). Für eine gerade Anzahl an Fahrspuren wird die Aufteilung gleichmäßig vorgenommen. Für eine ungerade Gesamtzahl erhält der Link deklariert mit ’TA’ die kleinere Anzahl an Spuren, was durch Ab- (Fall ’TA’) bzw. Aufrunden (Fall ’TW’) der Fließkommazahl realisiert ist.

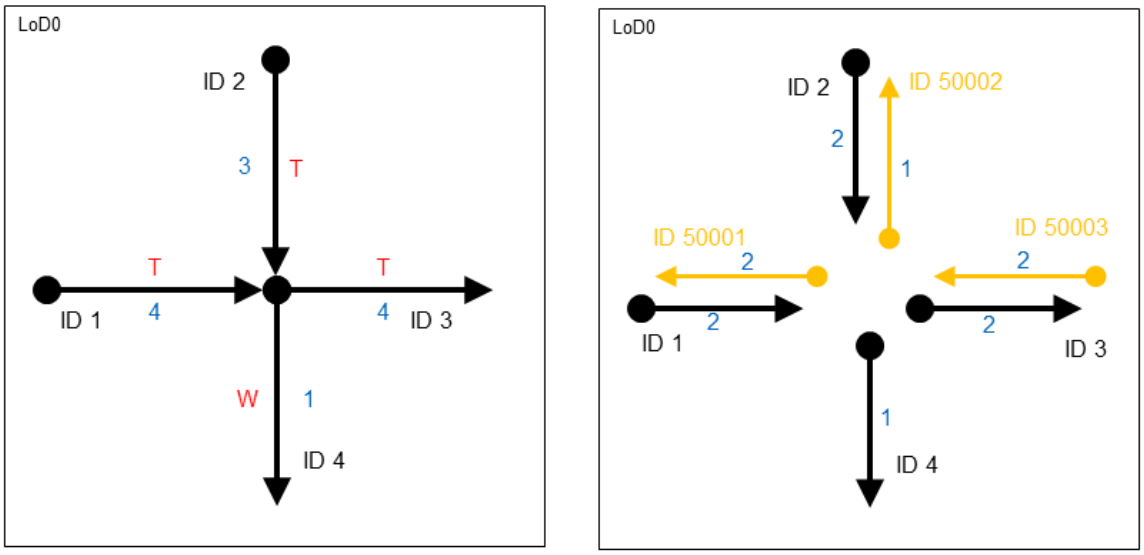


Abbildung 6.7 – Geometrie-Update für Link-Objekte in Form einer Verkürzung der Länge und Verschiebung senkrecht zur Bestandsachse bei hinzugefügter Gegenfahrtrichtung

Für die Berechnung des Offsets sind die Parameter Straßenbreite 'b', die Gesamtzahl an Fahrspuren 'N' und die Anzahl an Fahrspuren pro Fahrtrichtung 'n' notwendig, welche sich nach der Aufteilung ergibt. Abbildung 6.8 beinhaltet einen möglichen Fall mit insgesamt fünf Fahrspuren (angegeben in blau). Da der Link über eine beidseitige Fahrtrichtung verfügt ('T', angegeben in rot), erfolgt eine Aufteilung in zwei Links mit zwei bzw. drei Spuren. Wie aus Abbildung 6.8 hervorgeht, ist der Link mit mehr Spuren weniger weit zu verschieben. In Abhängigkeit der drei Parameter lässt sich der Offset allgemein durch folgenden Zusammenhang ausdrücken:

$$offset = (1 - \frac{n}{N}) \cdot \frac{b}{2} \tag{6.1}$$

Unter der Annahme, dass die Straße in Abbildung 6.8 eine Breite von 17,5 m (fünf Fahrspuren à 3,5 m) habe, ergibt sich ein Offset von 5,25 m für den Link 'TA', was einer Verschiebung von eineinhalb Fahrspuren entspricht. Für den Link 'TW' beträgt der Offset 3,5 m. Ein Vergleich mit Abbildung 6.8 bestätigt unter Orientierung an den weiß-gestrichelt dargestellten Fahrbahnmarkierungen und der grün-gestrichelten Mittelachse die Richtigkeit der berechneten Verschiebungswerte.

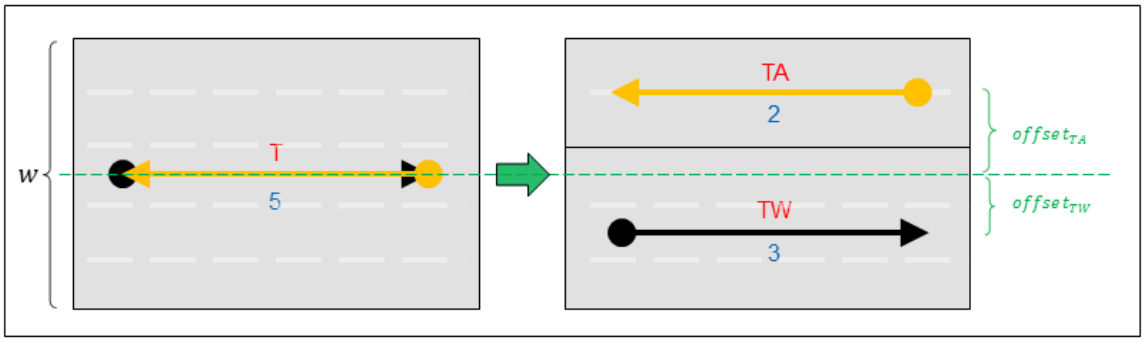


Abbildung 6.8 – Skizze zur Bestimmung des Offsets bei Verschiebung von Link-Objekten mit beidseitiger Fahrtrichtung

Die Implementierung erfolgt mit der PostGIS Funktion "ST_OffsetCurve", welche eine in 2D verschobene Kurve zum Ergebnis hat. Dabei gilt es zu beachten, dass ein positiver Offset einer Verschiebung nach links unter Beibehaltung der Orientierung entspricht, während ein negativer Offset eine Verschiebung nach rechts mit entgegengesetzter Orientierung zur Folge hat. Für den Anwendungsfall Rechtsverkehr ist demzufolge eine Umkehrung der verschobenen Links notwendig, was mit der PostGIS Funktion "ST_Reverse" erreicht wird.

Datenüberführung in Tabelle Konnektor

Damit ist die Vorverarbeitung der Link-Objekte abgeschlossen. Die Tabelle "Link" beinhaltet alle aus CityGML ableitbaren Daten zur Erstellung eines Link-Objektes in der dafür notwendigen Form. In die Tabelle "Connector" wurden bereits die abgeleiteten Informationen über Vorgänger und Nachfolger

eingefügt. Die fehlenden Spalten lassen sich per Fremdschlüssel “preID“ für Spalten mit dem Namensteil “pre“ oder “from“ bzw. über den Schlüssel “sucID“ für Spalten mit dem Namensteil “suc“ oder “to“ aus der Tabelle “Link“ überführen. Auf Details zur Überführung und zur Verwendung dieser Attribute sowie des Richtungswinkels “dirAngle“ wird später eingegangen. Die Spalte “conID“ enthält eine ID, um Konnektoren eindeutig zu identifizieren. Dazu wird in PostgreSQL eine Sequenz mit Startwert 10.000 angelegt. Der Startwert entspricht der in Vissim üblichen Deklaration von Verbindungsstrecken. Abbildung 6.9 gibt die Spalten der Tabelle “Connector“ und die zugehörigen Datentypen an.

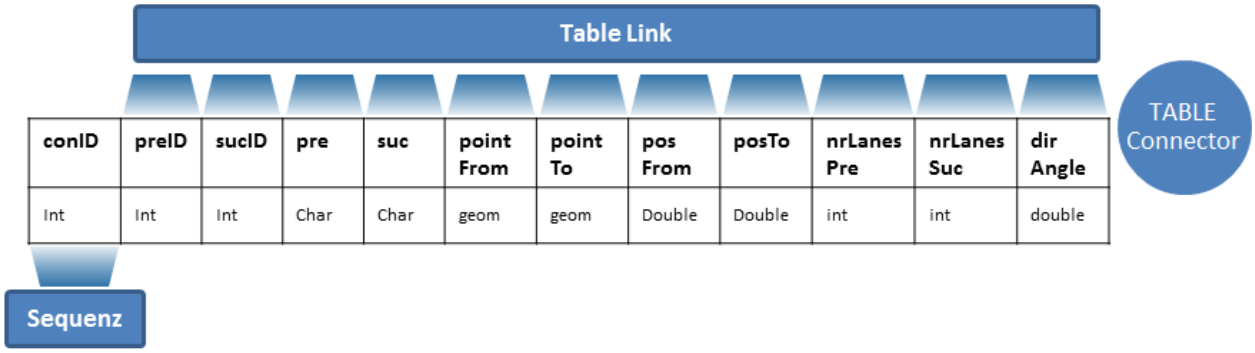


Abbildung 6.9 – Füllen der Tabelle Connector mit Daten abgeleitet aus der Tabelle Link

Konnektor-Geometrie

Für die Erstellung eines Konnektor-Objektes sind, neben der Information welche Strecken über den Konnektor verbunden werden, weitere Daten notwendig. Die Geometrie wird im einfachsten Fall durch einen Start- und einen Endpunkt beschrieben. Für den Startpunkt kann der Endpunkt des Vorgänger-Links und für den Endpunkt der Startpunkt des Nachfolger-Links verwendet werden. Diese Informationen können mittels der PostGIS Funktionen “ST_StartPoint“ bzw. “ST_EndPoint“ ermittelt werden und in den Spalten “point_from“ respektive “point_to“ abgelegt werden. Zusätzlich sind Start- und Endposition entlang des Vorgängers bzw. Nachfolgers zu bestimmen, welche in den Spalten “pos_from“ und “pos_to“ gespeichert werden. Die Startposition kann auf die Länge des Vorgänger-Links festgelegt werden. Die Endposition erhält den Wert ’0’, was dem Anfangspunkt des Nachfolgers entspricht. Dieser Vorgehensweise folgend sind die blau dargestellten Konnektoren in Abbildung 6.10 (links) nachempfunden.
Für den Fall LoD 1-4 können Konnektoren auch an Straßeneinmündungen ohne Kreuzungsfläche auftreten (Abbildung 6.10 - rechts). Die Bestimmung der Geometrie-Parameter gestaltet sich in diesem Fall etwas aufwendiger. Mit Hilfe der PostGIS Funktion “ST_ClosestPoint“ lässt sich der zum Vorgänger (’ID 11’) nächste Punkt finden, welcher in der Grafik rot markiert ist. Die Position entlang des Nachfolgers (’ID 10’) findet sich mit der PostGIS Funktion “ST_LineLocatePoint“.

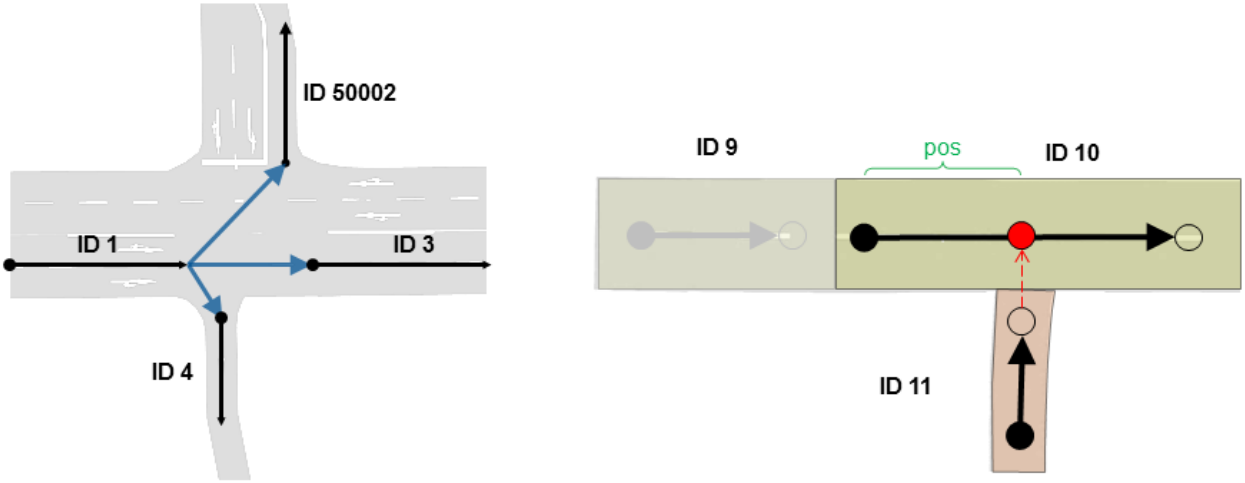


Abbildung 6.10 – Bestimmung der Konnektor-Geometrie: Konnektoren (blau) an Kreuzungspunkten (links) und Sonderfall für LoD1-4 ohne Kreuzungspunkt (rechts)

Fahrspuren und Richtungswinkel

Was für die vollständige Beschreibung eines Konnektors noch fehlt, sind Informationen, über wie viele Fahrspuren die Verbindungsstrecke verfügt und an welchen Fahrspuren des Vor- bzw. Nachfolgers der Konnektor anbindet.
Die Anzahl an Fahrspuren eines Konnektors ist sinnvollerweise abhängig von der Spurzahl seiner Vorgänger und Nachfolger, welche deshalb in den Spalten “nrLanesPre“ (Anzahl Spuren Vorgänger) und “nrLanesSuc“ (Anzahl Spuren Nachfolger) abgespeichert werden.
Mit Konnektoren werden in Vissim Abbiegerelationen modelliert. Neben den zu verbindenden Link-Objekten ist zusätzlich jeweils die Fahrspur anzugeben, an der der Konnektor anbindet. Auf diese Weise kann beispielsweise eine Linksabbiegerspur mit einer Linksabbiegerelation verknüpft werden. Eine automatische Auswahl der korrekten Fahrspuren an Vorgängern und Nachfolgern ist äußerst komplex, da eine Vielzahl von Sonderfällen zu berücksichtigen ist. Mit Hilfe der Richtungswinkel kann eine Kategorisierung der Konnektoren nach Abbiegerichtung vorgenommen werden, welche bei der Suche nach den korrekten Fahrspurverknüpfungen hilft. Der Richtungswinkel soll dabei als der im Uhrzeigersinn gezählte Winkel zwischen den Richtungsvektoren der zu verbindenden Links definiert werden. Da nicht der gesamte Verlauf eines Links, sondern lediglich die Richtung unmittelbar vor bzw. nach dem Kreuzungspunkt relevant ist, wird der Richtungsvektor des Vorgängers aus dessen vorletztem und letztem Punkt gebildet, der Richtungsvektor des Nachfolgers entsprechend aus dessen erstem und zweitem Punkt (vgl. Tabelle 6.2). Um den korrekten Richtungswinkel zu erhalten, ist bei der Berechnung eine Fallunterscheidung notwendig:

$$\begin{aligned} \alpha_{pre} < 180^\circ \quad &\rightarrow \quad \alpha = (180^\circ - \alpha_{pre}) + \alpha_{suc} \\ \alpha_{pre} > 180^\circ \quad &\rightarrow \quad \alpha = \alpha_{suc} - (\alpha_{pre} - 180^\circ) \end{aligned} \tag{6.2}$$

Tabelle 6.2 gibt für beide Fälle ein Beispiel. In der ersten Spalte sind die Azimute des Vorgänger-Link ("α_{pre}") und des Nachfolgers ("α_{suc}") visualisiert. Die zweite Spalte gibt den resultierenden Richtungswinkel an und die dritte Spalte zeigt, wie sich dieser gemäß Formel 6.2 herleiten lässt.

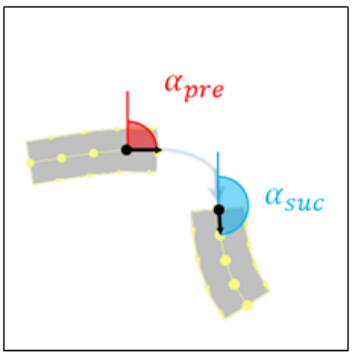
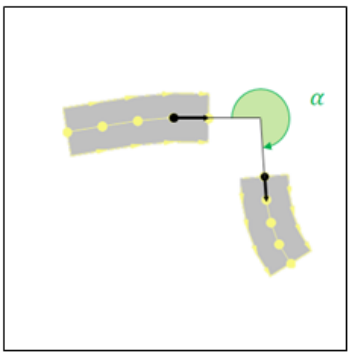
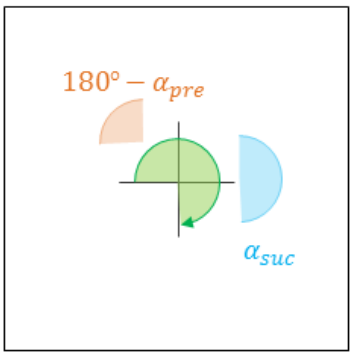
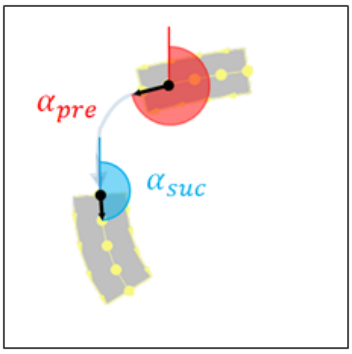
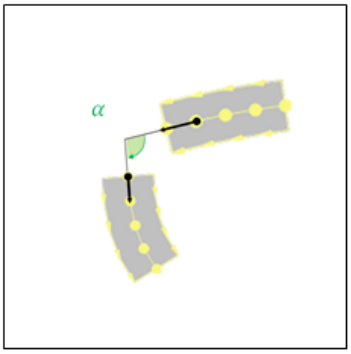
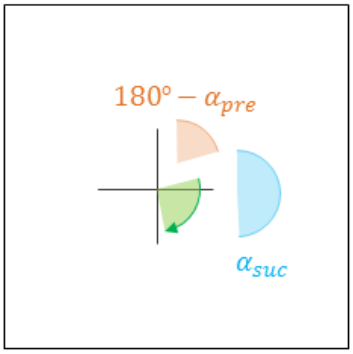
Azimut Vorgänger/ Nachfolger	**Richtungswinkel α**	**Herleitung von α**
		
		

Tabelle 6.2 – Richtungswinkelberechnung aus Azimuten der Richtungsvektoren von Vorgänger und Nachfolger

Damit sind alle grundlegenden Informationen in der Form aufbereitet, sodass mit der Erstellung eines Vissim-Netzwerkes begonnen werden kann.

6.2 Erstellung eines Vissim-Straßennetzwerkes als Plugin-Feature

Die Transformation wird durch das Schreiben einer Netzwerkdatei im Vissim-Format (Dateiendung ".inpx") abgeschlossen, was in der Java Umgebung des 3DCityDb Importer/Exporter umgesetzt ist. Bei der Erstellung der Datei wird auf den Inhalt der Tabellen "Link" und "Connector" zugegriffen. Für Abfragen an die Datenbank wird von der Java-Schnittstelle "Java Database Connectivity"(JDBC) Gebrauch gemacht. Ausgehend von den vorverarbeiteten Daten wird ein Vissim Netzwerk gebildet. Die vorgegebene Struktur wird dazu in Java Klassen abgebildet.

6.2.1 Links

Ein Link-Objekt wird aus den Informationen erzeugt, die im Laufe der Datenaufbereitung in der Tabelle Link gesammelt wurden. Die Abfrage an die Datenbank zeigt Abbildung 6.11. Selektiert werden alle Spalten der Tabelle Link (Zeilen 1-5). Die Geometrie wird dabei auf gesonderte Weise abgefragt. Mit der PostGIS Funktion “ST_DumpPoints“ lassen sich alle Punkte eines Linestrings zeilenweise ausgegeben (Zeile 8). Die einzelnen Punkte verfügen über ein Pfad-Attribut, welches die Stelle innerhalb eines Linestrings angibt. Durch Sortierung nach ID und Pfad ist eine sequenzielle, punktweise Verarbeitung der Link-Geometrien möglich.

```
1  SELECT id, path[1], gmlid, function, name,
2  ST_X(pts.geom) AS x, ST_Y(pts.geom) AS y, ST_Z(pts.geom) AS z,
3  lane_width_meter, number_parking_lanes, number_total_lanes,
4  number_travel_lanes, street_length_2D_meter, street_segment_width_meter,
5  street_speed_limit_km_h, traffic_direction
6  FROM (
7  -- extract points from linestring --
8    SELECT *, (ST_DumpPoints(lod0_network)).*
9    FROM vissim.link
10     ) AS pts
11 ORDER BY id, path;
```

Abbildung 6.11 – SQL-Abfrage zur Erstellung von Link-Objekten

In der Folge wird für jede ID ein neues Link-Objekt erzeugt. Der Konstruktor eines Link-Objektes erfordert die Angabe eines Namens und einer Identifikationsnummer. Als Name wird der Straßenname (gml:name) zusammen mit der gml:id übergeben. Die gml:id wird im Namen mit gespeichert, da die Identifikationsnummer in Vissim (“no“) numerisch zu sein hat. Der Link-Nummer wird die ID aus der CityDB zugewiesen. Die übrigen Attribute eines Vissim-Links werden beim Konstruktor-Aufruf zunächst mit Standardwerten belegt. In einem separaten Schritt wird eine Reihe weiterer CityGML Attribute überprüft und falls zweckdienlich zur Beschreibung von Link-Objekten hinzugezogen. Tabelle 6.3 gibt die Attribute mit zugehöriger Überführungsregel an. Aus dem generischen Attribut zum Tempolimit werden die Geschwindigkeit des Gegenverkehrs und die Meso-Geschwindigkeit hergeleitet. Aus der Codeliste des CityGML Attribut ’function’ lässt sich die Art der farbigen Darstellung in Vissim (“displayType“) und der Streckenverhaltenstyp (“linkBehavType“) ableiten. Auf diese Weise werden semantische Informationen über bestimmte Verkehrsbereiche aus CityGML nach Vissim übertragen.
Darüber hinaus könnten zukünftig weitere der in Tabelle Anhang B.1 aufgeführten Link-Attribute mit Werten belegt werden, die aus einem Stadtmodell abgeleitet wurden. Denkbar wäre beispielsweise eine Berechnung des Gradienten aus den Linienobjekten oder die Bestimmung der Vorausschauweite aus einer Sichtbarkeitsanalyse.

Link-Attribut	Uberführungsregel	Vissim Semantik
assumSpeedOncom	1,2 * street_speed_limit_km_h	
displayType	function < 1500	Road gray
	1800 < function < 1855	Rail (Road)
	function = 1220 ‖ 1500	Pedestrian area gray
isPedArea	function = 1220 ‖ 1500	'TRUE'
linkBehavType	function = 1000 ‖ 1200	Urban (motorized)
	function = 1040 ‖ 1050 ‖ 1060	Right-side rule (motorized)
	function = 1010 ‖ 1020 ‖ 1100	Freeway (free lane selection)
	function = 1220 ‖ 1500	Footpath (no interaction)
	function = 1240 ‖ 1280	Cycle-Track (free overtaking)
mesoSpeed	street_speed_limit_km_h	
name	name ⌒ (gmlid)	
no	id	

Tabelle 6.3 – Übersetzungstabelle von CityGML Attributen nach Vissim

Nach Überführung der semantischen Informationen steht die Geometrie-Definition eines Link-Objektes an. Mit den Attributen “nr_total_lanes“ bzw. “nr_travel_lanes“ werden einem Link Fahrspuren hinzugefügt, je nachdem, ob auch Parkspuren erzeugt werden sollen oder nicht. Das Attribut “lane_width_meter“ gibt die Breite der Fahrspuren an. Für jeden Link wird eine Punktliste (“Points3D“) angelegt (vgl. Abbildung 4.7). Für jede Zeile der Abfrage wird mittels der Link-ID überprüft, ob der Punkt zum aktuellen Link-Objekt gehört oder ob es sich bereits um ein neues Link-Objekt handelt. Im ersten Fall wird der Punkt, bestehend aus seinen X, Y und Z-Koordinaten, der Punktliste hinzugefügt, im zweiten Fall wird die vollständige Geometrie dem aktuellen Link-Objekt hinzugefügt und für den neuen Punkt ein neues Link-Objekt erzeugt. Diese Vorgehensweise wird wiederholt, bis das letzte Link-Objekt einer Abfrage abgearbeitet ist. Alle Link-Objekte werden in einer Liste namens “Links“ gesammelt und als solche einem Vissim-Netzwerk übergeben.

6.2.2 Konnektoren

Die Erstellung von Konnektor-Objekten erfolgt auf Basis der Tabelle “Connector“. Abbildung 6.12 gibt die zugehörige SQL-Abfrage an. Selektiert werden alle Spalten der Tabelle, Geometrien werden dabei koordinatenweise angefragt. Die Sortierung erfolgt nach der Konnektoridentifikationsnummer (“con_id“) sowie der Vorgänger bzw. Nachfolger ID.

```
1 SELECT con_id, pre_id, suc_id, pre, suc, ST_X(point_from) AS x_from,
2 ST_Y(point_from) AS y_from, ST_Z(point_from) AS z_from,
3 ST_X(point_to) AS x_to, ST_Y(point_to) AS y_to, ST_Z(point_to) AS z_to,
4 pos_from, pos_to, number_travel_lanes_pre, number_travel_lanes_suc,
5 number_parking_lanes_pre, number_parking_lanes_suc, dirAngle
6 FROM vissim.connector
7 ORDER BY con_id, pre_id, suc_id;
```

Abbildung 6.12 – SQL-Abfrage zur Erstellung von Konnektor-Objekten

Für jede Konnektor-ID (“con_id“) wird ein neuer Link vom Typ Konnektor erzeugt. Ein Konnektor erhält in der vorliegenden Implementierung einen Namen, der sich aus den IDs seines Vorgängers und Nachfolgers in der Form “preToSuc“ zusammensetzt. Zur Beschreibung der Geometrie wird eine Punktliste “Points3D“ bestehend aus den Punkten der Spalten “point_from“ und “point_to“ angelegt. Weitergehend wäre an dieser Stelle eine Modifizierung der bisher geraden Konnektor-Geometrien zum Beispiel mittels Spline-Interpolation möglich. Eine derartige Funktion ist in Vissim implementiert und kann per Mausklick, falls vom Nutzer als notwendig erachtet, manuell ausgeführt werden. Eine automatische Spline Interpolation ist bisher nicht Teil der Formatüberführung.
Die Anzahl an Fahrspuren eines Konnektors hängt davon ab, wie viele Spuren die jeweiligen Vor- bzw. Nachfolger besitzen. Es lassen sich zwei Fälle unterscheiden: Entweder besitzen beide die gleiche Anzahl, in welchem Fall die Attribute “nrLanesPre“ und “nrLanesSuc“ den gleichen Wert aufweisen oder die Anzahl an Spuren ist unterschiedlich. Sind die Anzahl an Fahrspuren ungleich, kann weiter differenziert werden in die Fälle “nrLanesPre“ ist größer oder ist kleiner als “nrLanesSuc“. Bei Gleichheit erhält der Konnektor die Anzahl an Spuren, die die beiden zu verbindenden Links aufweisen. Im zweiten Fall wird das Minimum des Zahlenpaares aus “nrLanesPre“ und “nrLanesSuc“ bestimmt und dem Konnektor zugewiesen. Beispiele für beide Fälle gibt Abbildung 6.13. Auf der linken Seite ist der Fall gleicher Spuranzahl hervorgehoben. Der Konnektor ’1 $\rightarrow$ 3’ hat wie sein Vorgänger und Nachfolger zwei Fahrspuren. Auf der rechten Seite ist der Fall mit ungleicher Spuranzahl visualisiert. Da der Nachfolger des Konnektors ’1 $\rightarrow$ 50002’ lediglich über eine Fahrspur verfügt, erhält auch der Konnektor nur eine Spur.

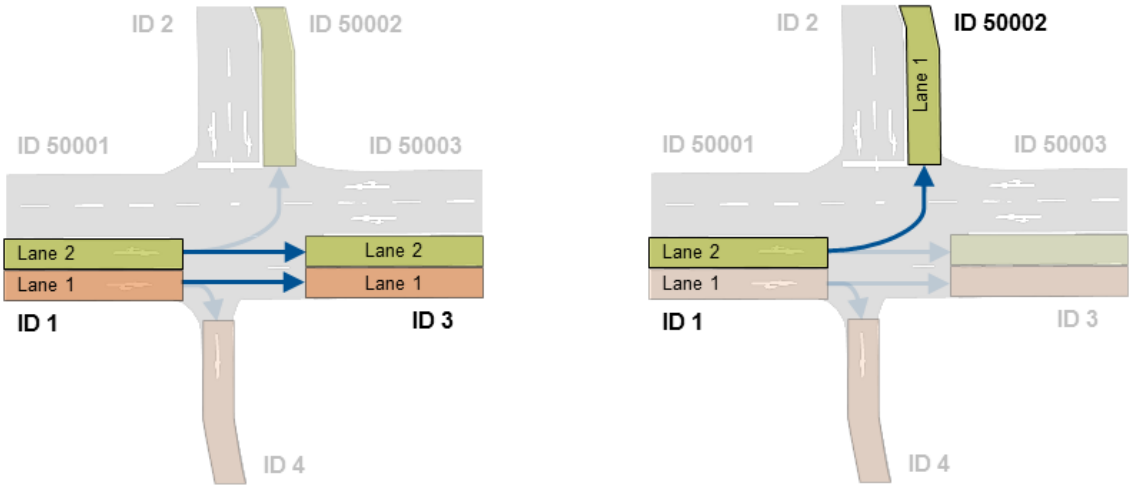


Abbildung 6.13 – Anzahl an Fahrspuren eines Konnektors abgeleitet aus der Spuranzahl der Vor- bzw. Nachfolger: Fall mit gleicher Spuranzahl (links) oder Fall mit ungleicher Spuranzahl (rechts)

Im Falle, dass einer der zu verbindenden Links mehr Fahrspuren besitzt als der andere Link, ist die Anbindung des Konnektors an der einen Seite mehrdeutig. Der Aufbau eines Konnektors beinhaltet die spurgenaue Verknüpfung zweier Links (vgl. Abbildung 4.8 und 4.9). Die Verknüpfung modelliert im XML-Attribut “lane“ der Elemente “<fromLinkEndPt>“ bzw. “<toLinkEndPt>“ wird in der Form “preID-Leerzeichen-laneFrom“ bzw. “sucID-Leerzeichen-laneTo“ beschrieben. Für den rechten Fall in

Abbildung 6.13 muss festgelegt werden, ob der Konnektor '1 $\rightarrow$ 50002' von Fahrspur ("Lane") '1' oder '2' beginnen soll. Um situativ die richtige Spur zu wählen, soll die Überlegung angestellt werden, wie sich Abbieger und Nicht-Abbieger in der Realität unter Rechtsverkehrbedingungen einordnen würden. Für das Beispiel des Links mit der 'ID 1' ist davon auszugehen, dass Linksabbieger Spur '2' wählen, Geradeausfahrer beide Fahrspuren nutzen und Rechtsabbieger sich auf Spur '1' einordnen. Abbildung 6.14 zeigt die Fahrspurnummerierung in Vissim sowie die Einteilung der Spuren nach dem den vorausgegangenen Überlegungen entsprechenden, zu erwartenden Abbiegeverhalten der Autofahrer.

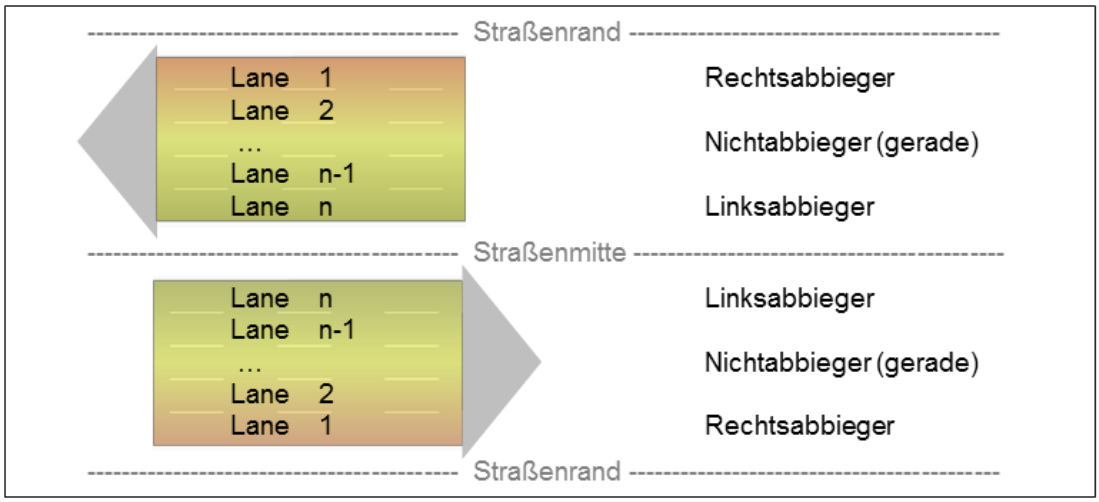


Abbildung 6.14 – Fahrspurnummerierung in Vissim und Einteilung der Fahrspuren nach zu erwartendem Abbiegeverhalten (Rechtsverkehr)

Um die Konnektoren automatisiert der richtigen Fahrspur zuordnen zu können, braucht es ein Unterscheidungskriterium. Da nach der Abbiegerichtung differenziert werden soll, liegt die Betrachtung der Richtungswinkel nahe. Diese wurden auf beschriebene Art und Weise berechnet (vgl. Formel 6.2) und liegen in der Spalte "dirAngle" vor. Für eine normale Vier-Weg-Kreuzung wie im Beispiel Abbildung 6.13 können folgende Unterscheidungskriterien anhand des Richtungswinkels α definiert werden:

$$\alpha < 175^\circ \quad \rightarrow \quad Linksabbieger \tag{6.3a}$$

$$175^\circ \leq \alpha \leq 185^\circ \quad \rightarrow \quad Nichtabbieger \quad (geradeaus) \tag{6.3b}$$

$$\alpha > 185^\circ \quad \rightarrow \quad Rechtsabbieger \tag{6.3c}$$

Mit der Information zu welcher Abbiegekategorie ('Links', 'Mitte' oder 'Rechts') ein Link-Paar zählt, kann die passende Spur gesucht werden. Nach Abbildung 6.14 ist für Linksabbieger die Spur des Vorgängers mit der höchsten Nummer gesucht. Rechtsabbiegern wird Spurnummer '1' zugeteilt, während Geradeausfahrer die mittleren Spuren zugeschrieben werden. Aus dieser Logik lassen sich allgemein die Formeln 6.4a-6.4c zur Spurermittlung für den Link-Vorgänger ("laneFrom") definieren.

Auf der Nachfolger-Seite ist die Spuranbindung eindeutig, da der Konnektor nur so viele Spuren besitzt wie der nachfolgende Link. Deshalb wird dem Element “<toLinkEndPt>“ im Attribut “lane“ der Wert “laneTo“ gleich eins zugewiesen.

$$Links \quad \rightarrow \quad nrLanesPre - nrLanesSuc + 1 \tag{6.4a}$$

$$Mitte \quad \rightarrow \quad ceil(\frac{nrLanesPre - nrLanesSuc + 1}{2}) \tag{6.4b}$$

$$Rechts \quad \rightarrow \quad 1 \tag{6.4c}$$

Bisher wurde nur der Fall betrachtet, dass die Anzahl an Fahrspuren des Vorgängers größer ist als die des Nachfolgers. Der umgekehrte Fall kann jedoch genauso auftreten. Dann gilt es die Spur zu finden, an die der Konnektor am Nachfolger anbinden soll. Dies kann ebenfalls mit den über die Richtungswinkel ermittelten Kategorien erfolgen. Die Formeln 6.5a-6.5c sind auf gleiche Weise aufgebaut wie die obigen mit dem Unterschied, dass die Attribute “nrLanesPre“ und “nrLanesSuc“ in umgekehrter Reihenfolge auftreten. Die Logik beider Ansätze ist die gleiche. In der Situation, in der ein Nachfolger mehr Spuren aufweist als sein Vorgänger, ist die Anbindung am Vorgänger (“<fromLinkEndPt>“) eindeutig. Die Spur im Element “<toLinkEndPt>“ bestimmt sich in diesem Fall wie folgt:

$$Links \quad \rightarrow \quad nrLanesSuc - nrLanesPre + 1 \tag{6.5a}$$

$$Mitte \quad \rightarrow \quad ceil(\frac{nrLanesSuc - nrLanesPre + 1}{2}) \tag{6.5b}$$

$$Rechts \quad \rightarrow \quad 1 \tag{6.5c}$$

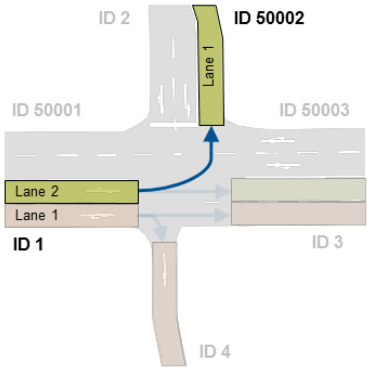
Beispiele für die Spurzuordnung zeigt Tabelle 6.4. In der ersten Zeile ist der Konnektor “1 $\rightarrow$ 50002“ beschrieben. Link ’1’ besitzt zwei Fahrspuren, während Link ’50002’ nur eine Fahrspur aufweist. Der Richtungswinkel zwischen den Vektoren der beiden Links beträgt 90 Grad. In Folge der Kategorisierung als Linksabbieger (Formel 6.3a) kann die Spur wie folgt hergeleitet werden.

nach 6.4a: $laneFrom = nrLanesPre - nrLanesSuc + 1 = 2 - 1 + 1 = 2$

Dies ergibt den Eintrag im Attribut “lane“ von ’1 2’ für das Element “<fromLinkEndPt>“. Auf der Nachfolgerseite tritt keine Mehrdeutigkeit auf, sodass dort das Attribut “lane“ ohne weitere Bestimmung mit dem Wert ’50002 1’ belegt werden kann.
Auf gleiche Art und Weise kann die Spurverknüpfung für die zweite Zeile der Tabelle hergeleitet werden. Da für den Konnektor “5 $\rightarrow$ 7“ gilt, dass “nrLanesPre“ kleiner ist als “nrLanesSuc“, sind die Formeln 6.5a bis 6.5c heranzuziehen.

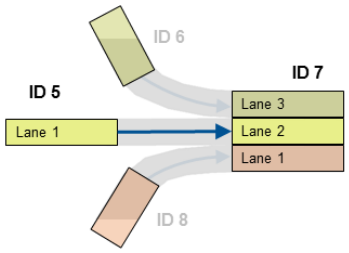
Im untersten Fall ist ein Beispiel wiedergegeben, in dem sich zwei Straßen zu einer vereinen. Korrekterweise liegen also nur zwei Bedingungen vor: von der Mitte kommend oder von rechts kommend. Wendet man die Formeln an, die den drei Bedingungen (Links, Mitte, Rechts) zu Grunde liegen, führt die Herleitung der Spurverknüpfung zu einem falschen Ergebnis. Mit den Fahrspurzahlen und einem Richtungswinkel von 180 Grad wird unter Anwendung von Formel 6.5b Spurnummer '1' gefunden. Das Konzept geht von einer weiteren Relation ('von links kommend') aus, für welche die am weitesten links liegende Spur freigehalten wird. Korrekt wäre Spurnummer '2'.
Das Konzept zur Kategorisierung der Abbiegerelationen mittels Richtungswinkel erlaubt es sinnvolle Vorschläge machen, welche Fahrspuren miteinander zu verknüpfen sind. Eine Angabe der Fahrspurverknüpfung mit hundertprozentiger Sicherheit wird aus den vorliegenden Daten nicht möglich sein. Eine zusätzliche Differenzierung in Abhängigkeit von der Anzahl an Abbiegerelationen könnte zu einer weiteren Erhöhung der Trefferquote bei der Zuordnung führen, welche einhergehen würde mit zunehmender Komplexität eines entsprechenden Formelapparates.

Beispiel	Ausschnitt XML-Code



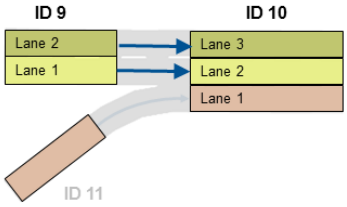
$\alpha = 90°$ → Formel 6.4a

```
<link ... name="1To50002" no="10000" ... vehRecAct="true"/>
        <fromLinkEndPt lane="1_2" pos="10.0"/>
        <geometry>
                <points3D>
                        <point3D x="5" y="0" zOffset="0"/>
                        <point3D x="5" y="5" zOffset="0"/>
                </points3D>
        </geometry>
        <lanes>
                <lane/>
        </lanes>
        <toLinkEndPt lane="50002_1" pos="0"/>
</link>
```



$\alpha = 180°$ → Formel 6.5b

```
<link ... name="5To7" no="10005" ... vehRecAct="true"/>
        <fromLinkEndPt lane="5_1" pos="5.5"/>
        <geometry>
                <points3D>
                        <point3D x="55" y="38" zOffset="0"/>
                        <point3D x="60" y="38" zOffset="0"/>
                </points3D>
        </geometry>
        <lanes>
                <lane/>
        </lanes>
        <toLinkEndPt lane="7_2" pos="0"/>
</link>
```



$\alpha = 180°$ → Formel 6.5b

```
<link ... name="9To10" no="10099" ... vehRecAct="true"/>
        <fromLinkEndPt lane="9_1" pos="6.0"/>
        <geometry>
                <points3D>
                        <point3D x="35" y="99" zOffset="0"/>
                        <point3D x="39" y="99" zOffset="0"/>
                </points3D>
        </geometry>
        <lanes>
                <lane/>
        </lanes>
        <toLinkEndPt lane="10_1" pos="0"/>
</link>
```

Tabelle 6.4 – Beispiele für Konnektoren: spurgenaue Verknüpfung Fall nrPreLanes > nrSucLanes (oben), Fall nrPreLanes < nrSucLanes (mittig) und fehlerhafte Verknüpfung (unten)

6.2.3 Parkspur

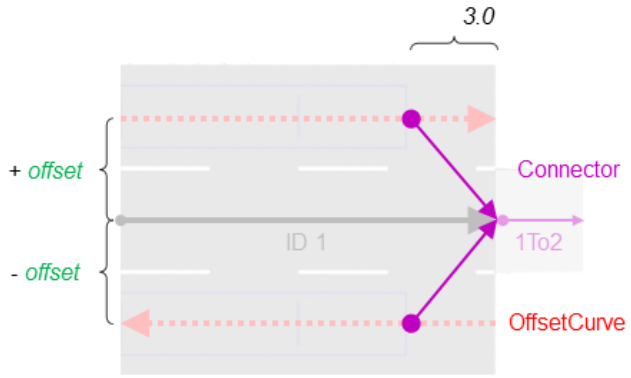
Parkspuren sind eine weitere Komponente, die sich in Vissim auf Grundlage der Links und Konnektoren modellieren lassen. Die Erzeugung von Parkplätzen ist optional und im vorliegenden Konzept für Straßen mit einer oder mehreren Parkspuren vorgesehen. Die notwendigen Informationen lassen sich - wie in Abbildung 6.15 wiedergeben - aus der Tabelle “Link“ abrufen. Zur Definition eines Stellplatzes in Vissim sind die ID und Fahrspurnummer des Links erforderlich, auf der sich der Stellplatz befindet. Hinzu kommen die Position, an der die Parkbucht beginnt, sowie die Länge.

```
SELECT l.id, l.name, l.number_parking_lanes, l.number_total_lanes,
ST_Length(l.lod0_network) AS length, ST_X(geom.point_from1) AS x_from1,
ST_Y(geom.point_from1) AS y_from1, ST_Z(geom.point_from1) AS z_from1,
ST_X(geom.point_from2) AS x_from2, ST_Y(geom.point_from2) AS y_from2,
ST_Z(geom.point_from2) AS z_from2, ST_X(ST_EndPoint(l.lod0_network)) AS x_to,
ST_Y(ST_EndPoint(l.lod0_network)) AS y_to,
ST_Z(ST_EndPoint(l.lod0_network)) AS z_to
FROM
  (SELECT id,
  -- connector geometry calculation: 1st parking lane
  MAX(CASE WHEN number_parking_lanes>0
    THEN ST_Line_Interpolate_Point(ST_OffsetCurve(lod0_network,
         -(number_total_lanes/2-0.5)*lane_width_meter, 'quad_segs=4 join=round'),
         3/ST_Length(lod0_network))
    END) AS point_from1,
  -- connector geometry calculation: 2nd parking lane
  MAX(CASE WHEN number_parking_lanes=2
    THEN ST_Line_Interpolate_Point(ST_OffsetCurve(lod0_network,
         (number_total_lanes/2-0.5)*lane_width_meter, 'quad_segs=4 join=round'),
         (ST_Length(lod0_network)-3)/ST_Length(lod0_network))
    END) AS point_from2
  FROM vissim.link
  GROUP BY id) AS geom
JOIN vissim.link l ON l.id=geom.id
WHERE number_parking_lanes > 0 AND number_parking_lanes < number_total_lanes;
```

Abbildung 6.15 – SQL-Abfrage zur Erstellung von Parkspuren

Im zugrunde liegenden Konzept sind Parkspuren nur für Link-Objekte vorgesehen. Konnektoren, die zwei Link-Objekte verknüpfen, besitzen keine Parkspuren. Zweck dieses Ansatzes ist, dass Parken in Kreuzungsbereichen und Abbiegen auf Parkspuren ausgeschlossen wird. Daraus resultiert jedoch die Notwendigkeit von zusätzlichen Konnektoren am Ende einer jeden Parkspur, was die Abfrage aus Abbildung 6.15 so umfangreich macht. Mit den zusätzlichen Konnektoren wird sichergestellt, dass Fahrzeuge, die bis zum Ende der Fahrspur fahren, wieder in das Straßennetz eingefädelt werden. Dargestellt ist die Problematik in Abbildung 6.16. Aufwendig zu finden sind dabei nicht die Verknüpfungsinformationen (“fromLinkEndPt“ bzw. “toLinkEndPt“), sondern die Beschreibung der Geometrien derartiger Konnektoren. Die Information zur Beschreibung der Verknüpfung (Link-ID und Fahrspur) besitzt jeder Stellplatz. Der Zielpunkt des Konnektors lässt sich ebenfalls unschwer festle-

gen. Definiert wird dieser durch den Endpunkt des Links, auf dem der Stellplatz liegt. Aufwendiger zu finden ist der Startpunkt. Gesucht ist ein Punkt, der in der Parkspurachse liegt. Zur Bestimmung eines solchen Punktes wird die Link-Achse um einen Offset verschoben. Der Offset berechnet sich nach Formel 6.6 aus der Gesamtzahl an Fahrspuren und der Fahrspurbreite. Mit der PostGIS Funktion “ST_OffsetCurve“ wird eine Hilfsachse pro Parkspur erzeugt, welche in Abbildung 6.16 rot dargestellt ist. Mit der PostGIS Funktion “ST_Line_Interpolate_Point“ lässt sich auf dieser Hilfsachse im Abstand von 3 m zum Ende der Parkspur ein Punkt interpolieren, welcher als Startpunkt eines Parkspur-Konnektors geeignet ist.



$$offset = (\frac{nrTotalLanes}{2} - \frac{1}{2}) \cdot laneWidth \quad (6.6)$$

Abbildung 6.16 – Erstellung von Konnektoren am Ende von Parkspuren

Da die Ausgangsdaten lediglich die Anzahl aber nicht die Spurnummer einer Parkspur umfassen, gilt es Regeln zur automatischen Erzeugung von Vissim Parkspuren festzusetzen: Verfügt ein Link über eine Parkspur, wird diese der rechten Fahrbahnseite und damit der Fahrspur mit der Nummer ’1’ zugewiesen. Im Fall von zwei Parkspuren wird die zweite Parkspur am linken Fahrbahnrand eingerichtet. In diesem Fall entspricht die zur Parkbucht gehörige Fahrspurnummer der Gesamtzahl an Fahrspuren. Veranschaulicht ist das Konzept in Abbildung 6.17. Die Stellplätze beginnen und enden jeweils in 3 Metern Entfernung vom Link-Anfang bzw. Ende.

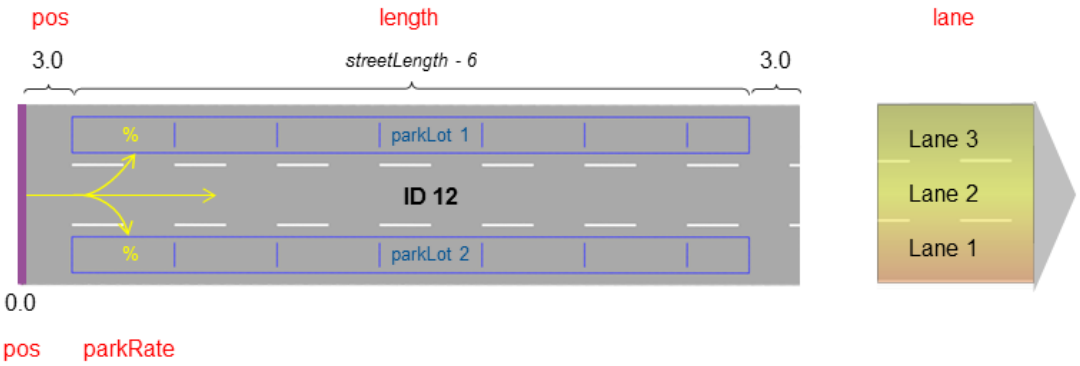


Abbildung 6.17 – Modellierung von Parkspuren mit den Vissim Netzwerk-Komponenten “parkingLot“ und “vehicleRoutingDecisionParking“

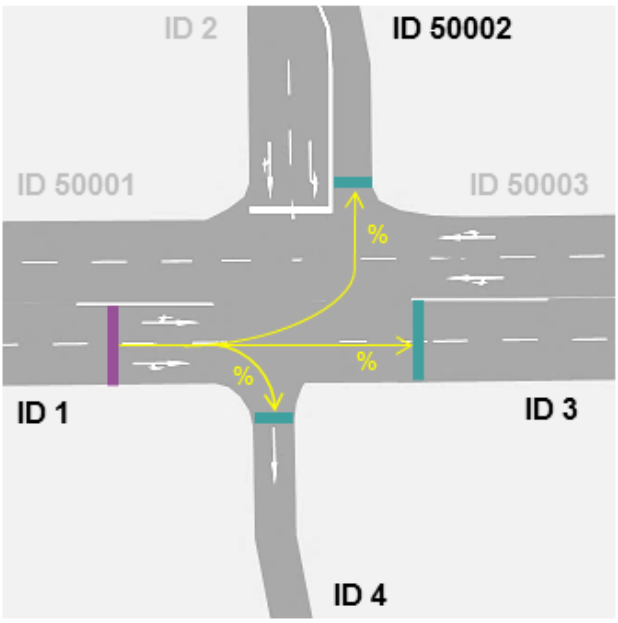
Damit tatsächlich Autos auf den Parkspuren anhalten, sind Routenentscheidungen vom Typ Parkplatz anzulegen, welche per Referenz (“ID“) an ein Link-Objekt gebunden sind. Der Beginn einer Routenentscheidung ist durch eine Position entlang des Links definiert, das Ende stellt der jeweilige Ziel-Parkplatz dar. Weitere beschreibende Attribute eines Parkplatzes sind die Haltedauer (20 Sekunden) und der Parker-Anteil (25%). Die in Klammern angegeben Standardwerte können vom Nutzer nachträglich manuell in Vissim verändert werden. In Abbildung 6.17 ist der Startquerschnitt der Routenentscheidung in Lila, die Routenoptionen in Gelb angedeutet.

6.2.4 Fahrzeugrouten

Fahrzeugrouten dienen der lokalen Verteilung von Fahrzeugen im Simulationsnetz [PTV, 2016]. Statische Routenentscheidungen führen Fahrzeuge von einem Startquerschnitt (Abbildung 6.19 - lila) zu einem der definierten Zielquerschnitte (türkis). Dabei kann für jeden Zielquerschnitt eine relative Belastung definiert werden. Aus dieser ergibt sich der prozentuale Anteil der Routenteile. Abbildung 6.18 gibt die zur Erstellung von Fahrzeugrouten zugehörige Datenbankabfrage an. Benötigt werden die ID des Vorgänger-Links für den Startquerschnitt, die Konnektor-ID für den Verlauf und die ID des Nachfolger-Links für den Zielquerschnitt. Um all diese Informationen in einer Abfrage zu vereinen, sind zwei Join-Aufrufe (Zeilen 9 und 10) notwendig. Für jeden neuen Startquerschnitt wird ein XML-Element “<vehicleRoutingDecisionStatic>“ angelegt (vgl. Abbildung 6.20). Für die exakte Position des Startquerschnittes wird die Länge des Vorgänger-Links berechnet und halbiert (Zeile 2).
Jeder Zielquerschnitt ist in einem XML-Element “<vehicleRouteStatic>“ definiert. Als Attribute sind die ID des Nachfolgers als Ziellink, eine Position entlang des Ziellinks (“destPos“ = 2 m), ein Name, eine Identifikationsnummer und der Fahrzeuganteil anzugeben. Der Name wird dabei in Abhängigkeit der Abbiegekategorie vergeben, welche sich aus dem Richtungswinkel herleiten lässt. Zugeteilt werden die Attributwerte ’L’ (Left), ’T’ (Through) und ’R’ (Right). Ein Wert für den Prozentanteil jedes Routenteils wird aus dem Verhältnis der Fahrspuren eines Ziellinks zur Summe an Fahrspuren aller Ziellinks hergeleitet (Formel 6.7). Die Summe der Fahrspuren wird aus einer nach der Vorgänger-ID gruppierten Teilabfrage bestimmt (vgl. Abbildung 6.18 Zeilen 4-7). Für das Code-Beispiel in Abbildung 6.20 ergibt sich nach Formel 6.7 ein Anteil von $\frac{1}{4} = 25\%$.

```
1  SELECT c.con_id, c.pre_id, c.suc_id, c.number_travel_lanes_suc,
2  nr.sum_lanes, c.dirangle, ST_Length(l.lod0_network)/2 AS dec_pos_from
3  FROM (
4    -- calculate total lane possibilities
5    SELECT pre_id, sum(number_travel_lanes_suc) AS sum_lanes
6    FROM vissim.connector
7    GROUP BY pre_id) AS nr
8  -- join connector with link information
9  JOIN vissim.connector c ON nr.pre_id=c.pre_id
10 JOIN vissim.link l ON nr.pre_id=l.id;
```

Abbildung 6.18 – SQL-Abfrage zur Erstellung von Fahrzeugrouten



$$percentage = \frac{nrTotalLanesSuc}{sumLanes} \tag{6.7}$$

Abbildung 6.19 – Fahrzeugroute mit Startquerschnitt (lila) und Zielquerschnitt (türkis)

```
<vehicleRoutingDecisionsStatic>
	<vehicleRoutingDecisionStatic . . . link="1" name="" no="1" pos="2.5">
		<vehRoutSta>
			<vehicleRouteStatic destLink="4" destPos="2" name="R" no="1" relFlow="2 0:0.25">
				<linkSeq>
					<intObjectRef key="10003"/>
				</linkSeq>
			</vehicleRouteStatic>
	. . .
		</vehRoutSta>
	</vehicleRoutingDecisionStatic>
</vehicleRoutingDecisionsStatic>
```

Abbildung 6.20 – XML-Code Beispiel für eine Fahrzeugroute ausgehend vom Startquerschnitt auf Link '1' zum Zielquerschnitt auf Link '4'

6.2.5 Erstellen der XML-Datei

Ein Vissim Netzwerk wird als XML-basierte Datei gespeichert. Sämtliche Netzwerkkomponenten (Links, Parkspuren, Routenentscheidungen, ...) und Einstellungen werden als Subelemente des XML-Elements "<network>" geführt. Eine valide Vissim Datei verfügt über eine Reihe von Standardeinstellungen. Darin sind beispielsweise grafische Komponenten definiert wie die Art der grafischen Repräsentation von Netzobjekten oder simulationsspezifische Voreinstellungen wie das Beschleunigungs- und Abbremseverhalten von Fahrzeugen. Die XML-Datei "defaults.inpx" umfasst insgesamt 1951 Zeilen.

Um aus den aus CityGML abgeleiteten Vissim-Objekten eine valide Datei zu formen, die von Vissim importiert werden kann, ist es notwendig die erstellten Komponenten mit den Voreinstellungen in einer Netzwerk-Datei zusammenzuführen. Dazu werden die Standardeinstellungen in einer eigenen Java-Klasse als String-Objekte angelegt. Mit der JAXB-API (“Java Architecture for XML Binding“) werden die Java-Objektzustände der einzelnen Netzwerkkomponenten auf XML-Dokumente übertragen. Die Zusammenführung mit den Standardeinstellungen wird mit Hilfe eines “BufferedWriter“ umgesetzt. Mit der Java-Klasse lassen sich mehrere in den Ausgabestrom geleitete Ausgaben puffern. Auf diese Weise werden mehrere Schreiboperationen zu einer zusammengefasst, was sich positiv auf die Performance auswirkt.

7 Ergebnisrückführung - Integration von Simulationsdaten in 3D-Stadtmodelle

Wie in Kapitel 5 erarbeitet lassen sich dynamische Daten in CityGML mit Hilfe des Dynamizer Konzeptes integrieren. Da dieser Ansatz derzeit noch keine Unterstützung durch die 3DCityDB erfährt, muss im Rahmen dieser Arbeit eine alternative Lösung gesucht werden. Ziel ist eine Visualisierung der Fahrzeuge und ihrer Bewegung in einem 3D-Stadtmodell. Der 3DCityDB Importer/Exporter beinhaltet für Stadtmodelle in CityGML eine Export Funktion nach KML, COLLADA respektive glTF, Formaten zur Visualisierung in virtuellen Globen wie Google Earth, ArcGIS Explorer oder Cesium [Kolbe et al., 2016]. Da die Ergebnisvisualisierung in einem derartigen virtuellen Globus erfolgen soll, liegt es nahe diese Formate für die Umsetzung heranzuziehen. Voraussetzung ist die Unterstützung zeitabhängiger Informationen, was auf KML zutrifft.
In diesem Kapitel wird vorgestellt, wie sich dynamische Simulationsergebnisse mit Hilfe von KML in einem 3D-Stadtmodell visualisieren lassen. Dazu wird zunächst auf den Aufbau einer KML-Datei mit dynamischen Fahrzeugdaten eingegangen, ehe beschrieben wird, wie die Überführung von einem Vissim Fahrzeugprotokoll nach KML vollzogen wird. Der letzte Abschnitt behandelt die Integration der Visualisierung von Fahrzeugdaten in den 3D-Web-Map-Client.

7.1 KML zur Visualisierung von Trajektoriendaten

Das Vissim Fahrzeugprotokoll enthält im Wesentlichen die Trajektorien, d.h. Sequenzen von Positionsdaten über den Simulationszeitraum, welche die Bewegung der Fahrzeuge beschreiben.
KML, seit 2008 offizieller OGC Standard, beinhaltet eine Möglichkeit Trajektorien zu visualisieren. Klassischer Weise werden Objekte mit zugehöriger Geometrieinformation in KML als “Placemark“ modelliert. Auf diese Weise lässt sich zum Beispiel ein Punkt auf der Erde beschreiben und als Icon visualisieren. Die Position wird in Form geographischer Koordinaten angegeben. Weiter kann ein Placemark mit Attributen wie einem Namen oder einer Beschreibung ausgestattet werden. Für Version 2.2 gibt es zusätzlich verschiedene Erweiterungen, welche in einem eigenen Namensraum gekennzeichnet mit dem Präfix “gx“ zusammengefasst sind. Zur Speicherung zeitabhängiger Positionsdaten wie beispielsweise (GPS-)Trajektorien stehen innerhalb des gx-Namensraumes die Elemente “MultiTrack“ und “Track“ zur Verfügung, welche eine statische Ortsmarkierung um dynamische

Aspekte erweitert[1]. Ein “MultiTrack“ wird verwendet, um mehrere “Tracks“ zu einer konzeptionellen Einheit zu kombinieren. Ein “Track“ beschreibt, wie sich ein Objekt über einen bestimmten Zeitraum bewegt. Definiert wird die Bewegung mit Hilfe der Subelemente “when“ (Zeitpunkt), “coord“ (korrespondierende Position in geographischen Koordinaten) sowie optional “angles“ (Orientierung zum entsprechenden Zeitpunkt). Zusätzlich kann ein Modell innerhalb eines Track-Elements spezifiziert werden, welches die standardmäßige Visualisierung mittels Icons ersetzt [KML Reference, 2016].
Abbildung 7.1 gibt die notwendige Struktur einer Visualisierung von Trajektoriendaten in KML wieder. Beispielhaft ist ein Fahrzeug mit der ID ’1’ und dem Namen ’car 1’ als Placemark spezifiziert. Das Aussehen der Ortsmarke referenziert auf einen Style namens ’invisible’, der im gleichen KML-Dokument definiert sein muss. Es folgen die dynamischen Bewegungsinformationen, schematisch dargestellt anhand eines Zeitpunktes, der korrespondierenden Position und Orientierung (Gier-, Nick- und Rollwinkel) sowie einer Modellspezifizierung.

```
<Placemark id="1">
        <name>car 1</name>
        <description>Model 1</description>
        <styleUrl>#invisible</styleUrl>
        <gx:MultiTrack>
                <altitudeMode>absolute</altitudeMode>
                <gx:Track>
                        <when>2017-05-05T12:00:04.40Z</when>
                        ...
                        <gx:coord>-73.9855067361291 40.7381796122102 0.0</gx:coord>
                        ...
                        <gx:angles>22.460794716431565 0 0</gx:angles>
                        ...
                        <Model id="model_1">
                                ...
                        </Model>
                </gx:Track>
        </gx:MultiTrack>
</Placemark>
```

Abbildung 7.1 – KML-Code Beispiel zur Visualisierung von Trajektoriendaten

7.2 Überführung Vissim Fahrzeugprotokoll nach KML

Um den Workflow aus Abbildung 2.7 zu vervollständigen, soll das Vissim Plugin für den 3DCityDB Importer/Exporter um eine Funktion erweitert werden, die es ermöglicht automatisch aus einem Vissim Fahrzeugprotokoll eine KML-Datei abzuleiten. Die KML-Datei dient dann als Grundlage zur Visualisierung der Simulationsergebnisse im 3D-Stadtmodell. Die Überführung erfolgt in drei Schritten: Dem Einlesen des Fahrzeugprotokolls in den Speicher folgt eine Prozessierung der Daten. Den Abschluss bildet das Erstellen und Schreiben der KML-Datei.
Das Vissim Fahrzeugprotokoll ist in Form einer CSV-Datei aufgebaut. Unterschiede im Vergleich zu einer normalen CSV-Datei sind die Dateiendung “.fzp“ und der mehrzeilige Dateikopf, welcher eine Beschreibung der ausgegebenen Attribute des Protokolls enthält. Welche Attribute ausgegeben

[1] In KML Version 2.3 wurden viele Elemente der gx-Erweiterung fest aufgenommen. Da sich Beispiele und Literatur nach wie vor weitestgehend auf KML 2.2 beziehen, wurde diese Version zur Implementierung herangezogen.

werden sollen, lässt sich in Vissim unter dem Menüpunkt Auswertung/ Konfiguration/ Direktausgabe/ Fahrzeugprotokoll/ Attribute einstellen. Tabelle 7.1 gibt die für eine Visualisierung notwendigen sowie optionale Attribute jeweils mit einem Beispiel an. Bisher werden für die Visualisierung keine optionalen Informationen herangezogen. Denkbar ist, zeitbezogene Informationen wie die aktuelle Geschwindigkeit ebenfalls abrufbar zu machen.

Attribut	**Beispiel**	**Angabe**
Simulationssekunde	4.40	erforderlich
Simulations-Uhrzeit	12:00:04.40	optional
Nummer	1	erforderlich
2D/3D-Modell	1	optional
Koordinate vorne	300852.878 63492.119 0.000	erforderlich
Koordinate hinten	300849.210 63494.186 0.000	erforderlich
Fahrzeugtyp	100	optional
Name	Auto 1	optional
Besetzung	1	optional
Geschwindigkeit	56,43	optional
Wegstrecke zurückgelegt (gesamt)	1,57	optional

Tabelle 7.1 – Attribute eines Vissim Fahrzeugprotokolls mit Beispiel und Angabe über die Erforderlichkeit für eine Visualisierung mittels KML

Das Plugin ist dabei nicht auf Fahrzeugprotokolle aus Vissim beschränkt, sondern auch auf Ergebnisse anderer Verkehrssimulationen anwendbar. So liefert die Open Source Lösung SUMO als Ergebnis ebenfalls Fahrzeugpositionen über den Simulationsverlauf. Standardmäßig liegen diese zwar im XML-Format vor, können aber mit Hilfe eines bereitgestellten Python-Skriptes in eine CSV-Datei umgewandelt werden[2]. Für die Verwendung im Plugin ist lediglich sicherzustellen, dass das Protokoll eine Zeile mit den Spaltennamen enthält, eingeleitet vom Identifikator “$VEHICLE:“. Abbildung 7.2 gibt ein Minimalbeispiel für den Anfang eines Fahrzeugprotokolls inklusive Spaltennamen und einer Zeile mit Werten. Ist die Position lediglich über ein Koordinatentupel beschrieben, kann an Stelle der Spalten “COORDFRONT“ und “COORDREAR“ eine Spalte namens “COORD“ angegeben werden. In diesem Fall entfällt die Berechnung einer Modell-Orientierung.

```
1 $VEHICLE:SIMSEC;NO;COORDFRONT;COORDREAR
2 2.40;1;300856.242 63567.573 0.000;300857.092 63571.697 0.000
```

Abbildung 7.2 – Ausschnitt aus einem Fahrzeugprotokoll: Zeile mit Spaltennamen und Beispiel für eine Zeile mit Werten

[2] http://sumo.dlr.de/wiki/Simulation/Output (Zugriff: 01.06.2017)

Das Fahrzeugprotokoll wird in der Java-Umgebung mit Hilfe eines “BufferedReader“ in den Speicher geladen. Dabei wird der Kopf weitestgehend übersprungen. Lediglich die Spaltennamen werden in einer HashMap gespeichert, die jedem Spaltennamen einen Index zuordnet. Zeilenweise wird daraufhin eine Datenprozessierung vorgenommen. Dazu wird aus vorderer und hinterer Koordinate eines Zeitpunktes der Mittelpunkt, d.h. die aktuelle Position (Formel 7.1), sowie die Orientierung (Formel 7.2) errechnet:

$$x = \frac{x_{front} + x_{rear}}{2} \qquad y = \frac{y_{front} + y_{rear}}{2} \qquad z = \frac{z_{front} + z_{rear}}{2} \tag{7.1}$$

$$angle = atan2(\frac{x_{front} - x_{rear}}{y_{front} - y_{rear}}) \tag{7.2}$$

Anschließend wird die Position vom projizierten System, welches in Vissim verwendet wurde (z.B. EPSG:3857 oder eine andere UTM-Projektion), in geographische Koordinaten des WGS84-Systems (EPSG:4326) transformiert. Dazu wird die Java-Bibliothek “Coordinate Transformation Suite“ (CTS) herangezogen. CTS stellt ein lizenzfreies Softwarepaket zur Durchführung von Koordinatentransformationen auf Basis gängiger geodätischer Formeln zur Verfügung[3].
Eine Formatumwandlung ist nur für drei Komponenten notwendig. Tabelle 7.2 gibt das Ursprungsattribut aus dem Fahrzeugprotokoll sowie die erforderliche Formatierung der drei KML-Elemente anhand eines Beispiels an. Die restlichen Attribute erfordern keinerlei Prozessierung und werden zusammen mit den abgeleiteten in einer Liste vom Typ “VehicleRecord“ gespeichert.

Quellattribut(e)	**KML-Element**	**Beispiel**
SIMSEC/ SIMTMOFDAY	<when>	2017-05-05T12:00:04.40Z
COORDFRONT/ COORDREAR	<gx:coord>	-73.98992 40.73844 0.0
COORDFRONT/ COORDREAR	<gx:angles>	29.402231503745242 0 0

Tabelle 7.2 – Ableitung und Formatierung der KML-Elemente

Was folgt, ist die Erstellung des KML-Dokuments gemäß eines vorgegebenen XSD-Schemas, welches als Java-Klassen abgebildet wurde. Ein KML-Dokument und die notwendigen Namensräume werden definiert. Dem Dokument wird der Name “TrafficSimVisualization“ zugewiesen. Als Style wird entweder ein “Icon“ oder ein Style “Model“ definiert. Ersterer dient zur Visualisierung der Fahrzeugpositionen in Form eines Google Icons (“placemark_circle“). Der zweite unterdrückt etwaige Icons und ermöglicht eine Visualisierung mit Hilfe eines 3D-Models.
Die Liste mit den Fahrzeugbewegungsdaten wird nach der Fahrzeugnummer sortiert, was den aufwendigsten Schritt der Überführung darstellt. Auf Basis der sortierten Liste kann nachfolgend für jede neue Fahrzeugnummer im Protokoll ein neues KML-Placemark-Element erstellt werden. Die Ortsmarkierung erhält als Namen den String “car“ in Verknüpfung mit seiner Fahrzeugnummer. Dem

[3] https://github.com/orbisgis/cts

Element “Description“ wird, falls vorhanden, die Modellnummer übergeben. Als Style wird eine der genannten Varianten festgelegt. Solange sich eine Fahrzeugnummer nicht ändert, werden der Ortsmarkierung die Bewegungsinformationen in Form eines Zeitpunktes sowie der korrespondierenden Position und Orientierung in das KML-Element “gx:Track“ mit den Kindelementen “when“, “gx:coord“ und “gx:angles“ überschrieben. Das Prozedere wird wiederholt, bis das Ende der Liste mit den Fahrzeuginformationen erreicht ist. Das aufgebaute Konstrukt, zwischengespeichert in einem Objekt der Klasse “Kml“, kann anschließend mit Hilfe eines Marshallers aus der JAXB API (vergleiche Kapitel 6) in ein XML respektive KML-Dokument überführt werden.

7.3 Integration in den 3D-Web-Map-Client

Ab Version 3.3.0 bietet das 3D City DB Softwarepacket eine hoch performante, web-basierte Lösung zur 3D-Visualisierung. Der 3DCityDB-Web-Map-Client basiert auf dem Cesium Virtual Globe (vgl. Kapitel 2) und ermöglicht eine interaktive Betrachtung und Exploration semantischer 3D-Stadtmodelle im Webbrowser. Verschiedene Erweiterungen zu Cesium erleichtern eine Visualisierung von Stadtmodelldaten. So lassen sich mit Hilfe des 3DCityDB Importer/Exporter im KML/glTF-Format exportierte CityGML Stadtobjekte einfach als zusätzlicher Layer einer Cesium Szene hinzufügen. Weiter besteht die Möglichkeit Stadtmodelle mit Tabelleninformationen, erzeugt mit dem Spreadsheet Generator Plugin, per Hyperlink zu verknüpfen [Kolbe et al., 2016].
Der Web Client unterstützt das Laden von KML-Dateien. Bisher keine Unterstützung finden dynamische Inhalte. Teil dieser Arbeit ist eine zusätzliche Erweiterung des Web-Clients, der die Integration der dynamischen Simulationsergebnisse einer Verkehrssimulation inklusive einer Visualisierung mit 3D-Automodellen im glTF-Format zulässt. Die Umsetzung in JavaScript zeigt Abbildung 7.3. Der Code wird unter dem Namen “TrafficSimulationUtil.js“ im Verzeichnis der bestehenden JavaScript Dokumente des Web-Map-Clients abgelegt. Im ersten Teil des Skripts erfolgt das Laden des KML-Inhaltes unter Verwendung der Cesium Funktionalität “KmlDataSource.load“ (Zeile 12). Dazu müssen Optionen für den Cesium Viewer definiert und der Pfad der KML-Datei angegeben werden. Es folgt das Hinzufügen der Daten zum Cesium Viewer (Zeile 15). Zur Zuordnung von 3D-Automodellen werden die einzelnen KML-Objekte per Schleife durchlaufen. Eine Standard-Visualisierung mittels Billboard wird ausgeschaltet. Stattdessen sollen die Fahrzeuge abhängig von der Kameraentfernung entweder als Punkt oder durch ein 3D-Modell visualisiert werden. Um die Darstellung einer Entität zu definieren, wird die im Zuge der Überführung des Fahrzeugprotokolls im KML-Element “description“ gespeicherte Modellinformation genutzt. Ein Switch-Befehl erlaubt die Festlegung einer beliebigen Anzahl unterschiedlicher Darstellungsformen (Zeile 32f). Anzugeben sind jeweils die Farbe einer Punktvisualisierung und ein Pfad zu einem glTF-Modell. Die Zuordnung erfolgt in Abhängigkeit von der in Zeile 26/27 extrahierten Modellnummer. Ab Zeile 45 werden die Punktrepräsentation und das Modell sowie die Regeln zur entfernungsabhängigen Wiedergabe (Zeilen 50/51 & 57) festgesetzt. Zur Bestimmung der Orientierung eines Modells gemäß der Fahrtrichtung bietet Cesium eine eigene Methode “VelocityOrientationProperty“ (Zeile 58).

```
var TrafficSimulationUtil = {

   loadDynamicKML: function() {

      var options = {
         camera : cesiumViewer.scene.camera,
         canvas : cesiumViewer.scene.canvas
      };

      /** define path of kml with dynamic content **/
      var promise = Cesium.KmlDataSource.load('http://localhost:8000/data/TrafficSim.kml', options);

      promise.then(function(dataSource) {
         cesiumViewer.dataSources.add(dataSource);

         // loop kml entities
         var entities = dataSource.entities.values;
         for (var i = 0; i < entities.length; i++) {
            var entity = entities[i];

            // disable billboard
            entity.billboard.show = false;

            // get model nr from kml description
            var desc = String(Object.values(entity.description)[0]);
            var m = desc.substring(desc.indexOf("Model")+6,desc.indexOf("</div>"));
            var color;
            var uri;

            /** define point color and model path **/
            switch(m){
               default:
                  color = Cesium.Color.RED;
                  uri = "http://localhost:8000/data/CarModels/AudiRS5/AudiRS5.gltf";
                  break;
            //  -- optional: alternative models --
                  case "1":
                  color = Cesium.Color.GOLD;
                  uri = "http://localhost:8000/data/CarModels/Porsche911/Porsche911.gltf";
                  break;
            /*  case ... */
            }

            // camera dstance > 250 m: point visualization
            entity.point = {
               pixelSize : 10,
               color : color
            };
            entity.point.distanceDisplayCondition =
               new Cesium.DistanceDisplayCondition(250.0, 100000.0);

            // camera dstance < 250 m: model viusalization
            entity.model = {
               uri : uri
            };
            entity.model.distanceDisplayCondition = new Cesium.DistanceDisplayCondition(0.0, 250.0);
            entity.orientation = new Cesium.VelocityOrientationProperty(entity.position);

         }
      });
   }
};
```

Abbildung 7.3 – JavaScript Code: Erweiterung des 3D-Web-Map-Clients um die Möglichkeit dynamische Inhalte einer KML-Datei zu laden und diese mit Hilfe von 3D-Modellen zu visualisieren

Die Erweiterung zum Laden dynamischer Daten und assoziierter Modelle muss in den 3D Web-Map-Client integriert werden. Dazu wird das neue Script “TrafficSimulationUtil.js“ als Quelle der Index HTML-Datei hinzugefügt. Die notwendige Zeile zeigt Abbildung 7.4.

```
1 <script src="../js/TrafficSimulationUtil.js"></script>
```

Abbildung 7.4 – Hinzufügen des Sourcecodes in der Index HTML-Datei

Um die Funktionalität im Kontext des 3D Web-Map-Clients ausführen zu können, ist außerdem eine Anpassung der Datei “script.js“ erforderlich. Abbildung 7.5 gibt die Einbettung an Zeile 178 des bestehenden JavaScripts an.

```
177         ...
178
179 // call traffic simulation util
180 TrafficSimulationUtil.loadDynamicKML();
181
182         ...
```

Abbildung 7.5 – Einbettung der Erweiterung in bestehende JavaScript-Umgebung des 3DCityDB-Web-Map-Client

Mit der vorgenommenen Erweiterung können dynamische Ergebnisse einer Verkehrssimulation visualisiert werden. Alle Funktionalitäten des 3D Web-Map-Clients sind weiter uneingeschränkt nutzbar. So lassen sich Straßen, Gebäude und weitere Objekte eines Stadtmodells als Layer hinzufügen, was zum zweiten großen Ziel der Arbeit führt: einer Visualisierung von dynamischen Simulationsergebnissen im 3D-Stadtmodell.

8 Vissim Plugin für den 3DCityDB Importer/Exporter

Die Implementierung der Konzepte zur automatischen Ableitung eines Simulationsgraphen aus CityGML (Kapitel 6) sowie zur Visualisierung von dynamischen Simulationsergebnissen im Stadtmodell (Kapitel 7) erfolgt als Plugin für den 3DCityDB Importer/Exporter. Als Verwaltungstool von 3D-Stadtmodellen im CityGML Format stellt der 3DCityDB Importer/Exporter eine ideale Umgebung dar. Ein großer Vorteil ist, dass im Laufe des Prozesses bereits existierende Komponenten und Funktionen des Importer/Exporters genutzt werden können. Dank der Plugin API ist eine unkomplizierte Erweiterung um zusätzliche Funktionalitäten möglich.

8.1 Benutzeroberfläche

Die Benutzeroberfläche des im Rahmen dieser Arbeit entwickelten Plugins mit dem Reiternamen “Vissim“ unterteilt sich in zwei Abschnitte. Der obere Teil enthält Funktionalitäten zum Export eines Simulationgraphen für Vissim. Der untere Teil stellt eine Oberfläche zur Erzeugung einer KML-Datei aus einem Vissim Fahrzeugprotokoll zur Verfügung, mit dessen Hilfe sich die dynamischen Simulationsergebnisse in einem Stadtmodell visualisieren lassen.

8.1.1 Vissim-Export

Abbildung 8.1 zeigt die Benutzeroberfläche des Exportteils, welche sich in seiner Struktur an den bestehenden Exportfunktionalitäten orientiert. Ähnlich zu den Reitern “Export“ und “KML/COLLADA/glTF Export“ stehen ein Bereich zur Auswahl des Exportpfades (1) sowie ein Bereich zur Auswahl des Export Inhaltes zur Verfügung. Letzterer umfasst die Möglichkeiten nach der gml:id (2), dem Namen (3) oder einer räumlichen Auswahl mit Hilfe einer Bounding Box Funktion (4) zu filtern. Die Geometrie betreffend lässt sich einstellen, welche LoD-Repräsentation als Grundlage des Exportes dienen (5) und ob eine Koordinaten Transformation in die sphärische Web Merkator Projektion (EPSG:3857) vorgenommen werden soll. Da Vissim ein kartesisches Koordinatensystem zugrunde liegt, sollte der Haken für Ausgangsdaten in nicht kartesischen Systemen (z.B. Daten in geographische Koordinaten) gesetzt werden. Zur rechten Seite lässt sich auswählen, welche Vissim-Netzwerkkomponenten erzeugt (6) werden sollen. Ausführen lässt sich die Ableitung eines Simulationsgraphen

per Klick auf den Export-Button (7). Die generierte Vissim Netzwerk-Datei findet sich daraufhin im angegebenen Verzeichnis.

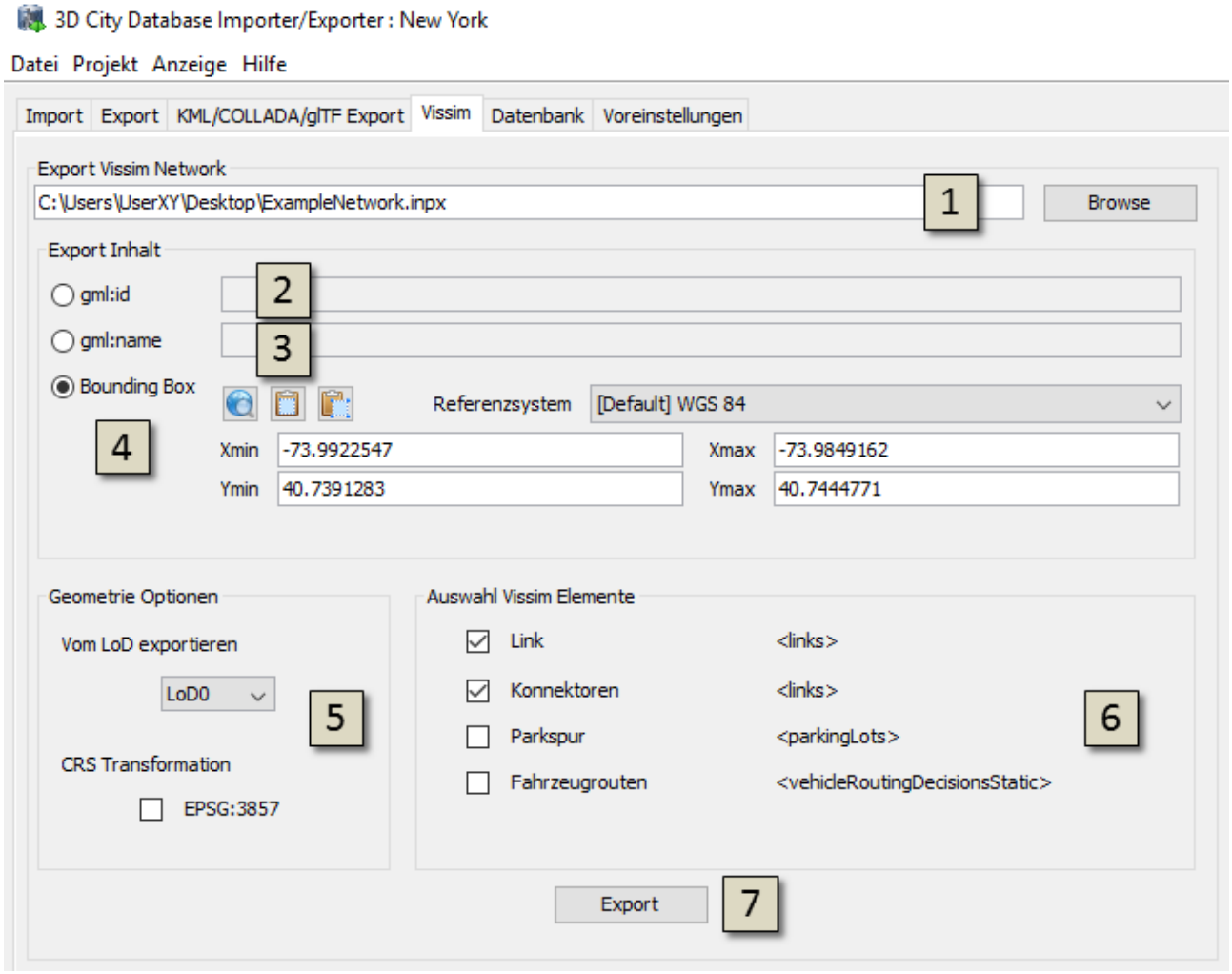


Abbildung 8.1 – Benutzeroberfläche des Vissim Plugins für den 3DCityDB Importer/Exporter: Export Bereich

8.1.2 Ergebnis Visualisierung

Abbildung 8.2 gibt die Benutzeroberfläche des Bereichs zur Visualisierung von Simulationsergebnissen im Stadtmodell wieder. Per Button lässt sich eine Vissim Fahrzeugprotokolldatei laden (1). Ein weiterer Button “Browse“ dient der Festlegung des Ausgabepfades der zu erzeugenden KML-Datei (2). Darunter folgen Optionen, welche die Visualisierung in KML definieren. Da die Protokolldatei keine Information über das Ausgangskoordinatensystem enthält, ist dieses explizit anzugeben (3). Durch Angabe eines Quellkoordinatensystems wird die erforderliche Koordinatentransformation in das WGS84-System möglich, auf welchem KML beruht. Per Checkbox lässt sich zwischen den KML-Styles “Icon“, “Model“ oder beiden Stilen wählen (4). Ist “Model“ selektiert, müssen in der Tabelle Modellinformationen hinterlegt werden (5). Dazu zählen die Modell Nummer, der Pfad und die

Grundorientierung des Modells. Für COLLADA-Modelle, die nach Norden orientiert sind (Motorhaube in Richtung durchgezogener, grüner SketchUp-Achse), beträgt der Azimut 0 Grad. Ist das Modell anders orientiert, kann ein entsprechender Azimut angegeben werden. Erstellt wird eine KML-Datei durch Klick auf den “Create KML“ Button (6).

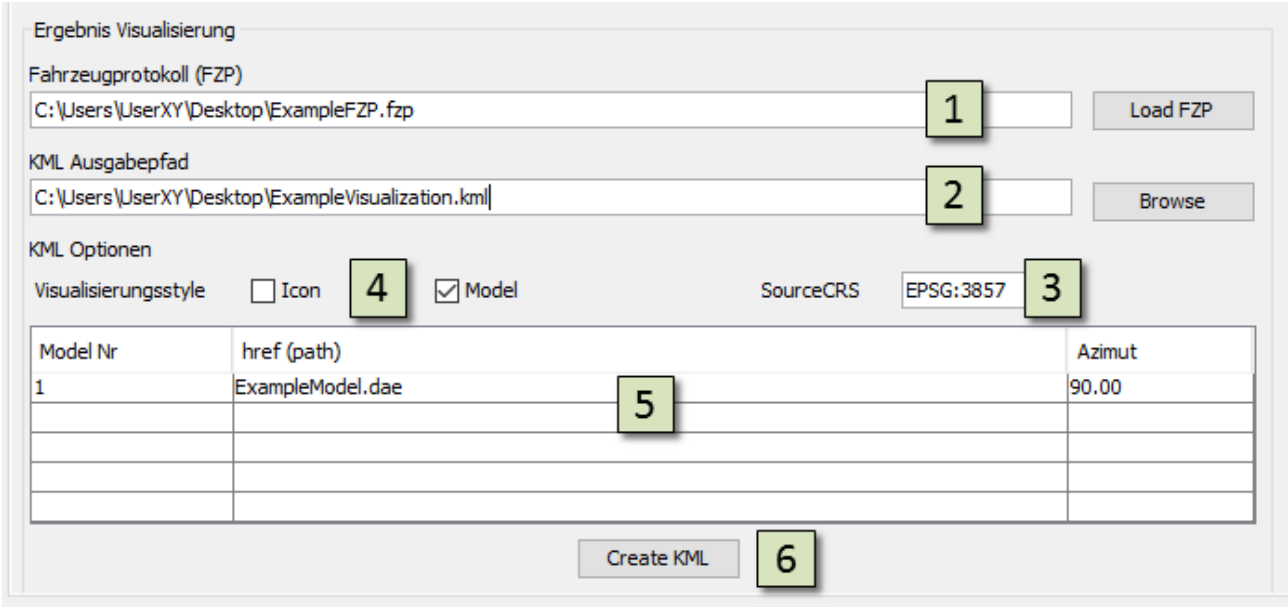


Abbildung 8.2 – Benutzeroberfläche des Vissim Plugins für den 3DCityDB Importer/Exporter: Visualisierungsbereich

8.2 Systemarchitektur und Ablauf

Die Struktur des Plugins ist durch vier Pakete beschrieben: Im Paket “gui“ sind die Klassen im Kontext der Benutzeroberfläche zu finden. Das “provider“-Paket enthält die Initialisierung des Plugins und damit die Schnittstelle zum 3DCityDB Importer/Exporter. Es bleiben ein Paket für den Export sowie eines für die Visualisierungsfunktionalitäten. Beide sind in mehreren Ebenen durch verschiedene Unterpakete strukturiert, welche sämtliche Klassen enthalten, die zur Bereitstellung der jeweiligen Funktionalität notwendig sind.

Abbildung 8.3 zeigt den Aufbau und Workflow des Export-Parts. “exp.db“ beinhaltet alle Klassen, die die Schnittstelle zur Datenbank bilden. Dazu zählt ein Datenbank-Kontroller (Klasse “DatabaseConnection“) zur Steuerung aller Datenbank-Interaktionen, welcher nach einem Konzept von Willenborg [2015] umgesetzt ist. Die Klasse “DatabaseUtil“ enthält Methoden, um ein neues Datenbankschema zu erstellen und die Daten nach der in Kapitel 6 beschriebenen Methodik vorzuverarbeiten. Die dazu notwendigen SQL-Aufrufe liegen im Subpaket “db.sql“. Die einzelnen Klassen u.a. zur Erstellung eines neuen Datenbankschemas (“SchemaQueries“), zur Datenaufbereitung (“PreprocessingQueries“) und zur Abfrage der zur Erstellung eines Straßennetzwerkes notwendigen Informationen (“NetworkQueries“) folgen einem Interface namens “Queries“, welche eine Methode “getSQL“ mit Rückgabewert String vorgibt. Im Paket “exp.xml“ und seinen Unterpakten ist die Struktur

eines Vissim-Netzwerkes bzw. der Komponenten “Links“, “ParkingLots“ und “VehicleRoutingDecisions“ abgebildet. In einer eigenen Klasse “XmlWriter“ sind verschiedene Methoden implementiert, um die Komponenten eines Vissim-Netzwerkes zu erzeugen. Eine weitere Methode “createXML“ fasst die Komponenten zu einem Netzwerk zusammen, fügt Standardeinstellungen hinzu und erstellt die Netzwerk-Datei.

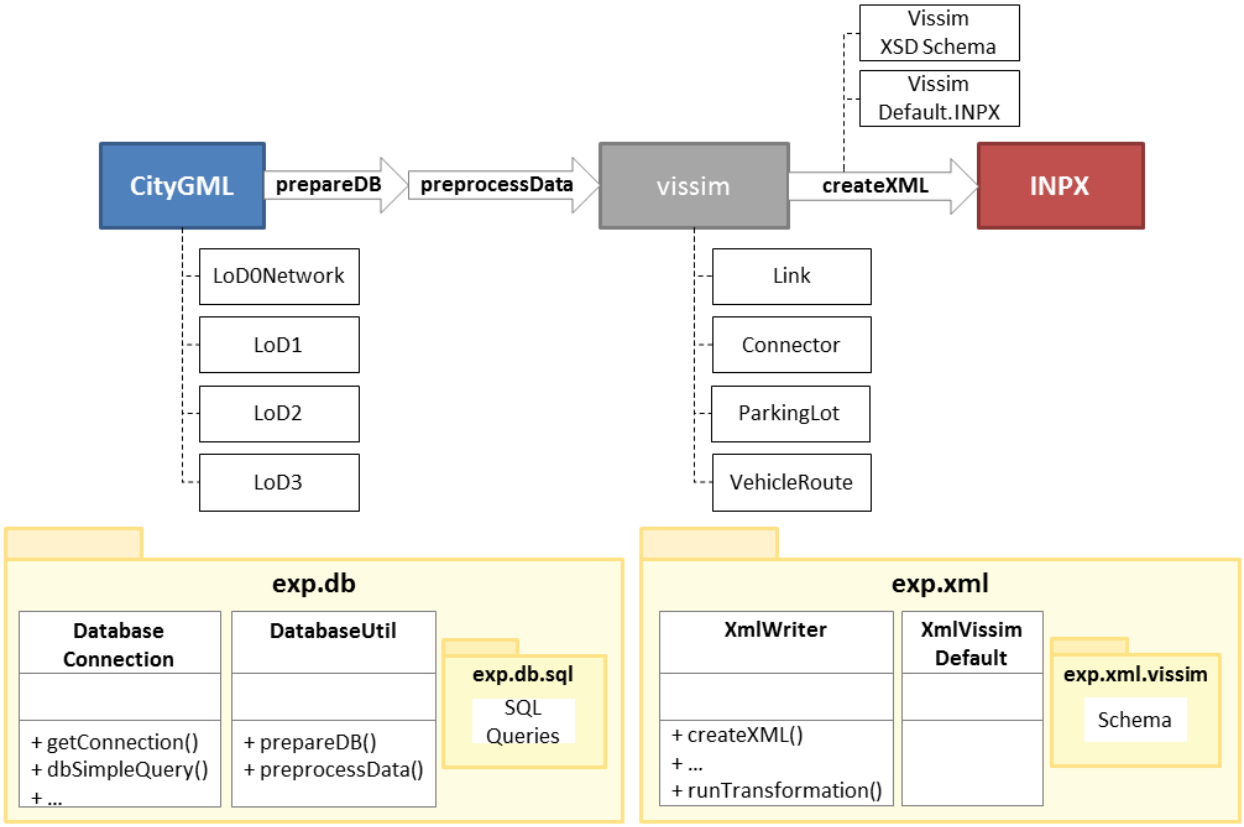


Abbildung 8.3 – Systemarchitektur und Workflow des Vissim-Export Plugins für den 3DCityDB Importer/Exporter

Das Paket “vis“ unterteilt sich in die zwei Subpakete “vis.data“ und “vis.kml“, deren Aufbau und Bezug zum Ablauf Abbildung 8.4 zeigt. In ersterem ist das Einlesen eines Fahrzeugprotokolls, die Zwischenspeicherung der Daten in einer Liste vom Typ “VehState“ sowie die Verarbeitung der Daten geregelt. Die Verarbeitung gemäß Kapitel 7 beinhaltet eine Koordinatentransformation, die Berechnung der aktuellen Position und Orientierung sowie eine Formatanpassung. Die Methoden “builtContent“ und “createKML“ der Klasse “KmlWriter“ aus dem “vis.kml“-Paket dienen der Erstellung der KML-Datei. Das notwendige KML-Schema ist in mehreren Klassen zusammengefasst im Unterpaket “kml22“ definiert.

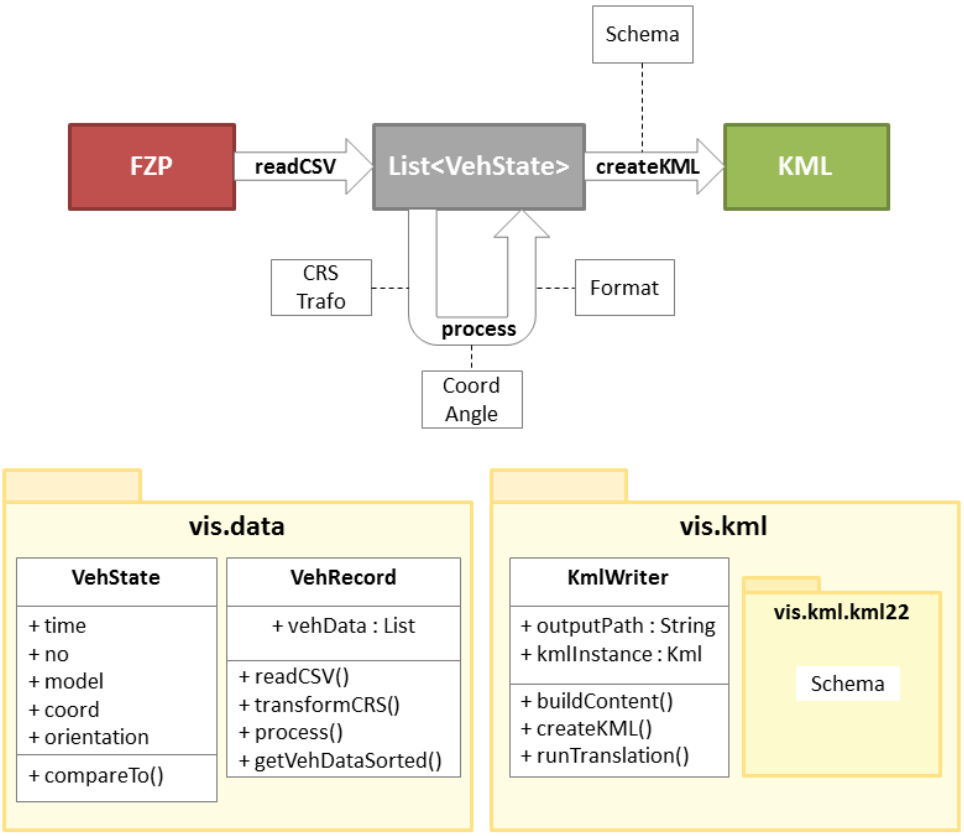


Abbildung 8.4 – Systemarchitektur und Workflow des Plugins zur Visualisierung von Simulationsergebnissen für den 3DCityDB Importer/Exporter

9 Durchführung und Ergebnisse

Die in der Theorie vorgestellten Konzepte sollen im Folgenden anhand von Beispielen getestet und deren Ergebnisse präsentiert und bewertet werden. Die Vorstellung der Ergebnisse folgt dem zu Beginn der Arbeit präsentierten Ablauf: aus einem Stadtmodell im CityGML Format wird automatisch ein Simulationsgraph für die Verkehrssimulation Vissim abgeleitet. Nach Durchführung der Simulation in Vissim werden die Ergebnisse im Stadtmodell visualisiert.
Nach einer Beschreibung der verwendeten Daten, werden zunächst die Resultate der automatischen Ableitung eines Simulationsgraphen diskutiert. Es folgen die Ergebnisse zur Visualisierung von Simulationsergebnissen im Stadtmodell.

9.1 Verwendete Datensätze

Die Bewertung der Methodik zur Ableitung eines Simulationsgraphen für Vissim aus CityGML sowie die auf eine Simulation folgende Visualisierung der Ergebnisse soll anhand zweier Beispiele durchgeführt werden, nämlich Stadtmodellen von New York City sowie der Stadt München.

9.1.1 Stadtmodell New York City

Das Stadtmodell von New York City ist im Rahmen mehrerer studentischer Arbeiten am Lehrstuhl für Geoinformatik der Technischen Universität München entstanden. Die Grundlage stellen offene Daten des Open Data Portals der Stadt New York dar. Das resultierende Stadtmodell im CityGML Format besteht u.a. aus mehr als einer Million Gebäuden und einer halben Million Straßenobjekten. Weiter sind Objekte wie Parkplätze, öffentliche Plätze, Wege und Parzellen verfügbar. Das Stadtmodell kann über einen Web-Map-Client aufgerufen und betrachtet werden. Sämtliche Datensätze stehen zudem als Download frei zur Verfügung [Lehrstuhl für Geoinformatik der TU München, 2017].
Zur Veranschaulichung und Evaluierung der Ableitung eines Simulationsgraphen werden ein Lod0-Network Datensatz und ein Datensatz mit LoD2-Straßenflächen herangezogen, welche jeweils im Rahmen der Masterarbeit von Beil [2017] entstanden sind. Zusätzlich findet ein älterer Datensatz in LoD 1 aus dem Jahr 2015 Eingang. Die Datensätze zeichnen sich durch eine hohe Informationsfülle aus. Neben der Linien- bzw. Flächengeometrie enthalten die Straßenobjekte über 30 Attribute, deren Bedeutung sich aus der Metadatentabelle entnehmen lässt. Für die vorliegende Arbeit von besonderem Interesse sind die ID, die Funktion, der Straßenname, die Anzahl an Fahrspuren (Verkehrsspu-

ren, Parkspuren und gesamt), Segmentlängen, Straßenbreiten, Geschwindigkeitsbegrenzungen und die Fahrtrichtung. Abgesehen von der Fahrspurbreite sind damit alle Informationen für die Ableitung nach dem in Kapitel 6 vorgestellten Konzept verfügbar. Für die Visualisierung der Simulationsergebnisse finden neben den LoD2-Straßenobjekten zusätzlich Gebäude in LoD 2 Eingang.
Mit der in das Plugin integrierten Bounding Box Auswahl wird ein Testgebiet mit einer Größe von ungefähr 1000 auf 1000 Meter in Manhattan ausgewählt und anschließend die Ableitung für beide LoD-Fälle ausgeführt. Abbildung 9.1 zeigt den Testausschnitt mit dem Broadway zur linken und dem Madison Square Park im Zentrum. Zur rechten Seite befindet sich die Park Avenue.

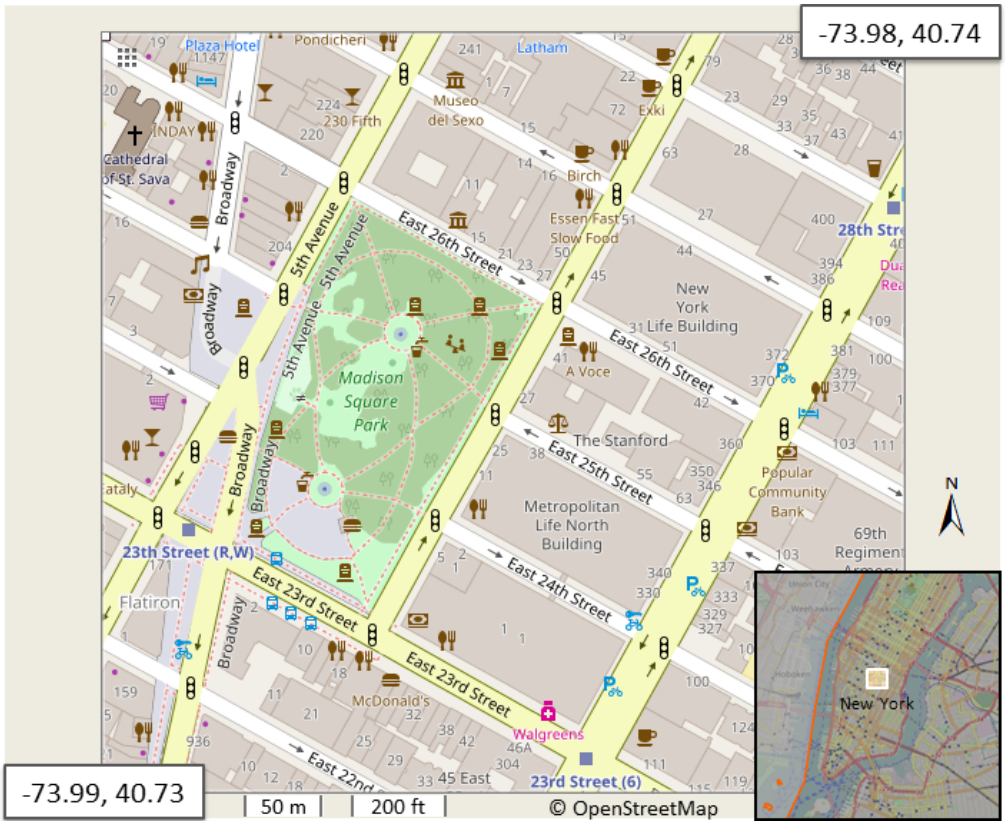


Abbildung 9.1 – Testgebiet New York City (Bounding Box = -73.9907312, 40.7391609, -73.9836931, 40.7442983)

9.1.2 Stadtmodell München in Verbindung mit OSM-Straßenachsen

Für das Beispiel München wurden die Informationen über die Straßenachsen von Open Street Map bezogen. Mit Hilfe der Feature Manipulation Engine (FME) von Safe Software[1] wird aus den exportierten OSM-Daten[2] eine valide CityGML Datei erstellt. Teil des Workflows ist eine Überführung semantischer Attribute in CityGML Attribute und generische Attribute.

[1] vgl. https://www.safe.com/
[2] Bezugsquelle: http://overpass.osm.rambler.ru/cgi/xapi_meta?*[bbox=11.5548167,48.1159309,11.6090916,48.1525364] (Zugriff 18.05.2017)

Dazu zählen Informationen zu der Anzahl an Fahrspuren (nur Verkehrsspuren ohne Parkspuren), die Fahrtrichtung (Einbahnstraßen) sowie falls vorhanden die Straßenbreite und das Tempolimit. Die automatische Formatumwandlung erschwert, dass die Attribute in OpenStreetMap als Liste vorliegen, welche keiner Sortierung folgt. Das bedeutet, Attribute treten für verschiedene Straßensegmente an unterschiedlicher Stelle der Liste auf, was für jedes Straßensegment und Attribut eine aufwendige Durchsuchung der Liste nach dem Attributnamen erfordert. Eine Koordinatentransformation überführt die OSM-Daten von geographischen Koordinaten ins Gauß-Krüger System (EPSG:31468). Aus den Linienobjekten wird mit Hilfe des FME-Transformers “TopologyBuilder“ ein Graph gebildet bestehend aus Knoten und Kanten. Dieser wird als CityGML LoD0-Network ausgegeben. Anhang D.1 gibt einen schematischen Überblick über den Ablauf der Erstellung eines LoD0-Networks aus OpenStreetMap-Daten. Für die Visualisierung werden LoD2-Gebäude eines CityGML Datensatzes der Landeshauptstadt München hinzugezogen. Straßenflächen sind in diesem Fall nicht verfügbar.
Auch für München wird ein ca. 1000 auf 1000 Meter großes Testgebiet ausgewählt. Der Ausschnitt (Abbildung 9.2) befindet sich in der Münchner Maxvorstadt mit Technischer Universität und den Pinakotheken im oberen Teil sowie Karolinenplatz und Königsplatz am südlichen bzw. westlichen Rand.

Abbildung 9.2 – Testgebiet München Maxvorstadt (Bounding Box = 11.5634537, 48.1442411, 11.5721655, 48.1505262)

9.2 Präsentation und Bewertung der Ergebnisse

Im Kontext der Kopplung von Verkehrssimulation und Stadtmodell auf Modellebene wurden Potenziale und Herausforderungen bei der Ableitung eines Simulationsgraphen für Vissim aus CityGML diskutiert. Im Zuge der Bewertung des vorgestellten Konzeptes soll auf die in Kapitel 5 angeführten Aspekte Bezug genommen werden. Auch die Potenziale eines Stadtmodells bei der Visualisierung von Simulationsergebnissen wurden mehrfach hervorgehoben und sollen an dieser Stelle anhand der beiden Testgebiete überprüft werden.

9.2.1 Simulationsgraph

Das entwickelte Plugin für den 3DCityDB Importer/Exporter ermöglicht eine automatische Ableitung eines Vissim Straßennetzwerkes aus einem CityGML Stadtmodell. Abbildung 9.3 zeigt ein in Vissim importiertes Ergebnis für das Testgebiet in New York abgeleitet aus LoD0-Daten.

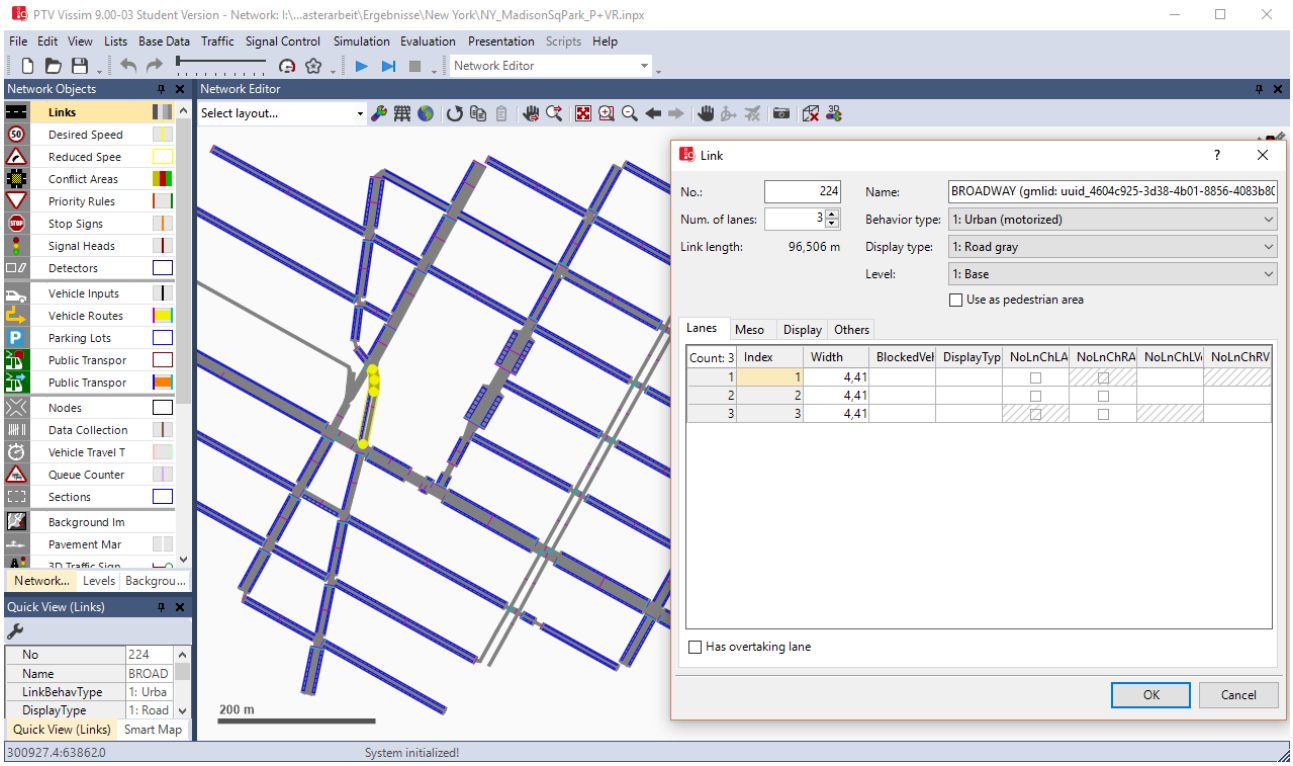


Abbildung 9.3 – Für das Testgebiet in New York automatisch abgeleiteter Simulationsgraph importiert in Vissim

Das Netzwerk zeichnet sich durch exakt überführte geometrische, topologische und semantische Informationen aus. Wie das Bearbeitungsfenster zur rechten Seite der Abbildung zeigt, enthält der selektierte Link Informationen zum Namen mit gml:id, über Funktion und Darstellungsform sowie die Anzahl und Breite an Fahrspuren. Weitere Informationen sind entsprechend Tabelle 6.3 überführt. Neben Link-Objekten sind Konnektoren, Parkspuren und Fahrzeugrouten Bestandteil dieses

Simulationsgraphen. Die Abbildung gibt den Exportzustand ohne jegliche, manuelle Nachbearbeitung wieder. Eine Bearbeitung des importierten Netzwerkes unter Verwendung sämtlicher Vissim Funktionalitäten ist selbstverständlich möglich.
Mit dem Vissim Plugin für den Importer/Exporter ist es möglich Simulationsgraphen aus verschiedenen Level of Detail abzuleiten. Tabelle 9.1 stellt verschiedene Ableitungsergebnisse für das Testgebiet in New York und die CityGML Ausgangsdaten in LoD 0, LoD1 und LoD 2 gegenüber. Als größte Herausforderungen bei der Überführung von CityGML nach Vissim wurden die unterschiedliche Geometrierepräsentation, die Fahrtrichtung und fehlende Informationen zu Abbiegerelationen genannt. Betrachtet man unter diesen Gesichtspunkten die Ergebnisse für das Testgebiet in New York aus Tabelle 9.1 kann zumindest für den LoD0-Fall von einer erfolgreichen Überwindung der Herausforderungen gesprochen werden. Der resultierende Simulationsgraph (Tabelle 9.1 - rechts oben) zeichnet sich durch eine Achse pro Fahrtrichtung und eine parametrische Beschreibung der Straßengeometrie ausgehend von Fahrspuren und deren Breite aus. Aus der Netzwerk-Topologie abgeleitete Abbiegerelationen (Konnektoren) vervollständigen ein für eine Simulation praktikables Vissim Straßennetzwerk.
Voraussetzung für ein optimales Ergebnis ist das Vorhandensein von Zusatzinformationen in Form generischer Attribute über Anzahl und Breite an Fahrspuren sowie die Fahrtrichtung. Im Falle der flächenhaften Datensätze (LoD 1 und 2) limitieren fehlende Informationen zur Fahrtrichtung das Ergebnis. Aus diesem Grund wird auf eine Herleitung von Konnektoren verzichtet. Abgesehen davon liefern die aus den Mittelachsen der Straßenflächen hergeleiteten Geometrien der Straßensegmente Ergebnisse mit vergleichbarer Qualität wie im LoD0-Fall. Für die Durchführung einer Verkehrssimulation auf Basis von CityGML-Daten in LoD 1-4 sind eine manuelle Überprüfung der Fahrtrichtung und die manuelle Abbildung der Abbiegerelationen in Form von Konnektoren erforderlich.
In der derzeitigen Umsetzung liefern linienhafte Ausgangsdaten in LoD 0 bessere Ergebnisse als flächenhafte Daten höherer Level of Detail. Ein Mangel der LoD0-Daten ist, dass Sperrzonen in Form von Straßenmarkierungen oder Mittelinseln meist keine Berücksichtigung finden, da den Daten Bestandsachsen zugrunde liegen. Dies kann zu unschönen Auswirkungen auf die Visualisierung im Stadtmodell führen. Ein Ausweg würde die Ableitung von Fahrbahnachsen aus flächenhaften Ausgangsdaten unter Berücksichtigung etwaiger Sperrzonen darstellen. Ein derartiges Konzept zur Ableitung von Fahrbahnachsen aus LoD 1-4 würde einen echten Mehrwert im Vergleich zu LoD 0 darstellen. Die Komplexität eines derartigen Ansatzes basierend auf den derzeitig verfügbaren Daten ist jedoch nicht zu unterschätzen. Voraussetzung ist in jedem Fall, dass Informationen zur Fahrtrichtung vorliegen bzw. eingebunden werden können.

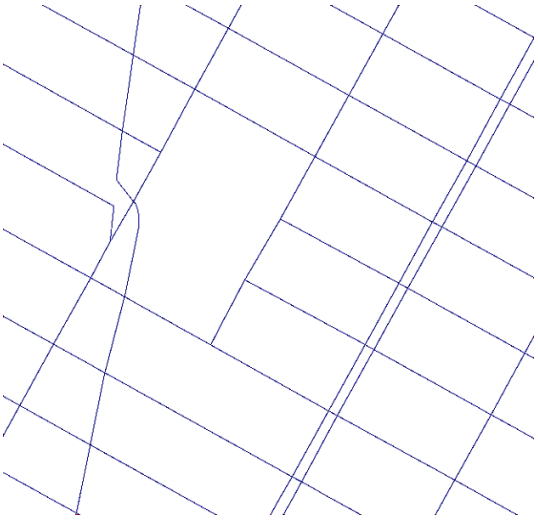
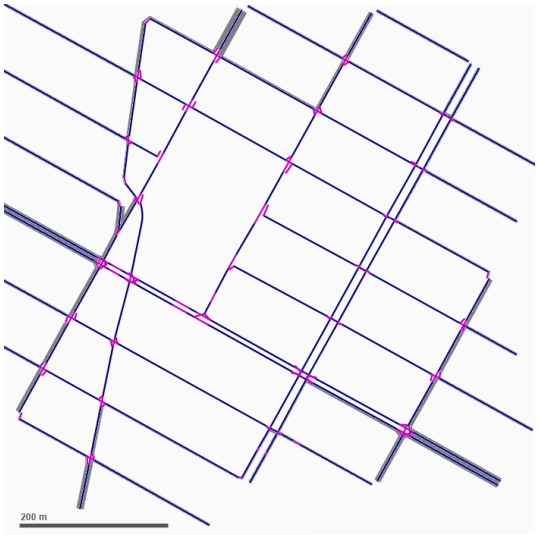
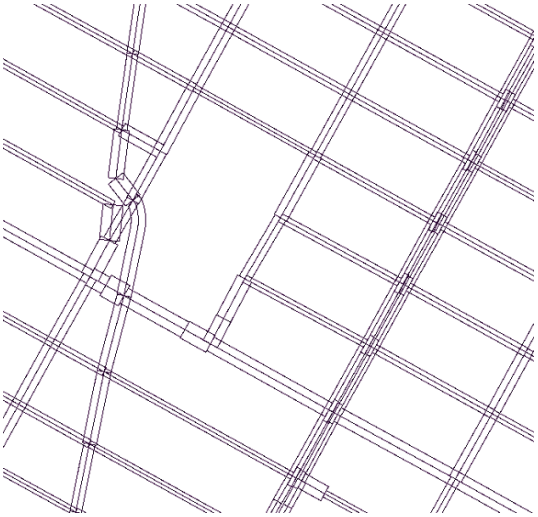
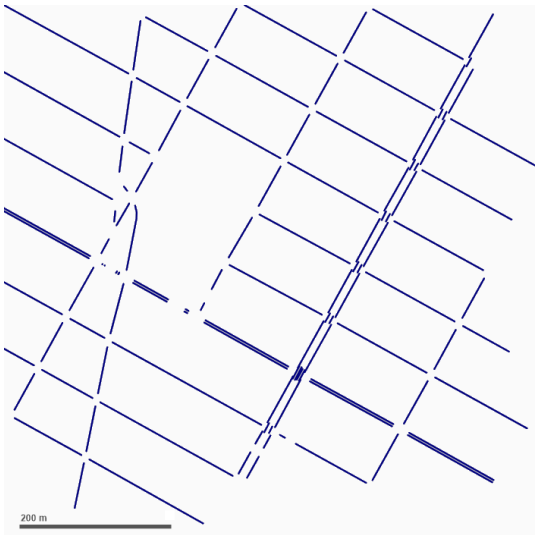
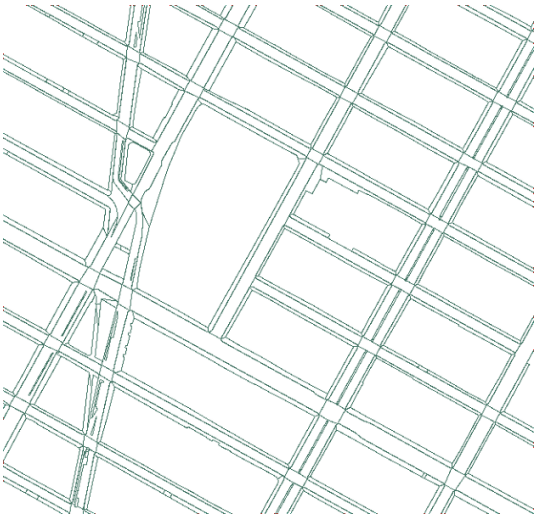
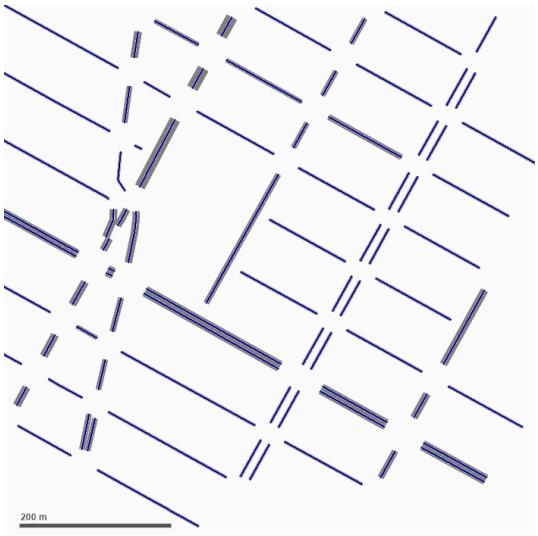
	Ausgangsdaten CityGML	**Ergebnis** Vissim Network
LoD 0		
LoD 1		
LoD 2		

Tabelle 9.1 – Vergleich der Ausgangsdaten mit dem daraus abgeleiteten Vissim Network

Die Analyse der Anforderungen an ein Straßenraummodell für Verkehrssimulationen in Kapitel 4 hat signifikante Unterschiede zwischen Verkehrsraummodellen aus dem Simulationsbereich und anderen Anwendungen wie beispielsweise Stadtmodellen ergeben. Das vorgestellte Konzept ist darauf ausgelegt, verfügbare Informationen bestmöglich zu verarbeiten und fehlende Informationen herzuleiten. Aufgrund der Diskrepanz zwischen den Modellen verwundert es jedoch nicht, dass Fehler bei der automatischen Ableitung auftreten können.
Ein Fehler sind unkorrekte Fahrspurverknüpfungen. Abbiegerelationen sind nicht Teil des CityGML Transportation Moduls. In Kapitel 6 wurde ein Konzept vorgestellt, wie aus der impliziten Topologie eines CityGML LoD0-Netzwerkes Verknüpfungsinformationen auf Fahrspurebene hergeleitet werden können. Bereits an dieser Stelle wurde darauf aufmerksam gemacht, dass das Konzept keine Lösung für sämtliche Sonderfälle umfasst. Tabelle 9.2 zeigt eine Kreuzung im New Yorker Testgebiet, an der eine fehlerhafte Fahrspurverknüpfung vorliegt. Neben dem Luftbild (Bildquelle: Google Earth), das die reale Situation mit Rechtsabbiegerspur zeigt, ist das automatisch abgeleitete Ergebnis und eine mögliche, manuelle Nachbearbeitung in Vissim angegeben. Der Fehler tritt auf, da das Konzept auf 4-Wege-Kreuzungen spezialisiert ist. Weil im vorliegenden Fall die von rechts kommende 'W 25 St' eine Einbahnstraße ist, fällt die Möglichkeit "Linksabbieger" weg, für die der Theorie einer 4-Wege-Kreuzung folgend eine Fahrspur freigehalten wird.
Im Zuge einer manuellen Nachbearbeitung wurde die Fahrspurverknüpfung des Konnektors korrigiert. Auf visueller Ebene wurden zwei Fahrbahnmarkierungen (ohne Effekt auf die Simulation) eingefügt, eine für die Rechtsabbiegerspur und eine für die Nichtabbiegerspur. Außerdem wurden für zwei weitere Konnektoren Spline-Geometrien erzeugt, welche die geraden Konnektorgeometrien ersetzen. Von den insgesamt 177 Konnektoren im Testgebiet sind entsprechend einer manuellen Überprüfung mittels Vergleich mit GoogleEarth-Luftbildern lediglich zwei Fahrspurverknüpfungen offensichtlich falsch und erfordern eine manuelle Nachbearbeitung.

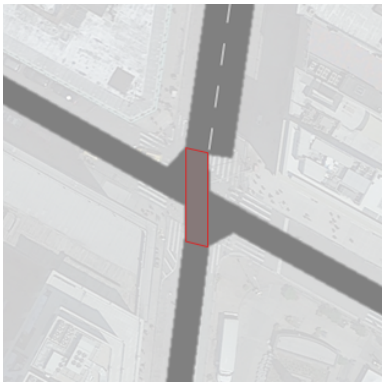
Luftbild	Ergebnis Ableitung	Ergebnis Nachbearbeitung
		

Tabelle 9.2 – Fehlerhafte Fahrspurverknüpfung am Beispiel der Kreuzung Broadway / W 25th St

Weitere Fehler in der Spurverknüpfung können durch fehlerhafte Spuranzahlen ausgelöst werden. Im Testgebiet tritt dieser Fall bei den Straßen auf, die die Park Avenue kreuzen. Die beiden Fahrtrichtungen der Park Avenue sind im Bereich des Gebietes durch eine Mittelinsel voneinander getrennt. Für das LoD0-Network sind den Achsstücken im Kreuzungsbereich zwei Fahrspuren zugewiesen.

Luftbild	Ergebnis Ableitung	Ergebnis Nachbearbeitung
		

Tabelle 9.3 – Fehlerhafte Spuranzahl und daraus resultierende Fehler am Beispiel der Kreuzung Park Avenue / E 25th Street

Tabelle 9.3 zeigt den Fall anhand der 25th Street. Das automatisch abgeleitete Ergebnis hat zur Folge, dass Autos beim Überqueren der Park Avenue einen Schlenker fahren, was durch die Korrektur auf eine Fahrspur verhindert wird.
Unzureichende Ausgangsdaten sind generell eine mögliche Fehlerquelle. Beispielsweise treten im Testgebiet Fälle auf, in denen in der Realität explizite Abbiegespuren vorliegen oder ein Stück der Parkspur als Abbiegespur verwendet wird. Diesen Detailgrad geben die Ausgangsdaten nicht her, weswegen auch an dieser Stelle eine manuelle Nachbearbeitung erforderlich sein kann.

Die Simulationsgraphen für das Stadtmodell von München müssen einer gesonderten Betrachtung unterworfen werden, da als Quelle der Straßenachsen Open Street Map Verwendung findet. Wie in Kapitel 3.2.1 erläutert, werden OSM-Daten als Datenquelle für Verkehrssimulationen allgemein durchaus kritisch gesehen. Der nahezu weltweiten Verfügbarkeit werden in der Literatur Fehlerhaftigkeit und ein unzulänglicher Detailgrad entgegengehalten.
Die aufbereiteten und nach CityGML überführten OSM-Daten liefern vergleichbare Informationen wie das LoD0-Network aus dem New Yorker Stadtmodell. Nicht verfügbar sind Informationen zu Parkspuren. Eine Straßen- bzw. Spurbreite existiert lediglich für 12 Straßen im Stadtgebiet. Bezüglich der Datenfülle besteht ein vernachlässigbarer Unterschied zwischen den Datensätzen. Hingegen unterscheiden sich die beiden Datensätze in ihrer Richtigkeit. Mehrere Straßensegmente müssen manuell korrigiert werden, weil die Information zur Fahrtrichtung nicht stimmt. Eine genauere Betrachtung des abgeleiteten Simulationsgraphen für das Testgebiet in der Münchner Maxvorstadt zeigt zudem auf, dass die Lagegenauigkeit zum Teil unzureichend ist. So führen die abgeleiteten Fahrbahnen am Königsplatz über den gemeinsamen Fuß- und Radweg, was einem Lagefehler im Meterbereich entspricht.

Im Vergleich der beiden Testgebiete von New York und München bleibt festzuhalten, dass das Münchner Beispiel basierend auf OSM-Daten eine aufwendigere Nachbearbeitung erfordert. Aufgrund der begrenzten Fläche beider Testgebiete lässt sich diese Feststellung allerdings nicht verallgemeinern. Eine wesentliche, manuelle Änderung erfordert beispielsweise der Kreisverkehr am Karolinenplatz. Ein derart komplexes Konstrukt kommt im Bereich des New Yorker Testgebietes nicht vor. Abbildung 9.4 zeigt ein Satellitenbild des Karolinenplatzes aus Google Earth, das Originalergebnis der Ableitung und das manuell editierte Ergebnis.

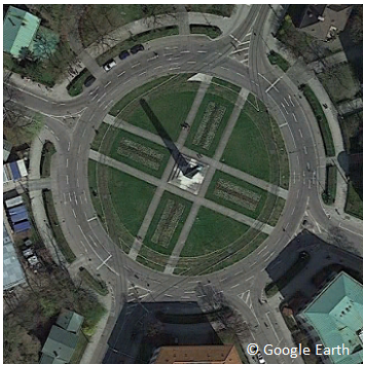
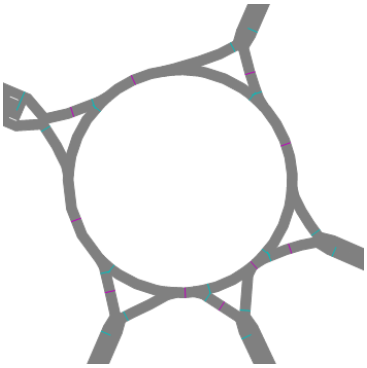
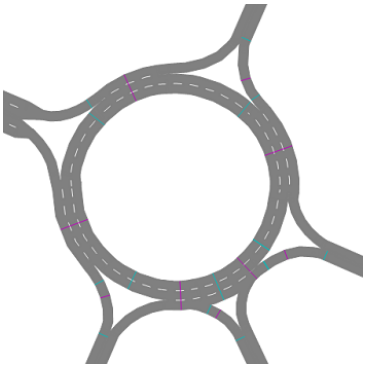
Luftbild	Ergebnis Ableitung	Ergebnis Nachbearbeitung
		

Tabelle 9.4 – Manuelle Nachbearbeitung eines Kreisverkehrs am Beispiel des Karolinenplatz in München

Der große Mehrwert einer automatischen Ableitung eines Simulationsgraphen aus einem existierenden Stadtmodelldatensatz liegt in der Reduzierung des Modellierungsaufwandes. Müssen bisher Simulationsgraphen mühsam manuell erstellt werden, liefert die vorliegende Arbeit ein Werkzeug, das per Knopfdruck ein Vissim Netzwerk aus CityGML Straßenobjekten erzeugt. Tabelle 9.5 gibt einen Überblick über die Ergebnisse der Ableitung verschiedener Simulationsgraphen. Neben der Anzahl an Eingangsobjekten wird die Zahl erstellter Objekte und die zur Ableitung benötigte Zeit aufgeführt. Zu beachten ist, dass die Exportdauer stark von der Größe der Datenbank abhängig ist. Die angegebene Exportdauer bezieht sich auf Tests mit einem Mittelklasse-Rechner (4GB RAM, Core2 Quad CPU 2.83 GHz).

Datensatz	LoD	Anzahl CityGML-Objekte	Anzahl ersteller Objekte				Exportdauer
			Links	Con	Parking	VehRoute	
NY Testgebiet	LoD 0	102	112	177	119	73	~1 sec
	LoD 0	102	112	-	-	-	< 1 sec
	LoD 1	104	114	-	-	-	~4 sec
	LoD 2	76	81	-	-	-	< 1 sec
Manhattan	LoD 0	433	476	826	636	352	~1 sec
	LoD 2	494	549	-	-	-	~1 sec
Muc Testgebiet	LoD 0	78	107	153	-	50	~1 sec
München	LoD 0	2768	4179	6947	-	5160	~23 sec

Tabelle 9.5 – Informationen zum Export verschiedener Simulationsgraphen aus CityGML-Daten der Stadtmodelle von New York City und München (mit OSM-Straßenachsen)

Selbst für den wahrscheinlichen Fall, dass eine manuelle Nachbearbeitung der Simulationsgraphen notwendig ist, birgt diese Methode zur automatischen Ableitung einen enormen Zeitgewinn gegenüber einer manuellen Erstellung von Straßennetzen. Dies gilt insbesondere auch für Simulationsgraphen, die aus Straßenflächen abgeleitet werden und damit ohne Konnektoren vorliegen. Im Gegensatz zur rein manuellen Modellierung müssen in diesem Fall lediglich die Abbiegerelationen abgebildet werden, wohingegen die Nachbildung der Straßengeometrien dem Nutzer abgenommen wird.

9.2.2 Visualisierung der Simulationsergebnisse

Die Visualisierung der Simulationsergebnisse kann entweder offline in Google Earth erfolgen oder per Server/Client-Technologie in Cesium. Die Grundlage für beide Varianten bildet die erstellte KML-Datei, welche die dynamischen Simulationsergebnisse beinhaltet.

Google Earth

In Google Earth kann die KML-Datei mit den Fahrzeugdaten direkt geladen werden. Für eine 3D-Visualisierung der Umgebung können CityGML-Stadtobjekte hinzugeladen werden, nachdem diese mit dem 3DCityDB Importer/Exporter im KML/COLLADA-Format exportiert wurden. Die Animation der dynamischen Verkehrssimulationsdaten lässt sich über ein eigenes Dialogfeld abspielen. Darin können einige Optionen festgelegt werden wie beispielsweise die Animationsgeschwindigkeit. Abbildung 9.4 zeigt ein Visualisierungsergebnis für das Testgebiet in New York. In der Google Earth Seitenleiste sind die geladenen KML-Dateien zu sehen (Fahrzeugdaten, Straßenobjekte, Gebäude). Das linke Fenster der Abbildung zeigt einen Überblick über das Testgebiet. Als Darstellungsstil der Fahrzeuge wurde auf einen Icon zurückgegriffen. Im rechten Teil ist eine Szene am Flatiron Gebäude in Manhattan in der Google Earth Bodenansicht zu sehen. Im Gegensatz zur linken Seite wurde hierfür beim Export der Fahrzeugdaten als Darstellungsstil “Model“ ausgewählt. Bei den 3D-Modellen handelt es sich um SketchUp-Modelle bezogen vom 3D Warehouse, der größten Online-Bibliothek für freie 3D-Modelle[3]. Anhang E.1 beinhaltet eine Liste der im Zuge der Visualisierungen verwendeten SketchUp-Modelle. Bei der Auswahl der Modelle ist auf deren Maße zu achten. Gegebenenfalls ist eine Skalierung auf die der Realität entsprechende Größe notwendig. Die Fahrzeugmodelle wurden zur Verwendung in Google Earth in KMZ respektive COLLADA umgewandelt, eine Funktion, die die Modellierungssoftware SketchUp umfasst. COLLADA ist ein offenes Datenformat zum Austausch von 3D-Modellen[4], welches auch von Google Earth unterstützt wird. Durch die Möglichkeit im Zuge des Erstellungsvorgangs einen Pfad zum COLLADA-Modell angeben zu können (vgl. Kapitel 8), lassen sich beliebige SketchUp-Modelle zur Visualisierung heranziehen. Unabhängig welche Option bei der Erzeugung der KML-Datei gewählt wurde, kann die Darstellung (Icon, Icon-Farbe, ...) jederzeit manuell in Google Earth geändert werden.

[3] https://3dwarehouse.sketchup.com/
[4] www.khronos.org/collada/

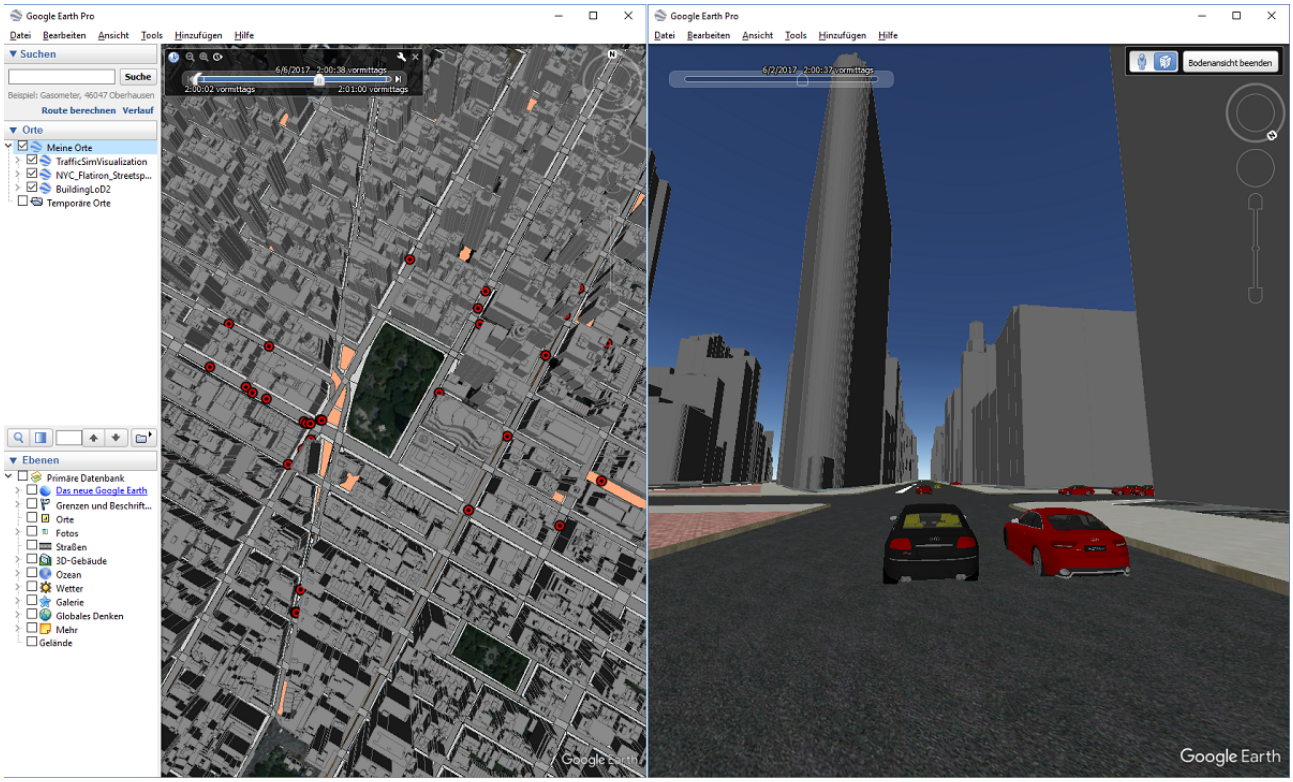


Abbildung 9.4 – Visualisierung der Simulationsergebnisse in Google Earth: Überblick über das New Yorker Testgebiet mit LoD2-Gebäuden und Icon Style (links) sowie in Bodenansicht am Flatiron Gebäude mit LoD2-Gebäuden und 3D-Automodellen aus SketchUp

Auch für das Münchner Testgebiet lassen sich Simulationsergebnisse visualisieren. Abbildung 9.5 (links) zeigt einen Überblick über das Testgebiet mit Icons als Darstellungsform. Auf der rechten Seite der Abbildung sind zwei Nahansichten mit 3D-Automodellen zu sehen. Zur Visualisierung der Stadtumgebung wurden einmal die LoD2-Gebäude der Vermessungsverwaltung (oben) und einmal die fotorealistischen 3D-Gebäude aus Google Earth (unten) herangezogen. Straßenflächen sind im Gegensatz zu New York nicht verfügbar. Da die Straßen im Satellitenbild nicht frei von Fahrzeugen sind, kommt es bei der Visualisierung mitunter zu unschönen Effekten, wenn 3D-Automodelle Fahrzeuge aus dem Satellitenbild “überfahren“. Im unteren Bild ist außerdem die angesprochene, unzureichende Lagegenauigkeit der OSM-Daten zu erkennen. Das rote Fahrzeug im Bildvordergrund bewegt sich auf dem Gehweg anstatt auf der Fahrbahn.

Abbildung 9.5 – Visualisierung der Simulationsergebnisse in Google Earth (2): Überblick über das Münchner Testgebiet mit LoD2-Gebäuden und Icon Style (links), am Karolinenplatz mit LoD2-Gebäuden und 3D-Automodellen aus SketchUp (rechts oben) sowie am Königsplatz mit Google Earth 3D-Gebäuden und 3D-Automodellen aus SketchUp (rechts unten)

Cesium/ 3D-Web-Map-Client

Der 3D-Web-Map-Client stellt Nutzern eine Toolbox zur Verfügung, welche Funktionen rund um das Laden und die interaktive Betrachtung von Stadtmodellen, thematischen Informationen, Web Map Service (WMS) Daten oder Digitalen Geländemodellen beinhaltet. Insbesondere lassen sich damit verschiedene Layer mit Stadtmodell-Objekten im KML/glTF Format hinzufügen [Kolbe et al., 2016].
Für die Visualisierung der Ergebnisse einer Verkehrssimulation existiert derzeit keine GUI-Erweiterung. Das Laden der dynamischen Fahrzeugdaten erfolgt durch Änderung der Pfade im JavaScript-Code (vgl. Kapitel 7). Gleiches gilt für 3D-Fahrzeugmodelle, die für eine Visualisierung verwendet werden sollen.
Abbildung 9.6 zeigt die Visualisierung der Simulationsergebnisse im Testgebiet von New York. Vergleichbar zu Google Earth besteht ein Widget zur Steuerung der Animation (unterer Bildrand). Ein Eingriff in die zeitliche Wiedergabe der Simulationsergebnisse ist entweder über die Buttons zur linken Seite oder über den Regler auf der Zeitleiste möglich. Während des Ablaufs der Simulationsergebnisse kann auf alle bestehenden Funktionen des 3D-Web-Map-Clients zurückgegriffen werden. In Abbildung 9.6 werden Informationen der gelb hervorgehobenen Straßenfläche abgerufen, welche dem Stadtmodell in Form von Spreadsheets hinterlegt werden können (für nähere Informationen vgl. [Kolbe et al., 2016, S.253f]). Für das Beispiel New York können auf diese Weise u.a. Informationen über den Straßennamen, die Anzahl an Fahrspuren oder die Länge, Breite sowie Fläche abgefragt werden.

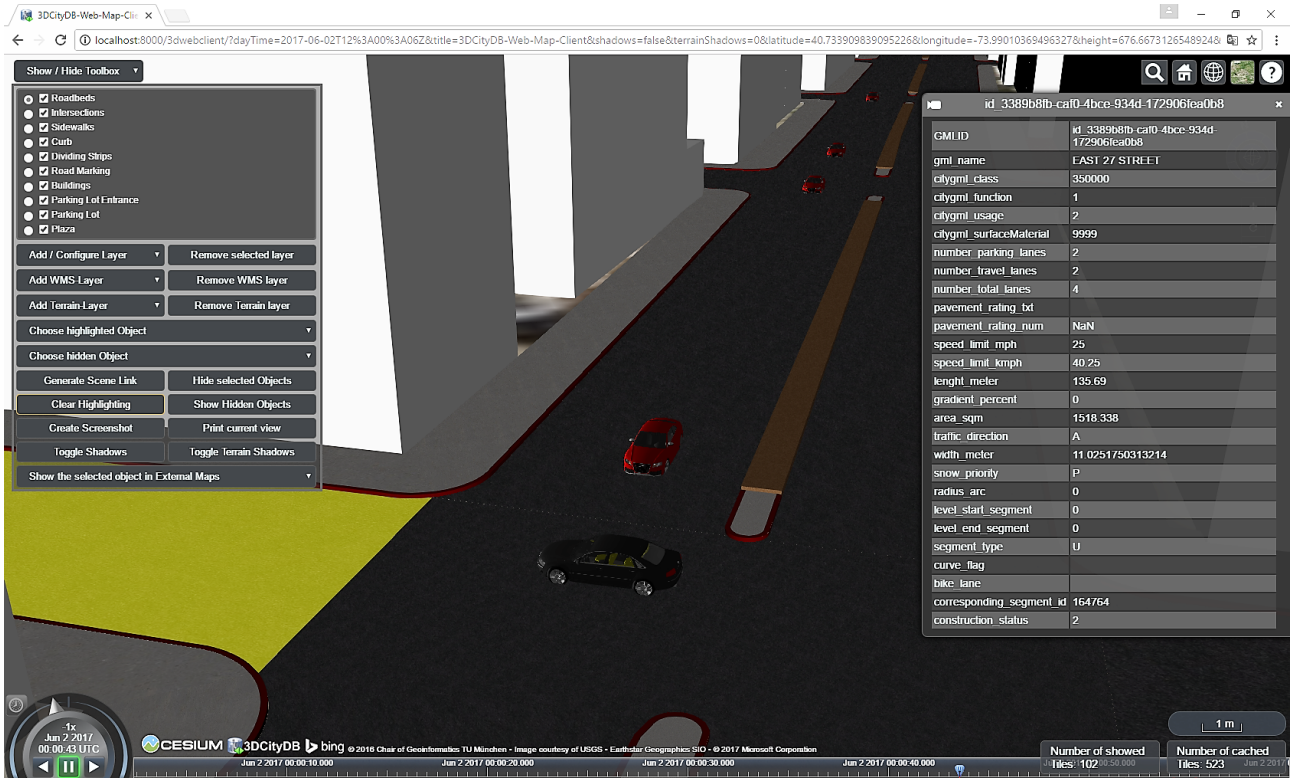


Abbildung 9.6 – Visualisierung der Simulationsergebnisse des Testgebietes von New York City im 3D-Web-Map-Client bei gleichzeitigem Abruf thematischer Informationen des Stadtmodells (Tabelle zur rechten Seite)

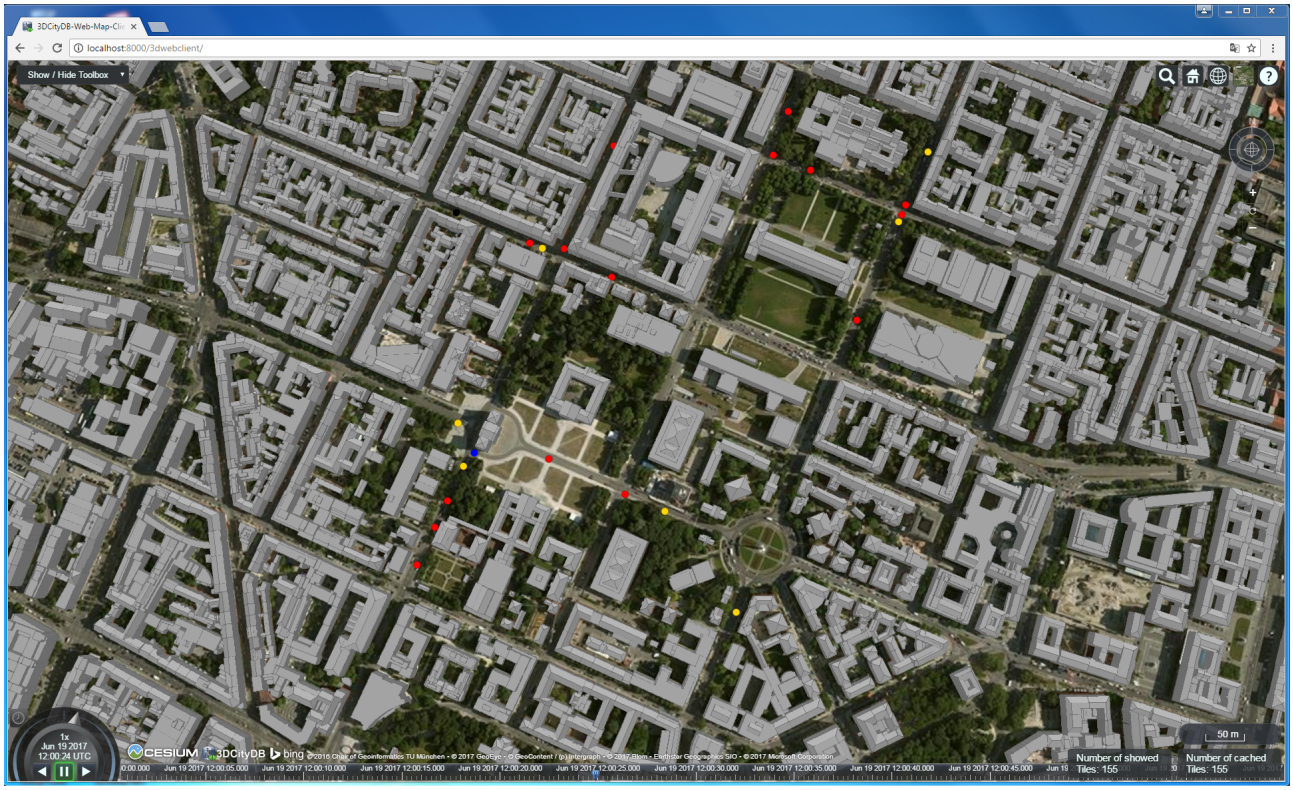


Abbildung 9.7 – Visualisierung der Simulationsergebnisse des Testgebietes von München im 3D-Web-Map-Client

In Abbildung 9.7 ist ein Beispiel für die Visualisierung von Ergebnissen einer Verkehrssimulation für das Testgebiet in München zu sehen. Entsprechend der Kameraentfernung dieser Überblicksszene sind die Fahrzeuge als Punkte in der Farbe des zugehörigen 3D-Modells dargestellt. Durch Heranzoomen wechselt die Visualisierung ab der voreingestellten Distanz von der Punktrepräsentation zum 3D-Modell.

10 Fazit und Ausblick

Informationen zum Verkehrsnetz stellen neben einem Simulationsmodell den Hauptbestandteil von Verkehrssimulationen. Simulationsgraphen müssen bisher meist manuell erstellt werden, was einem erheblichen Zeitaufwand entspricht. Gleiches gilt für Umgebungsmodelle, die für eine realitätsnahe Visualisierung von Simulationsergebnissen erforderlich sind. In beiden Fällen ist eine Kopplung von Verkehrssimulation mit einem 3D-Stadtmodell lohnend, wie die vorliegende Arbeit sowohl auf theoretischer Ebene als auch darauf aufbauend anhand einer praktischen Umsetzung mit zwei Beispielen zeigt.

Den eigentlichen Überlegungen zur Kopplung von Verkehrssimulation und Stadtmodell gehen theoretische Untersuchungen hinsichtlich allgemeiner Verkehrsraummodellierung voran. Die Betrachtung von Modellen verschiedener Bereiche unter dem Aspekt einer Eignung für die Anwendung Verkehrssimulation liefert Unterschiede wie Überschneidungen. CityGML, internationaler Standard für 3D-Stadtmodelle, zeigt sich im Vergleich zu anderen Verkehrsraummodellen aus den Bereichen Verkehr, Infrastruktur und Stadtmodellierung mehr als geeignet, um als Datengrundlage für Verkehrssimulationen zu dienen. Zum gleichen Ergebnis kommt die Verknüpfung von Verkehrssimulation und Stadtmodell auf Modellebene. Nach dem Konzept des Model Weaving werden dazu Korrespondenzen zwischen einzelnen Modellelementen hergestellt und Potenziale wie Herausforderungen einer Kopplung diskutiert.

Auf die theoretischen Überlegungen aufbauend werden Konzepte zur praktischen Umsetzung beider Ziele, der Ableitung eines Simulationsgraphen und der Visualisierung von Simulationsergebnissen, entwickelt. Beide Konzepte werden so allgemein wie möglich entworfen und prototypisch als Plugin für den 3DCityDB Importer/Exporter implementiert. Eine Evaluierung erfolgt anhand zweier Beispieldatensätze, einem Stadtmodell von New York City und einem Stadtmodell von München in Verbindung mit OSM-Straßenachsen.

Nach dem umgesetzten Konzept besteht die Möglichkeit innerhalb weniger Sekunden automatisch aus CityGML ein Vissim Netzwerk ganzer Stadtteile abzuleiten, welches für die Durchführung einer Verkehrssimulation geeignet ist. Die Qualität des Ergebnisses hängt dabei stark vom Vorhandensein und der Detailfülle der Ausgangsdaten ab. So konnten für das Beispiel New York qualitativ höherwertige Ergebnisse erzielt werden als für das Stadtmodell von München in Verbindung mit OSM-Mittelachsen. In den meisten Fällen wird eine manuelle Nachbearbeitung der automatisch erzeugten Simulationsgraphen notwendig sein, was das Potenzial dieses Ansatzes jedoch in keinster Weise schmälert.

Eine Visualisierung von Ergebnissen einer Verkehrssimulation wurde KML-basiert umgesetzt, was eine Darstellung in Google Earth und im 3D-Web-Map-Client der 3DCityDB ermöglicht.

Mit Stadtobjekten verschiedener CityGML Module in Verknüpfung mit 3D-Fahrzeugmodellen aus SketchUp lassen sich hochwertige Visualisierungsergebnisse erzielen. Ab einer gewissen Anzahl an (texturierten) Stadtobjekten und Fahrzeugen sinkt die Performance sowohl in Google Earth als auch im 3D-Web-Map-Client. Auch bei der Visualisierung macht sich die Qualität der Ausgangsdaten z.B. in der Lagegenauigkeit oder der Detailtreue des Straßenverlaufs bemerkbar, was eine fehlerhafte Positionierung der Fahrzeuge abseits der eigentlichen Fahrbahn zur Folge haben kann.

Die vorliegende Arbeit hat Chancen aufgezeigt, die in der Kopplung von Verkehrssimulation und Stadtmodell liegen. Das Potenzial dieser Synthese ist jedoch bei weitem noch nicht ausgeschöpft. Eine verbesserte Abbildung des Straßenverlaufs könnte durch einen Übergang von Bestandsachsen auf Fahrbahnachsen erreicht werden. Dazu wäre entweder eine Erweiterung des CityGML Standards um Achsrepräsentationen in höheren LoDs erforderlich oder ein Konzept, welches aus detaillierten Flächenrepräsentationen Fahrbahnachsen herleitet. Vorstellbar wäre beispielsweise ein Ausgleichungsansatz nach der Methode der kleinsten Quadrate, welcher unter Einbezug der Anzahl an Fahrspuren einzelne Fahrspurflächen in die Gesamtfläche einpasst, aus denen dann Fahrspurachsen bestimmt werden könnten. Im Zuge der Ausgleichung könnten auch zugehörige Fahrbahnbreiten bestimmt werden. Abgeleitete Fahrbahnachsen würden einem effektiven Informationsgewinn gegenüber einem einfachen LoD0-Network entsprechen. Allerdings bliebe der Nachteil, dass eine Ableitung der Fahrtrichtung aus Flächendaten nicht möglich ist. Dieses Problem könnte durch eine Kombination der flächenhaften Ausgangsdaten mit einem LoD0-Datensatz gelöst werden. In diesem Fall würde LoD 0 Informationen zur Fahrtrichtung liefern, während der genaue Achsverlauf aus den Flächendaten hergeleitet werden könnte. Das Konzept zur Ableitung eines Simulationsgraphen müsste dann ebenfalls erweitert werden, da Fahrbahnachsen einer anderen Behandlung bedürfen als Bestandsachsen.
Auch in Sachen Visualisierung von Simulationsergebnissen im 3D-Stadtmodell sind zukünftige Erweiterungen denkbar. So wurden bisher lediglich drei der dreizehn thematischen Module aus CityGML für eine Visualisierung herangezogen. Neben den verwendeten Modulen “Appearance“, “Building“ und “Transportation“ bieten sich praktisch alle weiteren Module an, darunter insbesondere “Bridge“, “CityFurniture“, “Tunnel“ und “Vegetation“.
Für eine Visualisierung im 3D-Web-Map-Client ist über eine Erweiterung der bestehenden Nutzeroberfläche nachzudenken, um das Laden dynamischer Fahrzeugdaten sowie von 3D-Fahrzeugmodellen im glTF-Format zu erleichtern. Aufgrund der beobachteten Performanceeinbußen ab einer gewissen Anzahl an Stadtobjekten und Fahrzeugen ist über eine Verlagerung der Visualisierung in die Performance-optimierte Umgebung der Verkehrssimulation nachzudenken. Vissim verfügt über ein eigenes Format für 3D-Modelle namens “V3D“. Mit einem Zusatzmodul (Vissim 3D Modeler) können eigene Modelle erstellt, sowie 3D-Modelle aus SketchUp oder Autodesk importiert und konvertiert werden [PTV, 2016]. An dieser Schnittstelle wäre eine Erweiterung hinsichtlich einer Integration von CityGML-Objekten denkbar, um ein hochwertiges Umgebungsmodell direkt in der Verkehrssimulation zu schaffen.

Literaturverzeichnis

[Analytical Graphics Inc. 2015a] ANALYTICAL GRAPHICS INC.: *Graphics Tech in Cesium - Renderer Architecture*. The Cesium Blog - Project news and developer tips [Internet]. http://cesiumjs.org/2015/05/15/Graphics-Tech-in-Cesium-Architecture//. Version: 2015a. – (Zugriff am 15.05.2017)

[Analytical Graphics Inc. 2015b] ANALYTICAL GRAPHICS INC.: *Cesium Wiki*. [Internet]. https://github.com/AnalyticalGraphicsInc/cesium/wiki. Version: 2015b. – (Zugriff am 15.05.2017)

[Applegate 2013] APPLEGATE, Chris: *The design and simulation of traffic networks in virtual environments*, University of East Anglia, Dissertation, 2013. https://ueaeprints.uea.ac.uk/42412/1/2013ApplegateCSPhD.pdf

[Asahara et al. 2015] ASAHARA, Akinori ; SHIBASAKI, Ryosuke ; ISHIMARU, Nobuhiro ; BURGGRAF, David: OGC® Moving Features Encoding Part I: XML Core. In: *OGC 14-083r2, Open Geospatial Consortium* (2015). http://www.opengeospatial.org/standards/movingfeatures

[Bailey & Chen 2011] BAILEY, John E. ; CHEN, Aijun: The role of Virtual Globes in geoscience. In: *Computers & Geosciences* 37.1 (2011), pp. 1 - 2. http://dx.doi.org/10.1016/j.cageo.2010.06.001. – DOI 10.1016/j.cageo.2010.06.001. – ISSN 0098–3004. – Virtual Globes in Science

[Behrisch et al. 2011] BEHRISCH, Michael ; BIEKER, Laura ; ERDMANN, Jakob ; KRAJZEWICZ, Daniel: SUMO - simulation of urban mobility: an overview. In: *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation* ThinkMind, 2011

[Beil 2017] BEIL, Christof: *Detaillierte Repräsentation des Straßenraums in 3D-Stadtmodellen*, Technische Universität München, Master Thesis, 2017. https://mediatum.ub.tum.de/1253230?query=geoinformatik&show_id=1350734&style=full_text

[Bézivin 2005] BÉZIVIN, Jean: On the unification power of models. In: *Software & Systems Modeling* 4.2 (2005), pp. 171-188. https://www.researchgate.net/profile/Jean_Bezivin/publication/220059240_On_the_Unification_Power_of_Models/links/0deec539d72145bfd6000000/On-the-Unification-Power-of-Models.pdf

[BMVI 2016] BMVI: *Bundesministerium für Verkehr und digitale Infrastruktur: Verkehr und Mobilität*. [Internet]. http://www.bmvi.de/SharedDocs/DE/Artikel/verkehr-und-mobilitaet.html. Version: 2016. – (Zugriff: 16.11.2016)

[Bröring et al. 2012] BRÖRING, Arne ; STASCH, Christoph ; ECHTERHOFF, Johannes: OGC® Sensor Observation Service Implementation Standard. In: *OGC 12-006, Open Geospatial Consortium* (2012). http://www.opengeospatial.org/standards/sos

[Campos et al. 2015] CAMPOS, Carlos ; LEITÃO, Joã. M. ; COELHO, António F.: Integrated Modeling of Road Environments for Driving Simulation. In: *GRAPP*, 2015, pp. 70-80

[Chaplier et al. 2010] CHAPLIER, Julien ; THAT, Thomas N. ; HEWATT, Marcus ; GALLÉE, Gilles: Toward a standard: RoadXML, the road network database format. In: *Proceedings of the Driving Simulation Conference Europe*. Paris, France, 2010, pp. 211-220

[Chaturvedi & Kolbe 2016] CHATURVEDI, K. ; KOLBE, T. H.: Integrating Dynamic Data and Sensors with semantic 3D City Modles in the context of Smart Cities. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* IV-2/W1 (2016), pp. 31-38. http://dx.doi.org/10.5194/isprs-annals-IV-2-W1-31-2016. – DOI 10.5194/isprs–annals–IV–2–W1–31–2016

[Chaturvedi & Kolbe 2015] CHATURVEDI, K. ; KOLBE, T.H.: Dynamizers - Modeling and implementing dynamic properties for semantic 3D city models. In: *Eurographics Workshop on Urban Data Modelling and Visualisation* (2015), November 23rd, pp. 43-48. http://dx.doi.org/10.2312/udmv.20151348. – DOI 10.2312/udmv.20151348. – Delft, the Netherlands

[Colditz et al. 2007] COLDITZ, Johannes ; DRAGON, Ludger ; FAUL, Rüdiger ; MELJNIKOV, Darko ; SCHILL, Volkhard ; UNSELT, Thomas ; ZEEB, Eberhard: Use of Driving Simulators within Car Development. In: *Conference Proc. DSC North America, Iowa, September 2007*, 2007

[Cox 2013] COX, Simon: Geographic information — Observations and measurements OGC Abstract Specification Topic 20. In: *OGC 10-004r3, Open Geospatial Consortium* v2. 0.0 (2013). http://www.opengeospatial.org/standards/om

[Dallmeyer 2014] DALLMEYER, Jörg: *Simulation des Straßenverkehrs in der Großstadt - Das Mit- und Gegeneinander verschiedener Verkehrsteilnehmertypen*. Wiesbaden : Springer Fachmedien, 2014 http://dx.doi.org/10.1007/978-3-658-05207-2_3

[Del Fabro et al. 2005] DEL FABRO, Marcos D. ; BÉZIVIN, Jean ; JOUAULT, Frédéric ; VALDURIEZ, Patrick et al.: Applying Generic Model Management to Data Mapping. In: *proceedings of BDA*. Saint-Malo, France, 2005, pp. 343-355

[Ducloux et al. 2016] DUCLOUX, P. ; CHAPLIER, J ; G., Millet ; GALLEE, G.: RoadXML – Road Network Description, XML Format Specification. (2016), Nr. Version 2.4.1. www.road-xml.org. – (Zugriff am 10.02.2017)

[Dupuis 2015] DUPUIS, Marius: OpenDRIVE® Format Specification. Rev. 1.4 Issue H (2015). www.opendrive.org. – (Zugriff am 10.02.2017)

[Esser & Schreckenberg 1997] ESSER, J. ; SCHRECKENBERG, M.: Microscopic Simulation of Urban Traffic Based on Cellular Automata. In: *International Journal of Modern Physics C* 08.05 (1997), pp. 1025-1036. http://dx.doi.org/10.1142/S0129183197000904. – DOI 10.1142/S0129183197000904

[Fan 2010] FAN, Hongchao: *Integration of time - dependent features within 3D city model*, Technische Universität München, Dissertation, 2010. http://mediatum.ub.tum.de/?id=982866

[Fellendorf & Vortisch 2010] FELLENDORF, Martin ; VORTISCH, Peter: Microscopic Traffic Flow Simulator VISSIM. Version: 2010. http://dx.doi.org/10.1007/978-1-4419-6142-6_2. In: BARCELÓ, Jaume (Hrsg.): *Fundamentals of Traffic Simulation*. New York : Springer, 2010. – DOI 10.1007/978–1–4419–6142–6_2. – ISBN 978–1–4419–6142–6, pp. 63-93

[Feng et al. 2010] FENG, T. ; LIANG, H. ; WU, J.: Ontology Driven Data Sharing between Microscopic Traffic Simulation and GIS. In: *2010 Second International Conference on Computer Modeling and Simulation* Bd. Vol. 3, IEEE, 2010, pp. 84-89

[Fischer & Hofer 2008] FISCHER, Peter ; HOFER, Peter: *Lexikon der Informatik*. Bd. 14. Springer-Verlag Berlin Heidelberg, 2008 http://dx.doi.org/10.1007/978-3-540-72550-3

[Greis 2016] GREIS, Friedhelm: *Zulassung autonomer Autos: Die längste Fahrprüfung des Universums*. Golem.de [Internet]. http://www.golem.de/news/zulassungautonomer-autos-die-laengste-fahrpruefung-des-universums-1611-124139.html. Version: Veröffentlicht: 3.11.2016, 12:03 2016. – (Zugriff am 08.11.2016)

[Gröger et al. 2012] GRÖGER, Gerhard ; KOLBE, Thomas H. ; NAGEL, Claus. ; HÄFELE, Karl-Heinz: OGC City Geography Markup Language (CityGML) En-coding Standard, Version 2.0.0. In: *OGC 12-019, Open Geospatial Consortium* (2012). www.opengeospatial.org/standards/citygml

[Gröger & Plümer 2012] GRÖGER, Gerhard ; PLÜMER, Lutz: CityGML – Interoperable semantic 3D city models. In: *{ISPRS} Journal of Photogrammetry and Remote Sensing* 71 (2012), pp. 12 - 33. http://dx.doi.org/10.1016/j.isprsjprs.2012.04.004. – DOI 10.1016/j.isprsjprs.2012.04.004. – ISSN 0924–2716

[Gruler et al. 2016] GRULER, Hans-Christoph ; STUBKJÆR, Erik ; AXELSSON, Peter ; WIKSTROM, Lars: OGC® Land and Infrastructure Conceptual Model Standard (LandInfra). In: *The Open Geospatial Consortium (OGC)* (2016)

[Haubrich et al. 2013] HAUBRICH, Tobias ; SEELE, Sven ; HERPERS, Rainer ; MÜLLER, Martin E. ; BECKER, Peter: Semantic road network models for rapid 3D traffic scenario generation. In: *Tagungsband ASIM/GI-Fachgruppentreffen STS/GMMS, Workshop Simulation technischer Systeme-Grundlagen und Methoden in Modellbildung und Simulation*, 2013, pp. 51-55

[Haubrich et al. 2014] HAUBRICH, Tobias ; SEELE, Sven ; HERPERS, Rainer ; MÜLLER, Martin E. ; BECKER, Peter: A Semantic Road Network Model for Traffic Simulations in Virtual Environments: Generation and Integration. In: *7th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)*. Minneapolis, MN, USA : IEEE, 30 March 2014, pp. 43-50

[Haunert et al. 2005] HAUNERT, Jan-Henrik ; BRENNER, Claus ; NEIDHART, Hauke: Using a gis system for the generation of driving simulator scenes. In: *Advances in Transportation Studies* (2005). https://www.researchgate.net/profile/Jan-Henrik_Haunert/publication/250312965_Using_a_GIS_System_for_the_Generation_of_Driving_Simulator_Scenes/links/54d2503a0cf25017917dc00f.pdf

[Hoffmann 2013] HOFFMANN, Silja: *Mikroskopische Modellierung und Bewertung von verkehrssicherheitskritischen Situationen*, Technische Universität München, Dissertation, 2013. http://mediatum.ub.tum.de/?id=1142214

[InGeoForum sa] INGEOFORUM: *3D-Stadtmodelle*. InGeoForum im ZGDV e.V. und Kommission 3D-Stadtmodelle der DGfK und DGPF [Booklet]. http://www.ingeoforum.de/3d-stadtmodelle/. Version: s.a.. – (Zugriff am 18.02.2017)

[INSPIRE 2014] INSPIRE, Thematic Working Group Transport N.: D2.8.I.7 Data Specification on Transport Networks – Technical Guidelines. Version 3.2 (2014). http://inspire.ec.europa.eu/id/document/tg/tn. – (Zugriff am 11.04.2017)

[ISO 14825 2011] ISO 14825: Intelligente Transportsysteme – Geographische Dateien (GDF) – GDF5.0 (ISO 14825:2011). (2011). https://www.iso.org/standard/54610.html. – Normenausschuss Automobiltechnik (NAAutomobil) im DIN

[KML Reference 2016] KML REFERENCE: *Keyhole Markup Language - Reference*. [Internet]. https://developers.google.com/kml/documentation/kmlreference. Version: 2016. – (Zugriff am 16.05.2017)

[Kolbe 2009] KOLBE, T.H.: Representing and Exchanging 3D City Models with CityGML. Version: 2009. http://dx.doi.org/10.1007/978-3-540-87395-2_2. In: LEE, J. (Hrsg.) ; ZLATANOVA, S. (Hrsg.): *3D Geo-Information Sciences*. Berlin Heidelberg : Springer, 2009. – DOI 10.1007/978–3–540–87395–2_2, pp. 15-31

[Kolbe et al. 2016] KOLBE, T.H. ; ZHIHANG, Y. ; NAGEL, C. ; REDWEIK, R. ; WILLKOMM, P. ; HUDRA, G. ; MÜFTÜOGLU, A. ; F., Kunde: 3D City Database for CityGML. [Documentation]. Version 3.3.0 (2016). http://www.3dcitydb.net/3dcitydb/fileadmin/downloaddata/3DCityDB_Documentation_v3.3.pdf. – Chair of Geoinformatics, Technische Universität München and virtualcitySYSTEMS GmbH, Berlin and M.O.S.S. Computer Grafik Systeme GmbH, Taufkirchen

[Krajzewicz et al. 2002] KRAJZEWICZ, Daniel ; HERTKORN, Georg ; RÖSSEL, Cristian ; WAGNER, Peter: SUMO (Simulation of Urban MObility) - an open-source traffic simulation. In: *Proceedings*

of the 4th Middle East Symposium on Simulation and Modelling (MESM20002). Sharjah (United Arab Emirates) : 4th Middle East Symposium on Simulation and Modelling, 09 2002, pp. 183-187

[Kreft 2012] KREFT, Sven: *Systematik zur effizienten Bildung geospezifischer Umgebungsmodelle fur Fahrsimulationen*, Universität Paderborn, Dissertation, 2012. https://portal.dnb.de/opac.htm?method=simpleSearch&cqlMode=true&query=idn%3D1036552977

[Lehrstuhl für Geoinformatik der TU München 2017] LEHRSTUHL FÜR GEOINFORMATIK DER TU MÜNCHEN: *Projekt: 3D City Model of New York City*. [Dataset]. https://www.gis.bgu.tum.de/de/projekte/new-york-city-3d/. Version: 2017. – Beteiligte Personen: Kolbe, Thomas H & Beil, Christof & Burger, Barbara & Cantzler, Berit & Yao, Zhihang

[Liang et al. 2008] LIANG, Hong ; WU, Jianping ; CHENG, Man ; FENG, Tao: A standardized data model for data sharing of GIS and microscopic traffic simulation system. In: *Service Operations and Logistics, and Informatics, 2008. IEEE/SOLI 2008. IEEE International Conference on* Bd. 1 IEEE, 2008, pp. 116-120

[Löwner et al. 2013] LÖWNER, Marc-O ; CASPER, Egbert ; BECKER, Thomas ; BENNER, Joachim ; GRÖGER, Gerhard ; GRUBER, Ulrich ; HÄFELE, Karl-Heinz ; KADEN, Robert ; SCHLÜTER, Sandra: CityGML 2.0–ein internationaler Standard für 3D-Stadtmodelle, Teil 2: CityGML in der Praxis. In: *Zeitschrift für Geodäsie, Geoinformation und Landmanagement* 2 (2013), S. 131–143. – 138. Jg.

[Löwner et al. 2012] LÖWNER, Marc-O ; BENNER, Joachim ; GRÖGER, Gerhard ; GRUBER, Ulrich ; HÄFELE, Karl-Heinz ; SCHLÜTER, Sandra: CityGML 2.0–ein internationaler Standard für 3D-Stadtmodelle, Teil 1: Datenmodell. In: *Zeitschrift für Geodäsie, Geoinformation und Landmanagement* 6 (2012), S. 340 – 349. – 137. Jg.

[Margreiter 2016] MARGREITER, Martin: Mobilität und Verkehr. In: *Forum Digitale Städte*. Nürnberg, 07.Oktober 2016

[Mozilla Developer Network 2017] MOZILLA DEVELOPER NETWORK: *WebGL (Web Graphics Library)*. Webtechnologien für Entwickler [Internet]. https://developer.mozilla.org/de/docs/Web/API/WebGL_API. Version: 2017. – (Zugriff am 15.05.2017)

[Nagel 2011] NAGEL, Claus: *Importer/Exporter - Plugin API*. 2011. – [unveröffentlichte Folien zur internen Diskussion]

[Negele 2007] NEGELE, Hans-Jürgen: *Anwendungsgerechte Konzipierung von Fahrsimulatoren für die Fahrzeugentwicklung*, Technische Universität München, Dissertation, 2007. http://mediatum.ub.tum.de/?id=620855

[Portele 2007] PORTELE, Clemens: OpenGIS Geography Markup Language (GML) Encoding Standard. In: *OGC 07-036, Open Geospatial Consortium* (2007). http://www.opengeospatial.org/standards/gml

[PostGIS Project Steering Committee 2016] POSTGIS PROJECT STEERING COMMITTEE, (PSC): *PostGIS 2.3.2dev Manual*, 2016. http://postgis.net/stuff/postgis-2.3.pdf

[PostgreSQL 2016] POSTGRESQL: The PostgreSQL Global Development Group: PostgreSQL 9.6.1 Documentation. (2016). https://www.postgresql.org/files/documentation/pdf/9.6/postgresql-9.6-A4.pdf

[PostgreSQL sa] POSTGRESQL: *The PostgreSQL Global Development Group: About PostgreSQL*. [Internet]. https://www.postgresql.org/about/. Version: s.a.. – (Zugriff am 04.01.2017)

[PTV 2016] PTV: *PTV VISSIM 9: Benutzerhandbuch*. Karlsruhe, 09 2016

[PTV sa] PTV: *PTV Vissim Broschüre*. [Booklet]. http://vision-traffic.ptvgroup.com/fileadmin/files_ptvvision/Downloads_N/0_General/2_Products/2_PTV_Vissim/BRO_PTV_Vissim_DE.pdf. Version: s.a.. – (Zugriff am 10.02.2017)

[Randt et al. 2007] RANDT, B. ; BILDSTEIN, F. ; KOLBE, T. H.: Use of Virtual 3D Landscapes for Emergency Driver Training. In: *Proceedings of the Int. Conference on Visual Simulation IMAGE 2007*. Scottsdale, Arizona, 2007

[Richter et al. 2016] RICHTER, Andreas ; FRIEDL, Hartmut ; SCHOLZ, Michael: Beyond OSM - Alternative Data Sources and Approaches Enhancing Generation of Road Networks for Traffic and Driving Simulations. In: *Proceedings of the SUMO2016*. Berlin, 23.-25. Mai 2016 (30), pp. 23-31

[Schüle et al. 2004] SCHÜLE, Michael ; HERRLER, Rainer ; KLÜGL, Franziska: Coupling GIS and Multi-agent Simulation – Towards Infrastructure for Realistic Simulation. Version: 2004. http://dx.doi.org/10.1007/978-3-540-30082-3_17. In: LINDEMANN, Gabriela (Hrsg.) ; DENZINGER, Jörg (Hrsg.) ; TIMM, Ingo J. (Hrsg.) ; UNLAND, Rainer (Hrsg.): *Multiagent System Technologies: Second German Conference, MATES 2004, Erfurt, Germany, September 29-30, 2004. Proceedings*. Berlin, Heidelberg : Springer, 2004. – DOI 10.1007/978–3–540–30082–3_17. – ISBN 978–3–540–30082–3, pp. 228-242

[Stachowiak 1973] STACHOWIAK, Herbert: *Allgemeine Modelltheorie*. Springer-Verlag, Wien, 1973 https://ia800904.us.archive.org/22/items/Stachowiak1973AllgemeineModelltheorie/Stachowiak%20(1973):%20Allgemeine%20Modelltheorie.pdf

[Stegemann & Ameling 1982] STEGEMANN, G. ; AMELING, W.: Simulation und Bewertung von Verkehr in signalgesteuerten Stadtstraßennetzen. Version: 1982. http://dx.doi.org/10.1007/978-3-642-68602-3_53. In: GOLLER, Manuel (Hrsg.): *Simulationstechnik: 1. Symposium Simulationstechnik Erlangen, 26. - 28. April 1982 Proceedings*. Berlin, Heidelberg : Springer, 1982. – DOI 10.1007/978–3–642–68602–3_53. – ISBN 978–3–642–68602–3, pp. 475-493

[Steierwald et al. 1994] STEIERWALD, Gerd ; KÜNNE, Hans D. ; VOGT, Walter ; STEIERWALD, Prof. Dr.-Ing. G. (Hrsg.) ; KÜNNE, Prof. Dr.-Ing. Hans-Dieter (Hrsg.): *Stadtverkehrsplanung*. Bd. 51. Springer Berlin Heidelberg, 1994 http://dx.doi.org/10.1007/978-3-662-10003-5

[Tamminga et al. 2013] TAMMINGA, Guus ; BRINK, Linda van d. ; VAN LINT, Hans ; STOTER, Jantien ; HOOGENDOORN, Serge: Toward GIS-Compliant Data Structures for Traffic and Transportation Models. In: *Transportation Research Board 92nd Annual Meeting*. Washington D.C., 2013

[Tomkins & Lowe 2016] TOMKINS, James ; LOWE, Dominic: Timeseries Profile of Observations and Measurements. In: *OGC 15-043r3, Open Geospatial Consortium* (2016). http://docs.opengeospatial.org/is/15-043r3/15-043r3.html

[Treiber & Kesting 2010] TREIBER, M. ; KESTING, A.: *Verkehrsdynamik und-simulationen: Daten, Modelle und Anwendungen der Verkehrsflussdynamik*. Bd. 43. Springer Berlin Heidelberg, 2010 http://dx.doi.org/10.1007/978-3-642-05228-6

[Wilkie et al. 2012] WILKIE, David ; SEWALL, Jason ; LIN, Ming C.: Transforming GIS data into functional road models for large-scale traffic simulation. In: *IEEE Transactions on Visualization and Computer Graphics* 18.6 (2012), 890–901. http://ieeexplore.ieee.org/document/5928342/

[Willenborg 2015] WILLENBORG, Bruno: *Simulation of explosions in urban space and result analysis based on CityGML-City Models and a cloud-based 3D-Webclient*, Technische Universität München, Master Thesis, 2015

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

A UML-Diagramme

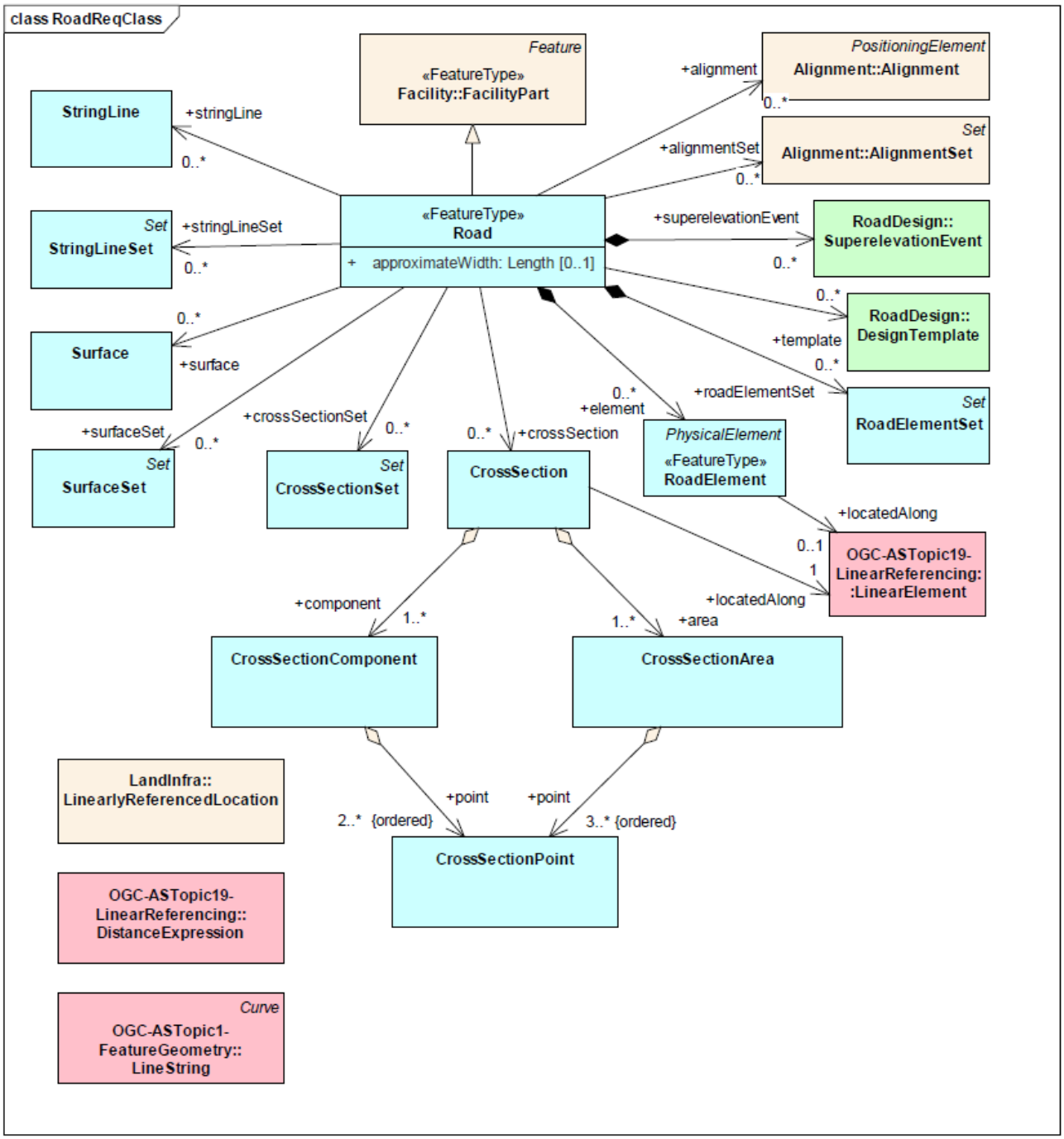


Abbildung A.1 – LandInfra: Road Requirements Class [Gruler et al., 2016, S.106]

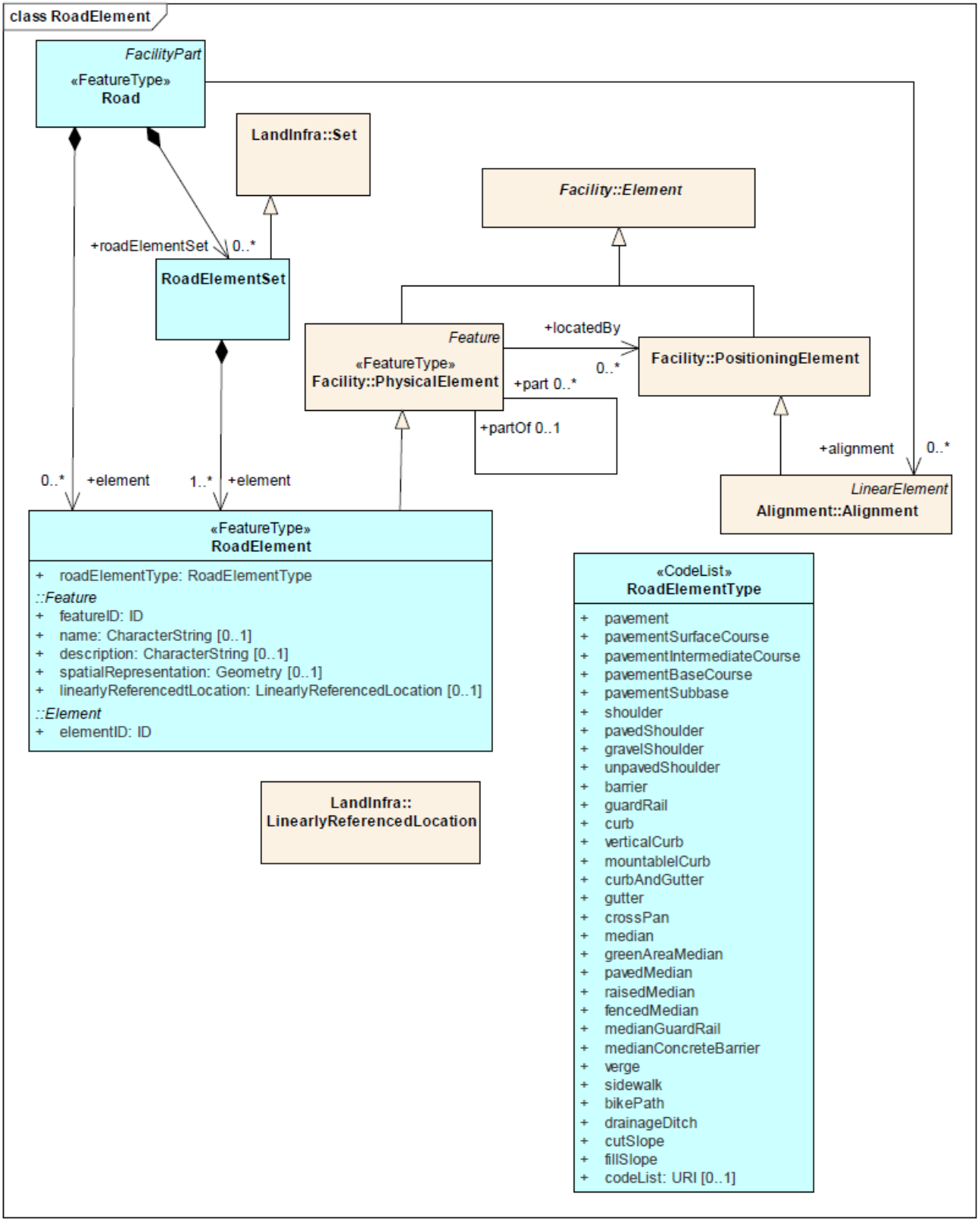


Abbildung A.2 – LandInfra: Road Element [Gruler et al., 2016, S.108]

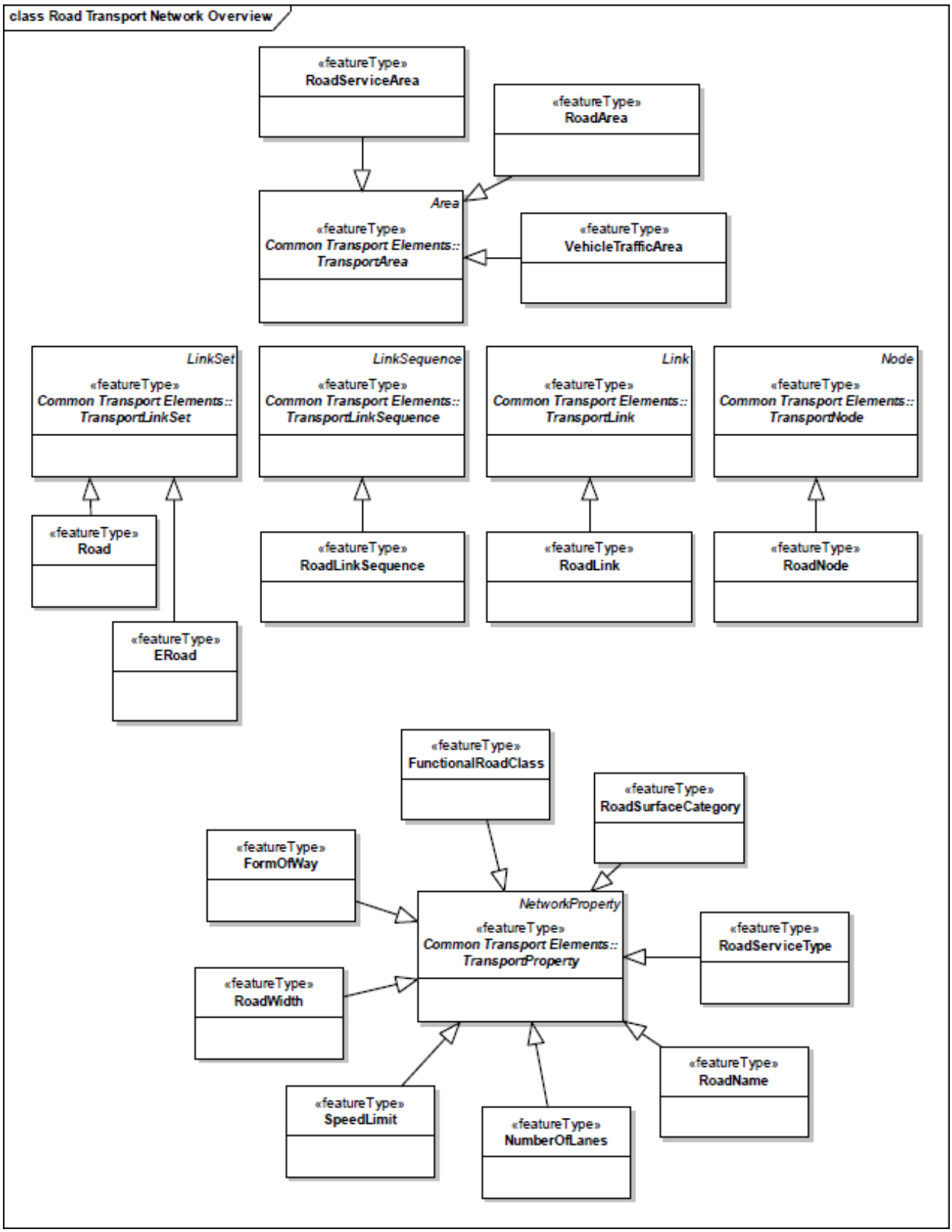


Abbildung A.3 – INSPIRE: Road Transport Networks Application Schema [INSPIRE, 2014, S.25]

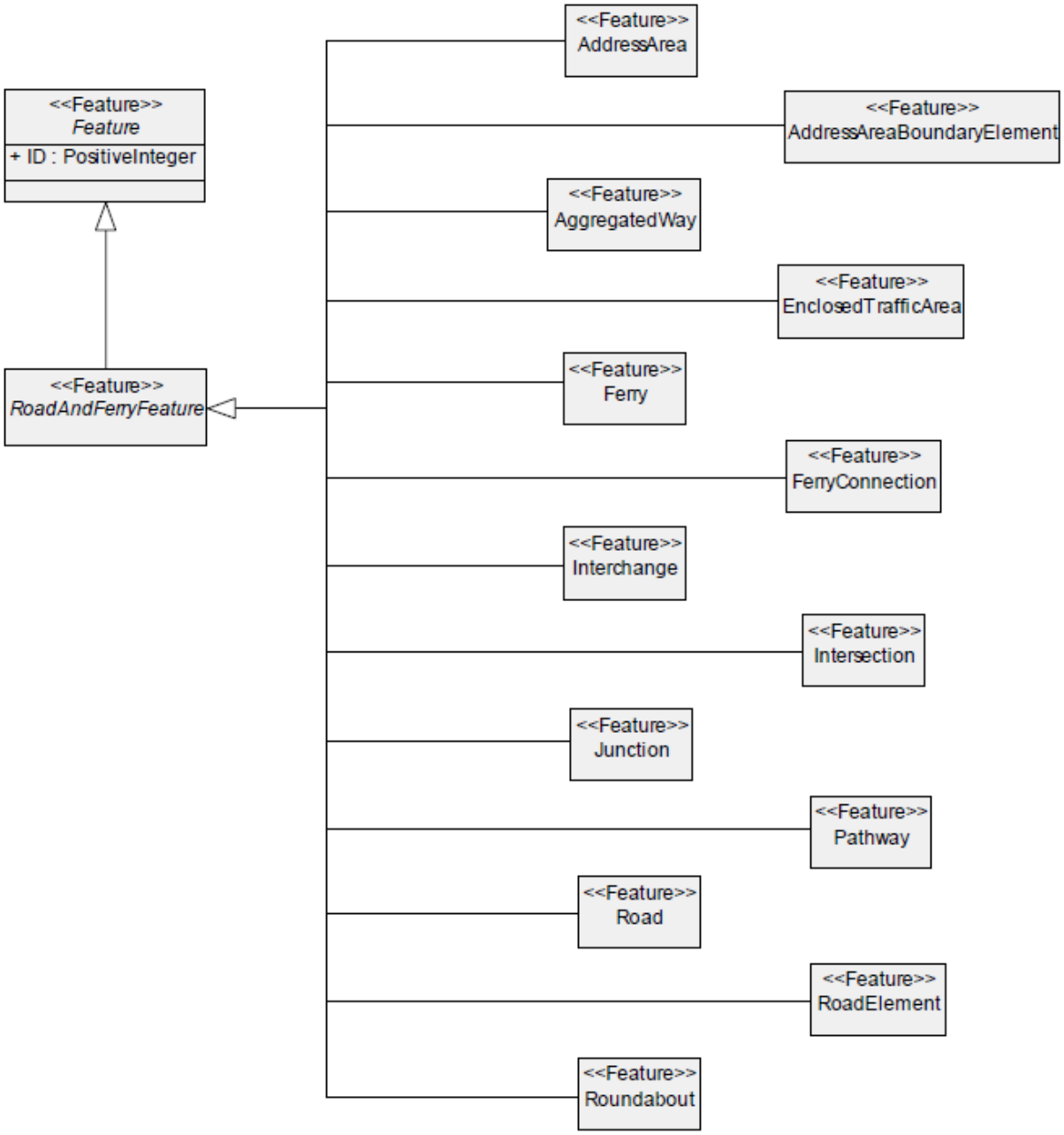


Abbildung A.4 – GDF: Conceptual Data Model for Road and Ferries [ISO 14825, 2011, S.42]

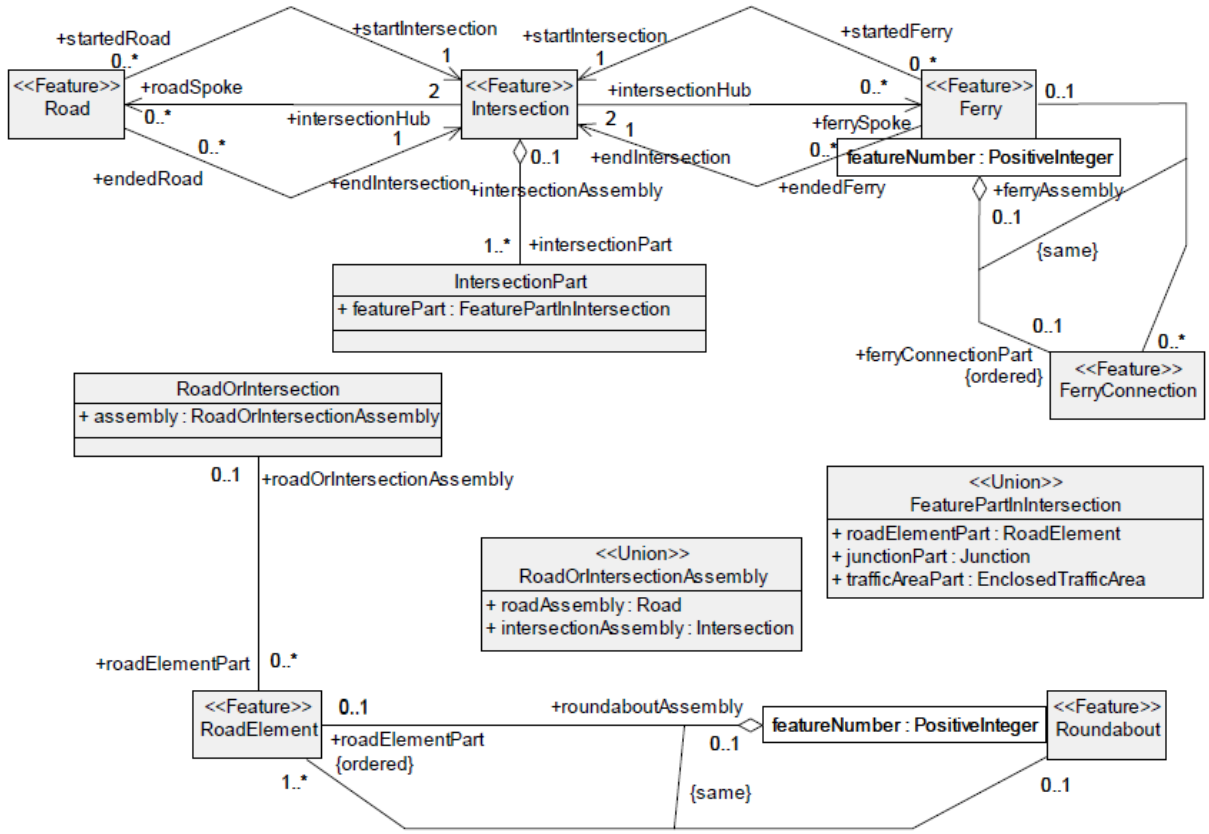


Abbildung A.5 – GDF: Conceptual Data Model for Road and Ferries (continued) [ISO 14825, 2011, S.43]

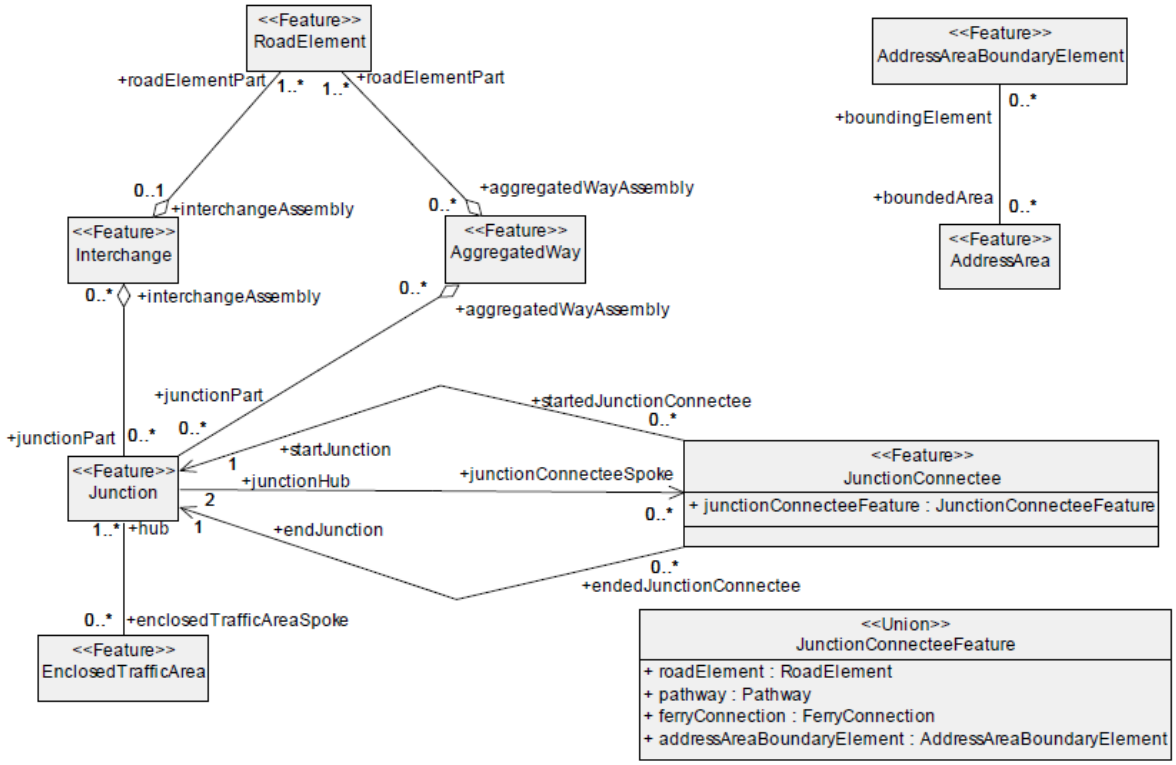


Abbildung A.6 – GDF: Conceptual Data Model for Road and Ferries (continued) [ISO 14825, 2011, S.44]

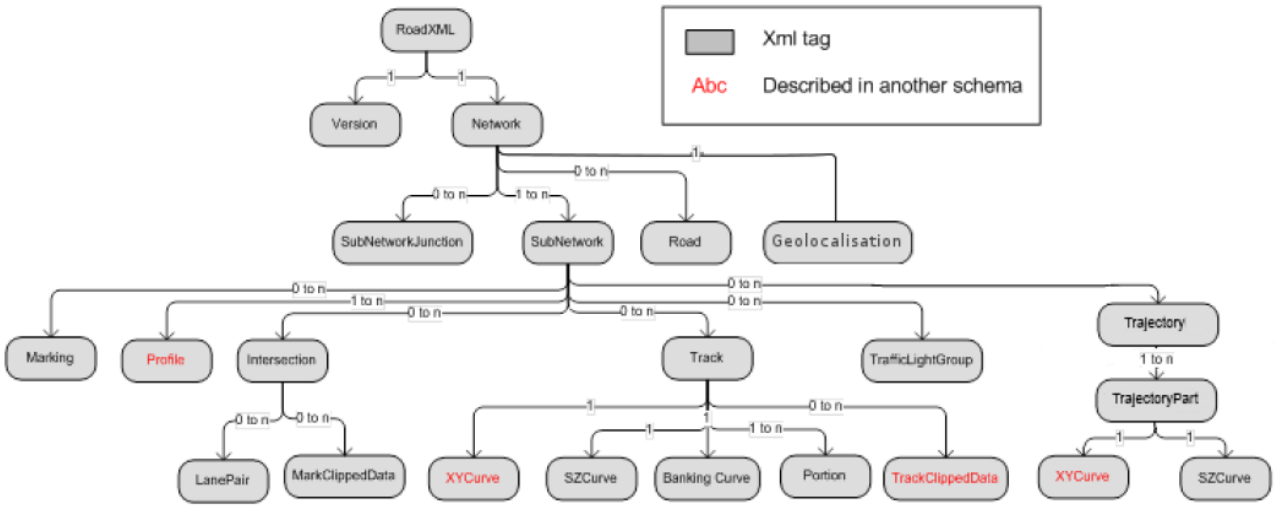


Abbildung A.7 – RoadXML: Network Graph [Ducloux et al., 2016, S.8]

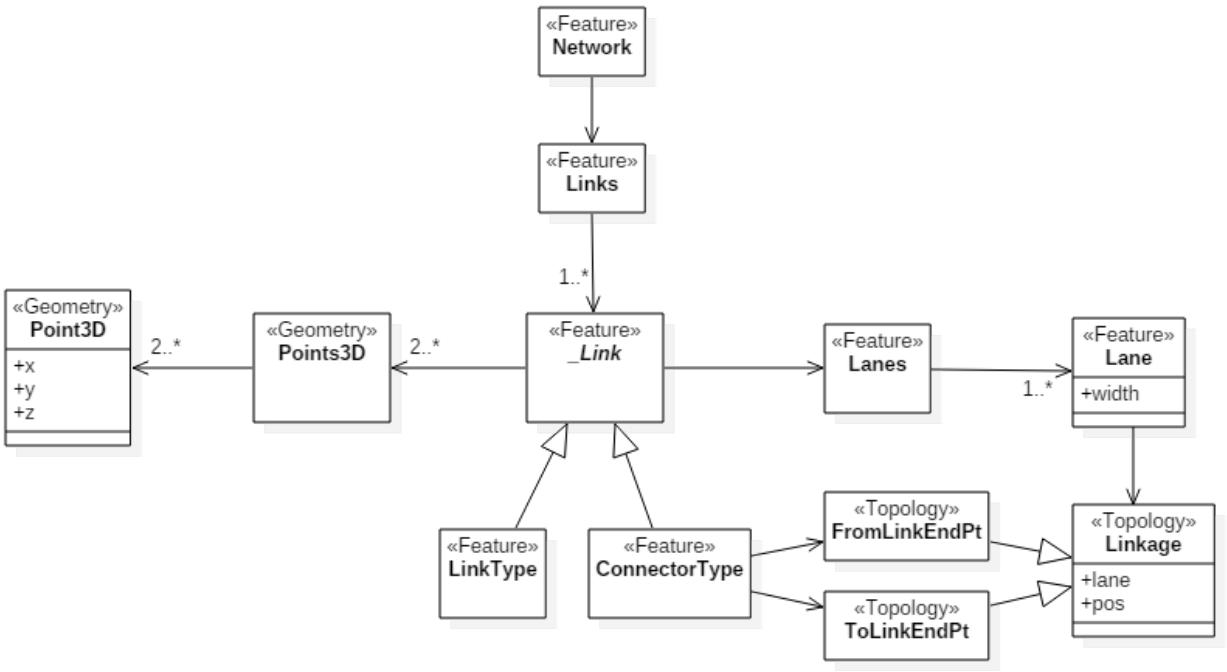


Abbildung A.8 – Vissim: Links als UML-Diagramm frei nach PTV [2016]

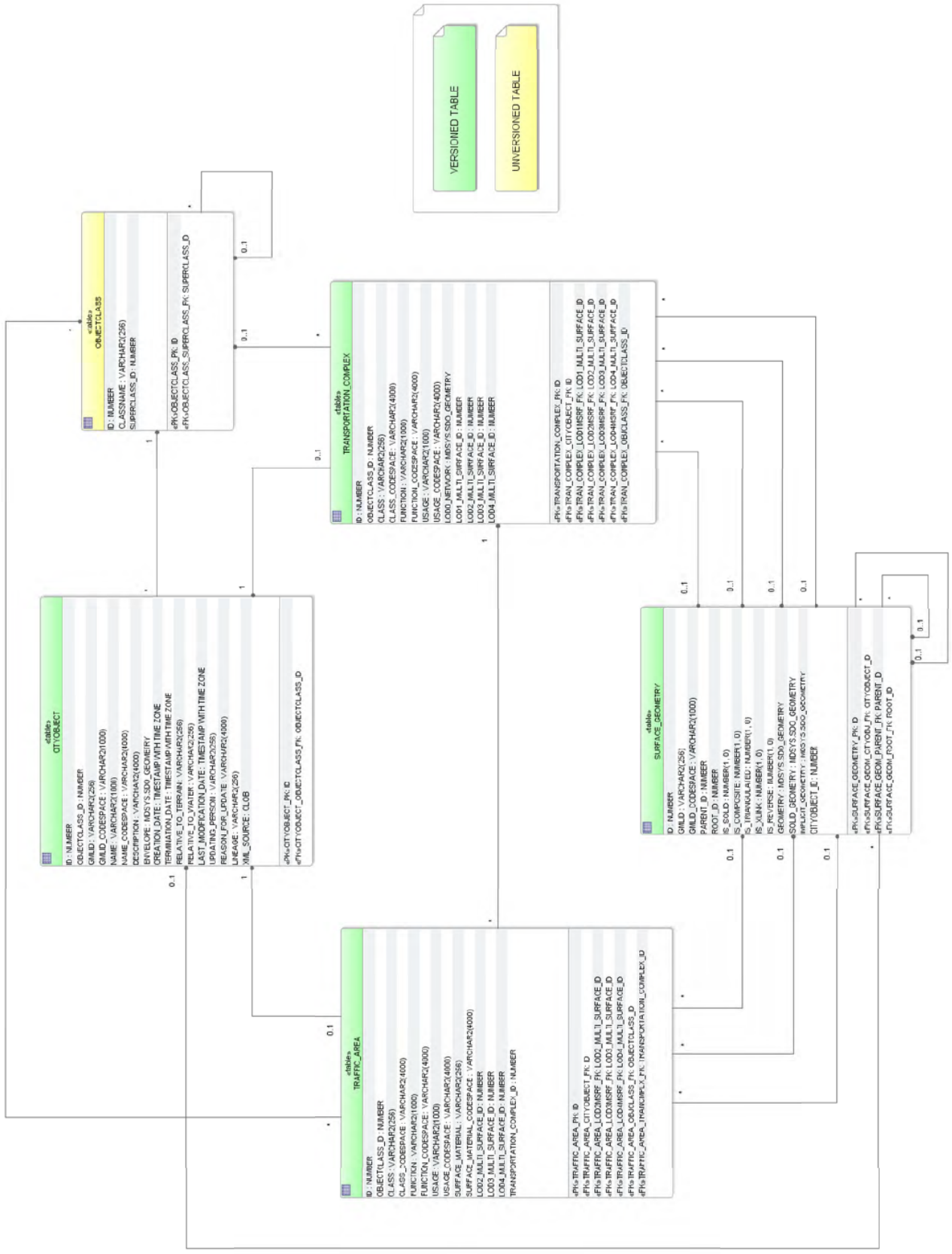


Abbildung A.9 – Datenbankschema der 3DCityDB für das Transportation Modul [Kolbe et al., 2016, S.84]

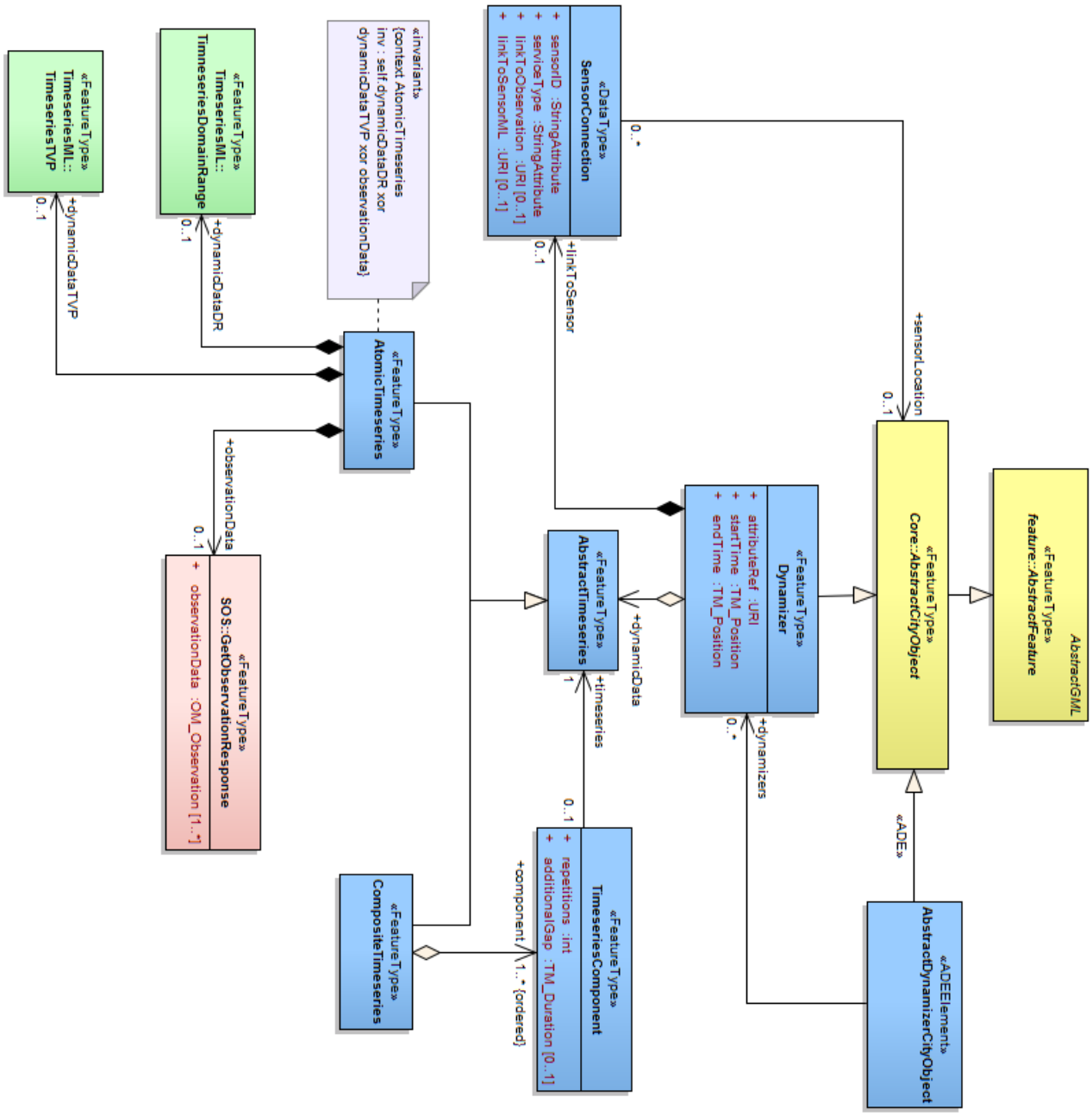


Abbildung A.10 – Dynamizer als UML-Diagramm [Chaturvedi & Kolbe, 2016, S.35]

B Vissim Link-Attribute

Tag	VISSIM-Attribut	Attribut-Beschreibung	Kategorie
<link>	assumSpeedOncom	Angenommene Geschwindigkeit des Gegenverkehrs	SIM
	costPerKm	Kosten je km	SIM
	direction	Richtungsanzeigende Blinker am Fahrzeug (“ALL“; “RIGHT“; “LEFT“)	SIM
	displayType	Farbige Darstellung der Strecke	SIM
	emergStopDist	Nothalteposition (Dient der Modellierung des Fahrstreifenwechselverhaltens)	SIM
	gradient	Steigung oder Gefälle in Prozent (Gefälle als negativer Wert)	NETZ
	hasOvtLn	Hat Überholfahrstreifen	SIM
	isPedArea	Ist Fußgängerfläche	NETZ
	level	Ebene, auf der die Strecke oder Verbindungsstrecke liegt	NETZ
	linkBehavType	Fahrverhalten für die Strecke	SIM
	linkEvalAct	Daten der Strecke für die Streckenauswertung erfassen	SIM
	linkEvalSegLen	Segmentlänge für die Streckenauswertung	SIM
	lnChgDist	Fahrstreifenwechseldistanz zur Modellierung des Fahrstreifenwechsels	SIM
	lnChgDistIsPerLn	Fahrstreifenwechseldistanz gilt je Fahrstreifen	SIM
	lnChgEvalAct	Daten der Fahrstreifen für die Auswertung Fahrstreifenwechsel erfassen	SIM
	lookAheadDistOvt	Vorausschauweite für Überholen	NETZ
	mesoFollowUpGap	Folgezeitlücke zwischen zwei Fahrzeugen desselben Verkehrsstroms	SIM
	mesoSpeed	Meso-Geschwindigkeit	NETZ
	mesoSpeedModel	Legt fest, wie die Geschwindigkeit von Fahrzeugen auf dieser Strecke bestimmt wird	SIM
	name	Bezeichnung der Strecke oder Verbindungsstrecke	NETZ
	no	Eindeutige Nummer der Strecke oder Verbindungsstrecke	NETZ
	ovtOnlyPT	Fahrzeuge können stehendes ÖV-Fahrzeug während dessen Fahrgastwechsels überholen	SIM
	ovtSpeedFact	Überholgeschwindigkeitsfaktor	SIM
	showClsfValues	Anzeigen klassifizierte Werte anstelle des ausgewählten Darstellungstypes	SIM

Tabelle B.1 – *Fortgeführt auf nächster Seite*

Tabelle B.1 – *Fortgeführt von vorheriger Seite*

	showLinkBar	Streckenbalken anzeigen	SIM
	showVeh	In der Simulation einzelne Fahrzeuge anzeigen	SIM
	surch1	Zuschlag 1: Daten zur Ermittlung der Kosten der Fahrzeuge bei der Dynamischen Umlegung	SIM
	surch2	Zuschlag 2: Daten zur Ermittlung der Kosten der Fahrzeuge bei der Dynamischen Umlegung	SIM
	thickness	Dicke des Konstruktionselements für die 3D-Darstellung	NETZ
	vehRecAct	Daten der Strecke für das Fahrzeugprotokoll erfassen	SIM
<point3D>	x	x-Koordinate des Punktes	NETZ
	y	y-Koordinate des Punktes	NETZ
	zOffset	z-Versatz: Höhe über der Ebene	NETZ
<lanes>	width	Breite der Fahrstreifen	NETZ
<fromLinkEndPt>	lane	Wenn die Strecke ein Verbinder ist: Fahrstreifen der Ausgangsstrecke	NETZ
	pos	Position der Ausgangsstrecke, an der die Verbindungsstrecke beginnt	NETZ
<toLinkEndPt>	lane	Wenn die Strecke ein Verbinder ist: Fahrstreifen der Folgestrecke	NETZ
	pos	Position der Folgestrecke, an der die Verbindungsstrecke endet	NETZ

Tabelle B.1 – Attribut-Tabelle von Strecken bzw. Verbindungsstrecken in Vissim

C Verwendete PostGIS Funktionen

Funktion	Kurzbeschreibung
ST_ApproximateMedialAxis	Berechnet die genäherte Mittelachse einer flächenhaften Geometrie.
ST_Azimuth	Gibt den nord-basierten Azimutwinkel im Uhrzeigersinn in der Einheit Radian zurück.
ST_Buffer	Gibt eine Geometrie zurück, die alle Punkte innerhalb einer vordefinierten Entfernung von der Eingangsgeometrie umfasst.
ST_ClosestPoint	Gibt den 2D-Punkt auf einer Geometrie g1 zurück, der am nächsten zu einer Geometrie g2 liegt.
ST_DumpPoints	Gibt reihenweise die Menge aller Punkte (Geometrie, Pfad) zurück, die die Eingabegeometrie bilden.
ST_Equals	Gibt WAHR zurück, falls die Eingabe-Geometrien die selbe Geometrie repräsentieren.
ST_EndPoint	Gibt den letzten Punkt eines LINESTRING- oder CIRCULARLINESTRING-Objektes zurück.
ST_GeometryN	Gibt die n-te Geometrie einer GEOMETRYCOLLECTION, (MULTI)POINT, (MULTI)LINESTRING, etc. zurück.
ST_Length	Gibt die 2D-Länge eines (MULTI)LINESTRING-Objektes in der Einheit des Referenzsystems zurück.
ST_Line_Interpolate_Point	Gibt einen Punkt interpoliert entlang einer Linie zurück.
ST_Line_Locate_Point	Gibt die Position eines Punktes entlang einer Linie als Fließkommazahl [0-1] zurück.
ST_LineMerge	Gibt ein oder eine Menge von LINESTRING-Objekt(en) zurück, erzeugt aus einem MULTILINESTRING-Objekt.
ST_LineSubString	Gibt eine Untermenge der Linie zurück, definiert durch Bruchteile [0-1] der Gesamtlänge.
ST_NPoints	Gibt die Anzahl an Punkten einer Geometrie zurück.
ST_OffsetCurve	Gibt eine Linie parallel zur Ausgangslinie in vordefiniertem Abstand zurück.
ST_PointN	Gibt den n-ten Punkt eines LINESTRING-Objektes zurück.
ST_Reverse	Gibt die Geometrie in umgekehrter Orientierung zurück.
ST_Simplify	Gibt eine vereinfachte Repräsentation der Ausgangsgeometrie unter Verwendung des Douglas-Peucker-Algorithmus zurück.
ST_StartPoint	Gibt den ersten Punkt eines LINESTRING- oder CIRCULARLINESTRING-Objektes zurück.
ST_Within	Gibt WAHR zurück, falls eine Geometrie g1 komplett innerhalb einer Geometrie g2 liegt.
ST_X	Gibt die X-Koordinate eines Punktes zurück.
ST_Y	Gibt die Y-Koordinate eines Punktes zurück
ST_Z	Gibt die Z-Koordinate eines Punktes zurück

Tabelle C.1 – Verwendete PostGIS Funktionen mit zugehöriger Beschreibung in Kurzform

D FME Workflow

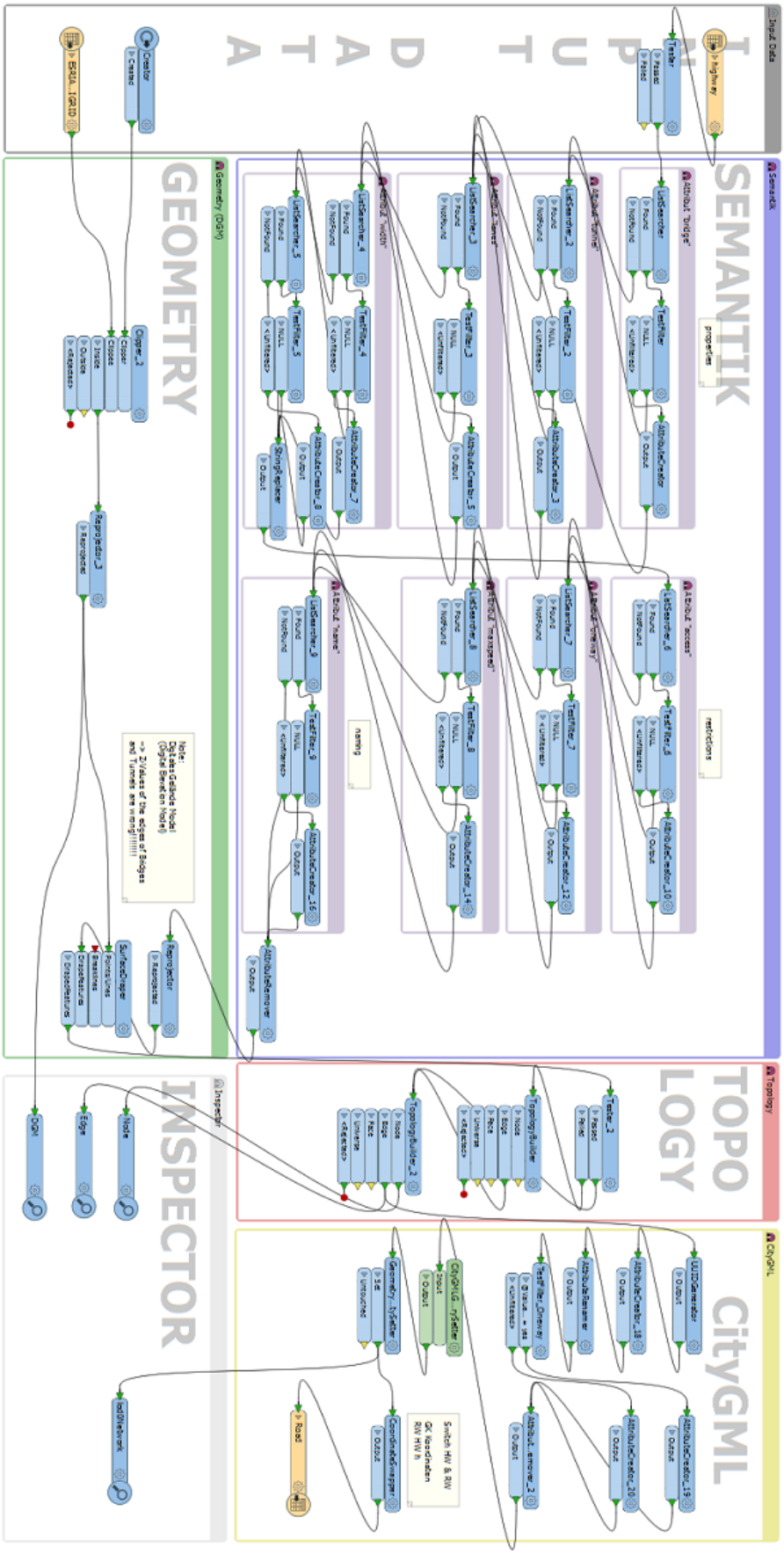


Abbildung D.1 – Schematischer Überblick über den Ablauf in FME zur Erstellung eines CityGML Lod0-Networks aus Open Street Map Straßenachsen

E Liste verwendeter SketchUp-Modelle

Fahrzeug	Link	Zugriff
Audi RS5	https://3dwarehouse.sketchup.com/model/ee3fad8103eba00e3dd12e2f8435ca98/Audi-RS-5	11.05.2017
Audi A8	https://3dwarehouse.sketchup.com/model/7e6f6101-79a7-4d17-aae4-94ff0c046e1a/2012-Audi-A8-42-Quattro-Luxury-Custom-Edition	11.05.2017
Porsche 911	https://3dwarehouse.sketchup.com/model/3784a123e454def616e828d0482766e/prosche-911	11.05.2017
Audi S4	https://3dwarehouse.sketchup.com/model/4d8a4808e9bf605b11647ffa4306609/Audi-S4	11.05.2017

Tabelle E.1 – Liste der zur Visualisierung der Ergebnisse einer Verkehrssimulation verwendeten SketchUp-Modelle und deren Bezugsquelle