Hochschule Merseburg

M.A. Informationsdesign und Medienmanagement

Masterarbeit

Vorverarbeitung mobilitätsrelevanter Daten zur Unterstützung nachhaltiger Verkehrsplanungsprozesse mittels 3D-Visualisierung

Zur Erlangung des Grades: *Master of Arts*

Vorgelegt von:	Franziska Schultz
Datum	14.01.2025
Erstgutachterin	Prof. Dr. Lisa Wenige
Zweitgutachter	Prof. Dr. phil. Michael Meng

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit in allen Teilen selbstständig angefertigt und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt habe, und dass die Arbeit in gleicher oder ähnlicher Form in noch keiner anderen Prüfung vorgelegen hat. Sämtliche wörtlichen oder sinngemäßen Übernahmen und Zitate, sowie alle Abschnitte, die mithilfe von KI-basierten Tools entworfen, verfasst und/oder bearbeitet wurden, sind kenntlich gemacht und nachgewiesen.

Ich versichere, dass ich keine KI-basierten Tools verwendet habe, deren Nutzung der Prüfer/die Prüferin explizit schriftlich ausgeschlossen hat. Ich bin mir bewusst, dass die Verwendung von Texten oder anderen Inhalten und Produkten, die durch KI-basierte Tools generiert wurden, keine Garantie für deren Qualität darstellt. Ich verantworte die Übernahme jeglicher von mir verwendeter maschinell generierter Passagen vollumfänglich selbst und trage die Verantwortung für eventuell durch die KI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate. Ich versichere zudem, dass in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt.

Im Rahmen der vorliegenden Arbeit wurden KI-Tools gezielt eingesetzt, um die Programmierarbeit zu unterstützen, insbesondere bei der Fehlerbehebung und Optimierung von Code. Zudem wurden sie genutzt, um komplexe Themengebiete effizient zu erschließen. Einzelne Formulierungen wurden mit Hilfe KI-gestützter Tools sprachlich verbessert und fachliche Begriffe präzise übersetzt. Eine explizite Verwendung von KI im entwickelten Workflow ist klar gekennzeichnet und wird im entsprechenden Abschnitt ausführlich reflektiert.

_____

Ort, Datum

_____

Franziska Schultz

Abstract

Diese Arbeit verfolgt das Ziel, einen Workflow zur Verknüpfung von 3D-Stadtmodellen mit mobilitätsrelevanten Daten zu entwickeln, der eine Vielzahl von Anforderungen und Herausforderungen berücksichtigt. Ausgangspunkt sind die Grundlagen zu 3D-Stadtmodellen sowie eine präzise Definition mobilitätsrelevanter Daten. Verschiedene Datenformate werden vorgestellt und potenzielle Anwendungsgebiete sowie deren Datenqualität ausführlich analysiert. Anhand eines praxisbezogenen Beispiels wird der Workflow erarbeitet, der die Integration von CityGML-, OSM- und GTFS-Daten umfasst. Zunächst werden geeignete Datenquellen identifiziert und eine systematische Literaturrecherche zu nachhaltiger Mobilität durchgeführt. Die daraus resultierenden Maßnahmen werden in OSM-Tags überführt und mittels der Overpass-API extrahiert. Anschließend erfolgt der Import der Rohdaten in PostGIS-Datenbanken innerhalb eines Docker-Netzwerks, gefolgt von der Datenverknüpfung mittels Python und festgelegter Verknüpfungsregeln.
Durch die Automatisierung des Workflows mittels Python-Skripten zeigte sich eine verbesserte Effizienz und Benutzerfreundlichkeit, trotz bestehender Importfehler bei osm2pgsql. Eine erstellte Anwenderdokumentation, welche den entwickelten Workflow verständlich erklärt, wird in einem durchgeführten Usability-Test positiv bewertet, was die Praktikabilität des Ansatzes bestätigt. Herausforderungen bestehen vor allem in der Komplexität des Prozesses, der Vielzahl der Schritte sowie der unterschiedlichen Datenformate und Importfehler. Diese Arbeit schließt eine Forschungslücke, indem sie praxisnahe Erkenntnisse und detaillierte Fehleranalysen zur Datenverknüpfung liefert. Zukünftige Arbeiten sollen den Workflow mit diversen Datensätzen testen und weiter optimieren, um die Flexibilität und Fehlerresistenz zu erhöhen. Insgesamt bietet der entwickelte Workflow eine solide Grundlage für die Integration von 3D-Stadtmodellen mit mobilitätsrelevanten Daten und erleichtert die Nutzung für Anwender ohne tiefgehende Vorkenntnisse im Bereich Geodaten.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

Abkürzung	Bedeutung
API	Application Programming Interface
CM / CDM	Conceptual Model/ Conceptual Data Model
CRS	Coordinate Reference System
DB	Datenbank
GTFS	General Transit Feed Specification
LODs	Levels of Detail
MIV	Motorisierter Individualverkehr
OGC	Open Geospatial Consortium
ÖPNV	Öffentlicher Personennahverkehr
OSM	OpenStreetMap
ÖV	Öffentlicher Verkehr
SRID	Spatial Reference Identifier

1 Einleitung

1.1 Motivation

Die fortschreitende Digitalisierung führt dazu, dass immer größere Datenmengen verfügbar werden, die durch das weltweit an Bedeutung gewinnende Open-Data-Prinzip in vielen Fällen frei genutzt und weiterverwendet werden können (vgl. BMI). In diesem Kontext stellen zahlreiche Städte und Gemeinden vermehrt 3D-Stadtmodelle zur Verfügung, die als digitale Abbilder urbaner Gebiete sowohl in unterschiedlichsten Anwendungsbereichen eingesetzt, als auch mit weiteren Daten kombiniert und weiterverarbeitet werden können. Besonders die Verknüpfung von 3D-Stadtmodellen mit Daten aus der Geodatenbank OpenStreetMap (OSM) eröffnet vielfältige Möglichkeiten. OSM bietet eine umfassende Sammlung frei zugänglicher, strukturierter Geodaten, die detaillierte Informationen über die räumliche Organisation und Eigenschaften urbaner Objekte enthalten. Durch die Integration dieser Daten können vielfältige räumliche und thematische Analysen durchgeführt werden, die sowohl in der Forschung als auch in praktischen Anwendungen, etwa im Bereich der Stadtplanung oder der Simulation, eine zentrale Rolle spielen.

Die Vielfältigkeit der vorhandenen Datenformate stellt Anwender*innen jedoch vor erhebliche Hürden, da 3D-Stadtmodelle häufig im CityGML-Format vorliegen, während andere strukturierte Geodaten in XML- oder CSV-basierten Formaten bereitgestellt werden. Abhängig von den Anforderungen der verwendeten Software und dem jeweiligen Verwendungszweck kann der Workflow zur Verarbeitung dieser Daten sehr komplex und unübersichtlich werden. Derzeit ist der Prozess noch durch eine hohe Anzahl an Teilschritten geprägt. Auf Konferenzen, in wissenschaftlichen Arbeiten und in Foren werden zwar verschiedene Verarbeitungsansätze und -schritte beschrieben und diskutiert, jedoch erfordern diese in vielen Fällen ein tiefgehendes technisches Verständnis, was die Nutzung der Daten besonders für Personen ohne entsprechende Kenntnisse erschwert.

Eine Handreichung oder Dokumentation, die einen Workflow zum Verbinden dieser Daten sowie zur Nutzung relevanter Programme in automatisierten Datenintegrationsprozessen verständlich erklärt, fehlt aktuell. Diese Lücke soll mit dieser Arbeit geschlossen werden. Da ein großer Teil der Daten über API[1]s abrufbar ist, bietet sich eine Automatisierung des Prozesses an.

1.2 Forschungsziele

In dieser Arbeit soll ein Workflow entwickelt werden, welcher die Vorverarbeitung und Verknüpfung eines 3D-Stadtmodells und weiterer mobilitätsrelevanter Daten bis hin zur Überführung in ein Unity-Projekt abdeckt. Ziel ist die Erarbeitung eines Workflows, der flexibel auf unterschiedliche Anwendungsfälle anpassbar ist und dabei möglichst viele potenzielle Anforderungen und Herausforderungen berücksichtigt. Dabei liegt ein besonderer Schwerpunkt auf der Vorverarbeitung der Rohdaten, um diese miteinander zu verknüpfen und in einem für die weitere Verarbeitung geeigneten Format bereitzustellen.

Die Entwicklung des Workflows geschieht anhand eines Beispiels, wobei das 3D-Stadtmodell der Leipziger Innenstadt verwendet wird (vgl. OpenData Leipzig). Es soll mit verschiedenen, für nachhaltige Fortbewegung relevanten Daten angereichert werden[2]. Der Fokus liegt auf Daten, die sich gut in 3D-Stadtmodellen darstellen lassen und direkt in die Modellierung einfließen können. Dazu

---

[1] APIs sind Schnittstellen für Anwendungsprogrammierungen (*application programming interfaces*). Diese standardisierten Schnittstellen ermöglichen eine Kommunikation zwischen Softwareanwendungen.

[2] Eine detaillierte Betrachtung des Anwendungsbeispiels erfolgt in Abschnitt 4.

zählen insbesondere räumliche und verkehrsrelevante Informationen. Es soll untersucht werden, wie der Prozess mit Hilfe der Programmiersprache Python und unter Nutzung von kostenloser oder Open-Source-Software so weit wie möglich automatisiert werden kann.

Dabei sollen folgende Forschungsfragen beantwortet werden:

- Welche mobilitätsrelevanten Daten sind für die Anreicherung von 3D-Stadtmodellen geeignet?
- Welche kostenfreien und Open-Source-Softwarelösungen sind geeignet, um 3D-Stadtmodelle mit weiteren Daten zu kombinieren? Wie kann der entstehende Workflow automatisiert werden?
- Welche Herausforderungen gibt es bei der Integration von strukturierten Daten in 3D-Stadtmodelle und wie können diese bewältigt werden? Welche speziellen Herausforderungen gibt es zwecks Datenqualität und Zuordnung?
- Wie können die einzelnen Schritte des Workflows, der verschiedene Softwarelösungen und ein Python-Skript integriert, so dokumentiert werden, dass sie für Anwender*innen verständlich und praxisnah aufbereitet sind, um eine effiziente und reproduzierbare Nutzung zu ermöglichen?

Als Ergebnis dieser Arbeit sollen eine Handreichung sowie Softwareskripte erstellt werden, welche dabei helfen, den Workflow zu überblicken und für Anwender*innen reproduzierbar zu machen.

1.3 Methodik

Zur Erreichung der Forschungsziele werden in dieser Arbeit verschiedene Methoden miteinander kombiniert. Nach einer einleitenden Beschreibung der relevanten Datenformate und des konkreten Anwendungsbeispiels wird eine systematische Literaturrecherche durchgeführt, um städtebauliche Einflussfaktoren auf nachhaltiges Mobilitätsverhalten zu identifizieren. Die dabei gewonnenen Erkenntnisse zu geeigneten mobilitätsrelevanten Daten werden strukturiert aufbereitet, sodass sie anschließend für gezielte Datenabfragen genutzt werden können.

Im darauf aufbauenden Praxisteil werden die Möglichkeiten zur Vorverarbeitung und Integration der gesammelten Daten literaturgestützt aufgezeigt. Für jeden Schritt wird die am besten geeignete Vorgehensweise bestimmt und praktisch umgesetzt, wobei auf die Nutzung von kostenloser und Open-Source-Software geachtet wird. Schritte, die sich für eine Automatisierung eignen, werden mit Hilfe der Programmiersprache Python[3] realisiert und verknüpft, wobei eine einfache grafische Benutzeroberfläche (GUI) erstellt und der Quellcode umfassend dokumentiert wird. Während der Umsetzung werden auftretende Herausforderungen und Hindernisse systematisch erfasst und beschrieben.

Nachdem der gesamte Prozess vorbereitet und getestet wurde, werden die gewonnenen Erkenntnisse in einer Anwenderdokumentation strukturiert zusammengeführt und aufbereitet. Diese Dokumentation soll die Nutzung der verschiedenen Softwarelösungen und des Python-Skripts verständlich erläutern und zusätzlich die Komplexität des Workflows reduzieren und dessen Nutzung vereinfachen. Um die Effektivität und Nutzerfreundlichkeit der Dokumentation zu bewerten, wird sie gemeinsam mit dem entwickelten Python-Skript im Rahmen eines Usability-Tests evaluiert. Dazu wird die Methode des *lauten Denkens* mit einem leitfadengestützten Interview kombiniert, um qualitative Daten zur Usability der Handreichung zu sammeln. Ziel ist es, herauszufinden, ob die entwickelten Materialien die zuvor identifizierten Hürden und Probleme im Workflow adressieren und beheben. Diese Untersuchung hilft sicherzustellen, dass die Anwenderdokumentation den Zugang zu

[3] Eine detaillierte Begründung für die Wahl von Python findet sich in Abschnitt 5.

komplexen technischen Prozessen erleichtert und somit die Relevanz des Workflows für die Praxis erhöht. Die Ergebnisse der Evaluation werden abschließend systematisch analysiert und dokumentiert.

2 Grundlagen

In diesem Kapitel werden die dieser Arbeit zu Grunde liegenden Konzepte und Begriffe genauer beschrieben. Weiterhin werden die für die Arbeit relevanten Datenformate und deren Anwendungsbereiche detailliert behandelt.

2.1 Mobilitätsrelevante Daten

Diese Arbeit beschäftigt sich mit der Vorverarbeitung von Daten, die für Mobilität relevant sind und im Folgenden als *mobilitätsrelevante Daten* bezeichnet werden. Da für diesen Begriff keine einheitliche Definition existiert, wurde auf Basis einschlägiger Literatur eine eigene Definition entwickelt (vgl. Gerike et al. 2020, S. 35-44; vgl. Münsch & Lell 2024; vgl. Scherhaufer et al. 2023; vgl. Wittwer et al. 2021; vgl. Buchholz & Flaig 2022).

Der Begriff mobilitätsrelevante Daten wird auf dieser Grundlage folgendermaßen definiert:

> Mobilitätsrelevante Daten umfassen alle Informationen, die einen Einfluss auf oder eine Relevanz für das Mobilitätsverhalten von Personen haben. Dazu zählen unter anderem Geodaten, Mobilitätsdaten sowie Daten zur öffentlichen Raumgestaltung. Auch soziodemografische Daten sowie Informationen zu wirtschaftlichen oder politischen Rahmenbedingungen können als mobilitätsrelevante Daten betrachtet werden.

Somit umfassen mobilitätsrelevante Daten verschiedene Datenarten, die jeweils spezifische Aspekte des Mobilitätsverhaltens adressieren. Diese können nicht immer strikt voneinander getrennt werden, so können Mobilitätsdaten und soziodemografische Daten auch zu den Geodaten gehören. Zum besseren Verständnis werden die zentralen Begriffe im Folgenden genauer erläutert.

Geodaten

Der Begriff der Geodaten (auch *Geoinformation*) ist sehr weit gefasst und beschreibt generell „Alle Daten mit direktem oder indirektem Bezug zu einem bestimmten Standort oder geografischen Gebiet“ (Inspire 2015). Sie umfassen verschiedenste Datenformate und -formen und können über Geoinformationssysteme (GIS) auf Karten visuell dargestellt werden. Je nach spezifischem Datentyp und Ursprung gibt es unterschiedliche Quellen, um Geodaten zu beziehen, beispielsweise im *Geoportal* des Bundes (vgl. Geoportal). Beispiele für Geodaten können sein (vgl. IBM):

- Vektordaten, in welchen Objekte durch Punkte, Linien und Polygone dargestellt werden
- Attribute, welche Positionen beschreiben
- Punktwolken, welche zu 3D-Modellen rekonstruiert werden können
- Umfragedaten im Zusammenhang mit geografischen Daten
- Technische Zeichnungen von Gebäuden und Orten

Geodaten können Informationen liefern, welche relevant für die Mobilität von Personen sind. Punktwolken können zu 3D-Stadtmodellen umgewandelt werden und Vektordaten können Informationen über die Positionierung von Straßen und Wegen liefern. Eine genaue Auswahl der für das Anwendungsbeispiel relevanten Geodaten erfolgt im weiteren Verlauf der Arbeit.

Mobilitätsdaten

Mobilitätsdaten umfassen „Informationen von Mobilitätsanbietern, Infrastrukturbetreibern und Verkehrsbehörden sowie Informationsanbietern“ (Mobilithek About). Ein großer Teil der Mobilitätsdaten beschreibt Gegebenheiten mit geografischem Bezug, weswegen sie zu den Geodaten

gezählt werden können. Mobilitätsdaten aus Deutschland werden unter anderem auf dem Onlineportal *mobilithek* hochgeladen (vgl. Mobilithek) und stehen dort zum Teil kostenfrei zur Verfügung. Beispiele für dort verfügbare Mobilitätsdaten sind:

- Liniennetzpläne des öffentlichen Personennahverkehrs (ÖPNV)
- Baustelleninformationen
- Echtzeit-Verkehrsdaten
- Straßen- und Verkehrswegenetz
- Statische Verkehrsregelungen wie Zufahrtsbedingungen für Tunnel

Fahrplandaten des *öffentlichen Verkehrs* (ÖV) in Deutschland stehen außerdem über GTFS online zum Teil unter freier Lizenz zur Verfügung.

Alle Mobilitätsdaten haben einen direkten Mobilitätsbezug und gehören deshalb zu den mobilitätsrelevanten Daten. Eine genaue Prüfung der für das Anwendungsbeispiel relevanten Mobilitätsdaten erfolgt im weiteren Verlauf der Arbeit.

Daten zur öffentlichen Raumgestaltung

Daten zur öffentlichen Raumgestaltung enthalten verschiedene Informationen zur Bebauung des urbanen Raums, ein großer Fokus liegt dabei auf der Barrierefreiheit. Zugänge zu solchen Daten bestehen unter anderem über die Geodatenbank OSM (vgl. OSM). Beispiele für Daten zur öffentlichen Raumgestaltung sind:

- Positionen von Bänken und Trinkwasserbrunnen
- Informationen über Begrünung und Parkanlagen
- Daten über barrierefreie Architektur wie Rampen und taktile Leitsysteme
- Informationen zu öffentlichen Toiletten und Wickelräumen

Diese Informationen haben unter Umständen eine Relevanz für das Mobilitätsverhalten von Personen. Der genaue Nutzen der Daten kann je Anwendungsfall evaluiert werden. Eine Prüfung der Relevanz für das hier bearbeitete Anwendungsbeispiel wird im weiteren Verlauf der Arbeit untersucht.

Soziodemografische Daten und wirtschaftliche Rahmenbedingungen

Soziodemografische Daten umfassen Informationen zu Alter, Geschlecht, Religion, Ausbildung, Beruf und Einkommen oder sozialer Schicht (vgl. Beckmann et al. 2016, S.8). Diese sind über verschiedene Umfragen und Statistiken verfügbar. Soziodemografische Daten können relevant für das Mobilitätsverhalten von Personen sein. Beispielsweise können finanzielle Mittel und Alter einen Einfluss darauf haben, ob ein PKW oder der ÖPNV für alltägliche Wege genutzt wird. Auch Informationen über die Lage von Wohn- und Arbeitsort bieten Aufschlüsse auf die täglich zurückgelegten Wege. In diesen Zusammenhang fallen auch wirtschaftliche Rahmenbedingungen wie Ticketpreise und Verfügbarkeiten von Ermäßigungen und politische Rahmenbedingungen wie die Subventionspolitik oder steuerliche Anreize zur Nutzung bestimmter Verkehrsmittel.

Innerhalb dieser umfassenden Definition und den verschiedenen Datenarten konzentriert sich die vorliegende Arbeit auf solche Informationen, die sich insbesondere für die Visualisierung und Analyse im Kontext von 3D-Stadtmodellen eignen. Die genauen Anforderungen an die Daten ergeben sich aus dem Anwendungsbeispiel, das in Abschnitt 4 beschrieben wird.

2.2 3D-Stadtmodelle

3D-Stadtmodelle sind digitale Abbildungen urbaner Gebiete. Sie beinhalten als komplexe Datenstrukturen geografische und geometrische Informationen zur Lage und Form von Objekten im städtischen Raum, dazu zählen unter anderem Gebäude, Straßen, Gewässer, Schienenwege und Brücken. Seit längerem umfassen sie neben der reinen Darstellung von Objekten auch weitere „ontologische Strukturen wie thematische Klassen, Attribute und deren Beziehungen zueinander" (vgl. Kolbe 2009, S.1-2), und werden deshalb als *semantische* 3D-Stadtmodelle bezeichnet. Aufgrund dieser Vielzahl an Informationen liefern semantische 3D-Stadtmodelle wertvolle Geodaten und fallen somit in die in Abschnitt 2.1 genannte Definition der mobilitätsrelevanten Daten. 3D-Stadtmodelle bilden die Grundlage für urbane digitale Zwillinge[4]. Ein solcher „digitaler Zwilling einer Stadt ist ein dynamisches digitales Abbild, welches sich auf die Ganzheit oder einen anwendungsfallspezifischen Teil einer Stadt bezieht." (Brandt et al. 2023, S. 10). Dabei wird diskutiert, ab wann ein 3D-Stadtmodell als digitaler Zwilling bezeichnet werden kann und welche zusätzlichen Daten erforderlich sind, um dieses Potenzial zu erfüllen.

3D-Stadtmodelle liegen in unterschiedlichen Detaillierungsgraden vor, welche durch die *Levels of Detail* (LODs) beschrieben werden. Diese reichen von stark abstrahierten bis hin zu sehr detaillierten Modellen. Die unterschiedlichen LODs und deren Eigenschaften sind in Tabelle 1 beschrieben (vgl. Kolbe et al. 2021, S. 3-4). Abbildung 1 stellt beispielhaft die verschiedenen LODs dar, wobei die Unterschiede nochmals genauer erkannt werden können.

LOD	Beschreibung
LOD0	stark abstrahiertes Modell
LOD1	Blockmodell/ Extrusionsobjekt
LOD2	realistisches, jedoch immer noch verallgemeinertes Modell
LOD3	sehr detailliertes Modell

Tabelle 1: Levels of Detail in 3D-Stadtmodellen

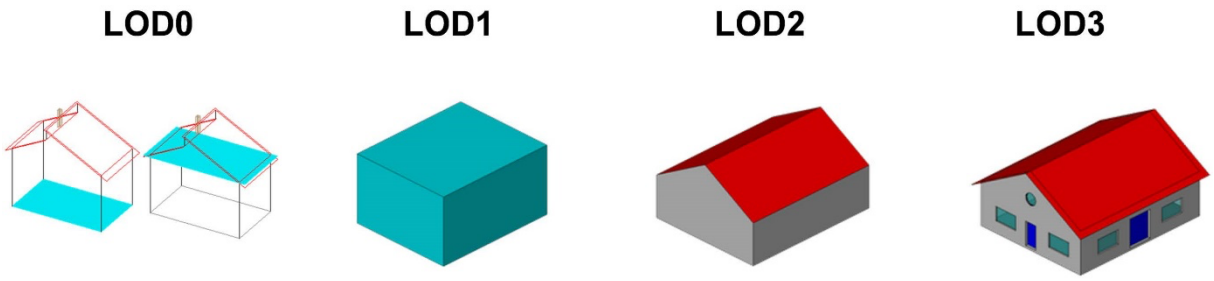


Abbildung 1: Darstellung eines Gebäudes in den LODs 0-3 (Kolbe et al. 2021, S. 36)

Die Daten zur Erstellung von 3D-Stadtmodellen werden durch verschiedene Methoden, beispielsweise Photogrammetrie oder Laser-Scan-Verfahren gesammelt. Das am häufigsten verwendete Datenformat für diese Modelle ist CityGML, ein auf XML basierendes Format, welches die Stadtobjekte in Typen definiert und ihre Geometrie beschreibt. Eine genauere Erklärung des Datenformats ist in Abschnitt 2.4.1 zu finden (vgl. Kolbe 2009, S. 1-2; vgl. Yao et al. 2016, S. 77; vgl. Biljecki et al. 2015, S. 2843).

[4] Digitale Zwillinge finden sich beispielsweise auch als Abbildungen von Maschinen in der Fertigungsindustrie und als digitale Repräsentationen einzelner Gebäude. In dieser Arbeit steht jedoch der Kontext digitaler Zwillinge im Fokus, die sich auf die 3D-Nachbildung von Städten oder Stadtteilen beziehen.

2.3 Anwendungsgebiete von mobilitätsrelevanten Daten und 3D-Stadtmodellen

Mobilitätsrelevante Daten sind aufgrund ihrer großen Vielfalt an Inhalten und Datentypen essentiell für zahlreiche Anwendungsbereiche. Sie spielen eine zentrale Rolle bei statistischen Auswertungen, in der intelligenten Stadt- und Verkehrsplanung, für eine wettbewerbsfähige Logistik, bei der Verringerung von Umweltauswirkungen sowie im Tourismus. Darüber hinaus unterstützen sie eine nachhaltige Entscheidungsfindung. So können beispielsweise Umfragedaten zu genutzten Verkehrsmitteln mit örtlichen Gegebenheiten verknüpft werden, um Potenziale zur Vermeidung von hohem Verkehrsaufkommen zu identifizieren. Auch die visuelle Aufbereitung dieser Daten, etwa in Form von Illustrationen oder Infografiken, bietet vielseitige Einsatzmöglichkeiten. Solche Grafiken können sowohl zur allgemeinen Wissensvermittlung als auch zur Schaffung von Akzeptanz in der Bevölkerung für städtebauliche Veränderungen genutzt werden. Darüber hinaus eignen sie sich, um Planungskonzepte anhand visueller Merkmale zu bewerten (vgl. Städtetag 2017, S. 6-10; vgl. Biljecki et al. 2015, S. 2851; vgl. EU).

Ein weiterer Anwendungsbereich für mobilitätsrelevante Daten liegt in der Nutzung persuasiver Technologien. Mithilfe von Gamification oder Framing können Menschen so beispielsweise dazu ermutigt werden, ein nachhaltigeres Verkehrsverhalten anzunehmen. Standortabhängig könnten beispielsweise nachhaltige Fortbewegungsmittel beworben oder Punkte für ökologisches Mobilitätsverhalten vergeben werden (vgl. Raunig & Hodzic-Srndic 2020, S. 1266-1267). Auch im Kontext von Smart Cities spielen mobilitätsrelevante Daten eine bedeutende Rolle, insbesondere in den Bereichen „smarte Stadtplanung“ und „Smart Transport/Mobilität“. Solche Daten werden von Nutzer*innen aus verschiedenen Bereichen wie Verwaltung, Politik oder der Bevölkerung eingesetzt, um Entscheidungen datenbasiert und effizienter zu gestalten (vgl. BSI).

Durch eine visuelle Aufarbeitung mobilitätsrelevanter Daten im dreidimensionalen Raum eröffnen sich weitere Nutzungspotenziale. So können sie genutzt werden, um Veränderungen zu dokumentieren und die Auswirkungen geplanter Maßnahmen zu simulieren.

Ein Praxisbeispiel ist das 3D-Stadtmodell „Energie, Umwelt & Klima“ (Abbildung 2) der Stadt Leipzig. Dieses Modell visualisiert verschiedene Umweltfaktoren, bewertet die Eignung von Hausdächern für die Installation von Solar- und Photovoltaikanlagen, stellt die Lärmkartierung dar und gibt Einblicke in die aktuelle Luftqualität. Es zeigt exemplarisch, wie 3D-Stadtmodelle zur Wissenskommunikation und Sensibilisierung genutzt werden können. So können Gebäudeeigentümer*innen das Modell beispielsweise nutzen, um zu prüfen, wie gut ihr Gebäude für die Nutzung nachhaltiger Energiequellen geeignet ist.

Im Bereich der Katastrophenprävention bieten 3D-Stadtmodelle einen direkten Mehrwert, etwa durch die Möglichkeit, Risikoanalysen durchzuführen und Präventionsmaßnahmen anschaulich darzustellen. Im Tourismus und der Bürger*innenpartizipation lohnt sich der Einsatz von 3D-Stadtmodellen besonders aufgrund des hohen Wiedererkennungseffektes der Stadt im dreidimensionalen Raum (vgl. deutscher_staedtetag_3d_geodaten, S. 5-6). Viele Städte nutzen bereits 3D-Modelle von Städten in Touristenattraktionen, beispielsweise um historische Veränderungen darzustellen wie TimeRide in Berlin (vgl. Visit Berlin).

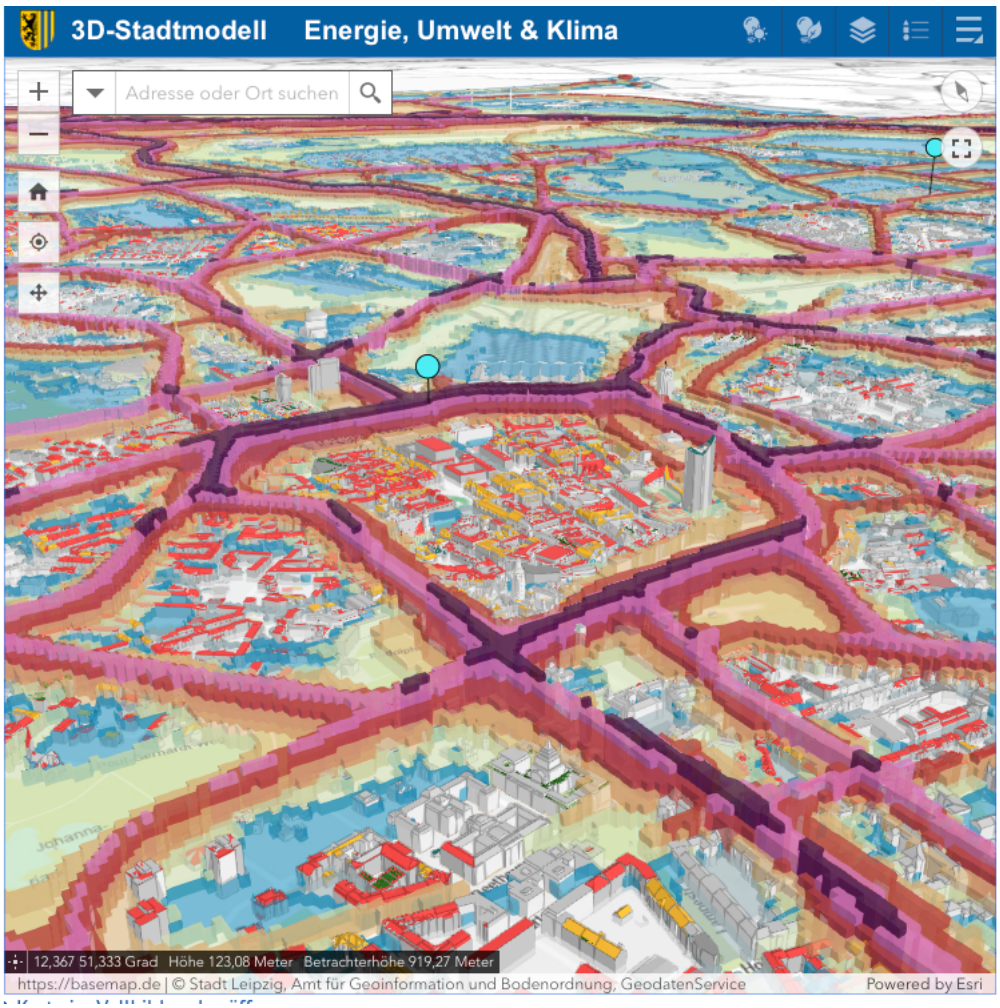


Abbildung 2: 3D-Stadtmodell "Energie, Umwelt & Klima" der Stadt Leipzig. (Leipzig 3D Energie)

2.4 Datenformate

Im Rahmen dieser Arbeit sollen mobilitätsrelevante Daten in Verbindung mit 3D-Stadtmodellen visuell aufbereitet werden. Dabei bieten sich insbesondere Geodaten und infrastrukturelle Informationen als Datengrundlage an. Soziodemografische, politische und wirtschaftliche Daten bleiben, wie in später in Abschnitt 4 erläutert wird, unberücksichtigt.

3D-Stadtmodelle sind häufig im CityGML-Format verfügbar, das eine standardisierte Grundlage für die Darstellung urbaner Räume bietet. Für infrastrukturelle Gegebenheiten stellt OSM aufgrund seiner Offenheit und Vielfältigkeit eine zentrale Datenquelle dar. Haltestellen- und Fahrplandaten können über das GTFS-Format genutzt werden. Dieses ermöglicht auch die Darstellung von Routen- und Mobilitätsnetz-Informationen.

Die genannten Datenformate und ihre spezifischen Eigenschaften werden in diesem Abschnitt detailliert beschrieben. Zusätzlich werden die beiden Game Engines Unity und Unreal Engine miteinander verglichen, und die Entscheidung für die Nutzung von Unity im Rahmen dieser Arbeit wird fundiert begründet.

2.4.1 CityGML

Das Format CityGML (und seine im Jahr 2021 vorgestellte Version 3.0) ist ein durch das Open Geospatial Consortium (kurz OGC) veröffentlichter, internationaler Standard für die Darstellung und

den Austausch von 3D-Stadtmodellen (Kolbe 2009, S. 3; Kolbe et al. 2021, S. 25-27). Es handelt sich dabei um ein XML-basiertes Format, welches für eine Vielzahl unterschiedlicher Anforderungen anpassbar ist.

CityGML-Dateien können aus unterschiedlichen Quellen bezogen werden, beispielsweise über Online-Portale von Städten und Gemeinden. Eine detaillierte Beschreibung der relevanten Datenquellen erfolgt später in Abschnitt 5.1, im Rahmen des ersten Schritts des Workflows zur Beschaffung des 3D-Stadtmodells. CityGML wird als Format nur von wenigen kostenlosen bzw. Open-Source-Softwaretools nativ unterstützt. Dies könnte unter anderem an der Komplexität des Formats liegen, das sowohl geometrische als auch semantische Informationen integriert.

Der Aufbau des CityGML-Formats wird im *CityGML Conceptual Model* (kurz CM) beschrieben, das festlegt, wie reale Objekte in einer CityGML-Datei strukturiert und klassifiziert werden. Das CityGML CM ist in ein *Core-Module* und mehrere *Extension-Modules* gegliedert (siehe Abbildung 3). Das Core-Module umfasst die grundlegenden Konzepte und Komponenten des Modells und ist daher für jedes System vorgeschrieben, das CityGML nutzt. Nicht alle Extension-Modules müssen verwendet werden, dabei ist es möglich, nur spezifische Module auszuwählen.

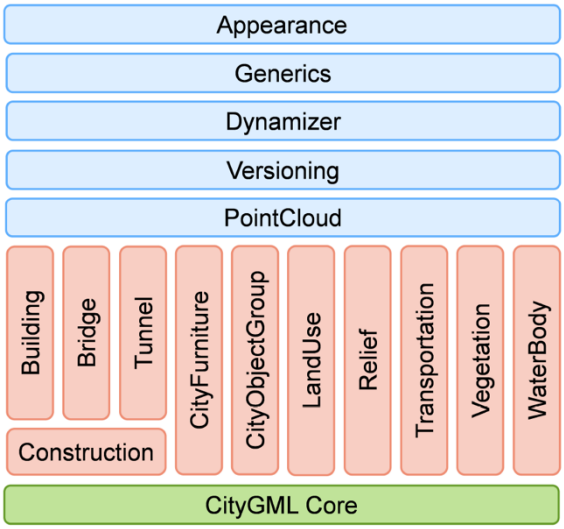


Abbildung 3: Modularisierung des CityGML CM (Kolbe et al. 2021, S. 26)

Die thematischen Extension-Modules (rot in der Abbildung) repräsentieren spezifische Themenbereiche virtueller 3D-Stadtmodelle. Ergänzend dazu gibt es fünf weitere Module (blau in der Abbildung), die Modellierungsaspekte beschreiben. Das Modul *Appearance* definiert das Aussehen von Objekten, wie Texturen oder Farben, während *PointCloud* die Geometrie von Objekten durch Punktwolken abbildet. *Generics* erlaubt die Definition generischer Objekte, Attribute und Beziehungen, die nicht von anderen Modulen erfasst werden. Mit *Versioning* können sowohl unterschiedliche Versionen eines Objekts als auch dessen Historie dargestellt werden. Das Modul *Dynamizer* ermöglicht die Verknüpfung von Sensordaten, um zeitliche Veränderungen oder dynamische Prozesse abzubilden (Kolbe et al. 2021, S. 25-27).

2.4.2 OSM

OpenStreetMap ist eine offene Geodatenbank, die von einer globalen Community im Rahmen eines Crowdsourcing-Projekts gepflegt wird (vgl. Keler 2016, S. 115). Die Plattform stellt Daten unter einer freien Lizenz zur Verfügung, wodurch sie sowohl für wissenschaftliche als auch praktische Anwendungen vielfältige Einsatzmöglichkeiten bietet. Mit über einer Million aktiven Mitwirkenden

weltweit hat sich OSM als zentrale Quelle für geografische Informationen etabliert (vgl. OpenStreetMap FAQ).

Der Zugriff auf OSM-Daten erfolgt über eine Vielzahl von Schnittstellen. Besonders hervorzuheben sind die Overpass API und Overpass Turbo, die für detaillierte Abfragen und die Analyse spezifischer Geodaten entwickelt wurden (vgl. OpenStreetMap WikiAPIs). Die Nutzung dieser Schnittstellen ermöglicht es, gezielt nach relevanten Daten zu suchen und diese für die Integration in eigene Anwendungen oder Systeme aufzubereiten.

Die strukturelle Grundlage von OSM-Daten wird im *Conceptual Data Model* (kurz CDM) beschrieben. Dieses Modell definiert die drei Grundelemente von OSM, *Nodes*, *Ways* und *Relations*:

- *Nodes* beschreiben Punkte mit Hilfe ihrer Koordinaten, einem Punkt wird mindestens eine ID zugeordnet. Nodes können alleinstehende geografische Punkte/ Orte oder Teil einer Relation sein. Außerdem können sie Punkte auf einem Way darstellen.
- *Ways* sind eine geordnete Liste aus mindestens einem bis maximal 2000 Punkten, welche eine Linie beschreiben. Sie können auch *Areas* umschließen.
- *Relations* beschreiben Verbindungen zwischen mindestens zwei Elementen, wobei die Teile der Relation als *Members* bezeichnet werden und eine bestimmte festgelegte Rolle übernehmen.

Diesen Grundelementen können mit Hilfe von *Tags* Eigenschaften zugewiesen werden. Ein solches Tag ist immer ein Key-Value-Pair (vgl. OpenStreetMap Elements).

Zur Nutzung und Weiterverarbeitung der Daten stellt OSM umfassende technische Ressourcen bereit. Neben detaillierter Dokumentation auf der Plattform bietet die Community zahlreiche Software-Bibliotheken, die den Zugriff auf die Daten erleichtern und die Entwicklung von Anwendungen unterstützen. So gibt es Online-Map-Editoren, Verarbeitungssoftware und Importer für Datenbanken (DBs) wie PostGIS-Datenbanken. Zu den unterstützten Programmiersprachen gehören unter anderem Python, C++, JavaScript und Java, wodurch sich OSM-Daten einfach in bestehende Systeme und Workflows integrieren lassen (vgl. OpenStreetMap Develop).

Ergänzend dazu stehen verschiedene Ressourcen zur Verfügung, um die Struktur und Nutzung der Daten zu verstehen. Besonders relevant sind das OSM-Wiki (vgl. OSM Wiki) und Taginfo (vgl. TagInfo). Während das Wiki grundlegende Informationen und Beschreibungen zu OSM-Daten und einzelnen, häufig verwendeten Tags bietet, stellt Taginfo eine umfassende Übersicht über alle in OSM verwendeten *Tags*, *Keys* und *Values* bereit. Dort finden sich zudem Angaben zur Häufigkeit der Nutzung sowie zu möglichen Kombinationen von *Tags*. Diese Informationen machen Taginfo zu einem essentiellen Werkzeug für die strukturierte Vorbereitung und gezielte Nutzung von OSM-Daten. Darüber hinaus sind, sofern verfügbar, relevante Wiki-Seiten direkt in Taginfo verlinkt, was die Recherche zusätzlich erleichtert.

Die in OSM verfügbaren Daten und Werkzeuge werden im Rahmen dieser Arbeit insbesondere für die Integration von Geodaten in 3D-Stadtmodelle genutzt. Eine detaillierte Beschreibung der entsprechenden Workflows erfolgt in den späteren Kapiteln.

2.4.3 GTFS

Die General Transit Feed Specification ist ein offener Standard, der Informationen zu öffentlichen Verkehrsmitteln bereitstellt. Er ermöglicht es Verkehrsunternehmen, Planungs- und Echtzeitdaten strukturiert aufzubereiten, sodass diese sowohl für Fahrgäste als auch für Entwickler*innen von Mobilitätsanwendungen zugänglich sind. GTFS gliedert sich in zwei Hauptkomponenten: *GTFS Schedule* und *GTFS Realtime*.

GTFS Schedule beschreibt als CSV-basiertes Format die statischen ÖPNV-Informationen und besteht aus einer Sammlung einfacher Textdateien. Ein Standarddatensatz umfasst mindestens sieben Dateien, die Informationen zu Verkehrsbetrieben (*agency.txt*), Routen (*routes.txt*), Verbindungen (*trips.txt*), Haltestellen (*stops.txt* und *stop_times.txt*), Kalenderdaten (*calendar.txt* und *calendar_dates.txt*) sowie zu weiteren planungsbezogenen Aspekten enthalten. Zusätzliche optionale Dateien können beispielsweise Preise oder Transfers beschreiben. Die Verknüpfung der Dateien erfolgt über eindeutige IDs, wodurch eine konsistente und flexible Datenstruktur gegeben ist.

GTFS Realtime erlaubt es Verkehrsunternehmen, dynamische Echtzeitdaten bereitzustellen. Hierzu zählen Fahrplanaktualisierungen wie geänderte Abfahrtszeiten oder Streckenverläufe, die Übermittlung von Fahrzeugpositionen oder Meldungen zu Verbindungsausfällen (vgl. GTFS Overview).

Auf der Website des GTFS-Standards wird ein Überblick über verschiedene Portale zur Suche nach GTFS-Daten bereitgestellt. Dabei stehen sowohl Crowdsourced-Daten als auch Daten, die direkt von Verkehrsunternehmen bereitgestellt werden, zur Verfügung (vgl. GTFS Resources).

2.5 Datenqualität

Verfügbare Daten sind nicht immer von hoher Qualität. Fehlende, veraltete oder komplizierte Datensätze können die Verarbeitung erschweren oder, abhängig vom Nutzungskontext, sogar zu falschen Ergebnissen führen. Daher ist es entscheidend, die Qualität von verwendeten Daten zu bewerten, um potenzielle Probleme zu vermeiden und Ergebnisse angemessen interpretieren zu können.

Dieser Abschnitt behandelt zunächst die Dimensionen der Datenqualität im Allgemeinen, bevor er sich den drei für diese Arbeit zentralen Datenformaten widmet. Die Datenqualität dieser Formate wird anhand der zuvor beschriebenen Einflussfaktoren bewertet. Anschließend werden typische Qualitätsprobleme und mögliche Lösungen anhand relevanter Literatur aufgezeigt.

2.5.1 Dimensionen und Einflussfaktoren

Eine etablierte Grundlagenquelle zur Datenqualität stammt von Wang & Strong (Wang & Strong 1996). Die Aspekte der Datenqualität können demnach in vier Dimensionen unterteilt werden, wie in Abbildung 4 zu sehen ist. Die *intrinsische Datenqualität* beschreibt den Fakt, dass *Accuracy* (Korrektheit) und *Objectivity* (Objektivität) nicht ausreichen, damit Daten als qualitativ hochwertig angesehen werden. *Believability* (Glaubwürdigkeit) und *Reputation* (Ansehen) sind ebenso wichtig. Die *kontextbezogene Datenqualität* beschreibt den Fakt, dass Datenqualität auch immer im Kontext der Datennutzung betrachtet werden muss. Die dabei relevanten Aspekte der Datenqualität sind *Value-added* (Mehrwert), *Relevancy* (Relevanz), *Timeliness* (Aktualität), *Completeness* (Vollständigkeit) und *Appropriate amount of Data* (angemessene Datenmenge). Die Dimension der *repräsentativen Datenqualität* bezieht sich auf das Format und die Bedeutung der Daten. Damit Daten als repräsentativ erachtet werden müssen sie die Eigenschaften *Concise Representation* (Prägnanz) und *Representational Consistency* (Konsistenz) aufweisen, gleichzeitig jedoch auch *Interpretability* (Interpretierbarkeit) und *Ease of Understanding* (leichte Verständlichkeit). Die Dimension der *Zugangsqualität* beschreibt, dass Daten eine hohe Qualität aufweisen, wenn *Accessibility* (Zugänglichkeit) und *Access Security* (Zugriffssicherheit) gegeben sind (vgl. Wang & Strong 1996, S. 20-21)

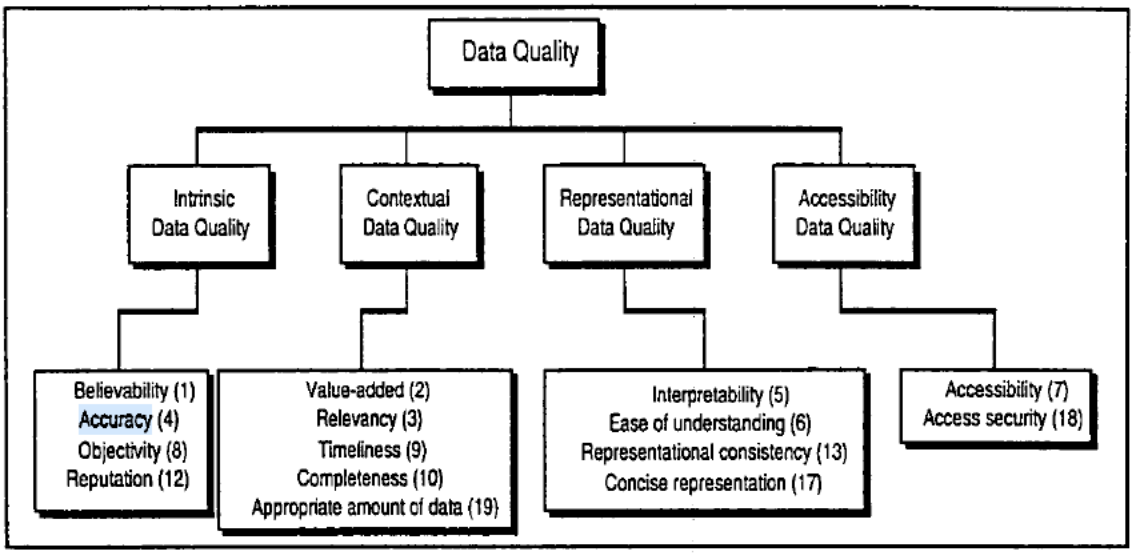


Abbildung 4: Dimensionen der Datenqualität (Wang & Strong, S. 20)

In Tabelle 2 werden, basierend auf (Wang & Strong 1996, S. 31–32), die relevanten Eigenschaften der Datenqualität mit einer kurzen Erklärung aufgeführt. Diese Eigenschaften werden im Folgenden auf Datenformate übertragen, um ihre Qualität zu bewerten. Ursprünglich ist das Modell darauf ausgerichtet, konkrete Datensätze aus der Perspektive der Nutzer*innen zu analysieren. Die Anwendung auf Datenformate stellt daher eine Abwandlung dar, bietet jedoch eine hilfreiche Orientierung, um die Qualität dieser Formate einzuschätzen.

Dimension	**Eigenschaft**	**Beschreibung**
intrinsische Datenqualität	Believability	das Ausmaß, in dem Daten als wahr, real und glaubwürdig akzeptiert oder angesehen werden.
	Accuracy	das Ausmaß, in dem die Daten korrekt, zuverlässig und als fehlerfrei zertifiziert sind.
	Objectivity	das Ausmaß, in dem die Daten unvoreingenommen und unparteiisch sind.
	Reputation	das Ausmaß, in dem Daten in Bezug auf ihre Quelle oder ihren Inhalt geschätzt oder hoch angesehen werden.
kontextbezogene Datenqualität	Value-added	das Ausmaß, in dem Daten nützlich sind und Vorteile aus ihrer Verwendung entstehen.
	Relevancy	das Ausmaß, in dem die Daten für die jeweilige Aufgabe anwendbar und hilfreich sind.
	Timeliness	das Ausmaß, inwieweit das Alter der Daten für die jeweilige Aufgabe angemessen ist.
	Completeness	das Ausmaß, in dem die Daten in Bezug auf Breite, Tiefe und Umfang für die jeweilige Aufgabe ausreichen.
	Appropriate Amount of Data	das Ausmaß, inwieweit die Menge oder der Umfang der verfügbaren Daten angemessen ist.
repräsentative Datenqualität	Interpretability	das Ausmaß, in dem die Daten in einer angemessenen Sprache und Einheit vorliegen und die Datendefinitionen klar sind.

	Ease of Understanding	das Ausmaß, in dem Daten klar und eindeutig und leicht verständlich sind.
	Representational Consistency	das Ausmaß, in dem die Daten immer im gleichen Format dargestellt werden und mit früheren Daten kompatibel sind.
	Concise Representation	das Ausmaß, in dem die Daten kompakt dargestellt werden, ohne überwältigend zu sein (d.h. kurz, aber vollständig und auf den Punkt gebracht).
Zugangsqualität	Accessibility	das Ausmaß, in dem Daten verfügbar oder leicht und schnell abrufbar sind.
	Access Security	das Ausmaß, in dem der Zugang zu den Daten eingeschränkt und damit sicher gehalten werden kann.

Tabelle 2: Eigenschaften der Datenqualität nach (Wang & Strong 1996, S. 31-32)

2.5.2 3D-Stadtmodelle im Format CityGML

Ein zentraler Aspekt bei der Nutzung von 3D-Stadtmodellen im Format CityGML ist die Sicherstellung einer hohen Datenqualität. Um dies zu gewährleisten, bietet das unabhängige Netzwerk der 3D-Stadtmodell-Experten „SIG3D" ein Modellierungshandbuch an, das die praktische Anwendung des CityGML-Standards unterstützt und großen Wert auf die Qualitätssicherung legt. Ergänzend dazu arbeitet das Netzwerk an einem Testhandbuch, das diesen Aspekt weiter vertiefen soll (SIG3D). Eine Möglichkeit, die 3D-Daten zu visualisieren und somit auf die Qualität der dargestellten Objekte zu prüfen ist SimStadt (vgl. SimStadt). Diese Anwendung steht jedoch nur auf Windows und Linux zur Verfügung.

Die intrinsische Datenqualität von 3D-Stadtmodellen im CityGML-Format ist generell eher hoch. Da diese Daten häufig von öffentlichen Stellen erstellt und verbreitet werden, können sie als glaubwürdig und vertrauenswürdig eingestuft werden, was eine hohe Believability und Reputation bedeutet. Das CityGML-Format ist ein international anerkannter Standard, der ebenfalls zur positiven Wahrnehmung beiträgt. Die Accuracy der Daten hängt jedoch stark von der Methode ab, mit der sie erfasst wurden. Da 3D-Stadtmodelle physische Gegebenheiten abbilden, sind sie wertungsfrei und unvoreingenommen, was eine hohe Objectivity gewährleistet.

Die kontextbezogene Datenqualität ist – wie der Name schon sagt – stark abhängig vom jeweiligen Anwendungsfall und Datensatz. CityGML bietet zwar ein solides Fundament für eine hohe Qualität, aber die Flexibilität des Formats bedeutet, dass dieses Potenzial nicht immer ausgeschöpft wird. 3D-Stadtmodelle haben durch ihre breite Anwendbarkeit einen hohen Value-added, wie bereits in Abschnitt 2.5.2 erläutert. Der CityGML-Standard erleichtert die Strukturierung und Weitergabe der Daten, wodurch zusätzliche Vorteile entstehen. Die Relevancy der Daten hängt von der jeweiligen Aufgabe und Aufbereitung ab. CityGML-Daten können für viele verschiedene Anwendungen nützlich sein, aber ihre Relevanz ist immer kontextspezifisch. Die Timeliness und Completeness müssen für jeden Datensatz individuell bewertet werden. Da die Erstellung von 3D-Stadtmodellen komplex und aufwändig ist, werden die Daten oft nur selten aktualisiert, was sich auf die Aktualität auswirken kann. Zudem erlaubt der CityGML-Standard eine gewisse Flexibilität bei der Ausgestaltung und Datenmenge. Das bedeutet, dass die Bewertung der Eigenschaft Appropriate Amount of Data stark von der jeweiligen Nutzung abhängt. Für manche Aufgaben könnten CityGML-Daten zu detailliert sein, während sie für andere Anwendungen nicht ausreichend detailliert erscheinen.

Die repräsentative Datenqualität ist bei 3D-Stadtmodellen im CityGML-Format mittel bis hoch einzuschätzen. Die Interpretability ist durch die Standardisierung des Formats gegeben. Die Datendefinitionen sind klar, und die Daten können – abhängig von der Ausgestaltung – sowohl von Maschinen als auch von Menschen verarbeitet werden. Reine Koordinaten beispielsweise sind eher maschinenlesbar, was dem Zweck des Formats entspricht. Die umfangreiche Dokumentation des

CityGML-Standards sorgt dafür, dass die Daten insgesamt leicht verständlich sind, was eine hohe Ease of Understanding bedeutet. Die Representational Consistency wird ebenfalls durch die Standardisierung sichergestellt, da die Daten immer im gleichen Format vorliegen. Im Gegensatz dazu ist die Concise Representation nicht durch das Datenformat vorgegeben und hängt stark von der konkreten Nutzung ab.

Die Zugangsqualität von CityGML-Daten ist insgesamt hoch. Die Daten sind unter verschiedenen Lizenzmodellen verfügbar, darunter finden sich offene Lizenzen sowie kostenpflichtige oder eingeschränkte Optionen. Die Accessibility ist gegeben, da die Daten online bereitgestellt und heruntergeladen werden können. Gleichzeitig wird die Access security gewährleistet, da nicht-öffentliche Daten durch Zugriffsbeschränkungen geschützt werden.

Zusammenfassend lässt sich sagen, dass CityGML-Daten durch ihre Standardisierung und Flexibilität eine solide Grundlage für zahlreiche Anwendungen bieten, jedoch von der Qualität und Aktualität der zugrunde liegenden Datensätze abhängen.

2.5.3 OSM

Der Crowdsourcing-Ansatz von OpenStreetMap (OSM) führt in vielen Fällen zu einer sehr hohen Datenqualität. Laut dem OSM-Wiki ist OpenStreetMap „often more up-to-date and of a higher quality than other commercial maps when dealing with New and Changed Ways“ (OpenStreetMap Quality). Gleichzeitig zeigen Studien, dass die Qualität der OSM-Daten stark vom geografischen Gebiet abhängt und von sehr vollständig bis sehr unvollständig reichen kann. Die Vollständigkeit der Daten ist weltweit unterschiedlich: Während in einigen Regionen über 80 % der Gebäudeattribute wie Stockwerkanzahl oder Gebäudehöhe vorhanden sind, fehlen in anderen Gebieten fast alle Informationen. Numerische Attribute wie Gebäudehöhe sind häufig konsistent, während textbasierte Attribute, wie Material oder Dachform, aufgrund uneinheitlicher Tags und subjektiver Interpretationen weniger konsistent sind. Daher wird empfohlen, die Daten vor der Nutzung auf ihre Qualität zu prüfen (vgl. Biljecki et al. 2015; vgl. Schrapp et al. 2019). Ein mögliches Werkzeug dafür ist das Tool „ohsome“, das von der BKG entwickelt wurde (vgl. ohsome). Aktuell ist es jedoch noch im Demo-Status und deckt nur wenige Gebiete ab. In diesen Prüftools wird die Datenqualität anhand verschiedener Kriterien wie zeitlicher Verlauf, Vollständigkeit, Aktualität und Erreichbarkeitsanalysen bewertet.

Die intrinsische Datenqualität von OSM ist überwiegend als hoch einzuschätzen. Der Crowdsourcing-Ansatz sorgt dafür, dass die Daten grundsätzlich glaubwürdig sind, da sie von einer großen und aktiven Community erstellt und überprüft werden. Dieser Ansatz birgt zwar das Risiko von Nutzerfehlern, jedoch wird die Believability durch die gegenseitige Kontrolle innerhalb der Community gestärkt. Die Accuracy der Daten ist stark von den individuellen Fähigkeiten und der Sorgfalt der mappenden Personen abhängig. Eine zusätzliche Überprüfung durch den Abgleich mit anderen Datensätzen kann hilfreich sein, um die Genauigkeit zu validieren. Die Daten von OSM beschreiben reale Gegebenheiten und sind daher in der Regel objektiv und unparteiisch. Fehlinterpretationen, etwa bei der Gebäudenutzung, können jedoch vorkommen, weshalb die Objectivity zwar hoch, aber nicht uneingeschränkt gegeben ist. Insgesamt hat OSM einen guten Ruf, was auf die hohe Verfügbarkeit von Daten und die ständige Kontrolle durch die Community zurückzuführen ist. Dies sorgt für eine hohe Reputation.

Die kontextbezogene Datenqualität von OSM-Daten ist stark anwendungs- und aufgabenspezifisch. Aufgrund der Vielzahl an verfügbaren Informationen und der freien Verfügbarkeit haben die Daten einen hohen Value-added und finden in unterschiedlichsten Szenarien Anwendung. Die Relevancy ist abhängig von der konkreten Aufgabe und der Aufbereitung der Daten, da OSM-Daten eine breite Grundlage für zahlreiche Anwendungen bieten, jedoch nicht immer direkt auf spezifische Anforderungen zugeschnitten sind. Die Timeliness variiert je nach Datensatz und geplanter Nutzung, da die Aktualisierung der Daten stark davon abhängt, wann und wie sie von der Community gemappt werden. Ebenso ist die Completeness der Daten von der Aktivität der Nutzer*innen abhängig. Da die

Daten nicht zentral erfasst werden, sondern von einzelnen Usern, besteht die Möglichkeit, dass nicht alle relevanten Informationen vollständig gemappt sind. Die Appropriate Amount of Data hängt ebenfalls von der Aufgabe ab. Für manche Anwendungen kann OSM zu viele Daten liefern, während für andere eventuell relevante Informationen fehlen.

Die repräsentative Datenqualität von OSM-Daten wird durch verschiedene Faktoren beeinflusst. Die Interpretability der Daten ist durch die klare Definition der Tags grundsätzlich gegeben. Allerdings existieren keine festen Standards, sodass manche Eigenschaften durch unterschiedliche Tags beschrieben werden können. Tools wie das OSM-Wiki oder Taginfo erleichtern es jedoch, häufig genutzte Tags zu identifizieren, was die Interpretierbarkeit fördert. Die Daten sind sowohl maschinen- als auch menschenlesbar, wobei dies von der Weiterverarbeitung abhängt. Die Ease of Understanding ist ebenfalls hoch, da OSM-Tags in einer klaren Sprache verfasst sind und durch umfangreiche Dokumentation im OSM-Wiki oder in Taginfo ergänzt werden. Die Representational Consistency wird durch das flexible Datenmodell von OSM gewährleistet, da Eigenschaften problemlos hinzugefügt werden können, ohne bestehende Daten zu beeinträchtigen. Im Gegensatz dazu ist die Concise Representation nicht immer gegeben und hängt von der Nutzung der Daten ab. Die repräsentative Datenqualität kann als mittel bis hoch eingeschätzt werden.

Die Zugangsqualität von OSM-Daten ist als sehr hoch einzustufen. Die Daten sind unter einer freien Lizenz verfügbar und können direkt über die OSM-Karte visualisiert oder über verschiedene APIs abgerufen werden. Während einfache Abfragen mit Tools wie Overpass Turbo leicht umsetzbar sind, erfordert der Zugriff auf umfangreichere Datenmengen mitunter spezifischere technische Kenntnisse. Somit ist die Accessibility der Daten grundlegend gegeben. Da alle Daten öffentlich sind, wird die Access security ebenfalls gewährleistet.

2.5.4 GTFS

Die intrinsische Datenqualität von GTFS-Daten kann als hoch eingeschätzt werden. Die Believability ist in der Regel gegeben, da viele Informationen direkt von Verkehrsbetrieben bereitgestellt werden, die als zuverlässige Datenquellen gelten. Bei Crowdsourcing-Daten sorgt die Kontrolle durch die Community zwar für eine Reduzierung von Fehlern, birgt jedoch ein gewisses Risiko für Nutzerfehler oder Manipulation. Die Accuracy der Daten ist aufgrund der Qualität der Primärquellen meist hoch. GTFS-Daten beschreiben reale Gegebenheiten, wodurch sie in der Regel objektiv und unparteiisch sind, was eine hohe Objectivity sicherstellt. Die Reputation der GTFS-Daten ist hoch, da sie häufig direkt von Verkehrsbetrieben bereitgestellt werden, die als Primärquelle als zuverlässige und vertrauenswürdige Datenquellen gelten.

Die kontextbezogene Datenqualität von GTFS-Daten hängt stark vom spezifischen Anwendungsfall ab. Durch ihre strukturierte und umfassende Bereitstellung bieten GTFS-Daten einen erheblichen Mehrwert, da sie die Grundlage für Anwendungen wie Fahrgastinformationssysteme, Routenplaner oder Verkehrsanalysen bilden. Der Value-added ist also sehr hoch. Die Relevancy der Daten variiert je nach Nutzung. Im Bereich der Verkehrsplanung, Fahrgastinformation und Echtzeitverkehrsmanagement sind GTFS-Daten jedoch oft von zentraler Bedeutung. Die Timeliness hängt von der Art der GTFS-Daten ab: GTFS Realtime liefert aktuelle Echtzeitinformationen, während GTFS Schedule auf Fahrplaninformationen basiert, die in regelmäßigen Abständen aktualisiert werden. Für die Bewertung der Aktualität ist die geplante Nutzung entscheidend. Die Completeness ist bei Daten, die direkt von Verkehrsverbünden bereitgestellt werden, in der Regel hoch. Bei Crowdsourcing-Daten kann die Vollständigkeit variieren, weshalb die Quelle der Daten überprüft werden sollte. Die Qualität gemäß der Eigenschaft Appropriate Amount of Data ist bei GTFS-Daten meist gegeben, da sie relevante Informationen enthalten, ohne zu umfangreich oder zu knapp zu sein.

Die repräsentative Datenqualität von GTFS-Daten ist insgesamt hoch. Die Interpretability ist durch die klare Definition der Datenelemente im GTFS-Standard gewährleistet. Beziehungen zwischen den einzelnen Dateien des Formats sind eindeutig und logisch aufgebaut. Die Ease of Understanding ist

ebenfalls gegeben, da die Daten in klarer Sprache verfasst sind. Allerdings kann die Handhabung durch die Vielzahl an Dateien und deren Verknüpfung komplexer werden, da oft mehrere Dateien gleichzeitig betrachtet werden müssen. Die Representational Consistency ist durch die klare Trennung der Dateien und die standardisierten Inhalte gewährleistet. Die Concise Representation ist nicht durch das Datenformat an sich gegeben und hängt stark von der konkreten Nutzung der Daten ab.

Die Zugangsqualität von GTFS-Daten variiert je nach Lizenzmodell. Einige Daten sind unter freier Lizenz verfügbar, andere erfordern eine kostenpflichtige Lizenz. Die Accessibility ist durch die Bereitstellung der Daten über das Internet und APIs grundsätzlich hoch. Die Access security wird ebenfalls gewährleistet, da für die Öffentlichkeit bestimmte Daten entsprechend zugänglich gemacht werden, während nicht-öffentliche Daten geschützt bleiben.

3 Forschungsstand

Es findet sich zahlreiche Literatur zu Einflussfaktoren für das Mobilitätsverhalten von Menschen. Betrachtet man die Ausführungen des Umweltbundesamtes, so findet man vermehrt Literatur, die Hinweise darauf gibt, wie nachhaltiges Mobilitätsverhalten gestärkt werden kann.

Buchholz & Flaig (2022) bieten eine detaillierte, aber übersichtliche Darstellung von Push- und Pull-Faktoren für nachhaltige Mobilität. Sie nennen zahlreiche Maßnahmen und ergänzen diese jeweils mit kurzen Begründungen oder Erklärungen, wobei die Maßnahmen nach Mobilitätsart (Fußverkehr, Radverkehr, ÖPNV und motorisierter Individualverkehr (MIV)) unterteilt werden. Scherhaufer et al. (2023) liefern aus dem europäischen Kontext, konkret aus Wien, eine stichpunktartige Liste möglicher Einflussfaktoren und Maßnahmen zur Förderung nachhaltiger Mobilität. Diese umfassen Ansätze wie Infrastruktur, Sensibilisierung und Aspekte der Gamification. Da die Maßnahmen auf den österreichischen Kontext zugeschnitten sind, ist nicht jede direkt auf Deutschland übertragbar, sodass zum Teil geprüft werden muss, ob vergleichbare Ansätze oder Rahmenbedingungen auch hierzulande existieren. Günther et al. (2023) fassen in ihrer Veröffentlichung 23 Maßnahmen aus einem Beispielprojekt in Chemnitz zusammen. Diese werden auf jeweils zwei Doppelseiten übersichtlich präsentiert, inklusive der Vorteile und Hintergründe der Maßnahmen. Schließlich liefern Rohs et al. (2023) in ihrem Abschlussbericht eine umfassende Darstellung verschiedener Maßnahmen. Sie gehen ausführlich auf Hintergründe, Erfolgsfaktoren, Hemmnisse sowie Praxisbeispiele ein und bieten damit eine besonders tiefgehende Betrachtung. Somit ist die Grundlage für die Spezifizierung der relevanten Faktoren für nachhaltiges Mobilitätsverhalten gegeben. Obwohl OSM weltweit für viele verschiedene Programme und Arbeiten von hoher Relevanz ist, fehlt in der Literatur jedoch ein Ansatz der Übertragung dieser Maßnahmen nach OSM. Diese Forschungslücke soll in der vorliegenden Arbeit geschlossen werden, indem die relevanten Daten strukturiert in OSM-Tags überführt werden, um somit eine automatische Abfrage über APIs möglich zu machen.

Der Standard CityGML ist ausführlich auf der Website des Open Geospatial Consortiums dokumentiert (vgl. OGC). Detailbeschreibungen einzelner Aspekte des Standards, wie den LODs oder der Visualisierung von Straßenzügen in OSM finden sich in verschiedenen Quellen (vgl. Biljecki et al. 2016, vgl. Beil & Kolbe 2018). OpenStreetMap und dessen API werden in einem eigenen Wiki ausführlich dokumentiert (vgl. OSM Wiki). Auch GTFS hat eine eigene technische Dokumentation, welche die Spezifikation ausführlich beschreibt (vgl. GTFS Overview).

Ledoux et al. (2019) erklären in ihrem Werk, wie CityGML in das Datenformat *CityJSON* umgewandelt werden kann. Dies kann die Nutzung der Daten vereinfachen, da es sich bei CityJSON um ein JSON-basiertes Format handelt. Für solche Daten gibt es in Python eine umfangreiche Unterstützung.

Weiler et al. (2018) beschreiben in ihrem Werk Möglichkeiten, CityGML- und OSM-Daten unter Nutzung des Open-Source-Tools *3DCityDB* zu verknüpfen. Dabei handelt es sich jedoch nur um eine Beschreibung, die praktische Umsetzung wurde in Ausblick gestellt. Dort wird auch auf die koordinatenbasierte Verknüpfung von OSM- und CityGML-Daten eingegangen. Die dort genannten Konzepte liefern hilfreiche Ansätze für diese Arbeit. Die Lücke, die dort durch die fehlende praktische Umsetzung entsteht, soll mit dieser Arbeit geschlossen werden.

Die von Weiler et al. verwendete 3DCityDB ist eine Datenbanklösung zur Nutzung von CityGML. Eine ausführliche Beschreibung dieser Schnittstelle findet sich auf ihrer Website (vgl. 3DCityDB) und in Yao et al. (2018). Dort werden mögliche Import- und Exportformate beschrieben. Es existieren wissenschaftliche Arbeiten, welche 3DCityDB nutzen, um Daten zu exportieren und in weitere Programme zur Weiterverarbeitung zu importieren, jedoch fehlt dort stets eine schrittgenaue Beschreibung. Einzelne Schritte und deren Hindernisse werden in Foren wie beispielsweise Stackoverflow oder der GitHub Community diskutiert. Diese Diskussionen können bei der

spezifischen Fehlerbehebung hilfreich sein, bieten jedoch keinen Mehrwert hinsichtlich eines konkreten Workflows.

Da eine spezifische Anleitung für einen solchen Workflow zur Verbindung von 3D-Stadtmodellen mit OSM- und GTFS-Daten fehlt, fehlt auch eine Darstellung möglicher Problemstellungen und Hürden. Diese werden in dieser Arbeit gesammelt und zusammengetragen.

4 Anwendungsbeispiel

Dieser Abschnitt beschäftigt sich mit dem Anwendungsbeispiel, das für die Umsetzung des Praxisteils genutzt wird. Zunächst wird das Anwendungsbeispiel allgemein beschrieben, bevor eine Zielgruppenanalyse durchgeführt wird.

Der Workflow wird anhand einer spezifischen Zielgruppe und deren Use-Case entwickelt, um die erforderlichen Schritte gezielt zu definieren und umzusetzen. Das Anwendungsbeispiel dient dabei als praktische Grundlage für die Entwicklung des Workflows. Dennoch ist der Workflow nicht ausschließlich auf diese Zielgruppe und den gewählten Use-Case beschränkt. Vielmehr wird er so konzipiert, dass er flexibel angepasst und in anderen Nutzungsbereichen und Anwendungen eingesetzt werden kann. So lassen sich beispielsweise alternative Daten aus OSM einbinden oder nach dem Unity-Import zusätzliche Verarbeitungsschritte integrieren, etwa zur 3D-Bearbeitung, zum Export oder zur Darstellung in immersiven Anwendungen. Es ist wichtig zu unterscheiden, dass der Use-Case für die Erstellung des Workflows ein anderer ist als der Use-Case für die Nutzung des fertigen Workflows.

4.1 Beschreibung und Use-Case

Die in dieser Arbeit formulierten Forschungsziele werden anhand des fiktiven Anwendungsbeispiels *„Nachhaltige Mobilität im Bereich der Leipziger Innenstadt“* verfolgt. Dieses speziell für die Arbeit entwickelte Anwendungsbeispiel zielt darauf ab, den Ist-Stand der mobilitätsrelevanten städtebaulichen Merkmale Leipzigs in einer immersiven Anwendung in Virtual Reality zu visualisieren. Die dafür nötige Datenvorverarbeitung wird in der vorliegenden Arbeit praktisch erprobt und durch Python-Skripte automatisiert. Eine solche Anwendung bietet je nach Ausgestaltung verschiedene mögliche Nutzungspotenziale. Für das fiktive Anwendungsbeispiel dieser Arbeit wurde die Anwendergruppe der Stadt- und Verkehrsplanung gewählt. Der gewählte Use-Case lautet:

„Förderung eines nachhaltigen Mobilitätsverhaltens in Leipzig: Analyse des Ist-Zustands zur Identifikation von Potenzialen, erfolgreichen Maßnahmen und Bereichen, in denen die infrastrukturellen Gegebenheiten nicht den wissenschaftlichen Empfehlungen entsprechen“.

Dieser Use-Case dient als Grundlage, um die für den Workflow benötigten Daten zu definieren und gezielt auszuwerten. Betrachtet werden dabei infrastrukturelle Gegebenheiten und Fahrpläne. Soziodemografische, wirtschaftliche und politische Daten werden nicht berücksichtigt, da sich diese nur schwer in 3D-Stadtmodellen darstellen und visualisieren lassen. Der Fokus liegt somit auf räumlichen und verkehrsrelevanten Daten, die direkt in die Modellierung einfließen können.

4.2 Zielgruppenanalyse

In der Literatur gibt es keine direkte Zielgruppenbeschreibung für Stadt- und Verkehrsplaner*innen. Um ein grundlegendes Bild dieser Berufsgruppe zu erhalten und ihre spezifischen Anforderungen zu analysieren werden verschiedene Quellen herangezogen und miteinander kombiniert.

Die Stadt- und Verkehrsplanung umfasst eine Vielzahl beruflicher Positionen, darunter Fachbereichs-, Amts- oder Geschäftsleitungen, Gruppen-, Sachbereichs- oder Projektleitungen sowie Sach- und Projektmitarbeiter*innen. Ebenso vielfältig sind die Tätigkeitsfelder, die von Stadterneuerung und -sanierung über Denkmalschutz, Freiraumplanung und Bauleitplanung bis hin zu Mobilitäts- und Verkehrsplanung reichen (vgl. Levin-Keitel et al. 2019, S. 120–121). Die Geschlechterverteilung zeigt mit 38,8% weiblichen und 61,2% männlichen Mitarbeiter*innen (vgl. Statista 2024, S. 12) eine deutliche Dominanz männlicher Beschäftigter.

Der Arbeitsalltag von Stadtplaner*innen ist geprägt von einer breiten Palette an Aufgaben, darunter Diskussionsrunden, Berechnungen, Sitzungen und Besprechungen (vgl. NachhaltigeJobs). Dies erfordert Flexibilität, Verhandlungsgeschick, Empathie sowie ein hohes Maß an Fachwissen. Rund 14% der Beschäftigten in Architekturbüros sind Absolventen eines Architekturstudiums und der Stadtplanung, während etwa 13% als technische Mitarbeiter wie Ingenieur*innen oder Bautechniker*innen tätig sind. Diese Fachkräfte bringen in der Regel einen hohen bis sehr hohen Bildungsstand mit, während der Bildungsstand der übrigen Mitarbeiter*innen nicht direkt ableitbar ist (vgl. Statista 2024, S. 24). Digitale Tools spielen eine zunehmend wichtige Rolle in der Stadtplanung. (vgl. NachhaltigeJobs), weshalb zumindest grundlegende technische Kenntnisse vorausgesetzt werden können. Bei technischen Mitarbeiter*innen ist das technische Verständnis aufgrund ihrer Ausbildung tendenziell stärker ausgeprägt.

Die Struktur der Architekturbüros in Deutschland ist stark von kleineren Betrieben geprägt: 34% der Büros beschäftigen zwei bis vier Personen, bei 30 % handelt es sich sogar um Einpersonenbüros mit vollzeitbeschäftigten Inhaber*innen. Lediglich 17% der Büros haben fünf bis neun und 11% zehn bis 19 Mitarbeiter*innen (vgl. Statista 2024, S. 23). Öffentliche Auftraggeber, die in der Stadtplanung oft als wichtigste Partner auftreten, machen je nach Größe des Architekturbüros zwischen 17% und 46% des Umsatzes aus (vgl. Statista 2024, S. 25). Größere Büros arbeiten tendenziell häufiger für öffentliche Auftraggeber. Öffentliche Aufträge unterliegen in der Regel festen Budgets und engen Zeitrahmen, was zu einem hohen finanziellen und zeitlichen Druck führt. Daher müssen eingesetzte Softwarelösungen einfach zu bedienen und schnell zu erlernen sein. Umfangreiche Einarbeitungszeiten oder unnötig komplexe Programme sind im Alltag nicht praktikabel. Der Aufwand für das Erlernen neuer Software muss in einem angemessenen Verhältnis zum erwarteten Nutzen stehen.

Werden Leitfäden für den Praxisalltag in der Verkehrsplanung entwickelt, so ist auf folgende Anforderungen zu achten (Reitberger et al. 2019, S. 3):

- Leitfäden müssen kompakt, einfach und leicht verständlich sein. Großer Wert ist auf Schaubilder und Ablaufschemata zu legen.
- Leitfäden und deren Überarbeitungen sind kostenfrei und aktiv zu verbreiten.
- Leitfäden sind regelmäßig zu überarbeiten und aufgrund von Personalfluktuation in den Verwaltungen turnusmäßig zu bewerben.

Auf Grundlage dieser Erkenntnisse werden im nächsten Schritt zwei Personas entwickelt. Diese dienen im weiteren Verlauf der Arbeit dazu, die Zielgruppenanforderungen, vor allem bei der Erstellung der Anwenderdokumentation, im Fokus zu behalten.

Personas

Thomas ist 53 Jahre alt und ein sehr gelassener Mensch. In seiner Freizeit angelt er, verbringt gerne Zeit mit seiner Familie und ist handwerklich aktiv. Er arbeitet als technischer Mitarbeiter für Geoinformationssysteme in einem Architektur- und Planungsbüro mit 15 Mitarbeiter*innen und verfügt über 35 Jahre Berufserfahrung in diesem oder vergleichbaren Tätigkeitsfeldern. Nach seiner Ausbildung zum Vermessungstechniker führte ihn eine Weiterbildung in Geoinformatik zu seiner aktuellen Position. Thomas versteht und spricht Englisch auf mittelgutem bis gutem Niveau und beherrscht den Umgang mit PCs aufgrund seiner beruflichen Tätigkeit äußerst sicher. Zudem hat er solide Programmierkenntnisse. In seinem Team für Sach- und Projektarbeit ist er auf Simulationen, Kartenerstellung und die 3D-Visualisierung von Plänen spezialisiert.

Thomas legt bei Software und deren Dokumentation großen Wert auf eine einfache Integration in bestehende Prozesse. Er bevorzugt Tools, die sich flexibel an verschiedene Anwendungsfälle anpassen lassen, und schätzt eine übersichtliche Navigation in Anwendungen und Dokumenten. Aufgrund des

hohen Zeitdrucks bei der Arbeit frustrieren ihn unnötig lange Texte. Ein schneller und einfacher Überblick über Prozesse ist ihm besonders wichtig.

Miriam ist 29 Jahre alt und eine energiegeladene Person. In ihrer Freizeit treibt sie gerne Sport und unternimmt Reisen mit ihren Freundinnen. Nach dem Abitur absolvierte sie zunächst ein Bachelor-, dann ein Masterstudium im Studienfach Stadtplanung. Seit ihrem Studienabschluss vor zwei Jahren arbeitet sie als Stadtplanerin in einem Architekturbüro mit 8 Mitarbeiter*innen. Miriam beherrscht die englische Sprache in Wort und Schrift sehr gut. Sie ist versiert im Umgang mit PCs, hat jedoch nur wenig Programmiererfahrung, die sie ausschließlich während ihres Studiums gesammelt hat. Als Projektmitarbeiterin verbringt sie den Großteil ihrer Arbeitszeit mit der Kommunikation mit Bürger*innen, der Koordination von Inhalten und der Unterstützung bei der Erstellung von Plänen. Ihr werden häufig Aufgaben übertragen, bei denen sie neue Prozesse erprobt und deren Praktikabilität für das Unternehmen bewertet.

Miriam wünscht sich Werkzeuge, die die Visualisierung bei der Kommunikation mit Bürger*innen, Verwaltungen und politischen Gremien erleichtern. Es ist ihr wichtig, bei Softwareprodukten schnell erkennen zu können, ob sie sich für einen bestimmten Anwendungszweck eignen, welche Rohdaten benötigt werden und welche Ergebnisse zu erwarten sind. Aufgrund ihrer noch geringen Berufserfahrung fehlt ihr in manchen Bereichen etwas Grundlagenwissen, weshalb sie es hilfreich findet, wenn Dokumentationen grundlegende Informationen enthalten.

In Tabelle 3 werden die relevanten Informationen aus den Personas noch einmal tabellarisch festgehalten.

Name	**Miriam**	**Thomas**
Alter/Geschlecht	29 Jahre / w	53 Jahre / m
Tätigkeits-bezeichnung und Bildungsstand	Stadtplanerin Absolventin (Bachelor + anschließend Master) Studium Stadtplanung	technischer Mitarbeiter Geoinformationssysteme Ausbildung Vermessungstechniker, Weiterbildung Geoinformatik
Firmengröße	8 Mitarbeiter*innen	15 Mitarbeiter*innen
Kenntnisse	Englisch: sehr gut PC: sehr gut, Programmieren: wenig Berufserfahrung: 2 Jahre	Englisch: mittel bis gut PC: sehr gut, Programmieren: gut bis sehr gut Berufserfahrung: 35 Jahre
Tätigkeits-bereiche	Sach-/Projektarbeiterin Kommunikation mit Bürger*innen Koordination von Inhalten Unterstützung bei der Erstellung von Plänen	Sach-/Projektarbeiter Simulation Kartenerstellung 3D-Visualisierung von Plänen
Bedürfnisse/ Wünsche	Vereinfachung der Visualisierungsmöglichkeiten für Kommunikation Überblick über Funktionen eines Softwareproduktes	einfache Orientierung in Anwendungen und Dokumenten Anpassbarkeit von Tools an verschiedene Anwendungsfälle leichte Integration in bestehende Prozesse
Frustration/ Hindernisse	fehlende Grundlageninformationen in Dokumentationen	lange Texte fehlende Übersichtlichkeit

Tabelle 3: Kurzübersicht über die erstellten Personas

5 Erarbeitung des Workflows

In diesem Abschnitt wird der Praxisteil der vorliegenden Arbeit beschrieben. Im Fokus stehen die einzelnen Schritte zur Kombination der OSM- und GTFS-Daten mit dem 3D-Stadtmodell. Dabei werden die notwendigen Verarbeitungsschritte inklusive möglicher Herangehensweisen detailliert erläutert. Auftretende Herausforderungen und Probleme werden dokumentiert. Ergänzend werden relevante Datenquellen, hilfreiche Literatur und unterstützende Dokumentationen aufgeführt, um die Umsetzung des Workflows zu erleichtern.

5.1 3D-Stadtmodell beziehen

Der erste Schritt bei der Erarbeitung des Workflows besteht darin, das 3D-Stadtmodell des gewünschten Gebiets zu beschaffen. Viele Städte und Gemeinden bieten mittlerweile 3D-Stadtmodelle direkt über ihre Websites an. Sollte dies nicht der Fall sein, können weitere Quellen wie die Online-Portale der jeweiligen Bundesländer in Betracht gezogen werden. Nachfolgend wird eine (nicht abschließende) Liste potenzieller Datenquellen vorgestellt, die für den Bezug von 3D-Stadtmodellen genutzt werden können.

Tabelle 4 zeigt zur Verfügung stehende, freie Datenquellen auf.

Institution/Autoren	**URL**	**Zugänglichkeit**	**Notiz**
verschiedene Beteiligte, unter anderem CityGML Wiki, TU Delft, Towards Data Science	https://github.com/OloOcki/awesome-citygml?tab=readme-ov-file#Germany	Open Source	- Liste verschiedener frei zugänglicher 3D-Stadtmodelle - nicht alle Modelle im CityGML-Format verfügbar
Mobilithek – Bundesministerium für Digitales und Verkehr	https://mobilithek.info/offers	abhängig vom Datensatz: - Open Source - Kostenfreie Lizenz - eingeschränkt oder kostenpflichtig	- verschiedene Suchbegriffe liefern relevante Daten, unter anderem: - „CityGML“ -„Gebäudemodell“ - „Stadtmodell“
Bundesamt für Kartographie und Geodäsie	https://www.geoportal.de	abhängig vom Datensatz: - Open Source - Kostenfreie Lizenz - eingeschränkt oder kostenpflichtig	- Verschiedene Suchbegriffe liefern relevante Daten, unter anderem: - „CityGML“ -„Gebäudemodell“ - „Stadtmodell“

Tabelle 4: Freie Datenquellen für 3D-Stadtmodelle

Dateien Tabelle 5 finden sich Datenquellen mit eingeschränkter oder kostengebundene Nutzungsberechtigung.

Institution/Autoren	URL	Zugänglichkeit	Notiz
Bundesamt für Kartographie und Geodäsie	https://gdz.bkg.bund.de/index.php/default/3d-gebaudemodelle-lod2-deutschland-lod2-de.html	Bereitstellung für Bundesbehörden und Nutzungsberechtigte nach V GeoBund Bereitstellung für Landesbehörden und Nutzungsberechtigte nach V GeoLänder	LOD2 3D-Gebäudemodelle aus ganz Deutschland

Tabelle 5: Eingeschränkte oder kostenpflichtige Datenquellen für 3D-Stadtmodelle

Im Rahmen dieser Arbeit wird das 3D-Stadtmodell der Stadt Leipzig als Beispiel genutzt, welches online über das OpenData-Portal der Stadt verfügbar ist (vgl. OpenData Leipzig). Das Modell wird in den Formaten DXF, Esri Shapefile und CityGML bereitgestellt, wobei die nachfolgenden Schritte auf Basis des CityGML-Formats durchgeführt werden.

Das Modell liegt im Detaillierungsgrad LOD2 vor und ist in 63 Teildateien unterteilt, die jeweils einen Teilbereich der Stadt abbilden. Zusätzlich werden zwei PDF-Dateien bereitgestellt, die eine Übersicht über die geografische Lage der Teilmodelle sowie wichtige Metadaten enthalten. Diese Metadaten umfassen unter anderem Angaben zum verwendeten Koordinatenreferenzsystem und eine Beschreibung der enthaltenen Attribute der CityGML-Dateien.

5.2 Analyse der mobilitätsrelevanten Daten

Für die Kombination des 3D-Stadtmodells mit den mobilitätsrelevanten Daten muss als Erstes recherchiert werden, welche Daten dafür infrage kommen. Für das Anwendungsbeispiel wurde sich dazu entschieden, Informationen zu infrastrukturellen Daten und Verkehrsplänen einzubauen. Daten zu den Verkehrsplänen werden durch das Format GTFS bezogen.

Daten zur öffentlichen Raumgestaltung sollen aus OSM extrahiert werden. Um im späteren Verlauf die relevanten OSM-Tags und Attribute zu identifizieren, muss als Erstes recherchiert werden, welche Maßnahmen der öffentlichen Raumgestaltung einen Einfluss auf das Mobilitätsverhalten von Personen haben. Dazu wird eine systematische Literaturrecherche durchgeführt, welche im Folgenden beschrieben wird.

5.2.1 Einflussfaktoren der öffentlichen Raumgestaltung

Die öffentliche Raumgestaltung hat, wie in Abschnitt 2.1 beschrieben, einen Einfluss auf das Mobilitätsverhalten von Personen. OSM hat sich als Quelle für die Sammlung und Bereitstellung detaillierter Daten zur öffentlichen Raumgestaltung etabliert. Um konkrete und relevante Maßnahmen zu sammeln, wurde eine systematische Literaturrecherche mit der Forschungsfrage „Welche städtebaulichen Maßnahmen beeinflussen das nachhaltige Mobilitätsverhalten der Bevölkerung?“ durchgeführt.

Bei der Durchführung der systematischen Literaturrecherche wurde sich an der Vorlage von Hirt & Nordhausen (2022) orientiert. Das Vorgehen inklusive genutzter Datenbanken, Suchbegriffe bzw. Suchstrings und die genauen erzielten Ergebnisse sind in Anhang 1 dokumentiert.

Bei der systematischen Literaturrecherche konnten 13 relevante Quellen ermittelt werden. Es zeigt sich, dass verschiedene städtebauliche Maßnahmen das nachhaltige Mobilitätsverhalten signifikant beeinflussen. Buchholz & Flaig (2022) sowie Raunig & Hodzic-Srndic (2020) konzentrieren sich vorrangig auf die detaillierte Beschreibung von Push- und Pull-Faktoren sowie die psychologischen Grundlagen hinter nachhaltigen Mobilitätsentscheidungen. Faßbender (2020) und Fender (2020) legen

einen besonderen Fokus auf spezifische Bereiche, wie den Radverkehr und Hochschulstandorte, wobei letztere dennoch übertragbare Maßnahmen für Innenstädte aufzeigen. Aichinger et al. (2020) bietet tiefgehende Einblicke in die rechtlichen Rahmenbedingungen und nutzt praxisnahe Beispiele zur Veranschaulichung. Rohs et al. (2021) und Rohs et al. (2023) erweitern diese Perspektive durch eine umfassende Analyse der gesellschaftlichen, politischen und infrastrukturellen Determinanten sowie durch ausführliche Handlungsempfehlungen in ihrem Abschlussbericht. Weitere Quellen wie Günther et al. (2023) und Münsch & Lell (2024) ergänzen diese Ergebnisse mit konkreten Projektbeispielen und detaillierten Wirkungsanalysen spezifischer Maßnahmen. Insgesamt verdeutlicht die ermittelte Literatur vielfältige Ansätze zur Förderung nachhaltiger Mobilität, wobei bestimmte Studien durch ihre praxisnahen Beispiele besonders wertvoll sind.

Die in den Quellen genannten Maßnahmen zur Förderung nachhaltiger Mobilität in Bezug auf die öffentliche Raumgestaltung wurden tabellarisch festgehalten, wobei wirtschaftliche und politische Maßnahmen nicht aufgenommen wurden. Zu jeder Textstelle wurde die implizierte Maßnahme bzw. eine Zusammenfassung der Textstelle notiert. Besonderheiten wurden schriftlich als Anmerkungen festgehalten. Dabei wurde zusätzlich auf die Übertragbarkeit der genannten Maßnahmen nach OSM geachtet. Allgemeine Handlungsempfehlungen wie „Begegnungszonen einrichten“ ließen sich gut interpretieren. In Fällen, in denen keine konkreten Maßnahmen genannt wurden, wurden relevante Textstellen interpretiert, um mögliche Maßnahmen abzuleiten. Durch die kontinuierliche Auseinandersetzung mit den relevanten Quellen und den darin ermittelten Maßnahmen konnte ein Grundverständnis für bestimmte Formulierungen entwickelt werden. Dieses Verständnis erleichterte es, neue Textstellen einzuordnen und bereits erfasste Maßnahmen gegebenenfalls zu ergänzen. Es wurde bei der Formulierung darauf geachtet, möglichst wenige redundante Maßnahmen zu formulieren.

Seite	Quelle	Zitat	implizierte Maßnahme/ Zusammenfassung
24	Rohs et al. 2021	Eine zentrale Bedeutung kommt der Reduzierung des öffentlichen Parkraums zu. Steht für Pkw weniger Parkraum zur Verfügung, sinkt die Attraktivität des MIV. Da Carsharing-Fahrzeuge einen Beitrag zu einer nachhaltigen Mobilität leisten, bietet es sich an, ihnen privilegierte Parkplätze aus dem bisherigen Bestand des öffentlichen Parkraums zuzuweisen.	**weniger Parkplätze**
			Carsharing-Parkplätze einrichten
		Für die Förderung des Fußverkehrs spielt neben dem Ausbau durchgängiger, ausreichend dimensionierter und attraktiver sowie beschilderter Fußwegenetze der Aspekt Sicherheit insbesondere beim Queren von Fahrbahnen eine wesentliche Rolle.	**Fahrbahnverengungen platzieren (zur Straßenüberquerung)**
			Fußverkehrsampeln errichten
			Fußgängerüberwege errichten
		Der Radverkehr profitiert von einer sicheren und bedarfsgerechten Radverkehrsinfrastruktur mit hinreichender Netzdichte und -qualität sowie sicheren Abstellanlagen.	**lange und verbundene Radwege**
			Gute Oberflächenbeschaffenheit von Radwegen
			Fahrradabstellanlagen
			Fahrradgaragen
		Ein qualitativ hochwertiger öffentlicher Verkehr bildet als attraktive, flächensparende und emissionsarme Alternative zum MIV das Rückgrat einer klimagerechten Stadtentwicklung und stellt die Mobilität in den Städten sicher. Ein attraktives ÖPNV-Angebot umfasst auch die Stadtränder und das Umland sowie kleinere Gemeinden. Wichtige Bestandteile sind unter anderem eine hohe Netz- und Haltestellendichte, separate Spuren für Busse und Straßenbahnen, umfangreiche Bedienzeiten in hoher Taktung, eine hohe Fahrzeugqualität, Barrierefreiheit, Sicherheit, Sauberkeit, Zuverlässigkeit, schnelle Umsteigemöglichkeiten an Verkehrsknotenpunkten, ein einfacher Zugang, umfassender Kundenservice mit Anschlussgarantien sowie der ergänzende Einsatz von Schnellbuslinien und flexiblen Bedienformen unter Berücksichtigung der Stadt-Umland-Beziehungen. Zudem zählen hierzu auch die kundenorientierte Informationsbereitstellung sowie nutzungsfreundliche Ticket- und Tarifsysteme.	**ausreichende Anzahl an ÖPNV-Haltestellen**
			Busspuren errichten
			Straßenbahnen auf eigenem Gleiskörper
			Schnellbuslinien

Abbildung 5: Auszug aus Excel-Datei zur systematischen Literaturrecherche

In Abbildung 5 ist eine beispielhafte Zuordnung von Textstellen und den genannten Maßnahmen zu sehen. Falls ein Soll-Wert genannt ist, so wird dieser in einer gesonderten Spalte aufgeführt. Dabei sind jeweils die Quelle und die Seitenzahl angegeben. Anschließend wurde das Zitat übernommen und die genannte oder implizierte Maßnahme zugeordnet.

Nicht alle Textstellen sind für die weitere Verarbeitung geeignet, da die beschriebenen Maßnahmen teils zu umfassend oder zu allgemein gefasst sind. Die Kombination mehrerer einzelner Maßnahmen erschwert zudem deren Abbildung in OSM. Ein Beispiel für eine solche Textstelle, die aufgrund ihrer schlechten Übertragbarkeit in OSM nicht aufgenommen wurde, ist: „Lebendigkeit bzw. Vitalität des öffentlichen Raumes entsteht dabei durch das Vorhandensein eines hohen Modal-Split-Anteils an Zufußgehenden und Radfahrenden. Dieser wird unter anderem durch die Vielzahl an Zielen (Nutzungsangebot) und den kurzen Distanzen zwischen Quelle und Ziel ermöglicht (Handy et al. 2014) - dementsprechend trägt die Kombination aus Nutzungsmischung und dichter Siedlungsstruktur zu einem aktiven Mobilitätsverhalten bei." (Gerike et al. 2020, S.38). Diese Textstelle behandelt zahlreiche breit gefasste Konzepte (z. B. „Modal Split" und „dichte Siedlungsstruktur"), deren konkrete Implikationen – wie kurze Wege zwischen Lern- und Arbeitsorten, Einkaufsmöglichkeiten und Zuhause sowie eine verbesserte Infrastruktur für Rad- und Fußverkehr und den ÖPNV – zwar ableitbar sind, sich jedoch aufgrund ihrer komplexen Zusammenhänge nur schwer systematisch in OSM abbilden lassen.

Insgesamt wurden 116 relevante Textstellen identifiziert und ausgewertet. Da in manche Textstellen mehr als eine Maßnahme konkret genannt oder impliziert wurden, ergaben sich daraus 181 konkrete Maßnahmen. Die systematische Überführung dieser in zugehörige OSM-Tags wird im folgenden Abschnitt erklärt.

Die 181 ermittelten Maßnahmen wurden anschließend in Kategorien eingeteilt, die basierend auf den Kategorien des Datenkatalogs der Mobilithek (vgl. Mobilithek Categories) festgelegt wurden. Obwohl diese Kategorien im hier entwickelten Workflow nicht direkt verwendet werden, können sie für die Anpassung des Workflows für andere Use-Cases wichtig sein, um irrelevante OSM-Objekte auszuschließen. Beispielsweise wäre denkbar, dass für gewisse Anwendungen Daten zu Sharing-Verkehrsmitteln nicht benötigt werden. Die entwickelte Tabelle hilft dabei, die zugehörigen Maßnahmen unkompliziert und schnell auszuschließen. Der Datenkatalog der Mobilithek teilt mobilitätsrelevante Daten in 24 Kategorien, von denen jedoch nicht alle für die hier identifizierten Maßnahmen zur Verbesserung der nachhaltigen Mobilität relevant sind. Tabelle 6 listet einen Ausschnitt der relevanten Datenkategorien und ihrer zugehörigen Maßnahmen auf. Die gesamte Übersicht befindet sich in Anhang 2. Eine Maßnahme kann dabei in mehr als einer Kategorie auftauchen, wenn dies angebracht ist. So sind abgesenkte Bordsteine beispielsweise für das Fußverkehrsnetz, aber auch für das Radverkehrsnetz von Relevanz. Für Maßnahmen, welche nicht eindeutig einer Kategorie zugeteilt werden können, wurden der Kategorie „Allgemeine Informationen zur Wegeplanung" zugeordnet.

Kategorie	**Maßnahmen**
allgemeine Informationen zur Wegeplanung	(digitale) Infotafeln
	Fahrplan-Anzeigetafeln mit barrierefreien Ansagen
	Bänke platzieren
Sharing-Verkehrsmittel	Bevorrechtigung Taxen & On-Demand-Verkehr
	Carsharing-Parkplätze einrichten
	E-Carsharing-Plätze
Dynamische Verkehrsregelungen	Bedarfsampeln für MIV

	Vorrang-Ampeln für ÖPNV
	Wetterabhängige Anpassung von Ampelschaltungen für Fußverkehr
Fracht und Logistik	Durchfahrtsverbote für Schwerlastverkehr
	Haltepunkte für Lieferverkehr errichten
	Nachtfahrverbote für Schwerlastverkehr
Fußverkehrsnetz	abgesenkte Bordsteine
	ausreichend Fußgängerüberwege
	breite Gehwege
Öffentlicher Verkehr: Gelegenheitsverkehr	Bevorrechtigung Taxen & On-Demand-Verkehr
	Haltepunkte für On-Demand-Verkehr errichten
	Haltepunkte für Taxen errichten
öffentlicher Verkehr: Linienverkehr	ausreichende Anzahl an ÖPNV-Haltestellen
	Busspuren errichten
	Fahrplanauskunft mit Lautsprecheransage
Parkplätze und Rastanlagen	Anwohnerparkausweise einführen
	Behindertenparkplätze
	E-Bike Abstellplätze einrichten
Radverkehrsnetz	Ampelgriffe anbringen
	Radwegweiser aufstellen
	steigungsarme Radspuren
Straßennetz	Radfahrstreifen rechts von Parkstreifen
	Sackgassen in Quartieren errichten
	Verkehrsflächen klar trennen
statische Verkehrsregelungen	Beschilderung beleuchten
	Bodenkissen platzieren
	Bodenschwellen platzieren
Tank- und Ladestationen	E-Auto-Ladesäulen errichten
	E-Bike-Ladestationen errichten

Tabelle 6: Kategorien der öffentlichen Raumgestaltung und zugehörige Maßnahmen (Auszug aus Anhang 1)

5.2.2 Überführung in OSM-Tags

Die Überführung spezifischer Maßnahmen in OpenStreetMap-Tags (OSM-Tags) ist ein zeitaufwendiger und komplexer Prozess. Bereits erprobte Ansätze oder Matching-Tabellen bzw. Schemata für dieses Vorhaben konnten nicht gefunden werden. Die Online-Anwendung Overpass Turbo, welche die Overpass-API nutzt, bietet einen Wizard, der das Erstellen von Anfragen erleichtern soll. Allerdings erkennt dieser Wizard nicht alle eingegebenen Suchbegriffe und erweist sich daher nicht in allen Fällen als hilfreich. Zusätzlich stehen das OSM-Wiki (vgl. OSM Wiki) und Taginfo (vgl. TagInfo) als Quellen zur Verfügung. Das OSM-Wiki ermöglicht durch die Eingabe bestimmter Suchbegriffe das Auffinden eines Teils der verfügbaren Tags und deren Beschreibungen. Dennoch beinhaltet es nicht alle Tags, insbesondere solche nicht, die weniger genutzt werden. Taginfo hingegen erlaubt die Eingabe spezifischer Tags und liefert detaillierte Informationen zu deren

Verwendung, wie beispielsweise die Anzahl der Nutzungen weltweit, eine Kartenansicht der Verteilung sowie Übersichten über zugehörige Werte.

Obwohl beide Tools nützlich sind, weisen sie Einschränkungen auf. Das OSM-Wiki deckt nicht alle Tags ab, während in Taginfo nur existierende Tags eingegeben werden können, was voraussetzt, dass grundlegendes Wissen über vorhandene Tags vorliegt. Zudem fehlen in Taginfo eine unscharfe Suche und umfassende Informationen in deutscher Sprache.

Aufgrund dieser Limitierungen wurde ein alternativer Ansatz erprobt, bei dem künstliche Intelligenz (KI) zur Überführung von Maßnahmen in OSM-Tags genutzt wurde. Diese Methode bietet vor allem dann Vorteile, wenn Unklarheit über die vorhandenen Tags oder deren potenzielle Nutzung besteht. ChatGPT wurde in diesem Kontext eingesetzt, da es auf einer breiten Datengrundlage basiert, mittlerweile sehr verbreitet ist und die Erstellung von Prompts einfach gestaltet. Bei der Verwendung ergaben sich folgende Erkenntnisse: Alle genutzten Modelle lieferten zu einem hohen Anteil falsche Ergebnisse, wobei die kostenpflichtigen Modelle GPT-5 und GPT-4o konsistenter waren. Aufgrund des Ziels der Arbeit, den Workflow mit Hilfe von kostenfreier und Open-Source-Software zu gestalten, wurde außerdem das kostenfreie Modell, GPT-3.5, verwendet. Dieses lieferte häufiger falsche Informationen als die kostenpflichtigen Modelle. Es wurde keine quantitative Analyse der korrekten und inkorrekten Ergebnisse durchgeführt, jedoch traten folgende Fehler wiederholt auf:

- Ausgabe von nicht existierenden Tags,
- Vorschlag von Key-Value-Paaren, die so nicht in OSM genutzt werden.

Selbst bei mehrfacher Aufforderung, nur existierende Tags auszugeben, wiederholte sich dieses Problem. Daher war es erforderlich, jede Ausgabe der KI sorgfältig zu überprüfen. Für die Verifizierung der Ergebnisse wurde Taginfo genutzt. Wie bereits beschrieben, ist es dort möglich, die Nutzungshäufigkeit von Tags, Keys und den zugehörigen Values zu finden. Zudem sind häufig direkt Wiki-Seiten direkt verlinkt, die eine detaillierte Nutzungserklärung enthalten. Sollten vorgeschlagene Tags oder Key-Value-Paare in Taginfo nicht auffindbar sein, gelten diese als fehlerhaft.

Um eine spätere Extraktion der ermittelten Maßnahmen mithilfe der Overpass-API aus OSM zu ermöglichen, ist es notwendig, die Daten bereits bei der Überführung so in OSM-Tags zu strukturieren, dass sie dem Schema der Overpass-API entsprechen. Um dies zu gewährleisten, wurden die ermittelten OSM-Tags jeweils in Overpass-Turbo getestet. In OSM liegen bestimmte Elemente üblicherweise nur als ein Typ — entweder als Node, Relation oder Way — vor. Aus Gründen der Einfachheit wird jedoch jedes Objekt in den Abfragen als „nwr“ (node, way, relation) behandelt. Dieser Ansatz berücksichtigt, dass es keinen einheitlichen Standard für das Mapping in OSM gibt und daher nicht garantiert werden kann, dass bestimmte Daten nicht abweichend gemappt wurden.

Von allen ermittelten OSM-Objekten werden, unabhängig von der eigentlichen Maßnahme, die gesamten Key-Value-Paare extrahiert. Diese Vorgehensweise verursacht keinen signifikanten Mehraufwand, erweitert jedoch die Flexibilität für detaillierte Analysen der Daten, die durch den in dieser Arbeit entwickelten Workflow exportiert werden. Als Beispiel lässt sich dafür der Node mit der `id=324569367` aufführen. Dieser Node ist Teil des Overpass-Ergebnisses, da alle Objekte mit dem Tag `nwr[„barrier“=“bollard“]` abgerufen werden. Zusätzlich zu dem spezifischen Tag werden jedoch auch alle anderen Eigenschaften der Node erfasst und gespeichert. Folgende Daten werden von Overpass exportiert:

```
  <node     id="324569367"     lat="51.3414164"     lon="12.3721169"
version="4" timestamp="2022-04-13T10:02:22Z" changeset="119659780"
uid="1198074" user="DoubleA">
    <tag k="access:conditional" v="yes @ (5:00-11:00)"/>
    <tag k="barrier" v="bollard"/>
    <tag k="bicycle" v="yes"/>
    <tag k="bollard" v="removable"/>
    <tag k="foot" v="yes"/>
  </node>
```

Das genaue Vorgehen bei der OSM-Datenabfrage über Overpass wird im Abschnitt 5.3 erläutert. Die Tabelle, welche zur Zuordnung der OSM-Tags zu den Maßnahmen genutzt wurde, kann in Anhang 3 betrachtet werden. Aufgrund des hohen Aufwands bei der Ermittlung der OSM-Tags wurde sich dazu entschieden, nur 50% der gefundenen Maßnahmen, also 90 Maßnahmen, in ihre zugehörigen OSM-Tags zu überführen. Der Zeitaufwand für die Überführung dieser 90 Maßnahmen in ihre zugehörigen Tags betrug ungefähr 15 Stunden. Die daraus entstandene Overpass-Query ist in Anhang 4 dokumentiert und kann in Overpass Turbo getestet werden.

5.2.3 Zusammenfassung der Hürden und aufgetretenen Probleme

Bei der Analyse der Einflussfaktoren der öffentlichen Raumgestaltung auf das Mobilitätsverhalten von Personen sowie bei der Überführung dieser Daten in OSM-Tags traten keine schwerwiegenden Probleme auf. Insgesamt lässt sich jedoch feststellen, dass die Überführung in OSM-Tags ein äußerst aufwendiger Prozess ist, der selbst durch den Einsatz künstlicher Intelligenz nur geringfügig erleichtert werden kann. Ein erprobter Workflow oder dokumentierte Hilfestellungen bei der Überführung fehlen. Der Prozess erfordert einen erheblichen manuellen Prüfaufwand, um die Genauigkeit und Konsistenz der Tags sicherzustellen. Aus diesem Grund wurden nur 90 der 181 gefundenen Merkmale in ihre zugehörigen OSM-Tags überführt. Dies macht die Ergebnisse zwar unvollständig, um die Praktikabilität des Workflows zu testen reichen die Daten jedoch aus, da trotzdem eine große Anzahl verschiedener Objekttypen vorhanden ist.

5.3 Exportieren/ Extrahieren der Daten aus OpenStreetMap

Die Extraktion von Daten aus OSM kann über diverse Tools und Schnittstellen erfolgen. Die OSM-API, die direkt von OSM bereitgestellt wird, ermöglicht das Einfügen, Modifizieren und Herunterladen von Daten. Allerdings ist das Herunterladen über die OSM-API aufgrund ihrer begrenzten Kapazität für kleine Kartenausschnitte und aufgrund ihrer geringen Performance für größere Abfragen ungeeignet. Daher wird häufig die Overpass-API bevorzugt, die zusammen mit ihrem Web-Interface Overpass Turbo eine effizientere Option darstellt. Overpass Turbo bietet einen einfach zu bedienenden Wizard für das Erstellen von Abfragen, was besonders für Einsteiger hilfreich ist. Diese Schnittstelle wird auch im OSM-Wiki aufgrund ihrer umfassenden Dokumentation empfohlen (siehe OpenStreetMap WikiAPIs).

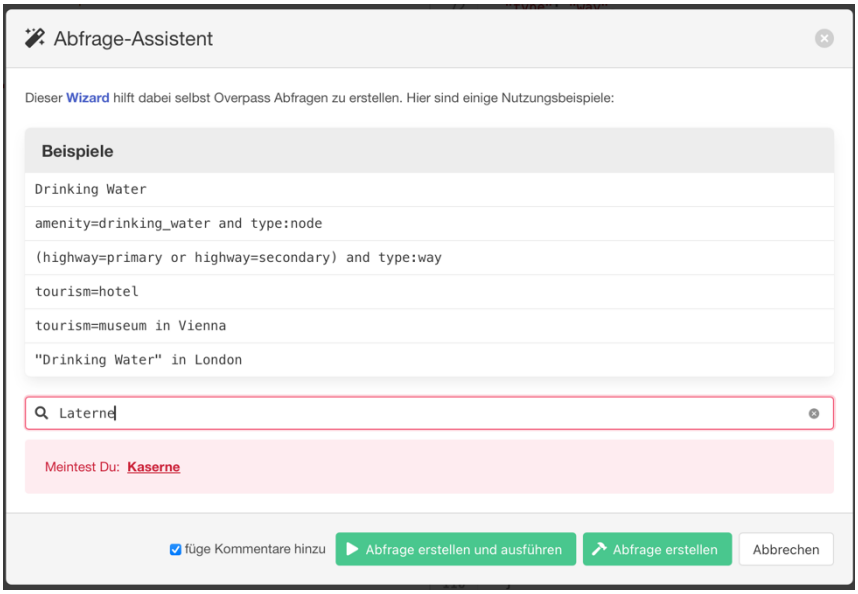


Abbildung 6: Overpass-Turbo-Wizard

Abbildung 6 zeigt den Overpass-Wizard. In das unten liegende Textfeld können Suchbegriffe eingegeben werden. Ist ein Suchbegriff unbekannt, werden alternative Vorschläge gemacht, die jedoch nicht immer exakt passen, wie das Beispiel in der Abbildung zeigt. Overpass Turbo ermöglicht das Herunterladen der gesuchten Tags im aktuell sichtbaren Kartenausschnitt (siehe Abbildung 7). Dies vereinfacht die Nutzung, wenn die genauen Koordinaten der relevanten Fläche nicht bekannt sind. Gleichzeitig besteht die Möglichkeit, spezifische Koordinaten anzugeben, was praktisch ist, da die Overpass-API ausschließlich die Eingabe exakter Koordinaten erlaubt. Dadurch können Overpass-API-Queries in Overpass Turbo getestet werden, wobei auftretende Fehler direkt im Tool angezeigt werden.

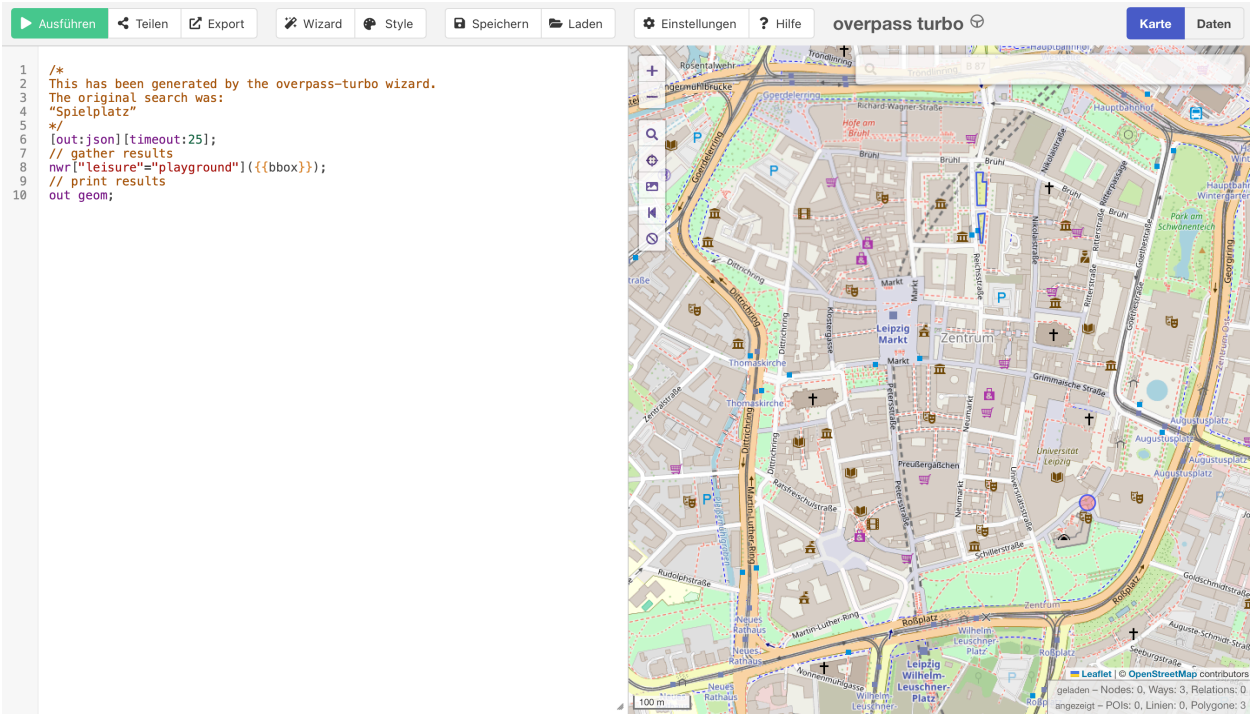


Abbildung 7: Overpass-Turbo Oberflächenübersicht

Aufgrund der besseren Performance und der einfachen Erstellung von Overpass-Queries mittels des Wizards wurde Overpass als bevorzugte Schnittstelle gewählt (vgl. Overpass Turbo). Die entwickelte Query[5] wurde, wie in Abschnitt 5.2 beschrieben, mit Unterstützung der KI ChatGPT erstellt und die Ergebnisse mit dem Tool Taginfo sowie dem OSM-Wiki validiert. Für den Datenabruf über die Overpass API kann dieselbe Query wie in Overpass Turbo verwendet werden, jedoch mit einer wichtigen Einschränkung: Während Overpass Turbo die Begrenzung des Suchbereichs durch die Platzhalterangabe `{{bbox}}` ermöglicht[6], erfordert die Overpass API die exakte Angabe von Koordinaten. Da für den hier entwickelten Workflow eine Automatisierung in Python und dementsprechend ein direkter Abruf der Overpass-API geplant ist, müssen Methoden zur Extraktion der Koordinaten der relevanten Gebiete entwickelt werden. Ansätze dazu werden in den folgenden Abschnitten beschrieben. Der Abruf der OSM-Daten aus der Overpass-API wird mit Hilfe von Python umgesetzt und später im Abschnitt 6 genauer beschrieben.

5.3.1 Festlegen der Koordinaten für den OSM-Export

Um die Begrenzungsfläche für den Overpass-API-Abruf zu erstellen, stehen verschiedene Methoden zur Verfügung. Dabei kann entweder ein Rechteck oder ein Polygon als Begrenzung verwendet

[5] Eine Query ist eine Anfrage an eine Datenbank, oder in diesem Fall an eine API.

[6] Die Verwendung der Variable (`{{bbox}}`) bewirkt, dass der in Overpass geöffnete Kartenausschnitt genutzt wird, um die Grenzen für den Overpass-Abruf festzulegen.

werden. Ein Rechteck bietet den Vorteil, dass lediglich zwei Koordinatenpunkte – die Süd-West- und die Nord-Ost-Ecke – benötigt werden (vgl. Overpass bbox). Dies vereinfacht die Erstellung der Abfrage erheblich. Allerdings kann ein Rechteck zu einer unnötigen Erfassung großer Datenmengen führen, wenn die tatsächliche Form des Suchgebiets stark von der rechteckigen Abgrenzung abweicht. Alternativ kann ein Polygon als Begrenzung genutzt werden, was den Vorteil hat, dass potenziell weniger irrelevante Daten abgerufen werden müssen. Dies führt zu effizienteren Datenverarbeitungsprozessen. Allerdings kann die Erstellung eines Polygons aufgrund der Vielzahl benötigter Punkte komplex sein und erfordert möglicherweise umfangreiche geometrische Operationen zur Überprüfung der Flächenüberlappung. Zudem muss sichergestellt werden, dass der erste und der letzte Punkt des Polygons identisch sind, um eine geschlossene Fläche zu bilden.

Zur Festlegung des Suchgebiets bieten sich verschiedene Vorgehensweisen an. Eine Möglichkeit besteht darin, die Begrenzungsfläche manuell zu erstellen und beispielsweise als GeoJSON-Datei in die Software zu importieren. Alternativ können GeoNames eingegeben oder die Begrenzung automatisch anhand einer hochgeladenen CityGML-Datei ermittelt werden. Eine Herausforderung stellt sich dabei, wenn mehrere, nicht zusammenhängende Flächen oder Punkte verwendet werden sollen. Hierfür könnten die Flächen entweder zu einer einzigen zusammengeführt oder einzeln genutzt werden. Aus Gründen der Einfachheit wurde entschieden, bei mehreren Flächen ein Rechteck zu erstellen, das sich aus den jeweils kleinsten und größten Längen- und Breitengraden zusammensetzt. Die möglichen Ansätze zur Verarbeitung von GeoJSON, CityGML und GeoNames zur Extraktion von geografischen Gebieten werden im Folgenden detailliert beschrieben.

CityGML-Datei

OpenStreetMap (OSM) arbeitet mit dem weitverbreiteten WGS84-Koordinatensystem[7], auch *Coordinate Reference System (CRS)*. Dieses geografische CRS unterteilt die Erde in Längen- und Breitengrade. Im Gegensatz dazu verwendet CityGML unterschiedliche lokale CRS[8]. Der CityGML-Standard empfiehlt die Angabe des Koordinatensystems im Tag `srsName` (vgl. OGC 2023, Recommendation 1). Aufgrund dieser Unterschiede ist für eine automatisierte Overpass-Abfrage basierend auf einer CityGML-Datei eine Koordinatenumwandlung notwendig. Ein besonderes Problem besteht darin, dass die verwendeten Koordinatensysteme nicht im Voraus bekannt sind. Daher muss eine Lösung gefunden werden, die eine automatische Umwandlung verschiedener CRS in WGS84 ohne großen Programmieraufwand ermöglicht. Dies kann in Python durch Bibliotheken wie pyproj und geopandas realisiert werden. Um die Umwandlung und das Zusammenführen der unterschiedlichen Geometrien aus CityGML nicht zu verkomplizieren, wurde folgendes Vorgehen gewählt:

Das 3D-Stadtmodell im CityGML-Format muss für den späteren Workflow in eine 3DCityDB importiert werden. Theoretisch könnte die importierte Datei genutzt werden, um die Koordinaten der Flächen und Punkte zu extrahieren. Eine Umsetzung dieses Ansatzes hätte den Workflow erheblich vereinfacht. Allerdings stellte sich bei der praktischen Umsetzung heraus, dass aufgrund der unterschiedlichen Konfigurationen der CityGML-Dateien die Geometrien beim Import in die 3DCityDB in verschiedene Tabellen abgelegt werden. Die Erfassung aller relevanten Tabellen mit Geometrien erfordert demnach eine vorherige Analyse der 3DCityDB, die Identifikation der relevanten Tabellen und die entsprechende Anpassung eines Skripts. Obwohl dies möglich ist, gestaltet sich dieser Prozess als wenig benutzerfreundlich.

Daher wurden alternative Methoden zur Begrenzungsdefinition mittels GeoNames oder GeoJSON als einfacher eingestuft und das Vorgehen mittels CityGML entsprechend nicht weiterverfolgt. Diese

---

[7] Eine genauere Beschreibung der Koordinatensysteme erfolgt in Abschnitt 4.6.3 unter dem Punkt „Koordinatensysteme“.

[8] CityGML-Dateien liegen projektionsbasierten Koordinatensystemen vor, welche Breiten- und Längengrade nutzen.

Alternativen ermöglichen eine unkomplizierte Erstellung der Begrenzungsflächen, ohne dass komplexe Skripte angepasst werden müssen.

GeoNames

GeoNames ist eine umfangreiche Datenbank, die eine Zuordnung von Ortsnamen zu geografischen Punkten im WGS84-Koordinatensystem bereitstellt. Über 25 Millionen geografische Namen stehen unter freier Lizenz zur Verfügung (vgl. GeoNames). Über die von GeoNames bereitgestellten Download-Ordner können ZIP-Dateien für jedes Land heruntergeladen werden. Diese enthalten CSV-Dateien, die unter anderem Ortsnamen, geografische Lage und Postleitzahlen umfassen. Die Daten sind sowohl für den Menschen als auch für Maschinen lesbar. Der Import der Daten in Tabellenkalkulationsprogramme wie Excel ist unkompliziert, sodass die Informationen direkt von den Nutzer*innen eingesehen werden können. Zudem lässt sich das Einlesen der CSV-Dateien für Automatisierungen, beispielsweise mit Python und Bibliotheken wie pandas und geopandas, problemlos durchführen.

Das Durchsuchen der GeoNames-Datenbank nach für das Anwendungsbeispiel relevanten Einträgen wie „Leipzig Zentrum“, liefert die in Tabelle 7 dargestellten Ergebnisse. Diese zeigen nicht den Stadtteil „Leipzig Zentrum“ selbst, sondern die Standorte zweier Hotels an. Die Daten der GeoNames sind also nicht sehr vollständig und nur unter Umständen für eine Verarbeitung geeignet. Für andere Orte könnte es jedoch bessere Daten geben, wie es beispielsweise für den Stadtteil „Leipzig Wahren“ der Fall ist. Die Qualität der gelieferten Daten muss also für jeden Anwendungsfall einzeln geprüft und daraus eine Entscheidung über die Nutzung von GeoNames getroffen werden.

Name	**geogr. Breite**	**geogr. Länge**
Days Inn Leipzig City Centre Prev Grand City Hotel Leipzig Zentrum	51.33783	12.39403
Ibis Leipzig Zentrum	51.3421	12.3811

Tabelle 7: Ergebnisse der GeoNames Suche nach "Leipzig Zentrum"

Ein Nachteil der Nutzung von GeoNames besteht darin, dass die bereitgestellten geografischen Daten lediglich als Punktdaten für die Mittelpunkte der jeweiligen Flächen vorliegen. Dies erfordert von den Nutzer*innen die Kenntnis des relevanten Suchradius um diese Punkte, was nicht immer gegeben ist. Um die Punktdaten in Flächen umzuwandeln, die für Overpass-API-Abfragen geeignet sind, wurde folgendes Vorgehen gewählt: Für jeden Punkt wird ein Suchradius in Metern festgelegt, aus dem anschließend ein Rechteck erstellt wird. Werden mehrere GeoNames angegeben, so werden, wie bereits beschrieben, die jeweils kleinsten und größten Längen- und Breitengraden zu einem Rechteck zusammengesetzt. Dieses Rechteck dient als Begrenzungsfläche für die Overpass-API-Abfragen, wodurch die Komplexität eines Polygons mit vielen Eckpunkten vermieden wird. Diese Methode ermöglicht eine effiziente Datenabfrage, indem unnötige Datenmengen, die bei der Verwendung eines Polygons entstehen könnten, reduziert werden. Die Verarbeitung der GeoNames ist aufgrund der nötigen Umrechnung des Radius in Metern in Längen- und Breitengrade kompliziert. Diese Umrechnung wurde in Python umgesetzt und wird später im Abschnitt 6 genauer beschrieben.

GeoJSON

GeoJSON ist ein einfaches, auf JSON basierendes, geografisches Austauschformat zur Darstellung von Punkten, Linien und Flächen sowie deren zugehörigen Eigenschaften. Es ermöglicht die strukturierte Speicherung und den Austausch geografischer Daten in einem leicht lesbaren und maschinenverarbeitbaren Format. Die Spezifikationen von GeoJSON sind standardisiert und können unter (vgl. GeoJSON) eingesehen werden. GeoJSON kann aus verschiedenen Anwendungen exportiert werden, darunter Geoinformationssysteme (GIS) und GIS-Bibliotheken in Programmiersprachen wie Python und JavaScript. Darüber hinaus bietet die Webseite Geojson.io (vgl.

GeoJSON.io) eine benutzerfreundliche Plattform, auf der Anwender*innen GeoJSON-Dateien durch Zeichnen von Flächen, Linien und Punkten auf einer Karte erstellen und herunterladen können. Dies erleichtert die Nutzung erheblich, da das relevante Gebiet visuell definiert werden kann.

Die maschinelle Verarbeitung von GeoJSON ist unkompliziert, da die Daten im WGS84-Koordinatensystem vorliegen, das von den meisten geografischen Anwendungen unterstützt wird. Für die Auswertung in Python können Bibliotheken wie geopandas, shapely und json verwendet werden, die eine einfache Handhabung und Analyse der GeoJSON-Daten ermöglichen. Für die Umsetzung der Koordinatenextraktion durch GeoJSON müssen keine speziellen Festlegungen getroffen werden. Bei der Angabe mehrerer Flächen werden, wie bereits mehrfach beschrieben, die größten und kleinsten Längen- und Breitengrade zu einem Rechteck zusammengefasst, anstatt komplexe Polygone mit vielen Eckpunkten zu verwenden. Da Rechtecke in GeoJSON genauso wie Polygone behandelt werden, liegt es im Ermessen der Nutzer*innen, ob sie ihre Flächen präzise als Polygone oder grob als Rechtecke abstecken möchten. Die Verarbeitung des GeoJSON wurde in Python umgesetzt und wird später im Abschnitt 6 genauer beschrieben.

5.3.2 Zusammenfassung der Hürden und aufgetretenen Probleme

Beim Exportieren der Daten aus OpenStreetMap fielen mehrere Herausforderungen auf, die die Effizienz und Genauigkeit der Datenextraktion beeinträchtigten. Ein Problem besteht darin, dass Overpass Turbo die Verwendung von `bbox` als Variable unterstützt, die Overpass API jedoch nicht. Dies erfordert für die später geplante Automatisierung eine Ersetzung der `bbox`-Variable durch exakte Koordinaten im Skript und verkompliziert den Prozess. Zudem stellt sich die Frage, wie mit zwei nicht zusammenhängenden Bereichen umgegangen werden soll. Die gewählte Lösung, die Kombination der Bereiche in ein gemeinsames Rechteck, erhöht jedoch die Komplexität der Datenverarbeitung.

Die Extraktion der Koordinaten aus einer CityGML-Datei musste verworfen werden. Trotz des standardisierten Formats erlaubt die hohe Flexibilität von CityGML-Dateien zahlreiche unterschiedliche Konfigurationen der Koordinatensysteme. Ein Import der Daten in die 3DCityDB sorgt dafür, dass die Objekte in viele verschiedene Tabellen importiert werden können, abhängig von der Dateikonfiguration. Dadurch entsteht ein hoher manueller Aufwand bei der Datenanalyse, was die Benutzerfreundlichkeit senkt.

Darüber hinaus weist die Verwendung von GeoNames einige Nachteile auf. Die bereitgestellten geografischen Daten sind lediglich Punktdaten für die Mittelpunkte der jeweiligen Flächen, wodurch die Nutzer*innen den relevanten Suchradius kennen müssen, was nicht immer der Fall ist. Zusätzlich sind die vorhandenen GeoNames unter Umständen ungenau, wie bei der Suche nach „Leipzig Zentrum“, wo statt des Stadtteils lediglich die Standorte zweier Hotels gefunden wurden. Dies macht eine individuelle Prüfung und selektive Nutzung der Daten erforderlich.

5.4 GTFS-Daten beziehen

Um GTFS-Daten in die Verknüpfungsoperationen einzubeziehen, müssen diese zunächst bezogen werden. GTFS-Daten sind online über verschiedene Plattformen verfügbar. Ein Beispiel hierfür ist der GTFS-Feed für Deutschland, der unter (GTFS DE) abgerufen werden kann. Allerdings sind nicht alle Feeds kostenfrei zugänglich. Auf der offiziellen Seite des GTFS-Standards werden weitere Datenquellen für den Bezug solcher Daten aufgeführt (vgl. GTFS Resources).

Für das in dieser Arbeit behandelte Praxisbeispiel wird der Feed „ÖPNV Deutschland“ verwendet (vgl. GTFS DE - NV). Dieser Feed enthält die Standarddaten des GTFS-Modells, darunter Informationen zu Verkehrsbetrieben (agency.txt), Routen (routes.txt), Verbindungen (trips.txt),

Haltestellen (stops.txt und stop_times.txt) sowie zu Kalenderdaten (calendar.txt und calendar_dates.txt).

5.5 Analyse der Datenqualität

Wenn alle Rohdaten aus den beschriebenen Quellen bezogen wurden, bietet es sich an, vor der eigentlichen Verknüpfung der Daten die Datenqualität zu überprüfen. Eine Analyse der Datenqualität ist nur teilweise sinnvoll. Sie kann genutzt werden, um mögliche Probleme bereits vor der Verknüpfung der Daten herauszufinden. Andererseits stehen für die benötigten Daten (3D-Stadtmodell, infrastrukturelle Daten und Fahrpläne) nur wenige oder keine anderen Datenquellen neben den oben genannten zur Verfügung. Eine Analyse der Datenqualität kann also weniger genutzt werden, um bessere Daten zu finden, sondern mehr, um potenzielle Probleme frühzeitig zu erkennen. Dies kann auch bei der Entscheidung helfen, ob ein Projekt, welches dem hier umgesetzten gleicht oder ähnelt, umsetzbar ist oder nicht.

Im Folgenden wird die Datenqualität der für das Anwendungsbeispiel verwendeten CityGML- und OSM-Daten unter Berücksichtigung des Anwendungsfalls bewertet.

5.5.1 CityGML

Die intrinsische Datenqualität der verwendeten Daten ist hoch. Sie stammen von der Website der Stadt Leipzig, sind dadurch glaubwürdig und haben einen guten Ruf. Die Genauigkeit der Daten kann nicht bewertet werden, da die Daten jedoch von der Stadt Leipzig bereits für Simulationszwecke genutzt werden ist von einer hohen Genauigkeit auszugehen (vgl. Leipzig 3D Energie).

Für den hier implementierten Workflow ist die kontextbezogene Datenqualität mittel bis hoch. Sie haben für das Praxisbeispiel einen hohen Value-added, da sie die einzige praktikable Möglichkeit sind, um die Gebäude der Stadt in 3D nachzubilden. Aus diesem Grund sind sie auch sehr relevant. Die Timeliness lässt sich nicht einschätzen, da für das 3D-Stadtmodell kein Erstellungsjahr angegeben ist. Die Completeness und Appropriate Amount of Data sind im Sinne des Anwendungszwecks als hoch einzuschätzen, alle geometrischen Daten zu Gebäuden sind vorhanden.

Die repräsentative Datenqualität ist hoch. Durch den Standard ist die Interpretation der Daten klar definiert. Die Daten sind aufgrund der guten Dokumentation leicht verständlich und die Representational Consistency ist ebenfalls durch den Standard sichergestellt. Für den Anwendungszweck sind die Daten vollständig genug und enthalten weder zu viele noch zu wenige Informationen. Aus diesem Grund ist die Concise Representation gegeben.

Die Zugangsqualität des Leipziger 3D-Stadtmodells ist hoch. Es ist direkt und kostenfrei aus dem Open Data Portal der Stadt herunterladbar.

Für die im Praxisbeispiel genutzten Daten gibt es einen Viewer auf der Website der Stadt Leipzig, dort ist erkennbar, dass die CityGML-Datei ausschließlich Gebäude repräsentiert. Auch die Datei zeigt, dass ausschließlich Polygone dargestellt werden.

5.5.2 OSM

Die intrinsische Datenqualität der OSM-Daten kann hoch eingeschätzt werden, wie bereits in Abschnitt 2.5.3 beschrieben.

Die kontextbezogene Datenqualität der OSM-Daten des Leipziger Zentrums kann als hoch eingeschätzt werden. Die gewählte Overpass-Query liefert eine sehr große Anzahl an Ergebnissen, die einen großen Mehrwert bieten. Da die Daten für eine Anwendung verwendet werden, die Infrastrukturdaten auswertet, ist die Relevancy auch sehr hoch. Die Timeliness variiert, wobei viele Einträge sehr aktuell (< 2 Jahre) sind. Die Datenqualität hinsichtlich der Vollständigkeit und Aktualität der OSM-Daten in Leipzig wird durch ohsome quality analysis (vgl. ohsome) als gut eingeschätzt,

wobei nur die Punktdaten leicht unvollständig sind (Mapping Saturation = 91,97%). Da die OSM-Query und somit die ausgegebenen Inhalte auf den Anwendungsfall zugeschnitten wurde ist der Appropriate Amount of Data gegeben. Auffällig ist, dass die gewählte Overpass Query größtenteils nur Punktdaten zurückgibt, was jedoch an der gewählten Query liegt.

Die repräsentative Datenqualität der OSM-Daten ist, wie bereits in Abschnitt 2.5.3 beschrieben, mittel bis hoch einzuschätzen und die Zugangsqualität ist hoch.

5.5.3 GTFS

Die intrinsische Datenqualität von GTFS-Daten kann, wie in Abschnitt 2.5.4 beschrieben, als hoch eingeschätzt werden.

Die kontextbezogene Datenqualität von GTFS-Daten für den gewählten Anwendungsfall ist hoch, denn den OSM-Objekten im gewählten Gebiet sind keine GTFS-IDs zugeordnet. Dadurch ist ohne die Nutzung der GTFS-Daten keine Auswertung der ÖPNV-Daten möglich. Die Daten für den deutschen ÖPNV werden regelmäßig aktualisiert. Alle nötigen Informationen sind enthalten.

Die repräsentative Datenqualität der GTFS-Daten ist, wie im Abschnitt 2.5.4 beschrieben, hoch. Auch die Zugangsqualität für das gewählte Modell „ÖPNV Deutschland“ ist hoch, da es online zur freien Verfügung steht.

5.5.4 Zusammenfassung

Die ermittelte Datenqualität der einzelnen Daten ist hoch, was bedeutet, dass die Rohdaten für die Weiterverarbeitung geeignet sind. Auffällig ist, dass die CityGML-Daten ausschließlich Flächen enthalten, während die OSM-Daten ausschließlich Punkte enthalten. Dieser Fakt könnte im weiteren Verlauf, bei der Kontrolle der Verknüpfung, eine Rolle spielen. So ist es möglich, dass keine Überschneidungen gefunden werden.

5.6 Datenverknüpfung

Dieser Abschnitt behandelt die Möglichkeiten der Datenumwandlung der Rohdaten in andere Formate, um die zuvor identifizierten mobilitätsrelevanten Daten effektiv zu verknüpfen. Dabei werden die Voraussetzungen, erforderlichen Schritte sowie die auftretenden Herausforderungen detailliert dokumentiert.

5.6.1 Import von CityGML in 3DCityDB

CityGML ist, wie bereits in Abschnitt 2.4.1 genauer beleuchtet wurde, ein komplexes Format mit vielfältigen Konfigurationsmöglichkeiten. Aufgrund dieser Komplexität ist eine Umwandlung der CityGML-Daten in ein handhabbareres Format unerlässlich, um eine effiziente Analyse und Integration mit den weiteren mobilitätsrelevanten Daten zu ermöglichen. Die 3DCityDB (vgl. 3DCityDB) ist ein vielgenutztes Tool zur Verarbeitung von CityGML-Dateien. Durch den Import der Daten in eine auf PostGIS basierende Datenbank wird das umfangreiche und komplex strukturierte CityGML-Format in ein übersichtlicheres und einfacher zu bearbeitendes Datenformat überführt. Dies ermöglicht die Anpassung, Erweiterung, Überprüfung und Korrektur der Daten. Da die Datenbank auf PostGIS[9] basiert, sind geografische Operationen, wie beispielsweise die Ermittlung von Überschneidungsflächen, sehr einfach und effizient durchführbar.

Der 3DCityDB Importer erleichtert den Import der Daten erheblich und unterstützt verschiedene Datenformate, unter anderem auch CityGML. Der 3DCityDB Exporter ermöglicht hingegen den

[9] „PostGIS erweitert die Fähigkeiten der relationalen Datenbank PostgreSQL um Unterstützung für die Speicherung, Indizierung und Abfrage von Geodaten.“(PostGIS).

Export in diverse Formate, sowohl visuelle als auch nicht-visuelle, was die Weiterverarbeitung der Daten vereinfacht. Aufgrund dieser Vorteile wurde entschieden, das 3D-Stadtmodell der Leipziger Innenstadt mithilfe von 3DCityDB weiterzuverarbeiten.

Um den 3DCityDB Importer/Exporter nutzen zu können, ist ein Java Runtime Environment[10] erforderlich. Zudem benötigt das Tool eine Datenbank. Die 3DCityDB kann entweder als Anwendung mit grafischer Benutzeroberfläche oder über ein Command Line Interface (kurz CLI) verwendet werden. Die Anwendung kann direkt von der Website heruntergeladen werden, alternativ besteht die Möglichkeit, Docker-Images[11] zu nutzen.

3DCityDB lässt den Anwender*innen eine gewisse Freiheit in der Auswahl, welches Java Runtime Environment und welche Datenbanksoftware genutzt werden soll. Bei der Auswahl der Datenbank fiel die Entscheidung in diesem Fall auf *PostgreSQL*, da sie als einzige vollständig kostenfrei ist und die Installation von PostGIS unterstützt. Dies erleichtert die Arbeit mit Geodaten erleichtert. Als Java Runtime Environment wurde *Oracle Java* gewählt. Weiterhin wurde die Verwendung von Docker-Images aus mehreren Gründen bevorzugt. Zum einen wurde der Workflow auf einem Rechner mit dem Betriebssystem *macOS* entwickelt, und die grafische 3DCityDB-Anwendung entspricht nicht den Sicherheitsanforderungen des Betriebssystems. Daher müssen vor der Nutzung die Sicherheitseinstellungen überprüft und die 3DCityDB explizit zugelassen werden. Zum anderen bietet Docker Vorteile hinsichtlich der Automatisierung, da Anwendungen in isolierten Umgebungen laufen, was die Installation einzelner Pakete schnell und unkompliziert macht.

Aufgrund fehlender Vorkenntnisse mit PostgreSQL, Docker und der 3DCityDB war der erste Start der Datenbank und der erste Datenimport mit zahlreichen Fehlern behaftet. Ein weiterer Grund hierfür ist die Vielzahl an Konfigurationsoptionen des 3DCityDB Importers (vgl. 3DCityDB Import), die zwar detailliert dokumentiert, aber unübersichtlich sind. Nachdem jedoch der erste erfolgreiche Import durchgeführt werden konnte, traten keine weiteren Probleme mehr auf. Für den Import wurden folgende Terminal-Befehle verwendet:

1. Erstellen der PostgreSQL-Datenbank

```
docker run -d --name cdb --network citydb-net \
    -e POSTGRES_PASSWORD=changeMe! \
    -e SRID=25833 \
    -p 5432:5432 \
    3dcitydb/3dcitydb-pg
7a64d31c9b98f1199da0ef61611db3caa3a676dac6ca8d5468cbd269f8e853f5
```

Dieser Befehl startet einen Docker-Container mit PostgreSQL und der 3DCityDB-Erweiterung, welche PostGIS beinhaltet. Wichtig ist dabei die Angabe des *Spatial Reference Identifiers* (SRID), welcher zuvor aus der CityGML-Datei ausgelesen wurde[12]. Der Container wird im Netzwerk citydb-net erstellt. Die letzte Zeile im Terminal zeigt die Docker-Container-ID an, was signalisiert, dass das Erstellen erfolgreich war.

2. Durchführen des Datenimports:

```
docker run -i -t --name impexp --rm --network citydb-net \
    -v /Users/Pfad/zu/CityGML/Ordner:/data \
    3dcitydb/impexp:latest \
    import -H cdb -P 5432 -d postgres -u postgres -p changeMe! \
```

[10] 3DCityDB verwendet Java-Basierten Code und benötigt aus diesem Grund ein Java Runtime Environment, welches die Ausführung von Java ermöglicht.

[11] Docker ist eine Open-Source-Plattform, die es ermöglicht, Anwendungen und ihre Abhängigkeiten in isolierten Containern zu verpacken, bereitzustellen und auszuführen, wodurch eine konsistente Umgebung über verschiedene Systeme hinweg gewährleistet wird. Docker-Container sind Linux-basiert.

[12] In der CityGML-Datei wird das räumliche Referenzsystem im Feld `srsName` spezifiziert, wie es durch den Standard festgelegt ist. Auf die Relevanz der CRS und die existenten Unterschiede wird im Abschnitt 5.6.3 genauer eingegangen.

```
/data/GEB3D_Leipzig_LoD2_00_Zentrum.gml
```

Dieser Befehl führt den Import der CityGML-Datei in die PostgreSQL-Datenbank durch. Wichtig ist, dass der Container im selben Netzwerk citydb-net gestartet wird. Der erfolgreiche Abschluss des Imports ist durch die Ausgabe `[19:46:40 INFO] Database import successfully finished` im Terminal erkennbar.

Zusammenfassung der Hürden und aufgetretenen Probleme

Während der Einrichtung und Nutzung der 3DCityDB traten mehrere Herausforderungen auf. Die Konfiguration des 3DCityDB Importers gestaltete sich aufgrund der Vielzahl an Optionen als komplex und fehleranfällig, insbesondere für Nutzer*innen ohne Vorkenntnisse in PostgreSQL und Docker. Die fehlende Erfahrung führte dazu, dass der erste Datenimport mehrfach fehlschlug, bevor eine erfolgreiche Implementierung erreicht werden konnte. Für Nutzer*innen, welche bereits Erfahrung mit der Arbeit mit Docker und PostgreSQL haben, wäre die Nutzung der 3DCityDB inklusive des Importer/Exporter-Tools vermutlich einfacher.

5.6.2 Import OSM in PostGIS

Da die Daten des 3D-Stadtmodells bereits in eine PostGIS-Datenbank importiert wurden, ist es sinnvoll, auch die OSM-Daten in eine solche Datenbank zu importieren, um die Vorteile von PostGIS, wie die effiziente räumliche Analyse, zu nutzen. Die OSM-Daten sollten jedoch in eine separate Datenbank importiert werden, um später Konflikte beim Export aus der 3DCityDB mit Hilfe des Exporters zu vermeiden. Aus diesem Grund wurde ein zusätzlicher PostgreSQL-Container mit PostGIS-Erweiterung in Docker erstellt. Dieser befindet sich im selben Netzwerk wie der 3DCityDB-Container, um die Kommunikation zwischen den Datenbanken zu ermöglichen.

Für den Import der OSM-Daten in die PostGIS-Datenbank bietet sich osm2pgsql an, welches auch im OSM-Wiki als „Programm für das Importieren von XML-Dateien in PostGIS-Datenbanken" vorgestellt wird (vgl. OpenStreetMapevelop). Osm2pgsql steht ebenfalls als Docker-Container zur Verfügung. Der Import erfolgte mit den folgenden Terminal-Befehlen:

1. Erstellen der PostgreSQL-Datenbank

```
docker run -d \
  --network citydb-net \
  -v osm-data:/var/lib/postgresql/data \
  -e POSTGRES_USER=postgres \
  -e POSTGRES_PASSWORD=postgres \
  -e POSTGRES_DB=osm_db \
  -p 5436:5432 \
  --name postgis_osm \
  postgis/postgis
579d4868eed709d4b97f4d29cd5ad6dfb17c4a2596238f3b3da8e829fa2e6ac9
```

Dieser Befehl startet in dem im letzten Schritt bereits erstellten Docker-Netzwerk citydb-net einen weiteren PostgreSQL-Container mit PostGIS-Erweiterung. Die letzte Zeile im Terminal zeigt auch hier wieder die Docker-Container-ID an, was signalisiert, dass das Erstellen erfolgreich war.

Anschließend können in der Theorie bereits die OSM-Daten mittels osm2pgsql in die Datenbank importiert werden. Dabei kam es jedoch zu Problemen: Einerseits fehlte anfangs die hstore[13]-Erweiterung in der Datenbank, diese kann jedoch einfach durch folgenden Terminal-Befehl hinzugefügt werden:

[13] Hstore ist eine PostgreSQL-Erweiterung, die es ermöglicht, Key-Value-Paare, wie sie in OSM vorkommen, in einer einzigen Tabellenzelle zu speichern und zu durchsuchen (vgl. PostgreSQL).

2. Hinzufügen der hstore-Erweiterung

```
docker exec -it postgis_osm psql -U postgres -d osm_db -c "CREATE
EXTENSION IF NOT EXISTS hstore;"
```

Anschließend sollte die OSM-Datei durch folgenden Befehl importiert werden:

3. Durchführen des Datenimports auf Basis der XML-Datei

```
docker run --rm \
  --network citydb-net \
  -v /Users/Pfad/zu/exports/Ordner:/osm-data \
  -e PGPASSWORD=postgres \
  iboates/osm2pgsql:latest \
  osm2pgsql \
  -H postgis_osm \
  -P 5432 \
  -d osm_db \
  -U postgres \
  --create \
  --slim \
  --hstore \
  /osm-data/overpass_result.xml
```

Beim Versuch, die OSM-Daten mit osm2pgsql zu importieren, trat jedoch folgende Fehlermeldung auf:

```
ERROR: No element inside <member> allowed
```

Dies war ungewöhnlich, da die OSM-Daten direkt aus der Overpass-API exportiert wurden und somit fehlerfrei sein sollten. Aufgrund der Größe der XML-Datei (über 20.000 Zeilen) war es nicht praktikabel, die Ursache des Fehlers manuell zu identifizieren. Der Import wurde verhindert, da osm2pgsql den Vorgang aufgrund der fehlerhaften Struktur abbrach. Eine Umwandlung der OSM-Daten mit Osmosis löste das Problem. Hierbei wurde die XML-Datei mit Hilfe des Programms, das von der offiziellen GitHub-Seite heruntergeladen werden musste, in das PBF-Format[14] konvertiert.

```
osmosis --read-xml file=overpass_result.xml
--write-pbf file=export.pbf
```

Der Import der so umgewandelten Daten funktionierte anschließend fehlerfrei mit osm2pgsql.

4. Durchführen des Datenimports auf Basis der PBF-Datei

```
docker run --rm \
  --network citydb-net \
  -v /Users/Pfad/zu/exports/Ordner:/osm-data \
  -e PGPASSWORD=postgres \
  iboates/osm2pgsql:latest \
  osm2pgsql \
  -H postgis_osm \
  -P 5432 \
  -d osm_db \
  -U postgres \
  --create \
  --slim \
  --hstore \
  /osm-data/export.pbf
```

[14] Bei dem Protocolbuffer Binary Format (PBF) handelt es sich um ein speziell für OSM-Daten entwickeltes Format, welches XML ablösen soll (OpenStreetMap PBF).

Dieser Befehl führt den Import der OSM-Daten im PBF-Format in die PostgreSQL-Datenbank durch. Der erfolgreiche Abschluss des Imports ist durch eine detaillierte Ausgabe im Terminal erkennbar, bei der beispielsweise folgende Informationen geliefert werden:

```
2025-01-04 16:01:56    Processed 1319 nodes in 0s - 1k/s
2025-01-04 16:01:56    Processed 747 ways in 0s - 747/s
2025-01-04 16:01:56    Processed 74 relations in 0s - 74/s
```

Auch wenn in dieser Ausgabe keine importierten Polygone aufgeführt wurden, schien der Import in die PostGIS-Datenbank zunächst erfolgreich, die ursprüngliche OSM-Query umfasste schlicht keine Polygone. Um jedoch den weiteren Workflow mit Polygonobjekten zu testen, wurde die Import-Query angepasst, sodass auch Flächen ausgewählt wurden. Erst dabei zeigte sich, dass der Import fehlerhaft war: Polygone wurden nicht als solche erkannt, sondern entweder gar nicht oder nur als Ways importiert. Da Polygone geschlossene Ways sind, bei denen der erste und letzte Punkt identisch sein müssen, wurde daraufhin eine Überprüfung auf geschlossene Ways in der OSM-Datenbank durchgeführt.

Query 1: Diese Abfrage überprüft, ob die Punkte eines Ways eine geschlossene Geometrie, also eine geschlossene Linie, bilden. Dafür werden die Koordinaten aller Knoten eines Ways zu einer Linie zusammengefügt, und mit der Funktion ST_IsClosed wird geprüft, ob der erste und der letzte Punkt der Linie identisch sind.

```
WITH node_points AS (
  SELECT
    p.id AS way_id,
    ST_SetSRID(ST_Point(n.lon, n.lat), 4326) AS point_geom
  FROM
    planet_osm_ways p
  JOIN
    planet_osm_nodes n ON n.id = ANY(p.nodes)
)
SELECT
  way_id,
  ST_IsClosed(ST_MakeLine(point_geom)) AS is_closed
FROM
  node_points
GROUP BY
  way_id;
```

Query 1 ergab folgendes Ergebnis (Auszug): Dort ist erkennbar, dass ein Teil der Ways geschlossen sind (`is_closed = t`).

```
   way_id   | is_closed
------------+-----------
    4362256 | t
    4362264 | t
    7947538 | t
    7947539 | t
    7947540 | t
    7947555 | t
    7947557 | t
    7947867 | t
    7947956 | t
    8084201 | f
   13480804 | f
...
```

Query 2: Diese Abfrage prüft, ob der erste und der letzte Knoten eines Ways in der Datenbank dieselbe node_id besitzen. Dies wird direkt anhand der Knotenreferenzen des Ways festgestellt, ohne die tatsächlichen Koordinaten der Punkte zu betrachten.

```
SELECT
  id AS way_id,
  CASE
    WHEN nodes[1] = nodes[array_length(nodes, 1)] THEN TRUE
    ELSE FALSE
  END AS is_closed
FROM
  planet_osm_ways;
```

Query 2 ergab folgendes Ergebnis (Auszug): Es ist erkennbar, dass ein Großteil der Ways geschlossen sind (`is_closed = t`).

```
   way_id   | is_closed
------------+-----------
    8031319 | t
   18920139 | t
   22734771 | t
   23566182 | t
   23732323 | t
   24376835 | t
   24377010 | t
   24386998 | t
   24391877 | t
   70288358 | t
   24594304 | t
...
```

Obwohl einige geschlossene Ways gefunden wurden, waren nicht alle erwarteten Objekte vorhanden. Von den insgesamt 634 Gebäuden, deren Import als Polygon erwartet wurde, wurden nur rund 70 als geschlossene Ways importiert, der Rest fehlte komplett. Ein automatisches Reparieren der Daten war nicht möglich, und auch der Import mit anderen Rohdaten führte zu denselben Fehlern. Die von osm2pgsql verwendete Style-Datei, welche darüber entscheidet, wann Ways zu Polygonen umgewandelt werden, wurde kontrolliert und auch diese war fehlerfrei. Es wurde zunächst vermutet, dass der frühere Fehler `ERROR: No element inside <member> allowed`, der bei der Konvertierung mit Osmosis auftrat, die PBF-Datei beschädigt haben könnte. Um dies zu prüfen, wurde der Prozess mit Daten aus einem anderen Stadtgebiet wiederholt. Zwar trat dort der ursprüngliche Fehler nicht auf, dennoch blieben die Probleme beim Import von Polygonen bestehen.

Auch nach längerer Analyse konnte die Ursache nicht identifiziert werden. Nutzer*innen mit mehr Erfahrung in osm2pgsql oder Osmosis könnten möglicherweise eine Lösung finden.

Zusammenfassung der Hürden und aufgetretenen Probleme

Der Import der OSM-Daten in die PostGIS-Datenbank führte zunächst zu einer Fehlermeldung, weshalb die XML-Datei mithilfe von Osmosis in das PBF-Format umgewandelt werden musste. Nach Erweiterung der OSM-Rohdaten um Polygonobjekte zeigte sich, dass manche erwartete Polygone als unvollständige Ways importiert wurden, ein Großteil wurde gar nicht importiert. Trotz Überprüfung der Datenstruktur, etwa durch Queries zur Erkennung geschlossener Ways, konnten die fehlenden Polygone nicht identifiziert werden. Wiederholte Importe mit anderen Rohdaten führten zu denselben Problemen.

5.6.3 Vorgehen beim Verknüpfen der Daten

Dieser Abschnitt beschreibt detailliert nötige Überlegungen und Ansätze sowie mögliche Probleme bei der Verknüpfung von OSM-, GTFS- und CityGML-Daten.

Koordinatensysteme

Im Verlauf der Arbeit wurde bereits mehrfach auf die unterschiedlichen Koordinatensysteme der für den Workflow relevanten Rohdaten hingewiesen. Diese CRS sind insbesondere bei der Datenverknüpfung von zentraler Bedeutung. Damit die Zuordnung zuverlässig und performant erfolgen kann, ist es wichtig, dass alle Daten in einem einheitlichen Koordinatensystem vorliegen.

Die OSM- und GTFS-Daten liegen standardmäßig im geografischen Koordinatensystem WGS84 vor, das die Positionen der Objekte in Breiten- und Längengraden angibt. Im Gegensatz dazu verwenden CityGML-Daten häufig projektionsbasierte Koordinatensysteme, wie beispielsweise EPSG:25833 (aus dem im Praxisbeispiel genutzten 3D-Stadtmodell), das metrische Koordinaten bereitstellt. Der Hauptunterschied besteht darin, dass geografische Koordinatensysteme auf einem Gradnetz basieren und die Erdoberfläche als Kugel modellieren, während projektionsbasierte Koordinatensysteme die Erdoberfläche auf eine Ebene projizieren und Distanzen direkt in metrischen Einheiten angeben.

Die Wahl eines geeigneten Koordinatensystems ist entscheidend, da räumliche Berechnungen wie Überschneidungs- und Abstandsfunktionen in PostGIS in einem projektionsbasierten Koordinatensystem deutlich schneller ausgeführt werden können als in einem geografischen Koordinatensystem (vgl. PostGIS FAQ). Aus diesem Grund wurde entschieden, die OSM- und GTFS-Daten in Python in das Koordinatensystem EPSG:25833 zu transformieren, um eine konsistente Grundlage für effiziente räumliche Analysen und die Umsetzung der Zuordnungsregeln zu schaffen.

Zuordnungsregeln

Die Datenzuordnung kann sich als komplexe Herausforderung erweisen. Abhängig von der Datenqualität und aufgrund des unterschiedlichen Informationsgehaltes der Quellen können Probleme auftreten. Nicht jede CityGML-Datei enthält beispielsweise Informationen zu Straßen oder Wegen, häufig sind nur Gebäude erfasst. Die Zuordnung wegebezogener Informationen aus OSM zu den CityGML-Daten gestaltet sich in solchen Fällen schwierig. Ebenso fehlen Points of Interest (POIs)[15] zum Teil in 3D-Stadtmodellen. Sollen die Attribute von OSM-Objekten zu CityGML-Objekten zugeordnet werden, so muss entschieden werden, in welchen Fällen Flächen, Punkte und Wege als einander zugehörig gewertet werden können. Weiler et al. (2018) beschreiben Ansätze zur Datenverknüpfung basierend auf der geografischen Lage und nennen dabei zwei Fälle:

- OSM-POIs innerhalb eines CityGML-Gebäudegrundrisses
- OSM-POIs außerhalb eines CityGML-Gebäudegrundrisses

Für den in dieser Arbeit betrachteten Anwendungsfall sind jedoch weitere Szenarien relevant. In CityGML gibt es grundsätzlich zwei Arten von Geometrien: Flächen und Punkte. Aus OSM werden folgende Datenarten extrahiert:

- Nodes – Punkte, welche markante Stellen (POIs) kennzeichnen
- Ways – Liniensegmente, die mindestens zwei Nodes verbinden
- Ways – geschlossene Liniensegmente, die eine Fläche umschließen (Anfangs- und End-Node sind identisch)

Für die Zuordnung der GTFS-Daten wurde entschieden, ausschließlich die GTFS-IDs den OSM-Nodes der Haltestellen zuzuordnen. Hierfür existieren etablierte OSM-Keys wie `gtfs_id` und `gtfs:stop_id`. Es wurde sich für die Nutzung des Keys `gtfs_id` entschieden, da er verbreiteter ist. Aufgrund der Komplexität der weiteren GTFS-Daten, wie etwa der

[15] Relevante Geografische Punkte, wie OSM-Nodes, können als Points of Interest, also interessante Punkte, bezeichnet werden.

Streckenverläufe und Abfahrtspläne, werden diese nicht direkt in den Datensatz integriert. Sie können jedoch später anhand der zugeordneten `gtfs_id` verarbeitet und genutzt werden.

Die möglichen Fälle für die Datenverknüpfung und ihre Auflösungsansätze werden in Tabelle 8 detailliert dargestellt. Zur Unterscheidung der beiden Arten von Ways wird in der Tabelle der Begriff *Areas* für geschlossene Liniensegmente, die eine Fläche umschließen, verwendet.

Fall	**mögliche Auflösung**	**gewählte Auflösung**
OSM-Node liegt genau auf einem CityGML-Point	• keine Auflösung nötig, Zuordnung aller OSM-Attribute zu CityGML-Objekt	• keine Auflösung nötig, Zuordnung aller OSM-Attribute zu CityGML-Objekt
OSM-Polygon liegt genau auf einer CityGML-Fläche		• keine Auflösung nötig, Zuordnung aller OSM-Attribute zu CityGML-Objekt
OSM-Polygon liegt komplett innerhalb einer CityGML-Fläche	• ganzer CityGML-Fläche die Attribute zuordnen • bis zu Schwellenwert: Zuordnung der OSM-Attribute zu gesamter CityGML-Fläche • bis zu Schwellenwert: Zuordnung der OSM-Attribute zu gesamter CityGML-Fläche, ansonsten Platzhalterobjekt erzeugen	• Anteil OSM-Polygon an CityGML-Fläche ≥ Schwellenwert: OSM-Attribute werden der CityGML-Fläche zugeordnet. • Anteil OSM-Polygon an CityGML-Fläche < Schwellenwert: keine Zuordnung findet statt
OSM-Polygon und CityGML-Fläche überschneiden sich teilweise		• Anteil OSM-Polygon an CityGML-Fläche ≥ Schwellenwert: OSM-Attribute werden der CityGML-Fläche zugeordnet. • Anteil OSM-Polygon an CityGML-Fläche < Schwellenwert: keine Zuordnung findet statt
OSM-Node liegt nicht genau auf einem CityGML-Point	• bis zu Schwellenwert: Zuordnung der OSM-Attribute zu CityGML-Punkt • Erzeugen eines Platzhalters	• Abstand OSM-Punkt zu CityGML-Punkt ≤ Schwellenwert: OSM-Attribute werden CityGML-Punkt zugeordnet • Abstand OSM-Punkt zu CityGML-Punkt > Schwellenwert: Keine Zuordnung findet statt
OSM-Way liegt innerhalb einer CityGML-Fläche	• Zuordnung der OSM-Attribute nicht zu CityGML-Flächen, stattdessen erzeugen von Platzhalterobjekten • falls Way durch eine CityGML-Fläche verläuft: Prüfen der Eigenschaften der CityGML-Fläche, falls Fläche eindeutig als Weg identifiziert werden kann: Zuordnung OSM-Tags zu CityGML-Fläche	• keine Zuordnung findet statt
CityGML-Fläche liegt komplett innerhalb eines OSM-Polygons	• Zuordnung der OSM-Attribute zu CityGML-Fläche	• Zuordnung der OSM-Attribute zur CityGML-Fläche

Fall	mögliche Auflösung	gewählte Auflösung
GTFS-Punkt liegt genau auf einem OSM-Node	• keine Auflösung nötig, Zuordnung der GTFS-ID zu OSM-Node	• keine Auflösung nötig, Zuordnung der GTFS-ID zu OSM-Node
GTFS-Punkt liegt genau auf einem OSM-Node	• bis zu Schwellenwert: Zuordnung der GTFS-ID zu OSM-Node	• Abstand GTFS-Punkt zu OSM-Node ≤ Schwellenwert: GTFS-ID wird OSM-Node zugeordnet • Abstand GTFS-Punkt zu OSM-Node > Schwellenwert: Keine Zuordnung findet statt

Tabelle 8: Regeln für die Datenverknüpfung

Ein Problem bei der Verknüpfung der Daten liegt darin, dass OSM-Ways Linien repräsentieren, während CityGML-Objekte Wege als Flächen darstellen. Eine direkte geometrische Überlagerung ist daher nicht möglich. Die festgelegten Verknüpfungsregeln werden mit Hilfe eines Python-Skripts umgesetzt.

Implementierung

Die 3DCityDB und die PostGIS-Datenbank, in der die OSM-Daten gespeichert sind, können aufgrund ihrer Trennung in unterschiedliche Docker-Container nicht direkt über SQL-Queries miteinander verknüpft werden. Dies ist nur innerhalb einer Datenbank möglich. Die Nutzung von PostGIS zur Datenüberschneidung ist jedoch sinnvoll, da es komplexe SQL-Abfragen zur Analyse geografischer Daten ermöglicht. Ein Import der OSM-Daten in ein neues Schema der 3DCityDB wäre zwar technisch machbar, könnte jedoch Probleme mit dem später benötigten 3DCityDB-Exporter verursachen. Daher wurde entschieden, die Verknüpfung über eine dritte PostGIS-Datenbank mit drei separaten Schemas für CityGML-, OSM- und GTFS-Daten vorzunehmen.

Beim Import der Daten in die PostGIS-Datenbank müssen die Koordinaten aufgrund der unterschiedlichen CRS in ein geeignetes Format umgewandelt werden. Da Überschneidungs- und Abstandsfunktionen in PostGIS bei der Verwendung von metrischen Koordinaten erheblich schneller ausgeführt werden können als bei WGS84-Koordinaten, wurde entschieden, das Koordinatensystem der CityGML-Datei zu übernehmen (EPSG:25833). Die OSM- und GTFS-Daten wurden hierfür in Python entsprechend transformiert, sodass alle Daten im gleichen Koordinatensystem vorliegen und effiziente räumliche Analysen möglich sind.

Für die CityGML-Daten wurde das Schema `citygml` mit den Tabellen `citygml.points` und `citygml.areas` erstellt. Dies ermöglicht eine gezielte Verarbeitung nach Objekttyp, wodurch beispielsweise Flächenverknüpfungen effizienter durchgeführt werden können, da nur die relevanten Flächenobjekte berücksichtigt werden müssen. Die OSM-Daten wurden aufgrund der engen Verknüpfung von Nodes und Ways in eine einzige Tabelle namens `osm.objects` im Schema `osm` importiert. Diese Tabelle enthält Spalten für die Geometrie, den Objekttyp (Point, Line, Polygon) und die zugehörigen Attribute. Für die GTFS-Daten wurde ein eigenes Schema `gtfs` mit der Tabelle `gtfs.stops` erstellt. Diese enthält Informationen zu Haltestellen, wie `stop_id` und `stop_name`, geografische Koordinaten und die Geometrie. Der Import der GTFS-Daten hat sich als sehr zeitaufwendig herausgestellt, dieser Prozess dauert zwischen 3 und 5 Minuten.

Die Verknüpfung der Daten erfolgte in einem vierten Schema `matching`, welches eine Verknüpfungstabelle mit dem Namen `matching.results` enthält. Diese Tabelle hat die Spalten `citygml_id` und `osm_id` für die eindeutige Identifizierung der verknüpften Objekte. Außerdem gibt es die Spalte `match_case`, die beschreibt, welcher der Fälle zur Verknüpfung geführt hat, und

die Spalte `attributes`, die die OSM-Tags beinhaltet, die dadurch dem CityGML-Objekt zugeordnet werden sollen.

Die Überschneidung der OSM-Objekte mit den CityGML-Daten anhand der Fläche erweist sich grundsätzlich als komplexes Unterfangen. Ein dreidimensionales CityGML-Objekt besteht aus mehreren Polygonen, beispielsweise für Wände, Grundfläche und Dach. Die Verknüpfung mit OSM-Daten hat also das Problem, dass sich die OSM-Area mit mehreren Polygonen eines CityGML-Objektes überschneiden kann. Jedes dieser Unterobjekte ist eindeutig einem CityGML-Objekt zugeordnet (So ist eine Wand zum Beispiel dem ganzen Haus zugeordnet), dennoch sollten Dopplungen der Zuordnung vermieden werden. Die Überschneidung mit Wandflächen wird in PostGIS automatisch ignoriert, da CityGML-Polygone, deren Eckpunkte zu nah beieinanderliegen, aufgrund ihrer Kolinearität von PostGIS (fehlerhaft) als invalide erkannt werden, und diese Objekte aus dem Prozess der Verknüpfung ausgeschlossen wurden[16]. Nun bleibt jedoch die Aufgabe, die vorhandenen Flächen eindeutig ihren OSM-Objekten zuzuordnen und die OSM-Tags eindeutig zu verknüpfen. Ob diese Zuordnung funktioniert, konnte anhand des Praxisbeispiels nicht geprüft werden. Dies liegt einerseits daran, dass die CityGML-Datei ausschließlich Flächen enthält, während die OSM-Daten ausschließlich Punkte enthalten, wodurch keine Überschneidungen möglich sind. Andererseits könnten, auch wenn die OSM-Daten Flächen enthalten würden, aufgrund der bereits genannten Probleme beim OSM-Import keine sich überschneidenden Flächen ermittelt werden. Ein ähnliches Problem trat bei der Überschneidung der GTFS-Punkte mit den Overpass-Punkten auf. Der osm2pgsql-Import hat zwar Nodes in die DB überführt, jedoch sind die überführten Nodes nicht vollständig. Viele OSM-Nodes, die Haltestellen repräsentieren, sind darum verloren gegangen und somit konnten keine Überschneidungen ermittelt werden.

Aufgrund dieser Probleme konnte die Zuordnung der durch das Matching als relevant erfassten OSM-Tags zu den CityGML-Grundobjekten nicht implementiert werden. Weiterhin konnten die Matching-Cases mit den vorhandenen Daten zwar in Python implementiert, jedoch nicht zuverlässig überprüft werden. Es ist möglich, dass mit korrekt umgewandelten und importierten Daten weitere Bugs auftreten und behoben werden müssen. Zur Fehleranalyse und Verbesserung ist eine detaillierte Untersuchung der Datenverluste beim Import über osm2pgsql erforderlich.

Für den Fall, dass solche Probleme bei der Datenverknüpfung häufiger auftreten, könnte ein weiterführender Workflow als Fallback implementiert werden. So wäre es möglich, alle OSM- und GTFS-Objekte, welche nicht mit einem anderen Objekt verknüpft werden konnten, durch Platzhalterobjekte zu repräsentieren. Ansätze dazu sind in Tabelle 9 beschrieben. Dies würde auch das bereits beschriebene Problem beheben, dass OSM-Ways kaum bis gar nicht zu CityGML-Objekten zugeordnet werden können. Die so erzeugten Platzhalterobjekte könnten wiederum in CityGML-kompatible Geometrien überführt und in 3DCityDB importiert werden, wobei ihnen die relevanten OSM-Tags zugeordnet werden können. Anschließend können sie gemeinsam mit dem 3D-Stadtmodell exportiert werden.

Fall	**mögliche Auflösung**
OSM-Area kann keiner CityGML-Fläche zugeordnet werden	Extrudieren der Grundfläche um 0,5m in die Höhe, um 3D-Platzhalter zu erstellen
OSM-Ways können keiner CityGML-Fläche zugeordnet werden	Verbreitern des Weges um 0,5m und anschließendes Extrudieren der entstandenen Grundfläche um 0,5m in die Höhe, um 3D-Platzhalter zu erstellen

[16] Die Objekte sind grundsätzlich gültige Polygone, sie haben mehr als 4 eindeutige Punkte und der Anfangs- und Endpunkt sind, gleich. Da die PostGIS Verarbeitung die Punkte jedoch auf eine 2D-Form umrechnet, liegen die Punkte (beispielsweise einer senkrecht stehenden Wand) auf einer Linie und werden darum mit einer Fläche von 0 als invalide erkannt.

Fall	mögliche Auflösung
OSM-Node kann keinem CityGML-Punkt zugeordnet werden	Erzeugen eines Würfels mit Kantenlänge 0,5m, dessen Mittelpunkt den Koordinaten des OSM-Nodes entspricht
GTFS-Punkt kann keinem OSM-Node zugeordnet werden	Erzeugen eines Würfels mit Kantenlänge 0,5m, dessen Mittelpunkt den Koordinaten des GTFS-Punkts entspricht

Tabelle 9: Mögliches Vorgehen beim Erstellen von Platzhalterobjekten

Zusammenfassung der Hürden und aufgetretenen Probleme

Das schwerwiegendste Problem bei der Verknüpfung der Daten ist der fehlerhafte Import der OSM-Daten durch osm2pgsql. Dadurch fehlten viele relevante OSM-Punkte und -Flächen, was zur Folge hatte, dass keine Überschneidungen mit den CityGML-Daten ermittelt werden konnten. Ohne diese Überschneidungen war eine Weiterverarbeitung der Daten nicht möglich.

Eine generelle Hürde in diesem Arbeitsschritt stellt die Überschneidung der Flächen dar. Da CityGML-Objekte aus mehreren Polygonen bestehen, beispielsweise für Wände, Dächer und Grundflächen, führt dies zu komplizierten Überschneidungen mit OSM-Flächen.

Ein weiteres Hindernis, welches die Verarbeitung verkompliziert, ist die Trennung der 3DCityDB und der PostGIS-Datenbank in unterschiedliche Docker-Container. Die Idee eines direkten Imports der OSM-Daten in die 3DCityDB wurde verworfen, da dies potenziell den Exporter beeinträchtigen könnte. Weiterhin müssen vor Beginn der Datenverknüpfung genaue Regeln festgelegt werden, die genutzt werden sollen, um Objekte einander zuzuordnen.

Der Datenimport, besonders der Import der GTFS-Daten, verursacht aufgrund der hohen Datenmenge lange Wartezeiten.

5.7 Export der Daten aus 3DCityDB

Der Export von Daten aus der 3DCityDB kann in verschiedenen Formaten erfolgen, darunter CityGML, CityJSON, KML, Collada und glTF. Wenn die Daten später in einer Game-Engine wie Unity verwendet werden sollen, empfiehlt sich das glTF-Format. Dieses ermöglicht die Kombination von semantischen und geometrischen Informationen und kann problemlos in Blender geladen und anschließend für die Verwendung in Unity bearbeitet und exportiert werden. Um glTF-Dateien mit dem 3DCityDB-Exporter zu erzeugen, muss zunächst ein Export im Collada-Format erfolgen.

Um den Export zu testen, sollte das Tool COLLADA2GLTF im Wurzelverzeichnis des Rechners installiert werden, damit es von überall aus aufgerufen werden kann. Da dies nicht wie geplant funktionierte, wurde stattdessen die ausführbare Binärdatei aus den Downloads genutzt.

```
docker run -i -t --name impexp --rm --network citydb-net \
  -v /Users/Pfad/zu/Exportdateien/3DCityDB_Exports:/output \
  -v ~/Downloads/COLLADA2GLTF-v2/COLLADA2GLTF-bin:/COLLADA2GLTF \
  3dcitydb/impexp:latest \
  export-vis -H cdb -P 5432 -d postgres -u postgres -p changeMe! \
  -o /output/citydb_export \
  --display-form collada \
  --lod 2 \
  -G --remove-collada \
  --gltf-converter=/COLLADA2GLTF
```

Dieser Befehl startet einen neuen Docker-Container innerhalb des bestehenden Netzwerks, führt den 3DCityDB-Exporter aus und konvertiert die exportierten Daten mithilfe von COLLADA2GLTF in das glTF-Format. Die ursprünglichen Exportdateien werden dabei durch die glTF-Dateien ersetzt.

Beim Export kam es jedoch zu dem Problem, dass jedes Gebäude als einzelne glTF-Datei exportiert wurde. Auch der Versuch, durch das Festlegen eines großen Tiles für den Export eine einzige Datei mit allen CityGML-Objekten zu erhalten, blieb erfolglos. Aus diesem Grund wird empfohlen, alle glTF-Dateien anschließend in Blender zu importieren, dort anhand eines Skriptes mit ihren Koordinaten zu positionieren und das Ergebnis wiederum als FBX-Datei aus Blender zu exportieren. Dies könnte das Problem lösen, der Ansatz wurde jedoch nicht praktisch umgesetzt und müsste deshalb auf seine Praktikabilität getestet werden.

5.7.1 Zusammenfassung der Hürden und aufgetretenen Probleme

Beim Export mit dem 3DCityDB Exporter und der Verwendung des COLLADA2GLTF-Konverters traten auf macOS Sicherheitsprobleme auf, die eine manuelle Freigabe des Konverters über die Sicherheitseinstellungen erforderlich machten. Zudem wurde jedes Gebäude als separate glTF-Datei exportiert, und der Versuch, durch ein großes Tile eine einzige Datei für alle CityGML-Objekte zu erzeugen, blieb erfolglos.

5.8 Fazit zur Erarbeitung des Workflows

Dieser Abschnitt hat den Workflow zur Verknüpfung mobilitätsrelevanter Daten detailliert betrachtet. Dabei wurden relevante Daten und deren Quellen identifiziert und beschrieben. Der Datenimport in Datenbanken wurde praktisch umgesetzt, und die Verknüpfung der Daten erfolgte auf Basis zuvor definierter Regeln. Der Prozess gestaltete sich insgesamt als komplex, was insbesondere auf die Vielzahl einzelner Schritte sowie die unterschiedlichen Import- und Exportformate zurückzuführen ist. Dabei traten verschiedene Hürden auf, von denen einige so gravierend waren, dass nachfolgende Schritte nicht korrekt ausgeführt werden konnten. Im Folgenden werden die zentralen Probleme zusammengefasst. Die Identifikation der mobilitätsrelevanten Daten war zwar zeitaufwändig, aber unkompliziert. Die Überführung in relevante OSM-Tags für die Overpass-Query hingegen erwies sich sowohl als zeitaufwändig, als auch kompliziert. Obwohl Tools zur Zuordnung und Kontrolle der Tags existieren ist die Benutzung aufwendig.

Die Festlegung der Koordinaten für den Overpass-Abruf erfordert eine sorgfältige Planung, ist anschließend jedoch problemlos umsetzbar. Hingegen wurde die Extraktion von Koordinaten aus der CityGML-Datei aufgrund des hohen Aufwands verworfen.

Die Verwendung der Tools 3DCityDB Importer/Exporter und osm2pgsql gestaltete sich trotz umfassender Dokumentation für Anfänger als herausfordernd. Die initiale Einarbeitung war zwar zeitintensiv, jedoch konnte die weitere Nutzung nach einer erfolgreichen Erstkonfiguration durch Wiederverwendung bereits dokumentierter Terminal-Befehle erheblich vereinfacht werden.

Der Import der OSM-Daten in die PostGIS-Datenbank mittels osm2pgsql war, wie bereits genannt, fehlerbehaftet. Trotz intensiver Bemühungen konnten die Fehler nicht behoben werden, was den weiteren Workflow erheblich beeinträchtigte.

Die Festlegung von Verknüpfungsregeln erfordert sorgfältige Planung, ihre Implementierung gestaltet sich jedoch dank der PostGIS-Erweiterung für Datenbanken effizient und unkompliziert.

Der gesamte Prozess ist aufgrund der zahlreichen manuellen Schritte zeitaufwendig und fehleranfällig, insbesondere durch die Eingabe spezifischer Daten. Sobald die Schritte zur Docker-Erstellung, zum Import und zur Verknüpfung einmal erfolgreich durchgeführt wurden, können jedoch getestete Terminal-Befehle wiederverwendet werden, was den Workflow beschleunigt. Müssen Docker-Ports, Dateipfade oder Datenbanknamen häufig angepasst werden, steigt sowohl der Zeitaufwand als auch das Fehlerrisiko erheblich.

In Abbildung 8 ist der gesamte entwickelte Workflow übersichtlich dargestellt.

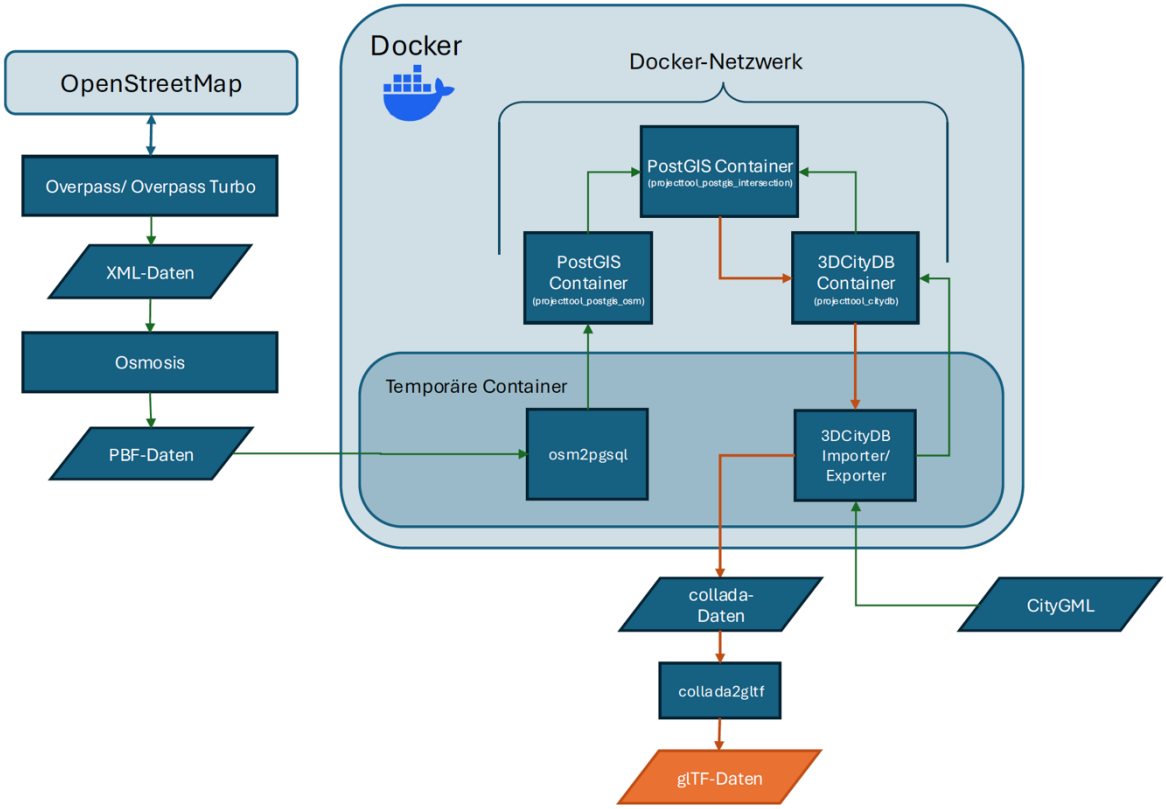


Abbildung 8: Übersicht zum entwickelten Workflow

Der Workflow wurde ausschließlich anhand des gewählten Praxisbeispiels genutzt. Die Verwendung von 3D-Stadtmodellen anderer Gebiete wurde nicht getestet. Dabei ist es theoretisch möglich, dass aufgrund unterschiedlicher Anwendung des CityGML-Standards weitere spezifische Fehler auftreten, welche analysiert werden müssen.

6 Automatisierung in Python

Der Prozess der Datenverknüpfung von 3D-Stadtmodellen, OSM-Daten und GTFS-Daten ist von vielen Schritten geprägt. Die nötigen Terminal-Befehle müssen genau angepasst werden, sodass Dateipfade, Tabellennamen und Netzwerk-IDs stimmen. Aufgrund dieser Kleinteiligkeit kann es bei manueller Ausführung des Datenimports schnell zu zeitaufwendigen Fehlern kommen. Ein fehlerhaftes Erstellen der Container führt zu hohem Aufwand durch nötige Wiederholungen.

Die Verknüpfung der Daten macht ein Kopieren der relevanten Daten in eine neue PostGIS-Datenbank nötig, auf welcher komplexe Berechnungen ausgeführt werden müssen. Um den Verschneidungsprozess zu vereinfachen, bietet sich eine Automatisierung an. Python ist für diesen Zweck eine gute Wahl, da es verschiedene Pakete (wie shapely, GeoPandas, Pyproj und viele weitere) unterstützt, welche sich speziell mit der Verarbeitung von Geodaten befassen. Auch Pakete, die eine direkte Kommunikation mit Docker ermöglichen, existieren, was die Arbeit mit den Containern vereinfacht.

Für die Automatisierung bietet es sich an, die nötigen Schritte zu modularisieren und somit in mehrere verschiedenen Skripte zu zerteilen, die jeweils voneinander abgegrenzte Funktionen ausführen. Daraus ergeben sich zwar mehr nötige Schritte für die Nutzung, jedoch bietet dies den Vorteil, dass fehlerhaft ausgeführte Skripte, unabhängig von der Fehlerursache, erneut gestartet werden können, ohne dass jeder einzelne Prozessschritt erneut durchlaufen werden muss. Besonders hinsichtlich der Dauer der Datenimporte ist es erheblich effizienter, beim Auftreten von Fehlern in bestimmten Schritten nicht alle Daten erneut importieren zu müssen.

Generell wurde sich bei der Automatisierung dafür entschieden, dass die einzelnen Skripte eine Mischung aus grafischem User-Interface (GUI) und Terminal-Ausgaben nutzen. Für die Implementierung des GUI wurde die Nutzung des Python-Packages *tkinter*[17] festgelegt. Über das GUI werden von Nutzer*innen kritische Eingaben abgefragt, beispielsweise ob Daten gelöscht werden sollen. Außerdem wird tkinter genutzt, um die Dateiauswahl durch Dialogfelder zu vereinfachen. Für die zeitgleiche Nutzung des Terminals für Programmausgaben wurde sich aus verschiedenen Gründen entschieden. Einerseits wäre es in Anbetracht des Designs nicht wirklich „schön", alle Terminal-Ausgaben in einem Textfeld des GUI auszugeben. Ein Abfangen aller Fehler in vernünftigen Fehlermeldungen, welche im GUI angezeigt werden, ist aber aufgrund der vielen verschiedenen möglichen Fehler nicht praktikabel. Weiterhin ist der Vorteil der Nutzung des Terminals, dass Fehlermeldungen auch erhalten bleiben, wenn ein Skript beendet wird, oder wenn das Programm bei der Ausführung abstürzt. Bei der Ausgabe im GUI wären die Ausgaben in diesen Fällen verloren, was auch die Fehleranalyse erschwert.

Bei der Automatisierung entstanden schlussendlich fünf Python-Skripte, welche nacheinander ausgeführt werden müssen. Sie führen alle einen bestimmten Schritt des Workflows aus und werden im Folgenden anhand von Sequenzdiagrammen erklärt. Relevante Ausschnitte des Quellcodes werden gezeigt. Obwohl alle Skripte für verschiedene Anwendungszwecke angepasst werden können, werden speziell vorgesehene Anpassungsmöglichkeiten, wenn vorhanden, aufgeführt.

Alle Python-Skripte liegen in einem gemeinsamen Projektordner. Dieser wurde mit dem Namen UrbanLink3D als Namen für das Softwarepaket versehen. Im Projektordner liegen außerdem noch zwei weitere Ordner, einer für die Exporte der Skripte und einen für die von den Skripten erstellten Logs sowie die Konfigurationsdatei, die die Query für den Overpass-API-Abruf bereitstellt (siehe Abbildung 9).

[17] Bei tkinter handelt es sich um die standardmäßige GUI-Bibliothek von Python (vgl. Python tkinter).

Die Skripte bauen aufeinander auf und müssen deswegen in folgender Reihe ausgeführt werden:
check_installation.py ➔ docker_creation.py ➔ overpass_api_call.py ➔ osm_import.py ➔ intersect_data.py

Abbildung 9: Inhalte des Softwarepakets

6.1 check_installation.py

6.1.1 Allgemeine Beschreibung

Das Skript check_installation.py prüft, ob die gesamte, für die Ausführung des Workflows zwingend nötige Software installiert ist. Dabei geschieht eine Unterscheidung in eigenständige Software und Python-Pakete. Zur Software gehören Java Runtime Environment und Docker. Die nötigen Python-Pakete sind:

- psycopg2 (ermöglicht Verbindung zu PostgreSQL)
- shapely (ermöglicht Analyse und Anpassung geometrischer Objekte)
- pyproj (ermöglicht Umrechnung von Koordinaten)
- requests (ermöglicht HTTP-Anfragen für den Overpass-API-Aufruf)
- lxml (ermöglicht Verarbeitung von XML-Dokumenten)
- pandas (ermöglicht Datenanalyse und komplexe Datenstrukturen in Python)
- docker (ermöglicht Interaktion mit Docker)

Das Skript überprüft die Software und Python-Pakete auf ihren Installationsstatus. Die Installation von Osmosis wird nicht geprüft, da die Nutzung von Osmosis im Workflow nicht zwingend erforderlich ist und nur im Falle eines Fehlers geschehen muss. Anschließend öffnet das Skript ein GUI, in dem die einzelnen nötigen Installationen aufgeführt werden. Nicht installierte Software und Pakete werden rot hinterlegt. Während fehlende Software manuell installiert werden muss, bietet check_installation.py die Möglichkeit, fehlende Python-Pakete auf Knopfdruck automatisch zu installieren.

In check_installaton.py kann weitere Software sowie benötigte Python-Pakete zur Prüfung hinzugefügt werden.

Abbildung 10 zeigt in einem Sequenzdiagramm den Ablauf der Prozesse in check_installation.py.

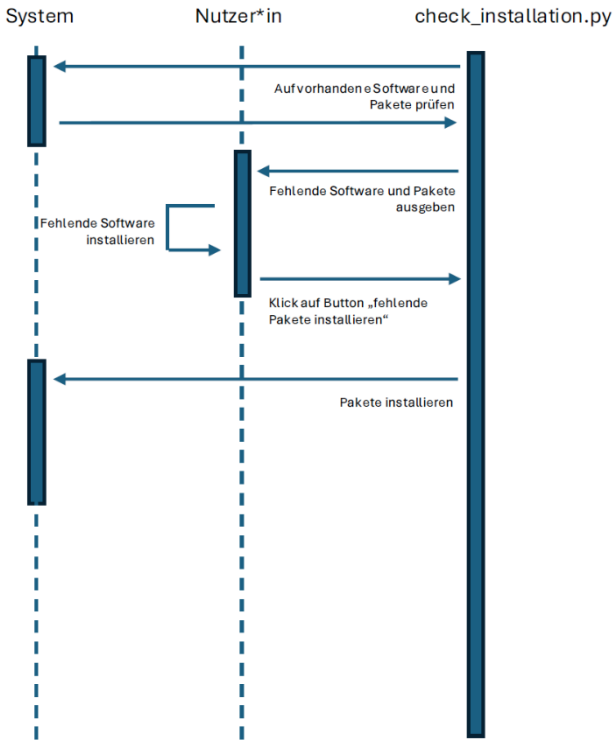


Abbildung 10: Sequenzdiagramm - check_installation.py

6.1.2 Relevante Codeausschnitte

Die benötigten Python-Pakete sind in einer Liste gespeichert. Diese Liste kann einfach durch Angabe der genauen Paketnamen erweitert werden:

```
# Liste der benötigten Python-Pakete
required_packages = [
    'psycopg2',
    'shapely',
    'pyproj',
    'requests',
    'lxml',
    'pandas',
    'docker'
]
```

Die fehlenden Python-Pakete werden auf Eingabe der Nutzer*innen hin durch einen Shell-Befehl installiert:

```
# Installationsbefehl ausführen
subprocess.check_call([sys.executable, "-m", "pip", "install"] + packages)
```

6.2 overpass_api_call.py

6.2.1 Allgemeine Beschreibung

Das Skript overpass_api_call.py ruft Daten aus der Overpass-API anhand einer Konfigurationsdatei auf. Die Konfigurationsdatei ist im Projektordner hinterlegt und enthält das Ergebnis der systematischen Literaturrecherche aus Abschnitt 5.2. In der Konfigurationsdatei sind die Grenzen des API-Abrufs mithilfe der Variable `({{bbox}})` festgelegt. Dies ermöglicht es den Nutzer*innen, die Ergebnisse des API-Abrufs in Overpass Turbo für einen gewählten Kartenausschnitt zu testen. Der Inhalt der Konfigurationsdatei overpass_request_config.txt ist im Anhang 4 hinterlegt. Sobald overpass_api_call.py die API-Ergebnisse erhalten hat, werden diese im Projektordner UrbanLink3D im Unterordner „exports" unter dem Namen overpass_result.xml abgelegt.

Die ausgeführte Overpass-API-Query kann angepasst werden, indem die Datei overpass_request_config.txt geändert wird. Sollte ein anderes Ausgabeformat als XML gewünscht sein, beispielsweise JSON oder CSV, so muss in der Konfigurationsdatei die Zeile 1:

```
[out:xml][timeout:90];
```

entsprechend angepasst werden, genau wie das Format der abgelegten Ergebnisse im Code. Der nachfolgende Workflow ist jedoch für die Nutzung von XML beziehungsweise PBF erstellt, sodass eine Änderung des Datenformats auch eine Anpassung im Skript osm_import.py nach sich zieht.

Das Python-Skript overpass_api_call.py ersetzt die Platzhaltervariable `({{bbox}})` automatisch durch von den Anwender*innen gewählte Koordinaten. Dazu stehen zwei Möglichkeiten zur Verfügung. Einerseits besteht die Möglichkeit, eine GeoJSON-Datei, die eines oder mehrere Polygone enthält, auszuwählen. Außerdem ist es möglich, eine GeoNames-Datei hochzuladen und einen oder mehrere GeoNames aus dieser für die räumliche Einschränkung der Abfrage zu nutzen. Da GeoNames aber immer nur einen Punkt und keine Fläche referenzieren, ist es nötig, einen Radius, um die Punkte einzugeben. Dieser Radius muss in eine Fläche umgerechnet werden.

Abbildung 11 zeigt in einem Sequenzdiagramm den Ablauf der Prozesse in overpass_api_call.py.

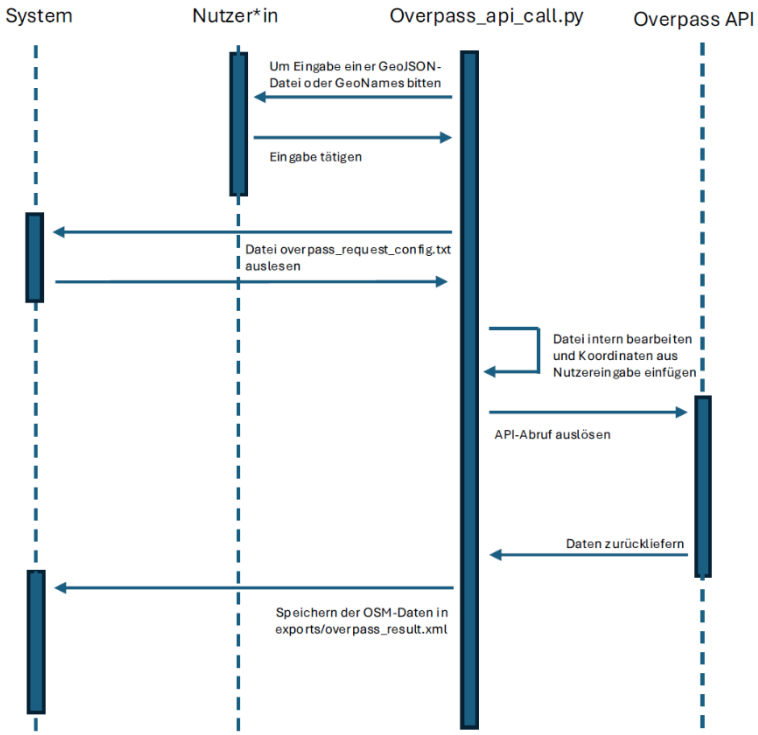


Abbildung 11: Sequenzdiagramm - overpass_api_call.py

6.2.2 Relevante Codeausschnitte

Da die Nutzer*innen den Radius um die GeoNames in Metern angeben, Overpass jedoch die räumliche Einschränkung in Längen- und Breitengraden erwartet, ist eine Umrechnung nötig. Die Umrechnung des Abstands in Metern in die Längen- und Breitengrade ist ein sehr komplizierter Prozess. Aus diesem Grund wurde sich dazu entschieden, die GeoNames-Koordinaten in UTM, also in ein meterbasiertes Format, umzuwandeln. Anschließend wird der Radius aufaddiert und die Koordinaten werden wieder in WGS84 umgerechnet.

```python
# Berechnet Bounding Box um einen Punkt mit Radius in Metern unter Verwendung von UTM
def calculate_bounding_box_utm(lat, lon, radius):
    transformer_to_utm, transformer_to_wgs84 = create_transformers(lon, lat)

    # Konvertiere geographische Koordinaten zu UTM
    x, y = transformer_to_utm.transform(lon, lat)
    # Berechne die Bounding Box in UTM
    min_x = x - radius
    max_x = x + radius
    min_y = y - radius
    max_y = y + radius

    # Konvertiere die BoundingBox zurück zu WGS84
    min_lon, min_lat = transformer_to_wgs84.transform(min_x, min_y)
    max_lon, max_lat = transformer_to_wgs84.transform(max_x, max_y)

    return min_lat, max_lat, min_lon, max_lon
```

Bevor eine Verbindung zur Overpass-API hergestellt werden kann, muss die Platzhaltervariable `({{bbox}})` in der Konfigurationsdatei durch die berechneten Koordinaten ersetzt werden. Anschließend kann die Overpass-API abgerufen werden.

```
    # Overpass API URL
    url = 'https://overpass-api.de/api/interpreter'
    # Query aus Datei lesen
    with open('overpass_request_config.txt', 'r') as file:
        body = file.read()

    # Koordinaten in der Abfrage aktualisieren
    updated_body = re.sub(r'\(\s*\{\{\s*bbox\s*\}\}\s*\)', f"({new_coordinates})",
body)

    # [Auslassung]

    # Verschlüsselung der Abfrage
    encoded_body = "data=" + urllib.parse.quote(updated_body)
    # Wechseln in das Verzeichnis 'exports'.
    exports_dir = "exports"
    os.makedirs(exports_dir, exist_ok=True)
    os.chdir(exports_dir)

    try:
        # POST-Anfrage senden
        response = requests.post(url, data=encoded_body)

        # Prüfung auf erfolgreiche Antwort
        if response.status_code == 200:
            try:
                # Parsen und Speichern der XML-Antwort
                root = ET.fromstring(response.text)
                file_name = "overpass_result.xml"
                with open(file_name, "w", encoding="utf-8") as file:
                    file.write(response.text)
                logging.info(f"Datei erfolgreich gespeichert:
{os.path.abspath(file_name)}")
                print(f"Datei erfolgreich gespeichert: {os.path.abspath(file_name)}")
                return root
            except ET.ParseError:
                logging.error("XML-Antwort konnte nicht geparst werden.")
                print("XML-Antwort konnte nicht geparst werden.")
        else:
            logging.error(f"HTTP Fehler: {response.status_code}")
            print(f"HTTP Fehler: {response.status_code}")
    finally:
        # [Auslassung]
        # Wechsel zurück ins ursprüngliche Verzeichnis
        os.chdir('..')
```

6.3 docker_creation.py

6.3.1 Allgemeine Beschreibung

Das Python-Skript docker_creation.py erstellt drei Docker-Container im gleichen Netzwerk:

- „projecttool_citydb“ für die 3DCityDB, in die die CityGML-Daten importiert werden
- „projecttool_postgis_osm“ für die PostGIS-DB in die die OSM-Daten importiert werden
- „projecttool_osm_intersection“ als dritte Datenbank in die alle Daten importiert werden, um dort verknüpft zu werden

Außerdem importiert docker_creation.py die Daten einer, von den Nutzer*innen über eine GUI ausgewählte CityGML-Datei. Dazu erstellt das Python-Skript einen temporären Docker-Container, in welchem der 3DCityDB Importer läuft.

Die Informationen zu den erstellten Docker-Containern werden in späteren Schritten von weiteren Skripten benötigt, beispielsweise um die korrekten Ports für die Verbindung oder den Docker-Netzwerk-Namen auszulesen. Aus diesem Grund werden die Informationen in der CSV-Datei „deployment_info.csv“ in den Unterordner „logs“ des Projektordners abgelegt.

Das Python-Skript docker_creation.py kann angepasst werden, indem Container-Namen, Nutzernamen und Passwörter der Docker-Container und den darin enthaltenen Datenbanken anders konfiguriert werden. Außerdem kann der Name des Docker-Netzwerks angepasst werden. Auch der Import-Befehl für den Import der CityGML-Datei in die 3DCityDB kann bei Bedarf abgeändert werden.

Abbildung 12 zeigt in einem Sequenzdiagramm den Ablauf der Prozesse in docker_creation.py.

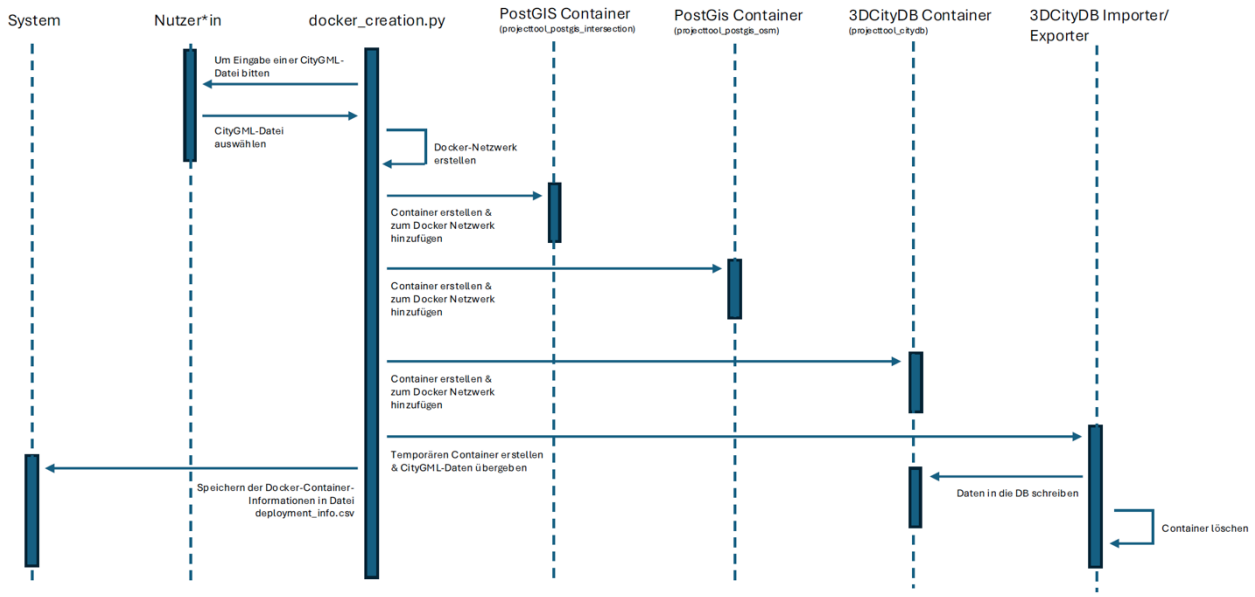


Abbildung 12: Sequenzdiagramm - docker_creation.py

6.3.2 Relevante Codeausschnitte

Die Konfiguration der erstellten Docker-Container und ihres Netzwerkes erfolgt im Programmcode. Dabei werden die Container-Informationen einem Type zugeordnet, welcher später auch in der CSV-Datei gespeichert wird. Dadurch können die nachfolgenden Skripte die richtigen Container für den jeweiligen Zweck erkennen, auch wenn die Namen geändert werden sollten.

```
    network_name = "projecttool_net"
    try:
        network = setup_network(client, network_name)
    # [Auslassung]
    # Port- und Container-Definitionen
    ports = []
    base_port = 5432
    used_ports = set()
    containers = [
        {"name": "projecttool_citydb", "image": "3dcitydb/3dcitydb-pg", "type":
"3DCityDB"},
        {"name": "projecttool_postgis_osm", "image": "postgis/postgis", "type":
"OSMPostgis"},
        {"name": "projecttool_postgis_intersection", "image": "postgis/postgis",
"type": "IntersectionPostgis"}
    ]

    # Container-Daten sammeln
    container_data = {}

    for container_config in containers:
        port = find_free_port(base_port, used_ports)
        ports.append(port)

        if container_config["type"] == "3DCityDB":
            env_vars = {
                "POSTGRES_USER": "postgres",
                "POSTGRES_PASSWORD": db_password,
                "POSTGRES_DB": "postgres",
                "SRID": srid
            }
        else:
            # PostGIS-Container für OSM
            env_vars = {
                "POSTGRES_USER": "postgres",
                "POSTGRES_PASSWORD": db_password,
                "POSTGRES_DB": "osm_db"  # <-- Wichtig
            }

        try:
            start_container(
                client,
                container_config["name"],
                container_config["image"],
                network_name,
                env_vars,
                {"5432/tcp": port}
            )
            logging.info(f"Container {container_config['name']} gestartet auf Port
{port}.")
        except Exception as e:
```

```
            logging.error(f"Fehler beim Starten des Containers
{container_config['name']}: {e}")
            continue

        # Daten für die CSV-Datei speichern
        container_data[container_config["name"]] = {
            "type": container_config["type"],
            "image": container_config["image"],
            "port": port,
            "network": network_name,
            "environment": env_vars
        }
```

Anschließend wird vom Python-Skript ein temporärer Container mit dem 3DCityDB Importer gestartet, welcher eine CityGML-Datei importiert, die über einen Dateidialog von den Nutzer*innen abgefragt wird. Der Import erfolgt dabei in den vorher erstellten Container „projecttool_citydb“.

```
    # IMPORT CityGML
    try:
        logging.info("Starte den Import der CityGML-Daten.")
        print("Starte den Import der CityGML-Daten.")
        citydb_container = client.containers.run(
            "3dcitydb/impexp:latest",
            name="projecttool_citydb_importer",
            detach=True,
            network=network_name,
            volumes={os.path.dirname(gml_file): {'bind': '/data', 'mode': 'rw'}},
            command=f"import -H projecttool_citydb -P 5432 -d postgres -u postgres -p
{db_password} /data/{os.path.basename(gml_file)}"
        )
        for line in citydb_container.logs(stream=True):
            logging.info(line.decode('utf-8').strip())
    except docker.errors.APIError as e:
        logging.error(f"Fehler beim Importieren der CityGML-Daten: {e}")
        print(f"Fehler beim Importieren der CityGML-Daten: {e}")
```

6.4 osm_import.py

6.4.1 Allgemeine Beschreibung

Das Skript osm_import.py importiert eine, von den Nutzer*innen hochgeladene Datei in den zuvor vom Skript docker_creation.py erstellten Docker-Container projecttool_postgis_osm. Die Datei darf den Typen XML oder PBF haben. Für den Import erstellt das Skript einen temporären Container, in dem das Tool osm2pgsql läuft.

Der Import-Befehl von osm2pgsql kann im Quellcode bei Bedarf angepasst werden.

Abbildung 13 zeigt in einem Sequenzdiagramm den Ablauf der Prozesse in osm_import.py.

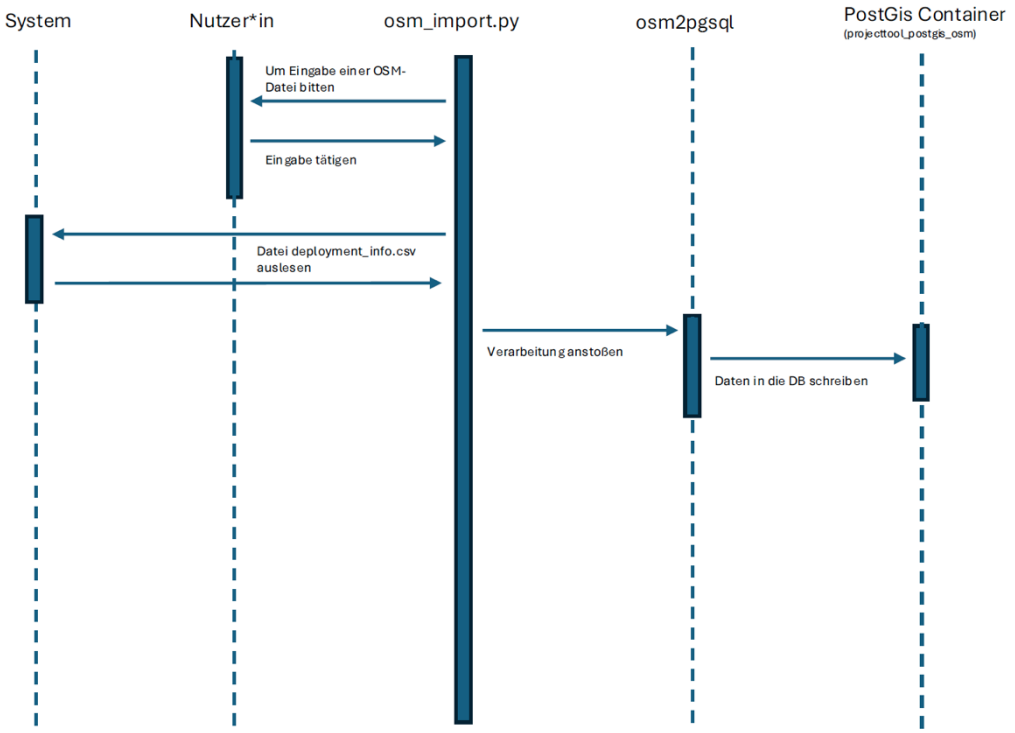


Abbildung 13: Sequenzdiagramm - osm_import.py

6.4.2 Relevante Codeausschnitte

Die Informationen über den Container, in den die OSM-Daten importiert werden sollen, wird aus der in docker_creation.py erstellten Datei deployment_info.csv ausgelesen.

```
# Hilfsfunktionen zum Auslesen aus der deployment_info.csv
def get_conn_info_from_csv(csv_path, container_name):
    with open(csv_path, 'r', encoding='utf-8') as file:
        reader = csv.DictReader(file)
        for row in reader:
            if row['container_name'] == container_name:
                # Port direkt aus Spalte 'port'
                port = row['port']
                env_dict = ast.literal_eval(row['environment'])
                # POSTGRES_USER, POSTGRES_PASSWORD, POSTGRES_DB extrahieren
                user = env_dict.get('POSTGRES_USER', 'postgres')
                password = env_dict.get('POSTGRES_PASSWORD', 'postgres')
                db_name = env_dict.get('POSTGRES_DB', 'postgres')
                return {
                    'dbname': db_name,
                    'user': user,
                    'password': password,
                    'host': 'localhost',
                    'port': port  # string
                }
    return None
```

Der Docker-Befehl zum Import einer PBF- oder XML-Datei erstellt einen temporären Container mit dem Image osm2pgsql.

```
        # Konstruiere den Docker-Befehl als Liste
        docker_command = [
            "docker", "run", "--rm",
            "--network", network,
            "-v", volume_mapping,
            "-e", f"PGPASSWORD={pgpassword}",
            "iboates/osm2pgsql:latest",  # Verwende das osm2pgsql-Image
            "osm2pgsql",
            "-H", host,
            "-P", str(internal_port),  # Verwende den internen Port 5432
            "-d", database,
            "-U", user,
            "--create",
            "--slim",
            "--hstore",
            f"/osm-data/{os.path.basename(pbf_file)}"
        ]
```

6.5 intersect_data.py

6.5.1 Allgemeine Beschreibung

Das Python-Skript intersect_data.py liest die Daten aus den Docker-Containern projecttool_postgis_osm und projecttool_citydb aus und importiert sie in die PostGIS-Tabelle des Docker-Containers zur Datenverknüpfung projecttool_postgis_intersection. Die relevanten Informationen über Docker-Container, Docker-Netzwerke, Ports, Passwörter und Nutzernamen werden aus der Datei deployment_info.csv ausgelesen. Anschließend werden die Nutzer*innen aufgefordert, eine GTFS-Datei auszuwählen. Auch diese wird in den Container zur Datenverknüpfung importiert. Anschließend müssen die Anwender*innen Schwellenwerte eingeben, mit denen bestimmt wird,

- wie viele Meter zwei Punkte voneinander entfernt sein dürfen, um verknüpft zu werden.
- wie viele Meter ein GTFS-Punkt von einem OSM-Punkt, der als Haltestelle gekennzeichnet ist, entfernt sein darf, um mit ihm verknüpft zu werden.
- zu wie viel Prozent sich zwei Flächen mindestens überschneiden müssen, um verknüpft zu werden.

Danach führt das Skript die Verknüpfungsoperationen anhand der in Abschnitt 5.6.3 beschriebenen Verknüpfungsregeln aus und schreibt die Ergebnisse in ein neues Schema der Datenbank. Dazu werden PostGIS-Funktionen wie beispielsweise ST_DWithin verwendet.

Im Python-Skript intersect_data.py können die zur Verknüpfung der Daten genutzten Regeln angepasst und erweitert werden.

Abbildung 14 zeigt in einem Sequenzdiagramm den Ablauf der Prozesse in intersect_data.py.

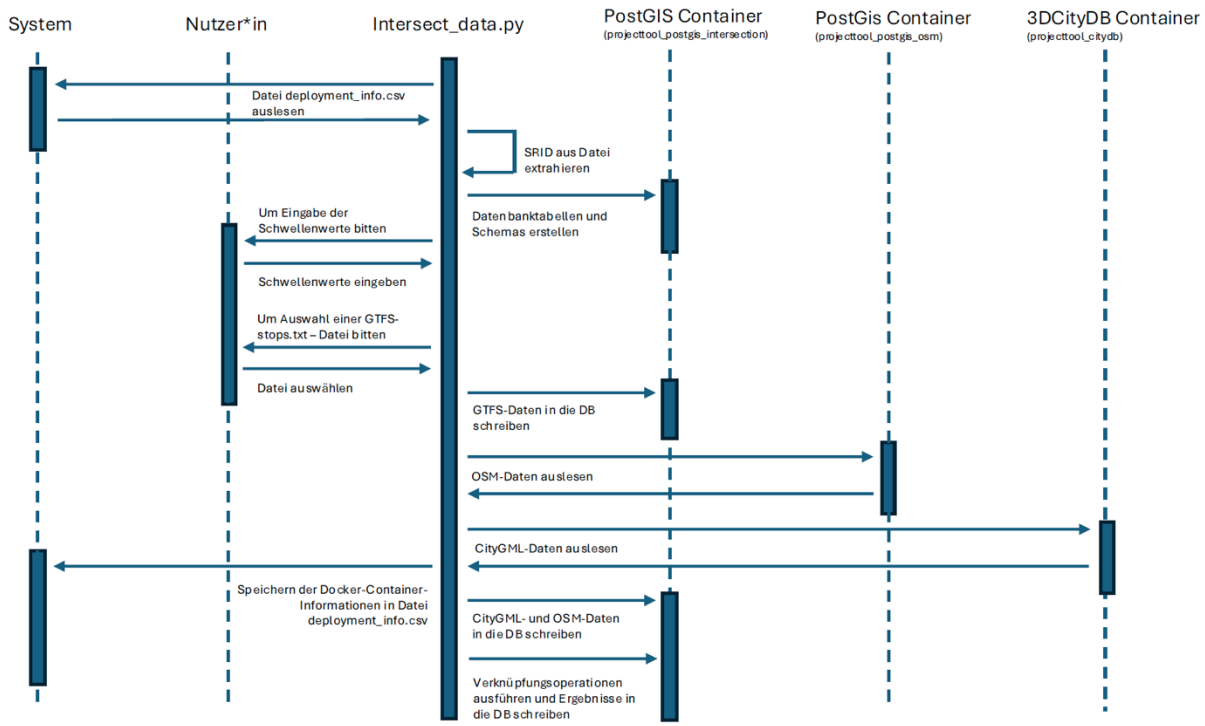


Abbildung 14: Sequenzdiagramm - intersect_data.py

6.5.2 Relevante Codeausschnitte

Das Skript erstellt für alle Daten, also für CityGML-Daten, OSM-Daten und GTFS-Daten eigene Schemas. Auch für die Ergebnisse der Verknüpfung wird ein eigenes Schema erstellt.

```
# Tabellen anlegen für matching-DB
try:
    with psycopg2.connect(postgis_conn_info) as conn:
        with conn.cursor() as cursor:
            cursor.execute(f"""
            CREATE EXTENSION IF NOT EXISTS postgis;

            CREATE SCHEMA IF NOT EXISTS citygml;
            CREATE TABLE IF NOT EXISTS citygml.points (
                id SERIAL PRIMARY KEY,
                cityobject_id BIGINT,
                geometry GEOMETRY(PointZ, {citydb_srid})
            );

            CREATE TABLE IF NOT EXISTS citygml.areas (
                id SERIAL PRIMARY KEY,
                cityobject_id BIGINT,
                geometry GEOMETRY(PolygonZ, {citydb_srid})
            );

            CREATE SCHEMA IF NOT EXISTS osm;
            CREATE TABLE IF NOT EXISTS osm.objects (
```

```
            id SERIAL PRIMARY KEY,
            osm_id BIGINT,
            geometry GEOMETRY(GeometryZ, {citydb_srid}),
            object_type TEXT,
            attributes JSONB
        );

        CREATE SCHEMA IF NOT EXISTS gtfs;
        CREATE TABLE IF NOT EXISTS gtfs.stops (
            stop_id TEXT PRIMARY KEY,
            stop_name TEXT,
            stop_lat DOUBLE PRECISION,
            stop_lon DOUBLE PRECISION,
            location_type INTEGER,
            parent_station TEXT,
            geometry GEOMETRY(PointZ, {citydb_srid})
        );

        CREATE SCHEMA IF NOT EXISTS matching;
        CREATE TABLE IF NOT EXISTS matching.results (
            id SERIAL PRIMARY KEY,
            citygml_id BIGINT,
            osm_id BIGINT,
            match_case TEXT,
            attributes JSONB
        );
        """)
    conn.commit()
logging.info(f"Intersection-DB Tabellen mit SRID={citydb_srid} angelegt.")
```

Im Folgenden wird beispielhaft die Umsetzung der dritten Überschneidungsabfrage gezeigt. Diese nutzt die PostGIS-Funktion ST_DWithin, um zu prüfen, ob zwei Punkte innerhalb einer von den Anwender*innen als Schwellenwert eingetragenen Entfernung zueinander liegen.

```
    # Regel 3: Punkte innerhalb der Entfernungsschwelle
    logging.info("Regel 3: Punkte innerhalb der Entfernungsschwelle.")
    inserted_count = do_insert_return_count(f"""
    INSERT INTO matching.results (citygml_id, osm_id, match_case, attributes)
    SELECT cg.id, o.osm_id, 'points_within_distance', o.attributes
    FROM citygml.points cg
    JOIN osm.objects o
    ON o.object_type = 'Point'
    AND ST_DWithin(cg.geometry, o.geometry, %s)
    """, (point_dist_m,))
    logging.info(f"Regel 3 eingefügt {inserted_count} trifft zu.")
```

6.6 Ergebnisse der Automatisierung

Die in diesem Abschnitt beschriebene Automatisierung sorgt dafür, dass der Workflow zur Verknüpfung mobilitätsrelevanter Daten anhand des Praxisbeispiels deutlich schneller ausgeführt werden kann.

Das in Abschnitt 5 aufgetretene Problem der Identifikation mobilitätsrelevanter Daten und der Überführung in OSM-Tags tritt nicht auf, da die relevanten OSM-Tags für Nutzer*innen vorgefertigt in einer Konfigurationsdatei vorhanden sind. Auch der Aufwand zur Festlegung der Koordinaten ist über eine einfache Eingabe einer GeoJSON-Datei oder GeoNames vereinfacht worden.

Die Nutzung der Tools 3DCityDB Importer/ Exporter sowie osm2pgsql ist nun auch für Anfänger ohne Probleme möglich, da eine funktionsfähige Konfiguration der nötigen Docker-Befehle im Quellcode hinterlegt ist. Diese sind trotzdem noch anpassbar, um somit für verschiedene Nutzungsfälle angepasst werden zu können.

Der Fehler beim Import der OSM-Daten in die PostGIS-DB mittels osm2pgsql konnte durch die Automatisierung nicht gelöst werden, da am in Abschnitt 5 erstellten Importbefehl keine Änderung vorgenommen wurde.

Das Ausführen von Verknüpfungsregeln ist nun automatisiert und dementsprechend vereinfacht. Eine sorgfältige Planung der Verknüpfungsregeln ist noch immer nötig, falls jedoch die in Abschnitt 5.6.3 vordefinierten Regeln ausreichen, besteht durch die Funktionalität des Skriptes die Möglichkeit, Schwellenwerte einzugeben. Dadurch wurden zwar die groben Verknüpfungsregeln vorgegeben, es besteht jedoch für Anwender*innen immer noch die Möglichkeit, die Regeln an ihre Bedürfnisse anzupassen.

Der Prozess wurde durch die Automatisierung stark vereinfacht, da keine spezifischen Daten wie Docker-Container-Name, Datenbankname, Passwort, Nutzername und Dateipfad mehr manuell im Shell-Befehl angepasst werden müssen. Die Daten zur Erstellung der Container sind vordefiniert und das Einfügen der Dateinamen geschieht durch ein GUI.

7 Zusammenfassung des Workflows in Form einer Anwenderdokumentation

7.1 Ziele der Dokumentation

Die in Abschnitt 5 erprobten und in Abschnitt 6 automatisierten Schritte des Workflows sollen nun in einer Anwenderdokumentation zusammengefasst werden. Ziel dieser ist es, die einzelnen Schritte mit Hilfe der erstellten Python-Skripte so zu dokumentieren, so dass sie für Anwender*innen verständlich und praxisnah aufbereitet sind. Die Dokumentation soll den Prozess der Datenintegration und der Verknüpfung effizient und reproduzierbar machen. Bei der Erstellung werden Informationen aus Kothes (2011) herangezogen, um die Anwenderdokumentation angemessen zu gestalten.

7.2 Inhaltliche Struktur

Eine vollständige Anleitung umfasst alle Informationen, welche Nutzer*innen brauchen, um ihnen zugesicherte Funktionen zu verstehen und das Produkt funktionsfähig zu halten. Texte und Bilder müssen so verständlich sein, dass sie ohne fremde Hilfe erfasst und im Produkt wiedergefunden werden können. Eine verständliche Formulierung ist ebenso wichtig (vgl. Kothes 2011, S. 37-38).

Die Vollständigkeit der Anleitung ist auch abhängig von den Anforderungen, welche die Zielgruppe an die Anleitung stellt. Die Zielgruppe und zwei Personas wurden in Abschnitt 4.2 ausführlich beschrieben. Sie können folgendermaßen zusammengefasst werden:

> Die Anleitung für die Software soll strukturiert, kurz und prozessorientiert sein. Es sollten keine unnötig langen Fließtexte enthalten sein, dafür jedoch übersichtliche Schaubilder und weiterführende Informationen zu relevanten Grundlagen. Der einer Software zugrundeliegende Prozess soll eindeutig und übersichtlich gestaltet sein, damit eine mögliche Integration in bereits bestehende Prozesse und eine Anpassung für verschiedene Anwendungsfälle vereinfacht wird. Generell ist in Leitfäden ein großer Fokus auf Kompaktheit, Einfachheit und Verständlichkeit zu legen, wobei Schaubilder und Ablaufschemata einen großen Vorteil bieten (vgl. Reitberger et al. 2019, S. 3).

Bei der Entscheidung für relevante Inhalte der Anleitung wurde sich an den Ausführungen aus Kothes (2011) orientiert. In der Einleitung wird, neben allgemeinen Informationen zur Dokumentenversion und dem Inhalt, beschrieben, welche Informationen das Dokument liefert und für welche Use-Cases es genutzt werden kann. Es werden keine Grundlagen zu den Datenformaten oder Docker geliefert, da davon ausgegangen werden kann, dass Nutzer*innen, welche die Anwenderdokumentation verwenden werden, dieses Grundwissen besitzen.

Es wurde sich dazu entschieden, zum Verständnis der einzelnen Skripte ausschließlich Schaubilder sowie kurze Textbeschreibungen einzufügen. Sollten Nutzer*innen die Skripte anpassen wollen, ist eine tiefergehende Auseinandersetzung mit dem Quellcode unerlässlich. Eingefügte Sequenzdiagramme erleichtern das Verständnis des Quellcodes.

7.3 Designentscheidungen

Die erstellte Anwenderdokumentation zum Softwarepaket UrbanLink3D wird in deutscher Sprache verfasst, um eine möglichst breite Nutzergruppe im deutschsprachigen Raum anzusprechen. Eine zukünftige Übersetzung ins Englische ist jedoch denkbar, um den potenziellen Nutzungskreis zu erweitern.

Bei der sprachlichen Gestaltung wird auf Substantivierungen verzichtet, um die Verständlichkeit zu erhöhen. Anweisungen werden in kurzen Sätzen formuliert, sofern Fließtext verwendet wird. Wo es

sinnvoll ist, werden Informationen in Stichpunkten dargestellt, um eine klare und übersichtliche Struktur zu gewährleisten. Um die Texte präzise und leserfreundlich zu gestalten, werden Weichmacher und Füllwörter konsequent vermieden. Verweise innerhalb der Anwenderdokumentation beschränken sich auf weiterführende Informationen oder Zwischenschritte. Externe Verweise beziehen sich auf nötige Softwaredownloads. Interne Verweise 
sind durch folgendes Icon gekennzeichnet:

Alle notwendigen Inhalte werden an den entsprechenden Stellen direkt aufgeführt, auch wenn dies gelegentliche Wiederholungen erfordert.

Auf Abkürzungen wird verzichtet, da kein Abkürzungsverzeichnis vorgesehen ist. Dies folgt der Empfehlung von Kothes 2011, da Nutzer*innen selten die Mühe auf sich nehmen, in solchen Verzeichnissen nachzuschlagen (vgl. Kothes 2011, S. 135).

Die Anrede erfolgt durchgehend in der „Sie"-Form, um die Lesenden höflich und direkt anzusprechen. Handlungsschritte werden aktiv formuliert und typografisch hervorgehoben, um sie klar von erläuternden Texten zu unterscheiden. Dazu wird die folgende Hervorhebung verwendet:

1. Anweisung erster Ordnung
    - Anweisung zweiter Ordnung
        - Anweisung dritter Ordnung
            - Anweisung vierter Ordnung

Nach erfolgreicher Durchführung von Handlungsschritten werden die Ergebnisse oder spezifische Kennzeichnungen deutlich ausgewiesen, um die Orientierung zu erleichtern. Die entsprechenden Icons werden dafür wie folgt eingesetzt:

➔ – Bestätigung des Erfolgs für Zwischenschritte

✔ – Bestätigung des Erfolgs für den gesamten Prozessschritt

Der Fokus der Anwenderdokumentation liegt klar auf dem Inhalt, nicht auf dem Design. Daher wurde ein einspaltiges Layout im DIN-A4-Format gewählt, das als PDF bereitgestellt wird. Für eine bessere Lesbarkeit am Bildschirm wird die serifenlose Schriftart Arial verwendet. Falls ein Code in der Dokumentation aufgeführt wird, erfolgt dessen Darstellung in der Schriftart Courier New, um ihn optisch von anderen Inhalten abzugrenzen.

Da die Benutzeroberflächen (UIs) der Anwendungen sehr einfach gehalten sind, wurden keine Screenshots des GUI in die Anleitung eingebunden. Wichtige Hinweise oder weiterführende Informationen sind durch das fett hervorgehobene Wort „Hinweis" gekennzeichnet.

Die auf Basis dieser Festlegungen entstandene Anwenderdokumentation kann in Anhang 5 gefunden werden.

8 Usability-Test

Das Konzept der Usability beschreibt, wie gut ein Produkt, sei es ein Softwareprodukt, eine Anleitung oder ein physisches Objekt, dazu geeignet ist eine vorgesehene Aufgabe zu erfüllen. Sarodnick und Brau definieren die Usability als „die Passung von System, Aufgabe und Nutzer aus der Perspektive einer vom Nutzer wahrgenommenen Qualität der Zielerfüllung“ (Sarodnick & Brau 2011, S. 20). Rubin und Chrisnell bezeichnen die Usability als Abwesenheit von Frustration bei der Nutzung eines Software-Produkts (vgl. Rubin & Chrisnell 2008, S. 3-4).

Ein Usability-Test ist eine empirische Methode, bei der repräsentative Nutzer*innen ein System verwenden, um spezifische Aufgaben zu lösen.

8.1 Testplan

Dieser Abschnitt beschreibt den für die Untersuchung erstellten Testplan. Ein strukturierter und durchdachter Testplan ist unerlässlich für einen systematisch geplanten und gut durchgeführten Usability-Test. Er unterstützt bei der Durchführung, indem er das genaue Vorgehen während des Tests beschreibt und sicherstellt, dass Forschungsfragen klar beantwortet werden können (vgl. Tullis & Albert 2008, S.15; vgl. Rubin & Chrisnell 2008, S. 65-68).

8.1.1 Ziele der Evaluation

Ziel dieser Untersuchung ist die Durchführung einer formativen Evaluation, um den Gesamteindruck des Produkts zu erfassen und mögliche Verbesserungspotenziale aufzuzeigen (vgl. Sarodnick & Brau 2011, S. 163). Dabei soll überprüft werden, ob die während der Erstellung des Workflows aufgetretenen Probleme durch die Automatisierung und die Anwenderdokumentation reduziert werden konnten. In Abschnitt 6.6 wurde bereits dargelegt, dass die meisten Probleme durch die Anwendung objektiv betrachtet behoben oder zumindest verbessert werden konnten. Die erstellte Anwenderdokumentation stellt eine zusätzliche Unterstützung für die Nutzer*innen dar. Das Ziel dieser Untersuchung besteht daher darin, die folgenden Fragen zu beantworten:

- Wie komplex ist die praktische Durchführung der Workflow-Schritte für Anwender*innen?
- Sind die Informationen in der Anwenderdokumentation verständlich dargestellt, und lässt sich der Prozess leicht überblicken?
- Ist die Anpassung von Konfigurationsdateien für die Nutzer*innen einfach und intuitiv möglich?
- Welche unerwarteten Probleme treten auf, wenn Anwender*innen die erstellten Softwaretools zum ersten Mal nutzen?

Forschungsfragen:

- Haben Nutzer*innen Verständnisprobleme beim Lesen der Anleitung? Sind alle Schritte eindeutig formuliert?
- Liefert die Anleitung einen angemessenen Ausgleich zwischen zu wenig und zu viel Information?
- Sind die Eingabeaufforderungen des Programmes eindeutig? Wissen Nutzer*innen stets, was zu tun ist?

Ziel des Usability-Tests ist es nicht, den Programmcode zu evaluieren. Dieser wurde zwar ausführlich mit Kommentaren versehen, sodass er einfach angepasst werden kann und er ist auch dazu da, um als Grundlage für weitere Programme genutzt zu werden, die Evaluation der Skripte würde jedoch zu tief greifen und eine längere Auseinandersetzung mit dem Quellcode erfordern.

8.1.2 Methodik

Um die genannten Forschungsziele zu erreichen und die Forschungsfragen zu beantworten, wird ein Usability-Testdurchgeführt, bei dem die Testpersonen den Workflow praktisch durchlaufen. Dabei werden sie gebeten, laut zu denken. Diese Methode ermöglicht es, während der Nutzung auftretende Probleme unmittelbar zu identifizieren. Ein Vorteil des lauten Denkens besteht darin, dass „auf diesem Wege mit wenig Probanden sehr hilfreiche, qualitative Informationen gewonnen werden [können]“ (Sarodnick & Brau 2011, S. 170).

Im Anschluss an den Test wird ein leitfadengestütztes Interview durchgeführt, in dem gezielt Fragen zur Verständlichkeit und Nutzbarkeit der Software sowie der Anwenderdokumentation gestellt werden. Das Interview bietet den Testpersonen die Möglichkeit, frei zu antworten und auf spezifische Aspekte einzugehen.

Durch die Kombination dieser beiden Methoden lassen sich umfassende qualitative Informationen gewinnen, die wertvolle Einblicke in die Stärken und Schwächen der Anwendung und Dokumentation liefern.

8.1.3 Testpersonen

Der Usability-Test wird mit nur zwei Testpersonen durchgeführt, da der entwickelte Workflow, obwohl er bereits einen komplexen Prozess abbildet, noch einen begrenzten Funktionsumfang aufweist und sich in einer Erstversion befindet. Ein Test mit wenigen Personen ermöglicht zunächst einen Gesamteindruck darüber, ob die Software und ihre Anwendungen grundsätzlich den geplanten Nutzungskontext unterstützen oder ob schwerwiegende Probleme bei der Nutzung auftreten.

Rubin & Chisnell (2008, S. 72) weisen darauf hin, dass bereits mit vier bis fünf Testpersonen rund 80 % der Usability-Probleme aufgedeckt werden können. Auch wenn die Anzahl der Testpersonen in der hier geplanten Untersuchung geringer ist, können durch die Kombination aus praktischen Tests und Interviews bereits wichtige Schwachstellen identifiziert werden. Diese vorläufigen Erkenntnisse bieten eine Grundlage für die Weiterentwicklung, bevor eine umfassendere Evaluation mit mehr Testpersonen durchgeführt wird. Da der Workflow schlank und geradlinig ist und nur wenige unterschiedliche Eingaben erlaubt, wird die kleinere Testgruppe als ausreichend erachtet. Für komplexere Anwendungen mit größerem Funktionsumfang wären jedoch entsprechend mehr Testpersonen erforderlich (vgl. Tullis & Albert 2008, S. 17).

Die zwei gewählten Testpersonen sind 26 und 27 Jahre alt und haben beide einen Bachelor of Science im Studienfach „Angewandte Informatik – Digitale Medien und Spieleentwicklung“. Testperson 1 (TP1) befindet sich derzeit im Masterstudium für einen Master of Arts im Fach „Online Kommunikation“ und hat 4 Jahre Erfahrung als Softwareentwickler. Testperson 2 (TP2) ist aktuell festangestellter Softwareentwickler mit einer Programmiererfahrung von 7 Jahren. Beide Testpersonen haben kein Vorwissen bei der Verwendung von Geodaten. Die Daten zu den Testpersonen sind nochmals übersichtlich in Tabelle 10 aufgeführt.

	TP1	**TP2**
Alter/Geschlecht	27/M	26/M
Höchster Bildungsabschluss	Bachelor of Science „Angewandte Informatik – Digitale Medien und Spieleentwicklung“	Bachelor of Science „Angewandte Informatik – Digitale Medien und Spieleentwicklung“
Aktuelle Tätigkeit	Master of Arts „Online Kommunikation“	Softwareentwickler
Kenntnisse Softwareentwicklung in Jahren	4	7

	TP1	TP2
Kenntnisse Verarbeitung Geodaten	Keine	Keine

Tabelle 10: Kurzbeschreibung zu den Testpersonen

8.1.4 Testumgebung

Die Tests wurden in einer kontrollierten Arbeitsumgebung durchgeführt. Um eine ruhige Testatmosphäre zu gewährleisten, wurden die Türen zum Raum jeweils geschlossen.

Die Untersuchung fand auf einem Laptop mit dem Betriebssystem macOS statt, da die Skripte ausschließlich auf diesem Betriebssystem getestet wurden. An den Laptop war ein 27-Zoll-Monitor angeschlossen, wobei nur mit einem Bildschirm gearbeitet wurde. Dies ermöglichte es, mehrere Fenster (PDF-Anleitung, PDF-Aufgabenstellung, Terminalfenster) gleichzeitig übersichtlich darzustellen.

Der Projektordner sowie die Rohdaten wurden auf dem Desktop abgelegt, um den Zugang zu den benötigten Dateien für die Testpersonen zu erleichtern. Da die Testpersonen mit der Dateistruktur des Systems nicht vertraut waren, wurde darauf geachtet, Suchzeiten zu minimieren.

Da keine der Testpersonen über Vorkenntnisse im Umgang mit Geodaten verfügte, wie aus den vor der Untersuchung erhobenen Informationen hervorging, wurden die Konfigurationsdateien (GeoJSON) vorab vorbereitet und bereitgestellt, um den Testablauf nicht unnötig zu erschweren. Zusätzlich war das Tool Osmosis, das während der Nutzung zur Fehlerbehebung erforderlich sein könnte, bereits installiert und leicht zugänglich auf dem Desktop abgelegt.

8.1.5 Testdauer und grober Ablauf

Für jeden Test waren 45 Minuten vorgesehen. Die Testpersonen wurden begrüßt und an den vorbereiteten Arbeitsplatz geführt. Dort waren die Aufgabenstellung, die Anleitung und ein Terminalfenster bereits geöffnet, wobei sich das Terminalfenster noch nicht im richtigen Verzeichnis befand. Zur Vorbereitung einer späteren Transkription wurde ein Smartphone mit aktivierter Tonaufnahme auf den Schreibtisch gelegt. Die Versuchsleiterin saß schräg hinter den Testpersonen, um mögliche Ablenkungen zu minimieren, während sie dennoch die Möglichkeit hatte, den Test zu beobachten und bei Bedarf einzugreifen.

Die Begrüßung und Einführung formulierte die Versuchsleiterin in folgenden Worten:

> Hallo und herzlich willkommen! Vielen Dank, dass du dir die Zeit nimmst, am Usability-Test teilzunehmen. Ich möchte dir zunächst kurz erklären, worum es in diesem Test geht und welches Ziel wir verfolgen. Da das Thema Geodaten für dich wahrscheinlich neu ist, gebe ich dir vorab einige grundlegende Informationen.
>
> Heute testen wir einen Workflow, den ich im Rahmen meiner Masterarbeit entwickelt habe. Dabei möchte ich herausfinden, ob die Softwareskripte und deren Ausgaben verständlich sind und ob die Anleitung klar formuliert ist. Ziel des Workflows ist es, ein 3D-Stadtmodell mit weiteren, für die nachhaltige Mobilität relevanten Daten zu verknüpfen. Ein 3D-Stadtmodell ist, wie der Name schon sagt, eine dreidimensionale Abbildung einer Stadt. Es zeigt beispielsweise Gebäude, Straßen, Flüsse oder markante Punkte. Diese Modelle werden im sogenannten CityGML-Format gespeichert, das auf XML basiert. Da die Daten stark miteinander verknüpft sind, beispielsweise gehören mehrere Wände zu einem Gebäude, sind die Dateien oft recht komplex zu verarbeiten.
>
> Unser Ziel der Evaluation ist es, beispielhaft das 3D-Stadtmodell der Leipziger Innenstadt mit Daten aus der General Transit Feed Specification (GTFS) und OpenStreetMap

(OSM) zu kombinieren. GTFS liefert Informationen zum öffentlichen Nah- und Fernverkehr, wie Haltestellenpositionen, Linienführungen und Abfahrtszeiten, die in einzelnen Textdateien gespeichert sind. OSM ist eine frei zugängliche, weltweite Geodatenbank, über die du Punkte, Linien und Flächen abfragen kannst – von Einbahnstraßen und verkehrsberuhigten Bereichen bis hin zu Parks. Sogar spezifische Details wie das Material einer Parkbank können dort verzeichnet sein. Zur Kombination dieser verschiedenen Datenquellen verwenden wir Docker-Container. Diese kannst du dir wie virtuelle Maschinen vorstellen, die auf Linux basieren und in denen spezielle Anwendungen laufen. Docker ermöglicht es, Software einfach zu verteilen und auszuführen.

Ich habe den Workflow mit Python-Skripten vereinfacht und eine Anwenderdokumentation erstellt, die ich dir gleich auf dem Laptop zeige. Deine Aufgabe wird es sein, die Anwenderdokumentation zu lesen und deren Schritte zu befolgen. Ich stelle dir alle notwendigen Rohdaten und Informationen zur Verfügung. Du wirst einmal den Prozess der Datenverknüpfung durchführen, der Export und die Beschaffung der Daten bleiben außen vor. Eine Übersicht deiner Aufgaben findest du zusätzlich auf einem Aufgabenblatt. Während des Tests bitte ich dich, laut zu denken. Erklär mir, was du gerade machst, was dein Ziel ist und ob dir etwas unklar erscheint oder Probleme bereitet. Ich bin hauptsächlich als Beobachterin hier, aber falls du gar nicht weiterkommst, kannst du mich jederzeit um Hilfe bitten. Im Anschluss werde ich dir in einem Interview noch ein paar Fragen stellen.

Aufgaben

Für die Untersuchung wurden den Testpersonen spezifische Aufgaben gestellt. Dabei wurden die erforderlichen Rohdaten bewusst bereitgestellt, um den Fokus auf die Nutzung des Workflows und nicht auf die Datenbeschaffung oder Qualitätsanalyse zu legen. Auch wenn der Workflow theoretisch den gesamten Prozess inklusive Datenbeschaffung und Qualitätsanalyse abbilden könnte, wurde darauf verzichtet, da aufgrund des fehlenden Grundwissens der Testpersonen nicht davon ausgegangen werden konnte, dass diese Schritte erfolgreich durchgeführt werden.

Zu Beginn sollten sich die Testpersonen die Informationen zum Dokument durchlesen, um einen Überblick darüber zu erhalten, welche Inhalte sie erwarten und wie die Dateien aufgebaut sind. Die Aufgabe bestand darin, die CityGML-, GTFS- und OSM-Daten zu verknüpfen. Darüber hinaus wurden die Testpersonen gebeten, der Overpass-Query eine weitere Zeile hinzuzufügen, um zu überprüfen, ob die Möglichkeit zur Konfiguration in der Anleitung verständlich beschrieben war.

Die Testpersonen konnten selbst entscheiden, in welchem Umfang sie die Anleitung nutzen und ob sie für den Overpass-Abruf GeoJSON oder GeoNames verwenden wollten. Um die Installationsfunktion zu testen, wurde ein Python-Paket absichtlich deinstalliert, sodass die Testpersonen dieses während der Untersuchung erneut installieren mussten. Zusätzlich wurde der komplette GTFS-Datenordner bereitgestellt, um zu prüfen, ob die Testpersonen die Anweisungen sorgfältig lesen und ausschließlich die stops.txt-Datei auswählen.

Das PDF-Aufgabenblatt ist im Anhang 6 zu finden.

Leitfadengestütztes Interview

Das Interview hat den Zweck, die oben genannten Forschungsfragen zu beantworten und Erkenntnisse über die allgemeine Usability der Software sowie der Anwenderdokumentation zu gewinnen.

Um die Usability der Software und der Anwenderdokumentation noch differenzierter zu bewerten, wurden weitere Leitfragen aus dem Fragebogen ISONORM 9241/110-S (vgl. Pümper) abgeleitet.

Dabei wurden nur die Items berücksichtigt, die eine direkte Anwendbarkeit auf den Workflow aufweisen. Beispielsweise wurden Fragen wie „Die Software bietet/bietet nicht alle Funktionen, um die anfallenden Aufgaben zu bewältigen“ (aa1) oder „Die Software ist schlecht/gut auf die Anforderungen der Arbeit zugeschnitten“ (aa3) ausgeschlossen, da sie von den Testpersonen nicht beantwortet werden können, da diese Aufgaben nicht ihrem beruflichen Alltag entsprechen. Ebenfalls als irrelevant eingestuft wurden die Items aa2, sb2, sb3, lf1, sk1, sk2, sk3, ft2, ft3, lk2 und lk3.

Die verbleibenden relevanten Items lauten:

- Die Software liefert in unzureichendem/zureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind
- Die Software erschwert/erleichtert die Orientierung durch eine einheitliche/uneinheitliche Gestaltung
- Die Software informiert in zureichendem/unzureichendem Maße darüber, was es gerade macht
- Die Software lässt sich/lässt sich nicht durchgehend nach einem einheitlichen Prinzip bedienen
- Die Software erfordert/erfordert nicht, dass man sich viele Details merken muss
- Die Software ist schlecht/gut ohne fremde Hilfe oder Handbuch erlernbar
- Die Software liefert schlecht/gut verständliche Fehlermeldungen
- Die Software lässt sich schwer/leicht erweitern, wenn für mich neue Aufgaben entstehen

Aus diesen Betrachtungen ergibt sich folgender Interviewleitfaden

Interviewleitfaden

Kurze Einleitung in das Interview

> So, nun wo wir die Durchführung geschafft haben, möchte ich dir noch ein paar Fragen stellen. Dabei gibt es keine richtigen oder falschen Antworten, teile mir einfach deine Erfahrungen mit.

Allgemeine Fragen:

- Wie hast du die Bearbeitung der Schritte im Workflow insgesamt empfunden?
- Empfandest du die Ausführung der Arbeitsschritte als einfach oder kompliziert?

Fragen zur Usability

- Wie fandest du die Gestaltung der Software, insbesondere die Kombination aus grafischer Eingabe (UI) und der Terminal-Ausgabe?
- Hast du die Software als einheitlich und konsistent in der Bedienung wahrgenommen?
- Gab es Situationen, in denen unklar war, welche Eingaben erforderlich sind? Falls ja, welche?
- Musstest du dir während der Nutzung viele Details merken? Hat die Anleitung dabei geholfen, dass du dir weniger merken musstest?
- Wäre die Software auch ohne die Anwenderdokumentation nutzbar gewesen?

Fragen zu Ausgaben

- Wusstest du während der Nutzung immer, was die Skripte gerade machen?
- Waren die Fehlermeldungen verständlich und hilfreich? Falls nein, wo lag das Problem?

- Hältst du die Software für leicht erweiterbar, falls neue Aufgaben oder Anforderungen entstehen?

Fragen zur Anleitung

- War die Anleitung einheitlich und leicht verständlich?
- Gab es Stellen in der Anleitung, die dir unklar oder zu wenig detailliert erschienen?
- Hast du sonst noch etwas, was du sagen möchtest?

Abschluss

Dann danke ich dir ganz herzlich für deine Teilnahme und deine Zeit!

8.1.6 Ergebnisse der Untersuchung

Für die Auswertung wurden die Interviews mit dem Transkriptionstool von Microsoft Word automatisch transkribiert. Zur besseren Referenzierbarkeit wurden den Transkripten Zeilennummern hinzugefügt. Die vollständigen Transkripte sind in Anhang 7 und Anhang 8 zu finden. Die Auswertung gliedert sich in zwei Teile: zunächst werden allgemeine Auffälligkeiten während der Untersuchung beschrieben, anschließend werden die Antworten auf die Interviewfragen analysiert.

Die Untersuchung hat gezeigt, dass die generelle Arbeit mit den Python-Skripten und der Anwenderdokumentation gut funktioniert. Beide Testpersonen benötigten anfänglich etwas Zeit, um sich mit der Arbeitsumgebung vertraut zu machen. Nach wenigen Minuten arbeiteten sie jedoch sicher am bereitgestellten Laptop.

Auffällig war, dass mehrere Probleme bei beiden Probanden aufgetreten sind. Dabei handelte es sich vor Allem um Probleme mit getroffenen Designentscheidungen.

In der Anleitung ist folgender Abschnitt integriert (siehe Abbildung 15):

> Um einzelne Python-Skripte auszuführen, öffnen Sie ein Terminal und wechseln Sie in den Projektordner. Starten Sie die gewünschten Programme mit dem Befehl:
>
> `python3 <programmname.py>`

Abbildung 15: Ausschnitt aus der Anwenderdokumentation - Python-Programme starten

Beide Testpersonen haben den Fehler gemacht, dass sie die eckigen Klammern mit in den Konsolenbefehl eingetippt haben. Diese waren eigentlich zur Symbolisierung eines Platzhalters gedacht, dies war aber anscheinend nicht eindeutig.

Auch bei der Umwandlung der XML-Datei in die PBF-Datei trat bei beiden Testpersonen der gleiche Fehler auf. In der Anleitung steht dazu folgender Text (siehe Abbildung 16):

> - Führen Sie im Ordner folgenden Befehl aus:
>
>   ```
>   bin/osmosis --read-xml file=overpass_result.xml --
>   write-pbf file=export.pbf
>   ```

Abbildung 16: Ausschnitt aus der Anwenderdokumentation - Osmosis-Befehl

Beide Testpersonen haben die Anleitung verstanden, jedoch entstand beim Kopieren des Befehls das Problem, dass auch der Zeilenumbruch mit kopiert wurde. Dadurch kam es bei beiden Testpersonen zu Fehlern in der Konsole, beide haben den Fehler jedoch allein gefunden.

Beide Testpersonen hatten das Problem, dass sie die Anpassung der Overpass-Query vergessen hätten. Sie wurden von der Testleiterin darauf hingewiesen, dass sie in Schritt 3 noch eine weitere Aufgabe haben. TP1 war sich unsicher, ob er die Schritte zur Anpassung vor dem Skript ausführen

muss, da die Erklärung dazu unter dem Skript steht. In der Anleitung ist zwar ein Verweis, der auf die Möglichkeit der Anpassung hinweist und intern zum Abschnitt verweist, dieser wurde aber von Testperson 1 und Testperson 2 überlesen. Testperson 1 sagte, dass sie aufgrund der Usability-Untersuchung weniger genau gelesen hat (Vergleich Transkript TP1, Zeile 300-307). Testperson 2 hat den gesamten Einleitungstext zum Skript overpass_api_call.py gelesen, aber dann zwei Zeilen vor dem Hyperlink aufgehört zu lesen. Ob die Hervorhebung der Verlinkung, wie sie in Abbildung 17 zu sehen ist, nicht ausreichend war und eine andere Verlinkung besser geholfen hätte, ist unklar. Beide Testpersonen merkten an, dass sie der Meinung sind, die Anleitung zum Anpassen der Query sollte über dem Arbeitsschritt positioniert sein, in welchem die Query verwendet wird (vgl. Transkript TP1, Zeile 118-128; vgl. Transkript TP2, Zeile 295-303).

Weiterhin hat Testperson 2 die Anleitung zum Anpassen der Query nicht vollständig gelesen, was dazu führte, dass er die neue Zeile an einer anderen Stelle als vorgegeben einfügen wollte. Dies hätte zwar zu keinem Fehler geführt, war jedoch trotzdem anders als von der Anleitung vorgesehen. Testperson 2 merkte dazu jedoch an, dass er nicht richtig gelesen hat und der Fehler deswegen aufgekommen ist (vgl. Transkript TP2, Zeile 86-93).

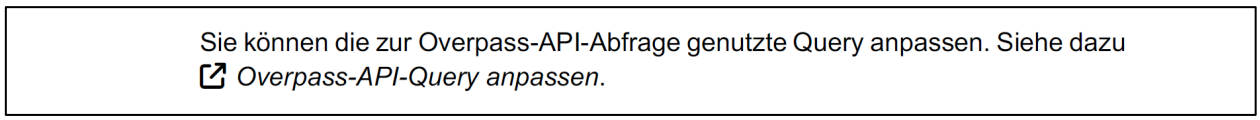


Abbildung 17: Ausschnitt aus der Anwenderdokumentation - Verlinkung Overpass-Query anpassen

Testperson 2 merkte während der Untersuchung an, dass er sich wünschen würde, dass die Relevanz der Schwellenwerte kurz erklärt wird. Testperson 1 hat diese Relevanz gar nicht hinterfragt. Mit Testperson 2 ist anschließend ein kurzes Gespräch entstanden, in dem erklärt wurde, wozu die Schwellenwerte sind, daraufhin konnte er mehr mit diesen anfangen (vgl. Transkript TP2, Zeile 161-182).

Die Fragen aus dem leitfadengestützen Interview wurden wie folgt beantwortet:

Beide Testpersonen empfanden die Bearbeitung der Schritte positiv. Sie empfanden die Schritte als verständlich und gut erklärt. Testperson 1 sprach in dem Zusammenhang noch Mal den Fakt an, dass er sich gewünscht hätte, dass die Anleitung zum Anpassen der Querys über der Ausführung des Skriptes erklärt worden wäre. Testperson 2 sprach noch Mal sein Verständnisproblem hinsichtlich der Schwellenwerte an (vgl. Transkript TP1, Zeile 214-221; vgl. Transkript TP2, Zeile 201-208). Beide Testpersonen empfanden die Ausführung der Arbeitsschritte als einfach, wobei Testperson 1 hinzufügte, dass die Bearbeitung ohne Anleitung kompliziert gewesen wäre (vgl. Transkript TP1, Zeile 221-227; vgl. Transkript TP2, Zeile 209-212).

Testperson 1 fand es gut, dass er nicht ausschließlich im Terminal arbeiten musste, da er dort auch eine eindeutige Rückmeldung erhalten hat, dass die Daten eingegeben wurden und alles funktioniert hat. Er hätte die Eingaben über das Terminal als komplizierter empfunden (vgl. Transkript TP1, Zeile 229-239). Testperson 2 fand das GUI als sehr gut, merkte jedoch an, dass er unsicher ist, ob er die Konsole von allein so viel genutzt hätte. Er meint, dass ihm die Ausgaben im Terminal möglicherweise gar nicht si aufgefallen wären und er wünscht sich einen Hinweis in der GUI, dass im Terminal Ausgaben passieren (vgl. Transkript TP2, Zeile 213-230). Testperson 1 empfand es als inkonsistent, aber trotzdem gut, dass zwischen Terminal und GUI hin und her gesprungen werden musste (vgl. Transkipt TP1, Zeile 242-249). Testperson 2 empfand die Software als einheitlich und konsistent (vgl. Transkript TP2, Zeile 233-236).

Testperson 1 hatte keine Probleme, die erforderlichen Eingaben zu verstehen, bemerkte jedoch, dass es schwierig war, die Daten auf einem fremden Rechner zu finden (vgl. Transkript TP1, Zeile 250-260). Testperson 2 äußerte Unklarheiten hinsichtlich der Bedeutung der Schwellenwerte (vgl. Transkript TP2, Zeile 237-245).

Beide Testpersonen warn der Meinung, dass sie sich während der Nutzung nur wenige Details merken mussten, da die Anleitung die Schritte klar und einfach erklärte. Testperson 2 betonte, dass die Anleitung eine souveräne Unterstützung darstellte (vgl. Transkript TP1, Zeile 261-266; vgl. Transkript TP2, Zeile 246-251). Hinsichtlich der Frage, ob die Software ohne Anwenderdokumentation nutzbar wäre, gaben beide Testpersonen an, dass dies wahrscheinlich nicht der Fall sei. Beide Testpersonen merkten an, dass Personen mit mehr Vorwissen möglicherweise weniger Probleme hätten (vgl. Transkript TP1, Zeile 267-271; vgl. Transkript TP2, Zeile 252-261).

Beide Testpersonen wussten durch die Anleitung stets, was die Skripte während der Nutzung taten. Sie gaben an, dass die Erklärungen in der Anleitung dabei hilfreich waren (vgl. Transkript TP1, Zeile 272-277; vgl. Transkript TP2, Zeile 262-266).

Die Fehlermeldungen wurden als verständlich und hilfreich wahrgenommen. Testperson 1 hob hervor, dass bekannte Fehler, wie der XML-Import per osm2pgsql, in der Anleitung gut erklärt waren (vgl. Transkript TP1, Zeile 278-283). Testperson 2 hatte hingegen nur Schwierigkeiten mit der Visualisierung in der Mac-Konsole, empfand die Fehlermeldung an sich aber als verständlich (vgl. Transkript TP2, Zeile 267-276). Zur Erweiterbarkeit der Software äußerte Testperson 1, dass sie dies nicht einschätzen könne, da ihr das Wissen über den Aufbau der Software fehlt (vgl. Transkript TP1, Zeile 284-289). Testperson 2 hielt die Software nach einer gewissen Einarbeitungszeit für leicht erweiterbar, da es sich um mehrere kleine Programme handelt (vgl. Transkript TP2, Zeile 279-291).

Beide Testpersonen empfanden die Anleitung als einheitlich und verständlich. Testperson 2 hatte jedoch Schwierigkeiten, die Anpassung der Overpass-Query zu verstehen, und verlor beim Klicken auf einen Hyperlink kurzzeitig die Orientierung im Dokument (vgl. Transkript TP1, Zeile 290-294; vgl. Transkript TP2, Zeile 292-303). Beide Testpersonen gaben an, dass es in der Anleitung keine unklaren oder zu wenig detaillierten Stellen gab (vgl. Transkript TP1, Zeile 295-299; vgl. Transkript TP2, Zeile 306-310). Auf die abschließende Frage, ob sie noch weitere Anmerkungen hätten, verneinten beide (vgl. Transkript TP1, Zeile 311; vgl. Transkript TP2, Zeile 312).

Fazit zu den Untersuchungsergebnissen

Die in den Interviews von den Testpersonen gegebenen Antworten zeichnen ein überwiegend positives Bild zur erstellten Anwenderdokumentation und den zugehörigen Softwareskripten. Die Ergebnisse des Usability-Tests zeigen auf, dass die Anleitung und die Softwareskripte insgesamt leicht verständlich sind. Auch wenn beiden Testpersonen das Vorwissen fehlte, konnten Sie mit der Anleitung gut umgehen. Es fielen Probleme hinsichtlich des Designs der Anwenderdokumentation auf. Die Beschreibung darüber, wie die Overpass Query angepasst werden kann, sollte vor dem Ausführen des Schrittes, der die Query nutzt, positioniert sein. Außerdem sollte Quellcode so in der Anleitung eingefügt werden, dass keine falschen Zeilenumbrüche entstehen, die bei korrektem Kopieren aus der Anleitung hinaus zu Fehlern in der Ausführung führen. Die eckigen Klammern als Platzhalter-Indikatoren sollten überdacht werden.

Die Usability-Untersuchung hat gezeigt, dass die Softwareskripte und die Anleitung dazu beigetragen haben, den sonst komplizierten Prozess auch für Personen ohne Vorwissen einfach umsetzbar und reproduzierbar zu machen. Die Informationen in der Dokumentation wurden als verständlich und klar eingeschätzt, den Testpersonen war zu jeder Zeit klar, was die Software gerade tut. Die Schritte zur Anpassung der Konfigurationsdatei waren klar, nur der Zeitpunkt der Erklärung in der Anwendung ist nicht optimal gewählt. Unerwartete Fehler traten nicht auf.

Es kam selten zu Verständnisproblemen, doch ein zusätzlicher Hinweis zur Bedeutung der Schwellenwerte wäre hilfreich gewesen. Die Anleitung wurde nicht als zu wenig detailliert wahrgenommen. Ob eine Reduktion der Inhalte dafür gesorgt hätte, dass die Schritte genauer gelesen werden, ist unklar. Den Testpersonen war mit Ausnahme dessen, dass Testperson 2 den Sinn der Schwellenwerte nicht verstand, zu jedem Zeitpunkt klar, welche Daten sie eingeben müssen und was das Programm gerade tut.

Insgesamt lässt sich also durch den Usability-Test sagen, dass die Anleitung und die Softwareskripte den erzielten Zweck erfüllt haben. Eine Untersuchung mit mehr Testpersonen wäre wünschenswert, um diese Ergebnisse weiter zu stützen und zusätzliche Erkenntnisse zu gewinnen.

9 Diskussion und Ausblick

Die vorliegende Arbeit hatte das Ziel, einen Workflow zur Verknüpfung von 3D-Stadtmodellen mit weiteren mobilitätsrelevanten Daten zu erarbeiten. Es sollte ein flexibler Workflow, welcher möglichst viele potenzielle Anforderungen und Herausforderungen berücksichtigt, erstellt werden.

Um dieses Ziel zu erreichen, wurden als erstes die Grundlagen zu 3D-Stadtmodellen behandelt und es wurde eine Definition für den Begriff *mobilitätsrelevante Daten* erarbeitet. Verschiedene Datenformate wurden vorgestellt. Anschließend befasste sich die Arbeit ausführlich mit möglichen Anwendungsgebieten der mobilitätsrelevanten Daten, den ihnen zugrunde liegenden Datenformaten und der Datenqualität. Nachdem das gewählte Praxisbeispiel genauer beschrieben wurde, begann die Erarbeitung des Workflows. Dabei wurden die nötigen Schritte zur Verknüpfung von CityGML-Daten, OSM-Daten und GTFS-Daten von Anfang an beschrieben. Als erstes wurden mögliche Datenquellen für CityGML-Daten beschrieben. Im Anschluss daran wurde eine systematische Literaturrecherche über Einflussfaktoren auf nachhaltige Mobilität durchgeführt. Die sich daraus ergebenen Textstellen wurden zu Maßnahmen zusammengefasst und diese Maßnahmen wurden in OSM-Tags übertragen. Die entstandene Overpass-API-Abfrage wurde genutzt, um die Rohdaten aus OSM zu extrahieren. Dabei wurden drei verschiedene Ansätze zum Festlegen der Begrenzung für die Overpass-Abfrage vorgestellt: die Datenextraktion aus einer CityGML-Datei, die Nutzung von GeoJSON und die Integration von GeoNames. Im Anschluss wurden die GTFS-Daten bezogen. Nachdem die Datenqualität auf Grundlage der in Abschnitt 2.5 erarbeiteten Erkenntnisse speziell für die vorliegenden Rohdaten überprüft wurde, war der nächste Schritt bereits die Datenverknüpfung. Für die Datenverknüpfung ist als erstes ein Import der Rohdaten in verschiedene Datenbanken nötig. In diesem Zuge wurde ein Docker-Netzwerk mit drei Containern erstellt. In die darin befindlichen, auf PostGIS beruhenden Datenbanken wurden unter Nutzung der Tools 3DCityDB Importer und osm2pgsql die Rohdaten importiert. Auch die GTFS-Daten wurden ausgelesen und in eine Datenbank importiert. Um die Verknüpfung der Daten möglich zu machen, fand eine Umwandlung in ein einheitliches, meterbasiertes Koordinatensystem statt. Vor der eigentlichen Datenverknüpfung wurden Verknüpfungsregeln aufgestellt und übersichtlich festgehalten. Die Implementierung der Datenverknüpfung fand mit Hilfe von Python statt.

Nachdem der Workflow einmal von Anfang bis Ende erarbeitet wurde, wurde sich zu einer Automatisierung des Prozesses entschieden. Dazu wurde die Programmiersprache Python verwendet. In 5 einzelnen Skripten wurden die Schritte getrennt voneinander automatisiert. Um die Nutzung der Skripte zu vereinfachen und den Workflow verständlich zu erklären wurde im Anschluss eine Anwenderdokumentation erstellt. Diese wurde, um ihre Nützlichkeit zu testen, zusammen mit den Softwareskripten in einem Usability-Test überprüft.

Die Bearbeitung dieser Arbeit sollte dazu genutzt werden, um vier Forschungsfragen zu beantworten. Das erste Ziel war herauszufinden, welche mobilitätsrelevanten Daten für die Anreicherung von 3D-Stadtmodellen geeignet sind. Um diese Frage zu beantworten, wurde zuerst der Begriff *mobiltätsrelevante Daten* auf Basis einschlägiger Literatur definiert. Außerdem wurde eine systematische Literaturrecherche über städtebauliche Maßnahmen zur Förderung nachhaltiger Mobilität durchgeführt. Im Ergebnis entstand eine Liste aus 181 Maßnahmen. Diese wurden den Datenkategorien der Mobilithek zugeordnet. Diese Zuordnung kann für Weiterentwicklungen des Workflows oder andere verwandte Arbeiten genutzt werden, um die verwendeten Tags zu filtern. Im Anschluss an die Kategorisierung erfolgte die Überführung der Maßnahmen in OSM-Tags. Die systematische Überführung dieser war jedoch sehr zeitaufwendig. Selbst unter Nutzung des KI-Tools ChatGPT konnte der Prozess kaum beschleunigt werden, sodass er nach 15 Stunden bei der Hälfte der Maßnahmen beendet wurde. Eine Liste der Maßnahmen inklusive zugeordneter OSM-Tags konnten im Ergebnis zur Verfügung gestellt werden.

Die zweite zu beantwortende Forschungsfrage beschäftigte sich damit, herauszufinden, welche kostenlosen oder Open-Source-Softwarelösungen geeignet sind, um 3D-Stadtmodelle mit weiteren

mobilitätsrelevanten Daten zu verknüpfen. Außerdem sollte herausgefunden werden, wie dieser Prozess anschließend automatisiert werden kann. Um diese Fragen zu beantworten, wurde eine strategische Vorgehensweise zur Erarbeitung des Workflows gewählt. Mögliche Schritte wurden als erstes auf ihre Praktikabilität geprüft, dabei auftretende Fehler und Probleme wurden dokumentiert. Durch dieses Vorgehen kann die vorgelegte Arbeit einen Mehrwert zu anderer relevanter Literatur liefern. Dort wurden Prozesse meist nur angerissen und das Konzept erklärt. Eine Auflistung möglicher Probleme und Hürden fehlt jedoch in der Literatur. An dieser Stelle reiht sich diese Arbeit ein und schließt diese Lücke. Der Prozess zur Verknüpfung der Daten konnte unter Nutzung mehrerer Docker-Container, der Tools 3DCityDB inklusive Importer/Exporter und osm2pgsql umgesetzt und mit Hilfe von Python automatisiert werden. Der beschriebene Prozess reicht von der Beschaffung der Rohdaten über die Erstellung der Docker-Container und den Datenimport bis hin zur Implementation der Verknüpfungsregeln mittels PostGIS. Durch die beschriebene Herangehensweise konnte in diesem Zuge auch die dritte Forschungsfrage beantworte werden, welche sich damit beschäftigt, welche Herausforderungen es bei der Integration der Daten gibt, auch speziell hinsichtlich der Datenqualität und Zuordnung. Zusammenfassend lässt sich feststellen, dass die geografische Zuordnung der einzelnen Datenpunkte grundsätzlich praktikabel ist. Dennoch können Importfehler und unterschiedliche Datenkonfigurationen zu Kompatibilitätsproblemen führen. Daher ist eine vorangehende Überprüfung der Rohdaten empfehlenswert, um beispielsweise festzustellen, ob sich Objekte überschneiden. Dies ermöglicht eine fundierte Bewertung der Ergebnisse der Verknüpfungsoperationen.

Die Erarbeitung des Workflows führte zu einigen Hürden. Insgesamt gestaltete sich der gesamte Prozess als komplex. Dies lag insbesondere an der Vielzahl der einzelnen nötigen Schritte sowie den unterschiedlichen Import- und Exportformaten. Die Überführung der als relevant erachteten Maßnahmen zur Förderung nachhaltiger Mobilität in OSM-Tags war sehr zeitaufwendig und wurde aus diesem Grund bei der Hälfte abgebrochen. Ein schneller Workflow für eine solche Überführung fehlt aktuell. Die Koordinatenfestlegung für den Overpass-Abruf erfordert sorgfältige Planung, damit das relevante Gebiet einbezogen wird.

Generell erwies sich die Nutzung der Tools 3DCityDB Importer/Exporter und osm2pgsql für Anfänger*innen als unübersichtlich und komplex. Durch die vielen Konfigurationsmöglichkeiten verliert man schnell den Überblick über nötige Eingaben und Fehlerquellen.

Der Import der OSM-Daten in die PostGIS-Datenbank mittels osm2pgsql war mit zahlreichen Fehlern behaftet. Trotz intensiver Bemühungen konnten diese Fehler nicht behoben werden, was den weiteren Workflow erheblich störte. Infolgedessen konnten die Verknüpfungsoperationen nicht durchgeführt werden, wodurch auch die anschließende Zuordnung der verknüpften OSM-Tags zu den CityGML-Daten in der 3DCityDB unmöglich wurde. Auch die Festlegung von Verknüpfungsregeln erfordert eine sorgfältige Planung. Dank der Erweiterungen von PostGIS gestaltet sich deren Implementierung jedoch effizient und unkompliziert, was den Integrationsprozess erleichtert. Der gesamte Prozess ist aufgrund der zahlreichen manuellen Schritte zeitaufwendig und anfällig für Fehler, insbesondere bei der Eingabe spezifischer Daten wie Dateipfade, Nutzernamen, Passwörter und Datenbankinformationen.

Der am Beispiel erstellte Workflow erleichtert die Schritte des Abrufs der mobilitätsrelevanten Daten von der Overpass API, der Erstellung der Docker-Container und Datenbanken. Auch die Schritte des Imports des 3D-Stadtmodells über den 3DCityDB Importer und der Import der relevanten Daten in eine dritte Tabelle zur Verknüpfung sind abgedeckt.

Die Automatisierung des Workflows, wie in Abschnitt 5.6 erläutert, konnte viele der zuvor auftretenden Probleme vollständig beheben oder zumindest verbessern. Allerdings wurde der Workflow ausschließlich am Beispiel der Leipziger Innenstadt entwickelt und mit Daten anderer Gebiete nicht getestet, wodurch die Fehlertoleranz für unterschiedliche Datensätze gering ist. Die Automatisierung beschleunigt die Verknüpfung mobilitätsrelevanter Daten erheblich, indem sie die Identifikation und Überführung in OSM-Tags durch vorgefertigte Konfigurationsdateien ersetzt und

die Festlegung der Koordinaten über einfache Eingaben erleichtert. Zudem ist die Nutzung der Tools 3DCityDB Importer/Exporter sowie osm2pgsql nun auch für Anfänger*innen zugänglich, da die notwendigen Docker-Befehle im Quellcode vorkonfiguriert sind und dennoch anpassbar bleiben.

Ein bestehender Fehler beim Import der OSM-Daten in die PostGIS-Datenbank mittels osm2pgsql konnte jedoch nicht durch die Automatisierung behoben werden, da der Importbefehl unverändert blieb. Die Automatisierung vereinfacht zudem die Ausführung der Verknüpfungsregeln, erfordert jedoch weiterhin eine sorgfältige Planung. Vordefinierte Regeln ermöglichen die Eingabe von Schwellenwerten, während Anwender*innen die Möglichkeit haben, diese an ihre spezifischen Bedürfnisse anzupassen. Insgesamt wurde der Prozess durch die Automatisierung stark vereinfacht, da keine manuellen Anpassungen spezifischer Daten wie Docker-Container-Namen, Datenbanknamen oder Dateipfade mehr erforderlich sind. Die Container-Erstellung erfolgt nun über vordefinierte Daten und die Dateinamen werden über eine grafische Benutzeroberfläche eingefügt, was den Workflow effizienter und benutzerfreundlicher macht.

Bei der erfolgten Automatisierung muss jedoch auch kritisch betrachtet werden, dass die Automatisierung zu einer verringerten Flexibilität führt, da nur die einprogrammierten Eingabe- und Ausgabeformate ohne weiteren Aufwand genutzt werden können. Anpassungen am Workflow sind nicht einfach und können dazu führen, dass spätere Schritte im Workflow nicht mehr problemlos funktionieren.

Aufgrund der hohen Fehleranfälligkeit des Prozesses, der durch zahlreiche Schritte und Tools bedingt ist, sind umfangreiche Tests mit vielfältigen Rohdaten notwendig, um unterschiedliche Möglichkeiten der Datenintegration, Import-Ergebnisse sowie die Validität der Verknüpfungsoperationen und weitere Fehlerquellen zu identifizieren. Der derzeit entwickelte Workflow funktioniert möglicherweise nur mit den verwendeten Daten, weshalb bei der Nutzung anderer Daten Verknüpfungsregeln angepasst und die Importschritte sorgfältig analysiert werden müssen, was die Effektivität der Automatisierung infrage stellt. Automatisierung eignet sich vor allem für die wiederholte Verarbeitung gleicher Daten. Die Programmierung ist jedoch sehr aufwendig und kann Importfehler sowie falsche Datenbankzuweisungen nicht vollständig vermeiden. Eine universelle Automatisierung für verschiedene Daten und Gebiete erfordert erheblichen Aufwand und ein umfangreiches Projekt, das die Prüfung unterschiedlicher Datensätze und die Recherche möglicher Fehlerquellen einschließt. Zudem ist ein zusätzlicher Workflow zur Kontrolle der Verknüpfungsergebnisse notwendig, wobei der Einsatz kostenpflichtiger Tools zur Überprüfung der Ergebnisse in Erwägung gezogen werden könnte.

Der im Rahmen dieser Arbeit entwickelte Programmcode ist für weitere Anwendungen vorgesehen und bildet eine solide Grundlage, auf der aufgebaut werden kann. Da ein solcher Prozess, wie in der Einleitung beschrieben, bislang nicht in dieser Ausführlichkeit dokumentiert wurde, bietet die Weiterverfolgung des Projekts, beispielsweise in Zusammenarbeit mit erfahrenen Softwareentwickler*innen, die sich mit den verwendeten Tools wie Osmosis oder osm2pgsql auskennen, die Möglichkeit, den Workflow fehlerfrei zu gestalten. Zudem wäre eine Anpassung der Skripte für weitere Betriebssysteme wie Linux und Windows sinnvoll. Verbesserungspotenziale bestehen hinsichtlich der Einheitlichkeit der Terminal- und Log-Ausgaben. Aufgrund der, trotz langjähriger Erfahrung in der Softwareentwicklung, begrenzten Programmiererfahrung der Verfasserin speziell in Python wäre es ratsam, den Code von erfahrenen Python-Entwickler*innen überprüfen zu lassen.

Die letzte Forschungsfrage dieser Arbeit beschäftigte sich damit, wie die entwickelten Skripte in einer Anwenderdokumentation festgehalten werden können, um den Prozess verständlich, praxisnah und reproduzierbar zu gestalten. Zur Beantwortung dieser Frage wurde auf Basis von Fachliteratur eine umfassende Anwenderdokumentation erstellt. Die Usability dieser Dokumentation in Verbindung mit den Softwareskripten wurde anschließend in einem Usability-Test überprüft. Die Ergebnisse zeigten insgesamt ein positives Bild, wobei nur wenige Probleme auftraten, die hauptsächlich auf gestalterische Entscheidungen in der Anleitung zurückzuführen waren. So wurden beispielsweise Platzhalterzeichen

nicht als solche erkannt und der Abschnitt zur Anpassung der Overpass-Query wurde an einer unerwarteten Stelle platziert. Zudem ist darauf zu achten, dass beim Einfügen von Quellcode in die Dokumentation keine unerwünschten Zeilenumbrüche entstehen, da dies sonst zu Fehlern bei der Nutzung des Codes führen kann.

Die zwei Testpersonen, die kein Vorwissen im Bereich Geodaten hatten, bewerteten den Gesamtprozess abgesehen von einigen Unklarheiten als verständlich. Die erforderlichen Eingaben und auftretenden Fehler wurden als klar und nachvollziehbar dargestellt bewertet, sodass die Anwenderdokumentation die Testpersonen erfolgreich bei der Bewältigung der Aufgaben unterstützte. Es ist jedoch zu beachten, dass eine Usability-Untersuchung mit nur zwei Testpersonen keine vollständig verlässlichen Ergebnisse liefert. Angesichts des aktuellen Entwicklungsstands des Workflows, der sich noch in der ersten Version befindet und nur begrenzte Eingaben sowie wenige Aktionen ermöglicht, war der Test mit zwei Teilnehmern dennoch sinnvoll. Dadurch konnte die allgemeine Nützlichkeit der Anwendung und Skripte bereits gut kontrolliert werden. Schwerwiegende Probleme in der Anleitung wären vermutlich auch bei nur zwei Testpersonen aufgetreten, sodass das Fehlen solcher Probleme darauf hindeutet, dass die Anwenderdokumentation ihren grundlegenden Zweck erfüllt.

Die weitere Bearbeitung des in dieser Arbeit untersuchten Themas wird empfohlen. Zunächst sollte der entwickelte Workflow mit Daten aus unterschiedlichen geografischen Gebieten getestet werden, um festzustellen, ob er universell einsetzbar ist oder für jeden Datensatz individuell angepasst werden muss. Zudem könnte der Workflow erweitert werden, um zusätzliche Datentypen in die Verknüpfung einzubeziehen. Die Datenverknüpfung von 3D-Stadtmodellen mit mobilitätsrelevanten Daten ist für Personen ohne Vorkenntnisse im Umgang mit Geodaten ein komplexer Prozess, der in der Literatur bisher nicht ausreichend dokumentiert ist. Diese Arbeit schließt diese Lücke, indem sie praxisnahe Erkenntnisse liefert, die als wichtige Grundlage für zukünftige Anwendungen dienen können.

Ob die aufbereitete Methodik auch für erfahrene Anwender*innen hilfreich ist, bleibt unklar. Für diese Nutzergruppe könnte die Arbeit entweder zu grundlegend sein oder genau die Problemstellen adressieren. Um die Nützlichkeit für verschiedene Nutzergruppen zu evaluieren, wäre es sinnvoll, Interviews mit Fachpersonal aus relevanten Bereichen durchzuführen. So könnten sowohl Mitarbeiter*innen, die intensiv mit 3D-Stadtmodellen arbeiten, als auch jene, die wenig bis gar keine Erfahrung damit haben, befragt werden, ob ihnen die entwickelten Workflows, die Anwenderdokumentation und die Softwareskripte als hilfreich erscheinen.

Weitere interessante Ansätze zur Anpassung und Erweiterung des Workflows umfassen die Implementierung semantischer Verknüpfungen neben den bereits bestehenden geografischen Verknüpfungen. Außerdem könnte der Exportprozess in die Automatisierung integriert werden. Ein spezifisches Problem bei der Nutzung von COLLADA2GLTF ist, dass für jedes Gebäude eine separate glTF-Datei erstellt wird. Diese könnten mithilfe eines Blender-Skripts basierend auf den zugeordneten Koordinaten korrekt positioniert werden, um die räumlichen Beziehungen der Gebäude zueinander zu erhalten. Eine anschließende Verarbeitung in Game-Engines wie Unity wäre durch den Export der Daten aus Blender möglich. Durch diese Erweiterungen und Anpassungen kann der Workflow weiter optimiert und vielseitiger einsetzbar gemacht werden, wodurch die Integration von 3D-Stadtmodellen mit mobilitätsrelevanten Daten noch effizienter und benutzerfreundlicher gestaltet wird.

Abschließend lässt sich sagen, dass die vorliegende Arbeit wertvolle Erkenntnisse in Probleme und Hürden bei der Datenverknüpfung von 3D-Stadtmodellen, OSM- und GTFS-Daten aufzeigt. Damit schließt sie eine Forschungslücke, da in der Literatur meist nur die generellen Ansätze, jedoch keine Fehlerquellen adressiert werden. Die erstellten Software-Skripte lassen sich, dank der formulierten Anleitung, leicht und zielführend verwenden und vereinfachen den Prozess dadurch stark.

Hilfsmittel und Literatur

Hilfsmittel

- **ChatGPT** (Version GPT-4o-mini & GPT-3.5, kostenfrei; Version GPT-4o, kostenpflichtig), OpenAI: https://openai.com
- **Duden Mentor** inklusive Microsoft Word-Plugin: https://mentor.duden.de
- **DeepL Translator**: https://www.deepl.com/de/translatorLiteratur
- **Microsoft Word Transkriptionstool**, Microsoft**:** https://support.microsoft.com/de-de/office/transkribieren-ihrer-aufnahmen-7fc2efec-245e-45f0-b053-2a97531ecf57

Literatur

Aichinger et al. 2020 – Aichinger, W., Frehn, M. and Pöpsel, L.; Quartiersmobilität gestalten - Verkehrsbelastungen reduzieren und Flächen gewinnen; Röthke-Habeck, P. (ed.). Umweltbundesamt, 2020

Beckmann et al. 2016 – Beckmann, K., Glemser, A., Heckel, C., von der Heyde, C., Hoffmeyer-Zlotnik, J. H., Hanefeld, U., Herter-Eschweiler, R. and Kühnen, C.; Demographische Standards: eine gemeinsame Empfehlung des ADM, Arbeitskreis Deutscher Markt- und Sozialforschungsinstitute e.V., der Arbeitsgemeinschaft Sozialwissenschaftlicher Institute e.V. (ASI) und des Statistischen Bundesamtes; Statistisches Bundesamt, 2016

Biljecki et al. 2015 – Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S. and Çöltekin, A.; Applications of 3D City Models: State of the Art Review; ISPRS International Journal of Geo-Information, MDPI AG, 2015, Vol. 4(4), pp. 2842-2889

BMI – Bundesministerium des Innern und für Heimat; Open Data; Online-Quelle: https://www.bmi.bund.de/DE/themen/moderne-verwaltung/open-government/open-data/open-data-node.html, letzter Abruf: 03.09.2024

Brandt et al. 2023 – Brandt, S., Henningsen, J., Hess, S., Jedlitschka, A., Hellmuth, R., Fricke, D., Gowers, K. and Blunk, S.; Digitale Zwillinge: Potenziale in der Stadtentwicklung; Bundesinstitut für Bau-, Stadt- und Raumforschung (BBSR) im Bundesamt für Bauwesen und Raumordnung (BBR), 2023

BSI – Smart City; Bundesamt für Sicherheit in der Informationstechnik; Online-Quelle: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Smart-City/smart-city_node.html, letzter Abruf: 09.10.2024

Buchholz & Flaig 2022 – Buchholz, P. and Flaig, S.; Push-Maßnahmen zur Reduzierung des motorisierten Individualverkehrs: Katalog zur nachhaltigen Veränderung des Modal Split; Bachhofer, M. (ed.); Bund für Umwelt und Naturschutz Deutschland (BUND), Landesverband Baden-Württemberg e.V., 2022

Wittwer et al. 2021 – Wittwer, R., Berger, M. and Nagel, T.; Einflussfaktoren des Mobilitätsverhaltens: Kurzüberblick zur Vorbereitung der Zukunftslabore; Technische Universität Dresden, Fakultät Verkehrswissenschaften "Friedrich List", Institut für Verkehrsplanung und Straßenverkehr, 2021

EU – Erschließung des Potenzials von Mobilitätsdaten; Europäische Kommission; Online-Quelle: https://digital-strategy.ec.europa.eu/de/policies/mobility-data, letzter Abruf: 10.10.2024

Faßbender 2020 – Faßbender, M.; Förderung nachhaltiger und aktiver Mobilität von Schülerinnen und Schülern - Möglichkeiten durch urbane infrastrukturelle Handlungsangebote wie die Wuppertaler

Nordbahntrasse und daran anknüpfende projektorientierte BNE in der Lehramtsausbildung im Fach Geographie; Bergische Universität Wuppertal, 2020

Fender 2020 – Fender, D. A.-C.; Förderung nachhaltiger Mobilität –Status quo des Mobilitätsverhaltens und Ableitungvon Handlungsansätzen in zwei sächsischen Hochschulen; Hochschule Osnabrück, 2020

Geoportal – Geoportal.de; suchen. finden. verbinden.; Online-Quelle: https://www.geoportal.de, letzter Abruf: 12.11.2024

GeoNames – GeoNames; About GeoNames; Online-Quelle: https://www.geonames.org/about.html, letzter Abruf: 01.01.2025

GeoJSON.io – geojson.io; geojson.io; Online-Quelle: https://geojson.io, letzter Abruf: 08.01.2025

GeoJSON – Internet Engineering Task Force (IETF); The GeoJSON Format; Online-Quelle: https://datatracker.ietf.org/doc/html/rfc7946, letzter Abruf: 06.01.2025

Gerike et al. 2020 – Gerike, R., Koszowski, C., Hubrich, S., Wittwer, R., Wittig, S., Pohle, M., Canzler, W. and Epp, J.; Aktive Mobilität: Mehr Lebensqualität in Ballungsräumen; Technische Universität Dresden, Professur für Integrierte Verkehrsplanung und Straßenverkehrstechnik, und Wissenschaftszentrum

GTFS DE – GTFS.DE; GTFS für Deutschland; Online-Quelle: https://gtfs.de, letzter Abruf: 26.12.2024

GTFS DE – NV – GTFS.DE; Öffentlicher Nahverkehr Deutschland; Online-Quelle: https://gtfs.de/de/feeds/de_nv/, letzter Abruf: 26.12.2024

GTFS Overview – General Transit Feed Specification; Overview; Online-Quelle: https://gtfs.org/documentation/overview/, letzter Abruf: 26.12.2024

GTFS Resources – General Transit Feed Specification; Resources: Data; Online-Quelle: https://gtfs.org/resources/data/, letzter Abruf: 18.09.2024

Günther et al. 2023 – Günther, M., Martinetz, S., Lessi, A. and Lotze, B.; 23 Maßnahmen zur Umsetzung aktiver und nachhaltiger Mobilität in Städten; Günther, M., Martinetz, S. & Lotze, B. (ed.); Fraunhofer-Institut für Arbeitswirtschaft und Organisation IAO, Fraunhofer IAO, 2023

Hirt & Nordhausen 2022 – Hirt, J. and Nordhausen, T.; Rechercheprotokoll für eine systematische Literaturrecherche; RefHunter. Systematische Literaturrecherche; RefHunter, 2022

Heldt & Hardinghaus 2022 – Heldt, B. and Hardinghaus, M.; Fact Sheet MUV Best-Practice: Umwidmung von Verkehrsflächen - Einfluss auf die lokale Ökonomie; 2022

IBM – IBM Think; Was sind Geodaten?; Online-Quelle: https://www.ibm.com/de-de/topics/geospatial-data, letzter Abruf: 10.10.2024

Inspire 2015 – Europäische Kommission; INSPIRE-Registry Geodaten; Online-Quelle: https://inspire.ec.europa.eu/glossary/SpatialData, 2015, letzter Abruf: 09.10.2024

Keler 2016 – Keler, A.; Öffentliche Geobasisdaten; Brand, K., Blankenbach, J. & Kolbe, T. H. (ed.); Mobile GIS – Hardware, Software, IT-Sicherheit, Indoor-Positionierung, Version 2.1; Runder Tisch GIS e.V., 2016, pp. 115-118

Kolbe 2009 – Kolbe, T.; Representing and Exchanging 3D City Models with CityGML; Lee, J. & Zlatanova, S. (ed.); Proceedings of the 3rd International Workshop on 3D Geo-Information; Springer Verlag, 2009, pp. 15-31

Kolbe et al. 2021 – Kolbe, T. H., Kutzner, T., Smyth, C. S., Nagel, C., Roensdorf, C. and Heazel, C.; OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard; Kolbe, T. H., Kutzner, T., Smyth, C. S., Nagel, C., Roensdorf, C. & Heazel, C. (ed.); Open Geospatial Consortium, 2021(20-010)

Kothes 2011 – Kothes, L.; Grundlagen der Technischen Dokumentation; Springer Berlin, Heidelberg, 2011, pp. XII, 308

Ledoux et al. 2019 – Ledoux, H., Arroyo Ohori, K., Kumar, K., Dukai, B., Labetski, A. and Vitalis, S.; CityJSON: a compact and easy-to-use encoding of the CityGML data model; Open Geospatial Data, Software and Standards, Springer Science and Business Media LLC, 2019, Vol. 4(4)

Leipzig 3D Energie – Stadt Leipzig; Thematisches 3D-Stadtmodell: Energie, Umwelt & Klima; Online-Quelle: https://www.leipzig.de/bauen-und-wohnen/bauen/geodaten-und-karten/3d-stadtmodell-energie-umwelt-klima, letzter Abruf: 12.12.2024

Levin-Keitel et al. 2019 – Levin-Keitel, M., Othengrafen, F. and Behrend, L.; Urban planning as a discipline. Planner's everyday routines and self-conceptions; Raumforschung und Raumordnung | Spatial Research and Planning, Oekom Publishers GmbH, 2019, Vol. 77(2), pp. 115-130

Mobilithek – Mobilithek; Deutschlands Plattform für Daten, die etwas bewegen; Online-Quelle: https://mobilithek.info, letzter Abruf: 10.12.2024

Mobilithek About – Mobilithek; Über - Deutschlands Plattform für Daten, die etwas bewegen; Online-Quelle: https://mobilithek.info/about, letzter Abruf: 20.11.2024

Mobilithek Categories – Mobilithek; Alle Themengebiete im Datenkatalog; Online-Quelle: https://mobilithek.info/all-categories, letzter Abruf: 30.12.2024

Münsch & Lell 2024 – Münsch, M. and Lell, O.; Für Mensch & Umwelt: Zwischenbericht Anreize zur Förderung eines nachhaltigen Mobilitätsverhaltens: Forschungskennzahl 3722 58 101 0, 2024(TEXTE 03/2024)

NachhaltigeJobs – NachhaltigeJobs; Berufsbild nachhaltige Stadtplanung: »Was lange bestehen soll, will gut geplant sein.«; Online-Quelle: https://www.nachhaltigejobs.de/berufsbild-nachhaltige-stadtplanung/m, letzter Abruf: 16.12.2024

OGC 2023 – Open Geospatial Consortium; OGC City Geography Markup Language (CityGML) Part 2: GML Encoding Standard; Online-Quelle: https://docs.ogc.org/is/21-006r2/21-006r2.html, letzter Abruf: 03.12.2024

OGC – Open Geospatial Consortium; CityGML; Online-Quelle: https://www.ogc.org/de/publications/standard/citygml/, letzter Abruf: 13.10.2024

Ohsome – ohsome quality analys; BKG OSM-Prüftools; Online-Quelle: https://giscience.github.io/oqt-bkg-demo/index_de.html?id=10&report=Report, letzter Abruf: 04.01.2025

OpenData Leipzig – Stadt Leipzig; 3D-Stadtmodell LoD2, Stadt Leipzig; Online-Quelle: https://opendata.leipzig.de/dataset/3d-stadtmodell, letzter Abruf: 10.10.2024

OpenStreetMap Develop – OpenStreetMap Wiki; DE:Develop; Online-Quelle: https://wiki.openstreetmap.org/wiki/DE:Develop, letzter Abruf: 13.11.2024

OpenStreetMap Elements – OpenStreetMap Wiki; Elements; Online-Quelle: https://wiki.openstreetmap.org/wiki/Elements, letzter Abruf: 08.09.2024

OpenStreetMap FAQ – OpenStreetMap Deutschland; FAQ: Was ist OpenStreetMap?; Online-Quelle: https://openstreetmap.de/faq/#was-ist-openstreetmap, letzter Abruf: 19.12.2024

OpenStreetMap PBF – OpenStreetMap Wiki; DE:PBF Format; Online-Quelle: https://wiki.openstreetmap.org/wiki/DE:PBF_Format, letzter Abruf: 20.12.2024

OpenStreetMap Quality – OpenStreetMap Wiki; Quality Assurance; Online-Quelle: https://wiki.openstreetmap.org/wiki/Quality_assurance, letzter Abruf: 19.12.2024

OpenStreetMap WikiAPIs – OpenStreetMap Wiki; Databases and data access APIs; Online-Quelle: https://wiki.openstreetmap.org/wiki/Databases_and_data_access_APIs, letzter Abruf: 19.12.2024

OSM – OpenStreetMap; OpenStreetMap; Online-Quelle: https://openstreetmap.org, letzter Abruf: 13.09.2024

OSM Wiki – OpenStreetMap Wiki contributors; OpenStreetMap Wiki; Online-Quelle: https://wiki.openstreetmap.org, letzter Abruf: 03.01.2025

Overpass bbox – Overpass API User's Manual; Bounding Boxes; Online-Quelle: https://dev.overpass-api.de/overpass-doc/en/full_data/bbox.html, letzter Abruf: 22.12.2024

Overpass Turbo – Overpass Turbo; Online-Quelle: https://overpass-turbo.eu, letzter Zugriff: 08.01.2025

PostgreSQL – PostgreSQL; F.1.17 hstore; Online-Quelle: https://www.postgresql.org/docs/current/hstore.html, letzter Abruf: 19.12.2024

Pümper – Pümper, J.; Fragebogen ISONORM 9241/110-S. Beurteilung von Software auf Grundlage der; Internationalen Ergonomie-Norm DIN EN ISO 9241-110.; Online-Quelle: http://www.uselab.tu-berlin.de/wiki/images/6/62/ISONorm_Kurzversion.pdf, letzter Abruf: 01.01.2025

PostGIS – PostGIS; About PostGIS; Online-Quelle: https://postgis.net, letzter Abruf: 17.12.2024

PostGIS FAQ – PostGIS; Should I use the geometry type or the geography type?; Online-Quelle: https://postgis.net/documentation/faq/geometry-or-geography/, letzter Abruf: 11.12.2024

Python tkinter – Python; tkinter — Python interface to Tcl/Tk; Online-Quelle: https://docs.python.org/3/library/tkinter.html, letzter Abruf: 29.12.2024

Raunig & Hodzic-Srndic 2020 – Raunig, K. and Hodzic-Srndic, N.; Die vielfältigen Vorzüge aktiver Mobilität auf Mensch und Umwelt (und wie wir sie erreichen); 2020; Berlin für Sozialforschung gGmbH, Im Auftrag des Umweltbundesamtes, 2020(TEXTE 226/2020)

Reitberger et al. 2019 – Reitberger, R., Hänsch, R. and Berlin, C. M.; Planungshilfen: Gut gemeint -- wenig erfolgreich?; Nahverkehr, 2019, Vol. 37(4), pp. 37-40

Rohs et al. 2021 – Rohs, M., Flore, G. and Cavagna, M.; Auf dem Weg zu einer nachhaltigen urbanen Mobilität in der Stadt für Morgen; Büttner, A. (ed.); Umweltbundesamt, 2021

Rohs et al. 2023 – Rohs, M., Flore, G., Schubert, D. M. and Schäfer, P. D. P. K.; Mobilitätskonzepte für einen nachhaltigen Stadtverkehr 2050: Metaanalyse, Maßnahmen und Strategien; PricewaterhouseCoopers GmbH Wirtschaftsprüfungsgesellschaft, Im Auftrag des Umweltbundesamtes, 2023(TEXTE 12/2023)

Rubin & Chisnell 2008 – Rubin, J. und Chisnell, D.; Handbook of Usability Testing. How to Plan, Design, and Conduct Effective Tests.; Wiley Publishing, Inc., Indianapolis, Indiana, 2. Auflage, 2008

Sarodnick & Brau 2011 – Sarodnick, F. und Brau, H.; Methoden der Usability Evaluation. Wissenschaftliche Grundlagen und praktische Anwendung; Verlag Hans Huber, Hogrefe AG, Bern, 2., überarbeitete und aktualisierte Auflage, 2011

Scherhaufer et al. 2023 – Scherhaufer, P., Braito, M., Hinterreiter, M., Schauppenlehner-Kloyber, E. and Wegener, S.; Nachhaltiges Mobilitätsverhalten von der Nische zur Norm: Maßnahmen zur Förderung nachhaltiger Mobilität; Universität für Bodenkultur (BOKU), 2023

Schrapp et al. 2019 – Schrapp, L., Tobisch, C., Schroth, O. and Blum, P.; Qualität und Nutzbarkeit von OSM-Daten für landschaftsplanerische Fragestellungen; GIS-Zeitschrift fü Geoinformatik, 2019, Vol. 32, pp. 77-86

SimStadt – SimStadt Dokumentation; CityGML Viewer; Online-Quelle: https://simstadt.hft-stuttgart.de/related-softwares/city-gml-viewer/, letzter Abruf: 06.01.2025

SIG3D – SIG3D; Qualitätssicherung und -management; Online-Quelle: https://www.sig3d.de/de/quality.html, letzter Abruf: 13.12..2024

Statista 2024 – Statista; Architekturmarkt: Statistik-Report zum Thema Architekturmarkt; 2024

Städtetag 2017 – 3D-Geodaten in der integrierten Stadtentwicklung; Deutscher Städtetag, 2017; Online-Quelle: https://www.staedtetag.de/files/dst/docs/Publikationen/Weitere-Publikationen/2019/3d-geodaten-integrierte-stadtentwicklung-handreichung-2017.pdf, letzter Abruf: 07.12.2024

TagInfo – OpenStreetMap Taginfo; Taginfo; Online-Quelle: https://taginfo.openstreetmap.org, letzter Abruf: 04.01.2025

Tullis & Albert 2008 – Tullis, T. und Albert, B.; Measuring the User Experience. Collecting, Analyzing, and Presenting Usability Metrics; The Morgan Kaufmann interactive technologies series, Elsevier Inc., Burlington, 2008

Visit Berlin – VisitBerlin; TimeRide: Eine Zeitreise in das geteilte Berlin; Online-Quelle: https://www.visitberlin.de/de/timeride, letzter Abruf: 20.12.2024

Wang & Strong 1996 – Wang, R. Y. and Strong, D. M.; Beyond Accuracy: What Data Quality Means to Data Consumers; Journal of Management Information Systems, Informa UK Limited, 1996, Vol. 12(4), pp. 5-33

Weiler et al. 2018 – Weiler, V., Würstle, P., Schmitt, A., Stave, J., Braun, R., Zirak, M., Coors, V. and Eicker, U.; Methoden zur Integration von Sachdaten in CityGML Dateien zur Verbesserung der energetischen Analyse von Stadtquartieren und deren Visualisierung; 2018

Yao et al. 2016 – Yao, Z., Chaturvedi, K. and Kolbe, T. H.; Browser-basierte Visualisierung großer 3D-Stadtmodelle durch Erweiterung des Cesium Web Globe; e.V., R. T. G. I. S. (ed.); Geoinformationssysteme 2016 - Beiträge zur 3. Münchner GI-Runde; Wichmann Verlag, 2016, pp. 77-89

Yao et al. 2018 – Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T. and Kolbe, T. H.; 3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML; Open Geospatial Data, Software and Standards, Springer Science and Business Media LLC, 2018, Vol. 3(1)

3DCityDB – 3DCityDB; 3D City DB; Online-Quelle: https://www.3dcitydb.org/3dcitydb/, letzter Abruf: 13.12.2024

3DCityDB Import – 3dcitydb-docs; 4.9.2. Import command; Online-Quelle: https://3dcitydb-docs.readthedocs.io/en/latest/impexp/cli/import.html, letzter Abruf: 28.12.2024

Anhang 1 Rechercheprotokoll

Rechercheprotokoll für eine systematische Literaturrecherche

Vorlage übernommen aus (vgl. Hirt & Nordhausen 2022).

Forschungsfrage

Welche städtebaulichen Maßnahmen beeinflussen das nachhaltige Mobilitätsverhalten der Bevölkerung?

Ein- und Ausschlusskriterien

Domäne	Einschlusskriterien	Ausschlusskriterien
Domäne 1:	Deutsch, Englisch	Andere Sprachen
Domäne 2:	10 Jahre	Älter als 10 Jahre
Domäne 3:	Studie nennt Kriterien die nachhaltiges Mobilitätsverhalten fördern oder behindern	Studie tut das nicht
Domäne 4:	Studie nennt konkrete Maßnahmen zur Förderung nachhaltiger Mobilität	Studie tut das nicht
Domäne 5:	Studie nennt Maßnahmen im Bereich der Geodaten, Mobilitätsdaten oder Daten zur öffentlichen Raumgestaltung	Studie bezieht sich nur auf wirtschaftliche oder politische Maßnahmen

Das maximale Alter von 5 Jahren wurde festgelegt, um möglichst nur aktuelle Trends im Mobilitätsverhalten zu betrachten.

Festlegung des Rechercheprinzips

Als Rechercheprinzip wurde eine Mischform aus sensitivem und spezifischem Rechercheprinzip festgelegt, heißt, es sollen möglichst viele relevante Treffer mit einem optimierten Arbeitsaufwand gefunden werden.

Festlegung der Suchkomponenten

Suchkomponente	Bezeichnung
Suchkomponente 1	Einflussfaktoren
Suchkomponente 2	Mobilitätsverhalten

Festlegung der zu durchsuchenden Datenbanken

Datenbank	Bezeichnung
	Begründung
Datenbank 1	EBSCO Discovery Service
	Liefert Ergebnisse aus vielen verschiedenen Datenbanken, Journals, Büchern, eBooks und Open Access Werken
Datenbank 2	Umweltbundesamt (https://www.umweltbundesamt.de/publikationen)
	Liefert Ergebnisse aus vielen verschiedenen Datenbanken, Journals, Büchern, eBooks und Open Access Werken

Für die Literaturrecherche wurde der EBSCO Discovery Service ausgewählt, da er eine Vielzahl von Datenbanken abdeckt und dadurch eine breite Basis für die Recherche mobilitätsrelevanter Daten bietet. Für relevante Literatur aus Deutschland wurde außerdem die Publikationsübersicht des Umweltbundesamtes genutzt. Für spezifischere Anwendungsfälle können zusätzliche Datenbanken herangezogen werden.

Identifikation von Stichwörtern

Identifikation der Stichwörter und deren Synonyme je Suchkomponente. Sofern eine Begründung für (einzelne) gewählte oder nicht gewählte Stichwörter notwendig erscheint, kann diese unter der Tabelle festgehalten werden.

Suchkomponenten	Stichwörter
Suchkomponente 1: Einflussfaktoren	EINFLUSSFAKTOR EINFLUSS PUSH-FAKTOR PULL-FAKTOR MAßNAHMEN FÖRDERUNG HINDERNIS
Suchkomponente 2: Mobilitätsverhalten	MOBILITÄT MOBILITÄTSVERHALTEN NACHHALTIG

Das Stichwort «Maßnahmen» wurde hinsichtlich der Formulierung «Maßnahmen zur Verbesserung nachhaltiger Mobilität» o.ä. gewählt

Identifikation von Schlagwörtern

Der EBSCO Discovery Service nutzt in seiner Suche zwar Schlagworte, jedoch gibt es dafür keine direkte Übersicht, weswegen es schwierig ist diese im Vornherein abzugrenzen. Da der Discovery-Service jedoch automatisch auch nach ähnlichen Begriffen sucht kann davon ausgegangen werden, dass ausreichend viele Suchergebnisse gefunden werden. Aufgrund dessen wurde er Schritt «Identifikation von Schlagwörtern» übersprungen.

Entwicklung des Suchstrings und Durchführung der Recherche (EBSCO Discovery Service)

Die hier genannten Suchstrings wurden jeweils getestet und rekursiv angepasst, um möglichst wenige dafür sehr relevante Quellen zu identifizieren.

Bei der Suche in Ebsco Discovery Service ist bereits öfter aufgefallen, dass eine Einschränkung der Sprache manchmal nicht richtig funktioniert, weswegen relevante Literatur nicht erscheint. Deshalb wurde sich dafür entschieden, die Sprache nicht in den Suchfiltern zu hinterlegen.

1 - 23.11.2024

Datenbank: EBSCO Discovery Service

Suchfilter:

- Vergangene 5 Jahre
- Alle meine Suchbegriffe suchen
- Auch innerhalb des Volltext-Artikels suchen
- Semantische Konzepte nutzen

Suchstring (Alle Felder):

Suchkomponente	Suchstring: Datenbank 1
Suchkomponente 1: Einflussfaktoren	EINFLUSSFAKTOR OR EINFLUSS OR PUSH-FAKTOR OR PULL-FAKTOR OR MAßNAHMEN OR FÖRDERUNG OR HINDERNIS
	AND
Suchkomponente 2: Mobilitätsverhalten	MOBILITÄT OR

XC

Suchkomponente	Suchstring: Datenbank 1
	MOBILITÄTSVERHALTEN OR NACHHALTIG

Ergebnisse:

- Anzahl der Ergebnisse: 34482
- Relevanz: Viele relevante Ergebnisse

13768 Ergebnisse sind zu viele, deswegen wurde der Suchstring angepasst, um die Anzahl der Ergebnisse zu verringern.

2 - 23.11.2024

Datenbank: EBSCO Discovery Service

Suchfilter:

- Vergangene 5 Jahre
- Alle meine Suchbegriffe suchen
- Auch innerhalb des Volltext-Artikels suchen
- Semantische Konzepte nutzen

Suchstring:

Suchkomponente	Suchstring: Datenbank 1
Suchkomponente 1: Einflussfaktoren	EINFLUSSFAKTOR OR EINFLUSS OR PUSH-FAKTOR OR PULL-FAKTOR OR MAßNAHMEN OR FÖRDERUNG OR HINDERNIS
	AND
Suchkomponente 2: Mobilitätsverhalten	MOBILITÄT OR MOBILITÄTSVERHALTEN
Neue Suchkomponente	AND NACHHALTIG

Anmerkungen/ Ergebnisse:

- Anzahl der Ergebnisse: 2119
- Relevanz: teils irrelevante Ergebnisse, relevante Ergebnisse überwiegen, jedoch

Immer noch zu viele Ergebnisse, darum weitere Einschränkung

3 - 23.11.2024

Datenbank: EBSCO Discovery Service

Suchfilter:

- Vergangene 5 Jahre
- Alle meine Suchbegriffe suchen
- Auch innerhalb des Volltext-Artikels suchen

Suchstring: (Nur noch in Titel und Abstract gesucht)

Suchkomponente	Suchstring: Datenbank 1
	EINFLUSSFAKTOR OR

Suchkomponente	Suchstring: Datenbank 1
Suchkomponente 1: Einflussfaktoren	EINFLUSS OR PUSH-FAKTOR OR PULL-FAKTOR OR MAßNAHMEN
	AND
Suchkomponente 2: Mobilitätsverhalten	MOBILITÄT OR MOBILITÄTSVERHALTEN
Neue Suchkomponente	AND NACHHALTIG

Ergebnisse:

- Anzahl der Ergebnisse: 81
- Relevanz: teils hoch relevante Ergebnisse, manche Ergebnisse tauchen häufiger auf

Festhalten der relevanten Ergebnisse

Die Titel und Abstracts der Ergebnisse wurden gesichtet und die relevanten Literaturen ausgewählt. Nachfolgend genannte Ergebnisse wurden aufgrund der Ein- und Ausschlusskriterien und hinsichtlich der Zugänglichkeit als relevant erachtet. Die Quelle von Kopal wurde heruntergeladen, jedoch als unwichtig erachtet, da es sich dabei um ein Forschungsplakat handelt, welches keine direkten Rückschlüsse hinsichtlich der Forschungsfrage bietet.

RAUNIG, K.; HODZIC-SRNDIC, N. Die vielfältigen Vorzüge aktiver Mobilität auf Mensch und Umwelt (und wie wir sie erreichen). *[s. l.]*, 2020. Disponível em: https://research.ebsco.com/linkprocessor/plink?id=d3900621-8fdf-3b3c-a1a7-3dbe22f0eb75. Acesso em: 28 nov. 2024.

FAßBENDER, M. Förderung nachhaltiger und aktiver Mobilität von Schülerinnen und Schülern - Möglichkeiten durch urbane infrastrukturelle Handlungsangebote wie die Wuppertaler Nordbahntrasse und daran anknüpfende projektorientierte BNE in der Lehramtsausbildung im Fach Geographie ; Promoting sustainable and active mobility of pupils – opportunities through urban infrastructural measures such as the Wuppertaler Nordbahn Route and related project-oriented SDO in the field of geography training. 2020. Europe, Europe, 2020. Disponível em: https://research.ebsco.com/linkprocessor/plink?id=e2d36da3-1f2b-3e86-b9cb-b3a0f6efc1f4. Acesso em: 28 nov. 2024.

GÜNTHER, M. *et al.* 23 Maßnahmen zur Umsetzung aktiver und nachhaltiger Mobilität in Städten. Germany, Europe: Fraunhofer IAO, 2023. DOI 10.24406/publica-1903. Disponível em: https://research.ebsco.com/linkprocessor/plink?id=2a4c7afe-8ad9-3725-99ab-cd1d1fafa0eb. Acesso em: 28 nov. 2024.

~~KOPAL, K.; WITTOWSKY, D. Gesunde und nachhaltige Stadt - Einflüsse der gebauten Umwelt auf gesundes und nachhaltiges (Mobilitäts)-Verhalten. Nachhaltige StadtGesundheit; 7. Konferenz "Stadt der Zukunft" - Gesunde, Nachhaltige Metropolen, *[s. l.]*, 2022. DOI 10.11576/nsg-1074. Disponível em: https://research.ebsco.com/linkprocessor/plink?id=e3f886b9-f185-3fde-9ce3-1c62ed8e18fd. Acesso em: 28 nov. 2024.~~

FENDER, A.-C. Förderung nachhaltiger Mobilität Status quo des Mobilitätsverhaltens und Ableitung von Handlungsansätzen in zwei sächsischen Hochschulen. Osnabrück *[s. n.]*. Disponível em: https://research.ebsco.com/linkprocessor/plink?id=a7d4f71a-89ae-3341-b821-fb5bc3ed35cb. Acesso em: 28 nov. 2024.

HELDT, B.; HARDINGHAUS, M. Fact Sheet MUV Best-Practice "Umwidmung von Verkehrsflächen - Einfluss auf die lokale Ökonomie". Verkehrliche und stadtplanerische

Maßnahmen zur Neuverteilung und Umwidmung von Verkehrsflächen des motorisierten Verkehrs zugunsten aktiver Mobilität und einer nachhaltigen urbanen Siedlungsstruktur mit hoher Lebensqualität (MUV). *[s. l.]*, 2022. Disponível em: https://research.ebsco.com/linkprocessor/plink?id=7c797c99-798f-3182-958b-8afe4eadc6ec. Acesso em: 28 nov. 2024.

LEITNER, S. Nachhaltige Mobilitäts- und Wohnraumkonzepte zur Förderung der Lebensqualität und Ressourcenschonung ; Mobilitätskonzepte. 2023. Austria, Europe, 2023. Disponível em: https://research.ebsco.com/linkprocessor/plink?id=3d2fd7a8-34f1-3680-9422-c09b6a9dc59f. Acesso em: 28 nov. 2024.

MÜNSCH, M. Anreize zur Förderung eines nachhaltigen Mobilitätsverhaltens Stand der Forschung zu Wirkung und Einsatzmöglichkeiten materieller, immaterieller und spielerischer Anreize : Zwischenbericht. Dessau-Roßlau: Umweltbundesamt, 2024. Disponível em: https://research.ebsco.com/linkprocessor/plink?id=917f42b0-a64f-33d6-b5b2-b6277b72f092. Acesso em: 28 nov. 2024.

SCHERHAUFER, P. *et al.* Nachhaltiges Mobilitätsverhalten von der Nische zur Norm: Maßnahmen zur Förderung nachhaltiger Mobilität. Germany, Europe: AUT, 2023. Disponível em: https://research.ebsco.com/linkprocessor/plink?id=9e26e992-ab9c-3f9e-a58d-dd4e0aab6827. Acesso em: 28 nov. 2024.

Entwicklung des Suchstrings und Durchführung der Recherche (Umweltbundesamt)

Die Eingabemaske für die Recherche auf der Seite des Umweltbundesamtes erlaubt keine direkten Suchstings mit AND und OR, einzelne Suchwörter müssen eingegeben werden. Folgende Suchbegriffe wurden gewählt:

- Mobilität
- Nachhaltige Mobilität

Folgende Ergebnisse wurden als relevant erachtet:

- Quartiersmobilität gestalten - Verkehrsbelastungen reduzieren und Flächen gewinnen
- Aktive Mobilität: Mehr Lebensqualität in Ballungsräumen
- Auf dem Weg zu einer nachhaltigen urbanen Mobilität in der Stadt für Morgen
- Mobilitätskonzepte für einen nachhaltigen Stadtverkehr 2050: Metaanalyse, Maßnahmen und Strategien (Abschlussbericht)

Eine weitere hoch relevante Quelle, welche in der allgemeinen Grundlagenrecherche über die Suchmaschine Google auffiel, ist die Quelle «Buchholz & Flaig 2022».

Anhang 2 Ermittelte Maßnahmen und zugehörige Kategorien

Kategorie	Maßnahmen
Allgemeine Informationen zur Wegeplanung	(digitale) Infotafeln
	Fahrplan-Anzeigetafeln mit barrierefreien Ansagen
	Bänke platzieren
	Bäume pflanzen
	Begrünung
	Begrünung von Übergängen in Straßen mit Tempo 30
	keine Multifunktionsflächen
	Kombination von Wohn- und Gewerbenutzung
	Mobilitätszentren errichten
	Mülleimer platzieren
	Multifunktionsflächen errichten
	Notfallüberquerung offen halten
	Notrufsysteme einrichten
	Öffentliche Plätze beleuchten
	öffentliche Toiletten errichten
	öffentliche Trinkwasserstationen errichten
	Parklets einrichten
	Parklets in Tempo-30-Zonen
	Parks errichten
	Parkstreifen einführen
	Pflanzkübel platzieren
	Photopoints einrichten
	Schließfächer für E-Roller
	Schließfächer für E-Scooter
	Spiegel anbringen
	Spielplätze errichten
	Sprühduschen platzieren
	Überwachungskameras anbringen
	Unterführungen vermeiden
	Wickelräume errichten
	Rampen
Sharing-Verkehrsmittel	Bevorrechtigung Taxen & On-Demand-Verkehr
	Carsharing-Parkplätze einrichten
	E-Carsharing-Plätze
	E-Lastenrad-Sharing-Plätze
	Radsharing-Plätze
	Ridepooling-Systeme einführen
	Rollersharing-Angebote einrichten

Dynamische Verkehrsregelungen	Bedarfsampeln für MIV
	Vorrang-Ampeln für ÖPNV
	dynamische Schilder mit Ampelinformation für Radverkehr
	elektronisch versenkbare Poller
	früheres Grün für Radfahrer vor MIV
	Fußverkehrsampeln ohne Gültigkeit für Radverkehr
	Fußverkehrsampeln vor/hinter ÖPNV-Haltestellen
	Grüne Welle für ÖPNV
	Grüne Welle für Radverkehr
	längere Grünphasen für Fußgänger
	Pförtnerampeln einrichten
	Radfahr-Ampeln
	Sommerstraßen einrichten
	Verkehrszugangskontrolle über Nutzungshäufigkeit
	Wetterabhängige Anpassung von Ampelschaltungen für Fußverkehr
	Wetterabhängige Anpassung von Ampelschaltungen für Radverkehr
	Zufahrtsbeschränkung mindestens 2 Personen pro Fahrzeug
	Priorisierung von Fußgängern an Ampel
Fracht und Logistik	Durchfahrtsverbote für Schwerlastverkehr
	Haltepunkte für Lieferverkehr errichten
	Lieferzonen einrichten
	Nachtfahrtsverbote für Schwerlastverkehr
Fußverkehrsnetz	abgesenkte Bordsteine
	Ampelsysteme mit akustischen Signalen
	Ampelsysteme mit an ÖPNV angepasster Schaltung
	Fußgängerüberwege errichten
	ausreichend Fußgängerüberwege
	Fußverkehrsampeln errichten
	ausreichend Gehwegbeleuchtung aufstellen
	baulich getrennter Rad- und Fußweg
	breite Gehwege
	Ebenerdige Ampelquerungen für Fußgänger
	Gehwegüberfahrten einrichten
	geteilte Nutzung von Fahrstreifen durch Fahrräder und MIV
	Gute Oberflächenbeschaffenheit der Fußwege sicherstellen
	Haltestellen mit abgesenkten Bordsteinen
	Haltestellen mit Leitlinien
	lange und verbundene Gehwege
	Leitstreifen

	Leitstreifen an abgesenkten Bordsteinen
	niveaugleiche Pflasterung
	steigungsarme Gehwege
	temporäre Spielstraßen errichten
	Übergänge baulich erhöhen
	Verkehrsflächen klar trennen
	Vorrang-Ampeln für Fußgänger
Öffentlicher Verkehr: Gelegenheitsverkehr	Bevorrechtigung Taxen & On-Demand-Verkehr
	Haltepunkte für On-Demand-Verkehr errichten
	Haltepunkte für Taxen errichten
	Mitfahrbänke
Öffentlicher Verkehr: Linienverkehr	ausreichende Anzahl an ÖPNV-Haltestellen
	Busspuren errichten
	Fahrplanauskunft mit Lautsprecheransage
	Haltestellenkap für Bushaltestellen
	Infotafeln (mit Infos zu nachhaltiger Fortbewegung)
	Infotafeln (mit Infos zu ÖPNV) an Haltestellen
	Netz aus Busspuren
	Schnellbuslinien
	Straßenbahnen auf eigenem Gleiskörper
	Überdachung von ÖPNV-Haltestellen
	Überdachung von Radwegen als Regenschutz
	viele ÖPNV - Haltestellen
Parkplätze und Rastanlagen	Anwohnerparkausweise einführen
	Behindertenparkplätze
	E-Bike Abstellplätze errichten
	Elterntaxi-Haltestellen
	Fahrradabstellanlagen
	Fahrradgaragen
	Gehwegparken nur in Einbahnstraßen
	Kurzparkzonen einführen
	Lastenradparkplätze errichten
	Parkhäuser errichten
	Parkpaletten errichten
	Radabstellplätze
	Radabstellplätze am Gehwegrand
	Radabstellplätze an Bahnhöfen einrichten
	Radabstellplätze errichten
	Radabstellplätze im Einmündungsbereich
	Radabstellplätze überdachen
	Sammelparkplätze errichten

	Stellplätze für Behinderte errichten
	Tiefgaragen errichten
	weniger Gehwegparken
	weniger Parkplätze
	Weniger Parkplätze an Straßen
Radverkehrsnetz	abgesenkte Bordsteine
	Ampelgriffe anbringen
	Anfahrtshilfen für Radverkehr
	Ausschilderung Radwegenetz
	baulich getrennter Rad- und Fußweg
	breite Radwege
	erhöhte Fahrradweg-Überfahrten an Kreuzungen
	Gehwegüberfahrten einrichten
	Fahrrad-Servicestationen
	Fahrradabstellanlagen
	Fahrradgaragen
	Fahrradstraßen errichten
	Gute Oberflächenbeschaffenheit von Radwegen
	Haltestellen mit abgesenkten Bordsteinen
	Kontaktschleifen für Radverkehr
	lange und verbundene Radwege
	Lastenradparkplätze errichten
	niveaugleiche Pflasterung
	Radabstellplätze
	Radabstellplätze am Gehwegrand
	Radabstellplätze an Bahnhöfen einrichten
	Radabstellplätze errichten
	Radabstellplätze im Einmündungsbereich
	Radabstellplätze überdachen
	Radaufstellflächen beim links abbiegen
	Radaufstellflächen vor dem MIV
	Radfahrstreifen baulich trennen
	Radfahrstreifen rechts von Parkstreifen
	Radschnellwege
	Radspielplätze einrichten
	Radwege beleuchten
	Radwegüberfahrten für PKW
	Radwegweiser aufstellen
	steigungsarme Radspuren
	Übergänge baulich erhöhen
	Verkehrsflächen klar trennen

	Vorfahrt für Fahrräder
	Vorrang-Ampeln für Fahrräder
	Zurückgesetzte Haltelinie für MIV
Straßennetz	Radarkontrollen
	Radfahrstreifen rechts von Parkstreifen
	Sackgassen in Quartieren errichten
	Verkehrsflächen klar trennen
Statische Verkehrsregelungen	Beschilderung beleuchten
	Bodenkissen platzieren
	Bodenschwellen platzieren
	Diagonalsperren nutzen
	Durchfahrtsverbote
	Durchfahrtsverbote für Schwerlastverkehr
	Einbahnstraßen (Außer Radverkehr)
	Einzelne Straßen ohne Geschwindigkeitsbegrenzende Maßnahmen belasssen
	Gehwegüberfahrten einrichten
	Fahrbahnverengungen platzieren
	Fahrbahnverengungen platzieren (zur Straßenüberquerung)
	Fußgängerüberwege an ÖPNV-Haltestellen
	Fußgängerzonen einrichten
	Fußwegweiser aufstellen
	Geschwindigkeitsverringerung MIV
	Grüner Pfeil für Radverkehr
	Hinweisschilder zur gegenseitigen Rücksichtnahme
	Parkverbotszonen einrichten
	Piktogramme zur gegenseitigen Rücksichtnahme
	Piktogramme zur Kennzeichnung von Verkehrswegen
	Plateaupflasterungen
	Plateaus platzieren
	Poller auf Gehwegen platzieren
	Poller platzieren
	Poller platzieren (speziell an Kreuzungen)
	Quersperren nutzen
	Radwegüberfahrten für PKW
	Radwegweiser aufstellen
	Schwellen platzieren
	Tempo 30 an Schulen, KiTas und sozialen Einrichtungen
	Tempo-10-Zonen errichten
	Tempo-20-Zonen errichten
	Tempo-30-Zonen einrichten

	temporäre Spielstraßen errichten
	verkehrsberuhigte Bereiche errichten
	Zurückgesetzte Haltelinie für MIV
Tank- und Ladestationen	E-Auto Ladesäulen errichten
	E-Bike Ladestationen errichten

Anhang 3 Zuordnungstabelle Maßnahme – OSM-Tag

implizierte Maßnahme/ Zusammenfassung	Objekt	Attribut
(digitale) Info-Tafeln	nwr["information"="board"] //Info-Tafeln	["board_type"="map"] //Karteninformationen ["board_type"="public_transport"] //ÖPNV-Informationen ["board_type"="architecture"] //Architektur-Informationen
abgesenkte Bordsteine	nwr["kerb"="lowered"] //abgesenkter Bordstein	
abgesenkte Bordsteine	nwr["kerb"="flush"] //ebenerdiger Bordstein	
Ampelgriffe anbringen	nwr["highway"="traffic_signals"] // Ampeln	["traffic_signals:grip"="yes"] // Ampelgriff angebracht
Ampelsysteme mit akustischen Signalen	nwr["highway"="traffic_signals"] // Ampeln	["traffic_signals:sound"="yes"] // Akustische Signale vorhanden ["traffic_signals:vibration"="yes"] // Vibrationssignal zusätzlich verfügbar
Ampelsysteme mit an ÖPNV angepasster Schaltung	nwr["highway"="traffic_signals"] // Ampeln	["traffic_signals:foot"="yes"] // Fußgängerampel ["traffic_signals:bicycle"="yes"] // Fahrradampel
Anfahrtshilfen für Radverkehr	nwr["highway"="traffic_signals"] // Ampeln	["traffic_signals:bicycle"="yes"] // Fahrradampel
Anwohnerparkausweise einführen	nwr["amenity"="parking"] // Parkplätze	["access"="private"] // Parkfläche nur für berechtigte Nutzer ["residents"="yes"] // Parkfläche speziell für Anwohner ["residents"="designated"] // Parkfläche speziell für Anwohner ["parking:condition"="residents"] // Parkregelung für Anwohner ["traffic_sign"="DE:1020-32"] // Kombiniertes Schild: „Nur mit Parkausweis Nr. ...“ (Anwohnerparkausweis)
Anwohnerparkausweise einführen	nwr["traffic_sign"="DE:314"] // Parkplatzschild in Deutschland	["access"="private"] // Parkfläche nur für berechtigte Nutzer ["residents"="yes"] // Parkfläche speziell für Anwohner ["residents"="designated"] // Parkfläche speziell für Anwohner ["parking:condition"="residents"] // Parkregelung für Anwohner ["traffic_sign"="DE:1020-32"] // Kombiniertes Schild: „Nur mit Parkausweis Nr. ...“ (Anwohnerparkausweis)
Anwohnerparkausweise einführen	nwr["traffic_sign"="DE:315"] // Eingeschränktes Halteverbot	["access"="private"] // Parkfläche nur für berechtigte Nutzer ["residents"="yes"] // Parkfläche speziell für Anwohner ["residents"="designated"] // Parkfläche speziell für Anwohner ["parking:condition"="residents"] // Parkregelung für Anwohner ["traffic_sign"="DE:1020-32"] // Kombiniertes Schild: „Nur mit Parkausweis Nr. ...“ (Anwohnerparkausweis)
Fahrplan-Anzeigetafeln mit barrierefreien Ansagen	nwr["public_transport"="stop_display"] // Anzeigetafel an Haltestellen	["accessibility"="yes"] // Allgemein barrierefrei ["speech_output"="yes"] // Akustische Ansagen vorhanden ["speech_output:purpose"="blind"] // Ansagen speziell für Sehbehinderte
Fußgängerüberwege errichten	nwr["highway"="crossing"] // Fußgängerüberwege	["crossing"="marked"] // Markierter Überweg ["crossing_ref"="zebra"] // Zebrastreifen (spezifische Referenz)
ausreichend Fußgängerüberwege	nwr["traffic_sign"="DE:350"] // Schild "Fußgängerüberweg" nach deutschem Standard	

implizierte Maßnahme/ Zusammenfassung	Objekt	Attribut
Fußverkehrsampeln errichten	nwr["highway"="traffic_signals"] //Ampeln	["traffic_signals:foot"="yes"] // Fußgängerampeln
Fußverkehrsampeln errichten	nwr["crossing"="traffic_signals"] // Fußgängerüberweg mit Ampelregelung	
ausreichend Gehwegbeleuchtung aufstellen	nwr["highway"="street_lamp"] // Beleuchtungspunkte (Laternen, Lampen)	["lit"="yes"] // Beleuchtung vorhanden
ausreichend Gehwegbeleuchtung aufstellen	nwr[highway]	["lamp:type"="street_lamp"] // Standard-Straßenlaterne ["lamp:type"="path"] // Beleuchtung entlang von Wegen (z. B. Gehwegen, Radwegen)
ausreichend Gehwegbeleuchtung aufstellen	nwr["type"="lighting"]	
ausreichende Anzahl an ÖPNV-Haltestellen	nwr["public_transport"="platform"] //Haltestellen	["bus"="yes"] // Haltestelle für Busse ["tram"="yes"] // Haltestelle für Straßenbahnen ["train"="yes"] // Haltestelle für Züge
Ausschilderung Radwegenetz	node["information"="route_marker"]	["bicycle"="yes"] //Wegweiser für Fahrradrouten
Bänke platzieren	nwr["amenity"="bench"] // Bank	
baulich getrennter Rad- und Fußweg	nwr["highway"="cycleway"] // Radweg	["segregated"="yes"] // baulich getrennt ["cycleway:separation"="kerb"] // Trennung durch Bordstein ["cycleway:separation"="barrier"] // Trennung durch physische Barriere (z. B. Hecke oder Zaun)
baulich getrennter Rad- und Fußweg	nwr["bicycle"="designated"]	["segregated"="yes"] // baulich getrennt ["cycleway:separation"="kerb"] // Trennung durch Bordstein ["cycleway:separation"="barrier"] // Trennung durch physische Barriere (z. B. Hecke oder Zaun)
baulich getrennter Rad- und Fußweg	nwr["bicycle"="yes"]	["segregated"="yes"] // baulich getrennt ["cycleway:separation"="kerb"] // Trennung durch Bordstein ["cycleway:separation"="barrier"] // Trennung durch physische Barriere (z. B. Hecke oder Zaun)
baulich getrennter Rad- und Fußweg	nwr["foot"="designated"]	["segregated"="yes"] // baulich getrennt
baulich getrennter Rad- und Fußweg	nwr["highway"="footway"] // Gehweg	["segregated"="yes"] // baulich getrennt ["cycleway:separation"="kerb"] // Trennung durch Bordstein ["cycleway:separation"="barrier"] // Trennung durch physische Barriere (z. B. Hecke oder Zaun)
Bäume pflanzen	node["natural"="tree"] //einzelne Bäume	
Bäume pflanzen	way["natural"="tree_row"] //Baumreihen	
Bedarfsampeln für MIV	nwr["highway"="traffic_signals"] // Ampeln	
Begrünung	way["landuse"="grass"] // Rasenflächen	
Begrünung	nwr["leisure"="park"] // Parkfläche	
Begrünung	way["natural"="tree_row"] // Baumreihen	
Begrünung	way["natural"="scrub"] // Buschwerk oder Sträucher	
Begrünung	way["landuse"="forest"] // Forstflächen oder Wälder	
Begrünung	way["landuse"="meadow"] // Wiesenflächen	

implizierte Maßnahme/ Zusammenfassung	Objekt	Attribut
Begrünung	node["amenity"="planter"] // Pflanzkübel	
Begrünung	node["natural"="tree"] // Einzelne Bäume	
verkehrsberuhigte Bereiche errichten	way["highway"="living_street"]; relation["highway"="living_street"];	
Behindertenparkplätze	nwr["amenity"="parking_space"] // Einzelne Parkplätze	["parking_space"="disabled"] //Behindertenparkplatz
Behindertenparkplätze	nwr["amenity"="parking"] // Parkflächen	["parking_space"="disabled"] //Behindertenparkplatz
Behindertenparkplätze	nwr["amenity"="parking"] // Komplexe Parkbereiche	["parking_space"="disabled"] //Behindertenparkplatz
Behindertenparkplätze	node["traffic_sign"="DE:314;DE:1044-10"] //Verkehrsschild für Behindertenparkplätze	
Beschilderung beleuchten	node["traffic_sign"~"DE:350|DE:239|DE:240|DE:241"] //Schilder für Fuß- und Radüberwege	["lit"="yes"] //beleuchtet
viele ÖPNV - Haltestellen	node["highway"="bus_stop"]; node["railway"="tram_stop"]; way["public_transport"="platform"];	
Bevorrechtigung Taxen & On-Demand-Verkehr	way["taxi:lanes"]	
Bodenkissen platzieren	node["traffic_calming"="cushion"]	
Bodenschwellen platzieren	nwr["traffic_calming"="bump"]; // Bodenschwelle	
breite Gehwege	way["highway"="footway"] //Fußwege	["width">"2"] //Breite über 2 Meter ["width"<"2"] //Breite unter 2 Meter ["width">"2.5"] //Breite über 2.5 Meter ["width"<"2.5"] //Breite über 2.5 Meter
breite Gehwege	way["highway"="path"]	["foot"="designated"] //Fußwege ["width">"2"] //Breite über 2 Meter ["width"<"2"] //Breite unter 2 Meter ["width">"2.5"] //Breite über 2.5 Meter ["width"<"2.5"] //Breite über 2.5 Meter
breite Radwege	way["highway"="cycleway"] //reiner Fahrradweg	
breite Radwege	way["highway"]	["cycleway"~"lane|track"] //Fahrradspur auf Fahrbahn oder baulich getrennt
Busspuren errichten	way["bus:lanes"~"designated"] // alle wege mit mindestens einer reservierten Busspur	
Carsharing-Parkplätze einrichten	nwr["amenity"="parking_space"]	["parking_space"="car_sharing"] ["parking"="car_sharing"]
Carsharing-Parkplätze einrichten	nwr["amenity"="car_sharing"]	
Diagonalsperren nutzen	nwr["barrier"="block"]	
Diagonalsperren nutzen	nwr["barrier"="bollard"]	
Durchfahrtsverbote	way["access"="no"] //Durchfahrtsverbot	
Durchfahrtsverbote	way["motor_vehicle"="no"] //kein MIV	
Durchfahrtsverbote	way["access:conditional"] //Zufahrtsbeschränkung	
Durchfahrtsverbote	way["access"="destination"] //Anlieger frei	
Durchfahrtsverbote für Schwerlastverkehr	way["hgv"="no"] //kein Schwerlastverkehr	

implizierte Maßnahme/ Zusammenfassung	Objekt	Attribut
Durchfahrtsverbote für Schwerlastverkehr	way["hgv"="destination"] //nur Anlieger	
Durchfahrtsverbote für Schwerlastverkehr	way["hgv:conditional"] //bedingte Beschränkung	
dynamische Schilder mit Ampelinformation für Radverkehr	nwr["highway"="traffic_signals"] //Ampel	["traffic_signals:bicycle"="yes"] //Fahrradampel ["traffic_signals:countdown"="yes"] //countdown
E-Auto Ladesäulen errichten	nwr["amenity"="charging_station"]	
E-Bike Abstellplätze errichten	nwr["amenity"="bicycle_parking"]["ebike"="yes"] //Abstellplatz für E-Fahrrad	["charging"="yes"] //mit Ladefunktion
E-Bike Ladestationen errichten	nwr["amenity"="charging_station"]["bicycle"="designated"] //Ladestationen für E-Bikes	
E-Carsharing-Plätze	nwr["amenity"="car_sharing"] //car-sharing Plätze	["capacity:electric">0] //mehr als 1 E-Auto Platz
E-Lastenrad-Sharing-Plätze	nwr["amenity"="bicycle_rental"] //Fahrradsharing	["capacity:electric" >0] //mehr als 1 E-Bike Platz ["electric_bicycle"] //E-Räder ["capacity:cargo" >0] //mehr als 1 Lastenrad-Platz ["cargo_bike"] //Lastenräder
Ebenerdige Ampelquerungen für Fußgänger	nwr["highway"="crossing"] //Fußgängerüberquerungen	["traffic_signals:foot"="yes"] //Fußgängerampeln ["kerb"="lowered"] //abgesenkter Bordstein ["kerb"="flush"] //ebenerdige Querung
Einbahnstraßen (Außer Radverkehr)	way["oneway"="yes"] //Einbahnstraße	["oneway:bicycle"="no"] //Radfahrer frei
Schwellen platzieren	node["traffic_calming"="bumps"]; way["traffic_calming"="bumps"]; node["traffic_calming"="hump"]; way["traffic_calming"="hump"]; node["traffic_calming"="table"]; way["traffic_calming"="table"];	
elektronisch versenkbare Poller	nwr["barrier"="bollard"]	["bollard"="automatic"]
Tempo-10-Zonen errichten	way["maxspeed"="10"]; relation["maxspeed"="10"];	
erhöhte Fahrradweg-Überfahrten an Kreuzungen	["highway"="cycleway"]	["crossing"="raised"]
Gehwegüberfahrten einrichten	["highway"="crossing"]	["crossing"="raised"]
Fahrbahnverengungen platzieren	nwr["traffic_calming"="choker"] //Fahrbahnverengung durch bauliche Maßnahmen	
Fahrbahnverengungen platzieren	nwr["traffic_calming"="narrow"] //allgemeine Verengung	
Fahrbahnverengungen platzieren (zur Straßenüberquerung)	nwr["traffic_calming"="island"] //Mittelinsel	
Fahrbahnverengungen platzieren (zur Straßenüberquerung)	["crossing:kerb_extension"] //Gehwegvorziehungen	
Fahrplanauskunft mit Lautsprecheransage	nwr["information"="board"]	["board_type"="public_transport"] //ÖPNV-Informationen ["announcement"="yes"] //Ansage

implizierte Maßnahme/ Zusammenfassung	Objekt	Attribut
		["announcement:automatic"="yes"] //automatische Ansage
Fahrrad-Servicestationen	nwr["service:bicycle:pump"="yes"]	
Fahrrad-Servicestationen	nwr["service:bicycle:repair"="yes"]	
Fahrradabstellanlagen	nwr["amenity"="bicycle_parking"]	
Fahrradgaragen	nwr["amenity"="bicycle_parking"]	["bicycle_parking"~"lockers|building"] //abschließbar oder Parkhaus
Fahrradstraßen errichten	nwr["bicycle_road"="yes"] //Fahrradstraße	
Tempo-20-Zonen errichten	way["maxspeed"="20"]; relation["maxspeed"="20"];	
Fußgängerüberwege an ÖPNV-Haltestellen	node["highway"="crossing"]	(around:10)["public_transport"="platform"] //ÖPNV-Haltestellen im Umkreis von 10 Metern
Fußgängerzonen einrichten	nwr["highway"="pedestrian"]	
Fußverkehrsampeln ohne Gültigkeit für Radverkehr	node["crossing"="traffic_signals"] //Kreuzung mit Ampel	["traffic_signals:foot"="yes"] ["traffic_signals:bicycle"="no"] //keine Fahrräder
Fußverkehrsampeln vor/hinter ÖPNV-Haltestellen	node["highway"="traffic_signals"]	["traffic_signals:foot"="yes"] (around:10)["public_transport"="platform"]
Fußwegweiser aufstellen	node["information"="guidepost"]	["foot"="yes"]
Gehwegparken nur in Einbahnstraßen	nwr["oneway"="yes"]	[~"^parking:lane:(left|right|both)$"~"sidewalk"] //gehwegparken
Rampen	way["ramp"="yes"]; way["highway"="footway"]["ramp"="yes"]; way["highway"="service"]["service"="driveway"];	
geteilte Nutzung von Fahrstreifen durch Fahrräder und MIV	way["cycleway"="shared_lane"]; way["cycleway"="lane"]; way["cycleway:right"="shared_lane"]; way["cycleway:left"="shared_lane"]; way["cycleway:both"="shared_lane"];	
weniger Gehwegparken	way["parking:lane"="sidewalk"]; way["parking:condition:right"="sidewalk"]; way["parking:condition:left"="sidewalk"];	
weniger Parkplätze	node["amenity"="parking"]; way["amenity"="parking"]; relation["amenity"="parking"];	
Grüner Pfeil für Radverkehr	node["traffic_sign"="DE:720"] way["traffic_sign"="DE:720"];	
Gute Oberflächenbeschaffenheit der Fußwege sicherstellen	way["highway"="footway"]["smoothness"="excellent"]["smoothness"="good"]; way["highway"="pedestrian"]["surface"~"^(paved|asphalt|concrete|concrete:lanes|concrete:plates|sett|compacted)$"]; way["highway"="pedestrian"]["smoothness"~"^(excellent\|good)$"]; //Fußgängerzonen	
Gute Oberflächenbeschaffenheit von Radwegen	way["highway"="cycleway"]["surface"~"^(paved|asphalt|concrete|concrete:lanes|concrete:plates|sett|compacted)$"]; way["highway"="cycleway"]["smoothness"~"^(excellent\|good)$"]; //Radweg	

implizierte Maßnahme/ Zusammenfassung	Objekt	Attribut
	way["cycleway"="lane"]["surface"~"^(paved|asphalt|concrete|concrete:lanes|concrete:plates|sett|compacted)$"]; way["cycleway"="lane"]["smoothness"~"^(excellent\|good)$"]; //Fahrradspuren auf Straßen way["cycleway"="shared_lane"]["surface"~"^(paved|asphalt|concrete|concrete:lanes|concrete:plates|sett|compacted)$"]; way["cycleway"="shared_lane"]["smoothness"~"^(excellent\|good)$"];	
Haltepunkte für Lieferverkehr errichten	node["amenity"="loading_dock"]; way["amenity"="loading_dock"]; node["amenity"="parking"]["access"="delivery"]; way["amenity"="parking"]["access"="delivery"]; way["highway"="service"]["service"="delivery"]; node["loading_zone"="yes"]; way["parking:condition:right"="delivery"]; way["parking:condition:left"="delivery"];	
Spiegel anbringen	node["man_made"="mirror"]; // Verkehrsspiegel	
Haltepunkte für Taxen errichten	node["amenity"="taxi"]; way["amenity"="taxi"]; node["taxi"="designated"]; way["taxi"="designated"]; way["highway"="service"]["service"="taxi"];	
Haltestellen mit abgesenkten Bordsteinen	node["highway"="bus_stop"]["kerb"="lowered"]; way["public_transport"="platform"]["kerb"="lowered"]; node["railway"="tram_stop"]["kerb"="lowered"]; node["highway"="bus_stop"]["wheelchair"="yes"]; way["public_transport"="platform"]["wheelchair"="yes"]; node["railway"="tram_stop"]["wheelchair"="yes"];	
Haltestellen mit Leitlinien	node["highway"="bus_stop"]["tactile_paving"="yes"]; way["public_transport"="platform"]["tactile_paving"="yes"]; node["railway"="tram_stop"]["tactile_paving"="yes"];	
Leitstreifen	node["tactile_paving"="yes"]	
Leitstreifen an abgesenkten Bordsteinen	node["tactile_paving"="yes"]["kerb"="lowered"];	
Lieferzonen einrichten	node["amenity"="loading_dock"]	
Mülleimer platzieren	node["amenity"="waste_basket"]	
Nachtfahrtsverbote für Schwerlastverkehr	way["hgv"="no"]["conditional"~"@.*(22:00-06:00)"]	
niveaugleiche Pflasterung	way["kerb"~"^(lowered|flush)$"]	
öffentliche Toiletten errichten	node["amenity"="toilets"]; way["amenity"="toilets"]; relation["amenity"="toilets"];	
öffentliche Trinkwasserstationen errichten	node["amenity"="drinking_water"]; way["amenity"="drinking_water"]; relation["amenity"="drinking_water"];	
Parkhäuser errichten	node["building"="parking"]; way["building"="parking"]; relation["building"="parking"];	
Parks errichten	node["leisure"="park"]; way["leisure"="park"]; relation["leisure"="park"];	

implizierte Maßnahme/ Zusammenfassung	Objekt	Attribut
Parkstreifen einführen	way["parking:lane:both"="painted_area_only"]; way["parking:lane:left"="painted_area_only"]; way["parking:lane:right"="painted_area_only"]; node["amenity"="parking"]["parking"="street_side"]; way["amenity"="parking"]["parking"="street_side"]; // ehr Parkbuchten	
Parkverbotszonen einrichten	way["parking:condition:right"="no_parking"]; way["parking:condition:left"="no_parking"]; way["parking:lane:both"="no"];	
Pflanzkübel platzieren	node["barrier"="planter"]; way["barrier"="planter"]; relation["barrier"="planter"];	
Plateaupflasterungen	way["traffic_calming"="table"]["surface"~"^(paving_stones|sett|concrete:plates|asphalt)$"]; way["traffic_calming"="raised_crossing"]["surface"~"^(paving_stones\|sett\|concrete:plates)$"]; way["traffic_calming"="plateau"]["surface"~"^(paving_stones|sett|concrete:plates)$"]	
Tempo-30-Zonen einrichten	way["maxspeed"="30"]; relation["maxspeed"="30"];	
Poller auf Gehwegen platzieren	nwr["barrier"="bollard"]	
Poller platzieren	nwr["barrier"="bollard"]	
Poller platzieren (speziell an Kreuzungen)	nwr["barrier"="bollard"]	
Quersperren nutzen	nwr["barrier"="block"]; nwr["barrier"="bollard"]	

Anhang 4 Overpass-Query

```
[out:xml][timeout:90];
(
  way["building"]({{bbox}});
  nwr["information"="board"]  ["board_type"="map"]({{bbox}});
  nwr["information"="board"]
["board_type"="public_transport"]({{bbox}});
  nwr["information"="board"]
["board_type"="architecture"]({{bbox}});
  nwr["kerb"="lowered"]({{bbox}});
  nwr["kerb"="flush"]({{bbox}});
  nwr["highway"="traffic_signals"]
["traffic_signals:grip"="yes"]({{bbox}});
  nwr["highway"="traffic_signals"]
["traffic_signals:sound"="yes"]({{bbox}});
  nwr["highway"="traffic_signals"]
["traffic_signals:vibration"="yes"]({{bbox}});
  nwr["highway"="traffic_signals"]
["traffic_signals:foot"="yes"]({{bbox}});
  nwr["highway"="traffic_signals"]
["traffic_signals:bicycle"="yes"]({{bbox}});
  nwr["amenity"="parking"]  ["access"="private"]({{bbox}});
  nwr["amenity"="parking"]  ["residents"="yes"]({{bbox}});
  nwr["amenity"="parking"]  ["residents"="designated"]({{bbox}});
  nwr["amenity"="parking"]
["parking:condition"="residents"]({{bbox}});
  nwr["amenity"="parking"]  ["traffic_sign"="DE:1020-32"]({{bbox}});
  nwr["traffic_sign"="DE:314"]  ["access"="private"]({{bbox}});
  nwr["traffic_sign"="DE:314"]  ["residents"="yes"]({{bbox}});
  nwr["traffic_sign"="DE:314"]
["residents"="designated"]({{bbox}});
  nwr["traffic_sign"="DE:314"]
["parking:condition"="residents"]({{bbox}});
  nwr["traffic_sign"="DE:314"]              ["traffic_sign"="DE:1020-
32"]({{bbox}});
  nwr["traffic_sign"="DE:315"]  ["access"="private"]({{bbox}});
  nwr["traffic_sign"="DE:315"]  ["residents"="yes"]({{bbox}});
  nwr["traffic_sign"="DE:315"]
["residents"="designated"]({{bbox}});
  nwr["traffic_sign"="DE:315"]
["parking:condition"="residents"]({{bbox}});
  nwr["traffic_sign"="DE:315"]              ["traffic_sign"="DE:1020-
32"]({{bbox}});
  nwr["public_transport"="stop_display"]
["accessibility"="yes"]({{bbox}});
  nwr["public_transport"="stop_display"]
["speech_output"="yes"]({{bbox}});
  nwr["public_transport"="stop_display"]
["speech_output:purpose"="blind"]({{bbox}});
  nwr["highway"="crossing"]  ["crossing"="marked"]({{bbox}});
  nwr["highway"="crossing"]  ["crossing_ref"="zebra"]({{bbox}});
  nwr["traffic_sign"="DE:350"]({{bbox}});
  nwr["highway"="traffic_signals"]
["traffic_signals:foot"="yes"]({{bbox}});
  nwr["crossing"="traffic_signals"]({{bbox}});
  nwr["highway"="street_lamp"]  ["lit"="yes"]({{bbox}});
  nwr["highway"]["lamp:type"="street_lamp"]({{bbox}});
  nwr["highway"]["lamp:type"="path"]({{bbox}});
  nwr["type"="lighting"]({{bbox}});
  nwr["public_transport"="platform"]  ["bus"="yes"]({{bbox}});
  nwr["public_transport"="platform"]  ["tram"="yes"]({{bbox}});
```

```
  nwr["public_transport"="platform"]  ["train"="yes"]({{bbox}});
  node["information"="route_marker"]  ["bicycle"="yes"]({{bbox}});
  nwr["amenity"="bench"]({{bbox}});
  nwr["highway"="cycleway"]  ["segregated"="yes"]({{bbox}});
  nwr["highway"="cycleway"]
["cycleway:separation"="kerb"]({{bbox}});
  nwr["highway"="cycleway"]
["cycleway:separation"="barrier"]({{bbox}});
  nwr["bicycle"="designated"]  ["segregated"="yes"]({{bbox}});
  nwr["bicycle"="designated"]
["cycleway:separation"="kerb"]({{bbox}});
  nwr["bicycle"="designated"]
["cycleway:separation"="barrier"]({{bbox}});
  nwr["bicycle"="yes"]  ["segregated"="yes"]({{bbox}});
  nwr["bicycle"="yes"]  ["cycleway:separation"="kerb"]({{bbox}});
  nwr["bicycle"="yes"]  ["cycleway:separation"="barrier"]({{bbox}});
  nwr["foot"="designated"]  ["segregated"="yes"]({{bbox}});
  nwr["highway"="footway"]  ["segregated"="yes"]({{bbox}});
  nwr["highway"="footway"]
["cycleway:separation"="kerb"]({{bbox}});
  nwr["highway"="footway"]
["cycleway:separation"="barrier"]({{bbox}});
  node["natural"="tree"]({{bbox}});
  way["natural"="tree_row"]({{bbox}});
  nwr["highway"="traffic_signals"]({{bbox}});
  way["landuse"="grass"]({{bbox}});
  nwr["leisure"="park"]({{bbox}});
  way["natural"="tree_row"]({{bbox}});
  way["natural"="scrub"]({{bbox}});
  way["landuse"="forest"]({{bbox}});
  way["landuse"="meadow"]({{bbox}});
  node["amenity"="planter"]({{bbox}});
  node["natural"="tree"]({{bbox}});
  nwr["amenity"="parking_space"]
["parking_space"="disabled"]({{bbox}});
  nwr["amenity"="parking"]  ["parking_space"="disabled"]({{bbox}});
  nwr["amenity"="parking"]  ["parking_space"="disabled"]({{bbox}});
  node["traffic_sign"="DE:314;DE:1044-10"]({{bbox}});
  node["traffic_sign"~"DE:350|DE:239|DE:240|DE:241"]
["lit"="yes"]({{bbox}});
  way["taxi:lanes"]({{bbox}});
  node["traffic_calming"="cushion"]({{bbox}});
  nwr["traffic_calming"="bump"]({{bbox}});
  way["highway"="footway"]  ["width"="2"]({{bbox}});
  way["highway"="footway"]  ["width"="2.5"]({{bbox}});
  way["highway"="path"]  ["foot"="designated"]({{bbox}});
  way["highway"="cycleway"]({{bbox}});
  way["highway"]  ["cycleway"~"lane|track"]({{bbox}});
  way["bus:lanes"~"designated"]({{bbox}});
  nwr["amenity"="parking_space"]
["parking_space"="car_sharing"]({{bbox}});
  nwr["amenity"="parking_space"]
["parking"="car_sharing"]({{bbox}});
  nwr["amenity"="car_sharing"]({{bbox}});
  nwr["barrier"="block"]({{bbox}});
  nwr["barrier"="bollard"]({{bbox}});
  way["access"="no"]({{bbox}});
  way["motor_vehicle"="no"]({{bbox}});
  way["access:conditional"]({{bbox}});
  way["access"="destination"]({{bbox}});
  way["hgv"="no"]({{bbox}});
  way["hgv"="destination"]({{bbox}});
```

```
  way["hgv:conditional"]({{bbox}});
  nwr["highway"="traffic_signals"]
["traffic_signals:bicycle"="yes"]({{bbox}});
  nwr["highway"="traffic_signals"]
["traffic_signals:countdown"="yes"]({{bbox}});
  nwr["amenity"="charging_station"]({{bbox}});
  nwr["amenity"="bicycle_parking"]["ebike"="yes"]
["charging"="yes"]({{bbox}});

nwr["amenity"="charging_station"]["bicycle"="designated"]({{bbox}});
  nwr["amenity"="car_sharing"]  ["capacity:electric"="0"]({{bbox}});
  nwr["amenity"="bicycle_rental"]
["capacity:electric"="0"]({{bbox}});
  nwr["amenity"="bicycle_rental"]  ["electric_bicycle"]({{bbox}});
  nwr["amenity"="bicycle_rental"]  ["capacity:cargo"="0"]({{bbox}});
  nwr["amenity"="bicycle_rental"]  ["cargo_bike"]({{bbox}});
  nwr["highway"="crossing"]
["traffic_signals:foot"="yes"]({{bbox}});
  nwr["highway"="crossing"]  ["kerb"="lowered"]({{bbox}});
  nwr["highway"="crossing"]  ["kerb"="flush"]({{bbox}});
  way["oneway"="yes"]  ["oneway:bicycle"="no"]({{bbox}});
  nwr["barrier"="bollard"]  ["bollard"="automatic"]({{bbox}});
  nwr["highway"="cycleway"]  ["crossing"="raised"]({{bbox}});
  nwr["highway"="crossing"]  ["crossing"="raised"]({{bbox}});
  nwr["traffic_calming"="choker"]({{bbox}});
  nwr["traffic_calming"="narrow"]({{bbox}});
  nwr["traffic_calming"="island"]({{bbox}});
  nwr["information"="board"]
["board_type"="public_transport"]({{bbox}});
  nwr["information"="board"] ["announcement"="yes"]({{bbox}});
  nwr["information"="board"]
["board_type"="public_transport"]({{bbox}});
  nwr["information"="board"]
["announcement:automatic"="yes"]({{bbox}});
  nwr["service:bicycle:pump"="yes"]({{bbox}});
  nwr["service:bicycle:repair"="yes"]({{bbox}});
  nwr["amenity"="bicycle_parking"]({{bbox}});
  nwr["amenity"="bicycle_parking"]
["bicycle_parking"~"lockers|building"]({{bbox}});
  nwr["bicycle_road"="yes"]({{bbox}});
  node["highway"="crossing"]
(around:10)["public_transport"="platform"]({{bbox}});
  nwr["highway"="pedestrian"]({{bbox}});
  node["crossing"="traffic_signals"]
["traffic_signals:foot"="yes"]({{bbox}});
  node["crossing"="traffic_signals"]
["traffic_signals:bicycle"="no"]({{bbox}});
  node["highway"="traffic_signals"]
["traffic_signals:foot"="yes"]({{bbox}});

node["highway"="traffic_signals"](around:10)["public_transport"="pla
tform"]({{bbox}});
  node["information"="guidepost"]  ["foot"="yes"]({{bbox}});
  nwr["oneway"="yes"]
[~"^parking:lane:(left|right|both)$"~"sidewalk"]({{bbox}});
  nwr["traffic_calming"="island"]({{bbox}});
);
// Ausgabe der Ergebnisse
out tags meta geom;
```

Anhang 5 Anwenderdokumentation

Anwenderdokumentation zur Software UrbanLink3D

Version

Version 1.0 – 03.01.2025 – Autorin: Franziska Schultz

Use-Case

Befolgen Sie bei der ersten Benutzung von UrbanLink3D die **Arbeitsschritte zur Datenverknüpfung**.

Dieses Dokument beschreibt das Softwarepaket UrbanLink3D zur Verknüpfung eines 3D-Stadtmodells im Format CityGML mit mobilitätsrelevanten Daten. Diese stammen aus OpenStreetMap und der General Transit Feed Specification (GTFS).

Der Workflow ist weitestgehend mit Hilfe von Python automatisiert, ist jedoch an einigen Stellen anpassbar gestaltet. Der Workflow und die zugehörigen Softwareskripte können für verschiedene Anwendungszwecke genutzt, bearbeitet und erweitert werden.

Ziel des Dokuments

Dieses Dokument hat folgende Ziele:

- Den Prozess zur Kombination von CityGML-, GTFS- und OpenStreetMap-Daten übersichtlich darstellen
- Verfügbare Datenquellen aufzeigen
- Eine Schritt-für-Schritt-Anleitung bereitstellen
- Anpassungs- und Konfigurationsmöglichkeiten des Workflows und der Python-Skripte erläutern

Inhalte des Dokuments

Dieses Dokument beschreibt den Prozess zum Verknüpfen der Daten inklusive der Nutzung der relevanten Software und mitgelieferter Python-Skripte.

Für den Fall, dass die Python-Skripte für einen Anwendungsfall zu spezifisch sind oder andere Datenanalysen oder Verknüpfungen durchgeführt werden sollen, werden außerdem die manuell durchführbaren Schritte bis zum Datenimport erklärt.

Die Funktion der einzelnen Python-Skripte wird mit Hilfe von Schaubildern erklärt.

Inhaltsverzeichnis

Symbole und Bezeichnungen

„Schritt 1:“ – Beschreibt einen Prozessschritt

1. Beschreibt einen Zwischenschritt erster Ordnung
    - Beschreibt einen Zwischenschritt zweiter Ordnung
        - Beschreibt einen Zwischenschritt dritter Ordnung
            - Beschreibt einen Zwischenschritt vierter Ordnung

– Zeigt Verlinkungen innerhalb des Dokuments an

➔ – Bestätigung des Erfolgs für Zwischenschritte

✓ – Bestätigung des Erfolgs für den gesamten Prozessschritt

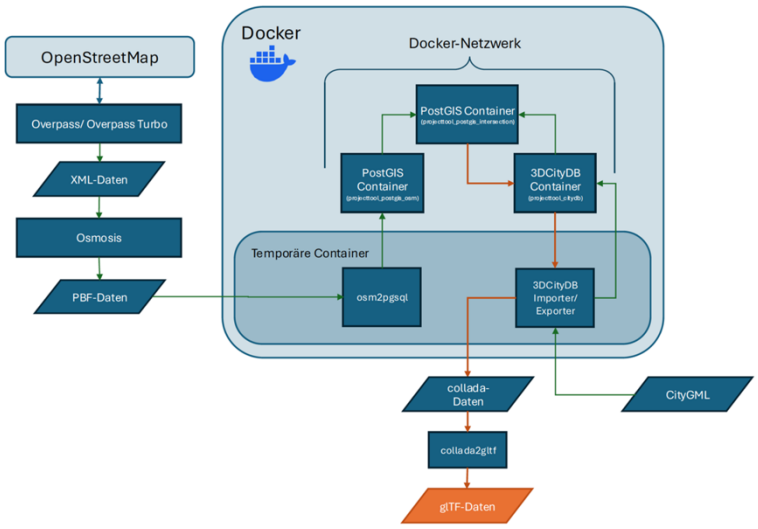
3

Prozessübersicht

UrbanLink3D hat den Prozess der Datenverknüpfung von 3D-Stadtmodellen, OpenStreetMap-Daten und GTFS-Daten mit Hilfe von Python-Skripten automatisiert.

Der grundlegende Prozess ist in Abbildung 1 dargestellt. Der Prozess nutzt Docker-Container zur Speicherung und Verarbeitung der Daten. Es ruft mithilfe der Overpass-API Daten aus OpenStreetMap ab und importiert diese über das Tool osm2pgsql in eine PostGIS-Datenbank. Eine von Ihnen zur Verfügung gestellte CityGML-Datei wird mithilfe des 3DCityDB Importers in eine 3DCityDB-Datenbank importiert. Beide Datenbanken liegen in einzelnen Docker-Containern im gleichen Netzwerk. Eine dritte Datenbank im Netzwerk wird genutzt, um die Daten miteinander zu verknüpfen. Die verknüpften Daten werden in die 3DCityDB-Datenbank geschrieben und können von dort mit dem 3DCityDB Exporter unter Nutzung des Tools collada2gltf als glTF-Dateien exportiert werden.

Abbildung 1: Übersicht über den Gesamtprozess



Technische Voraussetzungen

Betriebssystem

Dieser Prozess ist für das Betriebssystem **macOS** entwickelt. Die Durchführung auf Rechnern mit anderen Betriebssystemen wurde nicht getestet und kann fehlerbehaftet sein.

Benötigte Tools

Die Anwendung verknüpft Daten automatisiert mit Python. Dafür muss Python auf Ihrem Gerät installiert sein.

Beim Start der Anwendung prüft das System automatisch, ob die benötigte Software und Python-Pakete vorhanden sind. Fehlende Software wird angezeigt und muss manuell installiert werden, fehlende Python-Pakete werden automatisch hinzugefügt.

Erforderliche Software (manuelle Installation nötig):

- Docker-Installation
- Java Runtime Environment
- Osmosis (via GitHub: https://github.com/openstreetmap/osmosis)

Erforderliche Python-Pakete (automatische Installation):

- psycopg2
- shapely
- pyproj
- requests
- lxml
- pandas
- docker

Beschreibung der Paketinhalte

Der Paketordner enthält folgende Daten (siehe Abbildung 2):

- Python-Skripte, die für die Verarbeitung benötigt werden
- Ein Ordner für die von den Python-Skripten exportierten Dateien (Name: **exports**)

- Ein Ordner für die von den Python-Skripten ausgegebenen Protokolldateien (Name: **logs**)
- Eine Konfigurationsdatei zur Festlegung der relevanten OpenStreetMap-Tags (Dateiname: **overpass_request_config.txt**)
- Eine Datei zur Speicherung der Informationen zu den von den Python-Skripten erstellten Docker-Containern (Dateiname: **deployment_info.csv**)

Abbildung 2: Paketinhalte

Tabelle 1 beschreibt die generellen Funktionen der Skripte. Eine ausführliche Beschreibung aller Skripte inklusive Anpassungsmöglichkeiten finden Sie im Abschnitt ↗ Anpassungsmöglichkeiten der Skripte

Tabelle 1: Funktionen der Skripte

Skript	Funktion
check_installation.py	Prüft die installierte Software und installiert automatisch fehlende Python-Pakete.
docker_creation.py	Erstellt die Docker-Umgebung inklusive Docker-Containern und Docker-Netzwerk. Speichert die Informationen zur Docker-Umgebung in logs/deployment_info.csv. Importiert eine CityGML-Datei in die 3DCityDB.
intersect_data.py	Kombiniert CityGML-, OpenStreetMap- und GTFS-Daten.
osm_import.py	Importiert OpenStreetMap-Daten mithilfe von osm2pgsql in eine Datenbank.
overpass_api_call.py	Ermittelt Koordinaten aus einer GeoNames-Datei oder GeoJSON-Datei und ruft OpenStreetMap-Daten über die Overpass-API ab.

Vorbereiten der Daten

Rohdaten beschaffen

Falls Sie noch keine Rohdaten haben, laden Sie sich diese herunter. Sie benötigen eine CityGML-Datei und GTFS-Daten.

3D-Stadtmodell (CityGML)

3D-Stadtmodelle finden Sie häufig auf den offiziellen Webseiten von Städten oder Bundesländern. Sollten Sie dort nicht fündig werden, können Sie in der Mobilithek (https://mobilithek.info) oder im Geoportal (https://www.geoportal.de) nach frei lizenzierten 3D-Stadtmodellen suchen. **Hinweis:** Nicht alle Modelle sind kostenfrei nutzbar.

GTFS

Die GTFS-Daten für Deutschland können Sie online über https://gtfs.de/de/feeds/ herunterladen. GTFS-Feeds für andere Länder sind ebenfalls online verfügbar. Eine Übersicht finden Sie beispielsweise unter https://gtfs.org/resources/data.

OpenStreetMap

Sie müssen keine OpenStreetMap-Daten herunterladen. OpenStreetMap-Daten werden automatisch über die Overpass-API abgerufen und gespeichert, basierend auf den von Ihnen gewählten Koordinaten.

Datenqualität prüfen

Um die Ergebnisse der Datenverknüpfung richtig einzuordnen und Fehler schnell zu identifizieren, ist es entscheidend, die Qualität der Ausgangsdaten zu prüfen.

Achten Sie besonders darauf, welche Objekte in den Dateien vorhanden sind. Diese Informationen sind später hilfreich, etwa wenn Verknüpfungen fehlschlagen, weil eine Datei bestimmte Objekte wie Punkte oder Flächen nicht enthält.

3D-Stadtmodell (CityGML)

Visuelle Prüfung: wenn möglich, öffnen Sie das 3D-Stadtmodell in einem Viewer wie SimStadt (https://simstadt.hft-stuttgart.de/related-softwares/city-gml-viewer/) **Hinweis:** nur für Windows und Linux

1. Überprüfen Sie, ob die Datei folgende Objekte enthält:
    - Gebäude
    - Straßen
    - Punktdaten

Prüfung im Texteditor: Falls kein visueller Viewer verfügbar ist, öffnen Sie die Datei in einem Texteditor.

1. Suchen Sie nach den Begriffen:
    - Polygon (Flächen)
    - Point (Punktdaten)

OpenStreetMap

1. Öffnen Sie die Datei **overpass_request_config.txt** in einem Texteditor
2. Kopieren Sie den gesamten Inhalt der Datei overpass_request_config.txt in die Zwischenablage
3. Öffnen Sie die Website Overpass Turbo: https://overpass-turbo.eu
4. Fügen Sie den kopierten Text aus der Datei overpass_request_config.txt in das Textfeld auf der linken Seite von Overpass Turbo ein
5. Navigieren Sie im rechten Kartenbereich von Overpass Turbo zu dem Gebiet, dessen Daten Sie verknüpfen wollen
6. Klicken Sie in Overpass Turbo oben links auf den Button „Ausführen“
7. Wechseln Sie in Overpass Turbo oben rechts in die Ansicht „Daten“
8. Überprüfen Sie auf das Vorhandensein folgender Objekte:
    - "type": "node" (Punktdaten)
    - "type": "way" (Linien)

GTFS

Eine Überprüfung der GTFS-Daten ist nicht erforderlich. Da diese Daten direkt von Verkehrsbetrieben stammen, kann von einer hohen Datenqualität ausgegangen werden.

Arbeitsschritte zur Datenverknüpfung

Dieser Abschnitt beschreibt eine Schrittweise das Vorgehen zum Verknüpfen der Daten mit Hilfe der mitgelieferten Python-Skripte.

Um einzelne Python-Skripte auszuführen, öffnen Sie ein Terminal und wechseln Sie in den Projektordner. Starten Sie die gewünschten Programme mit dem Befehl:

```
python3 <programmname.py>
```

Während der Ausführung zeigt das Terminal Statusmeldungen an. Kritische Eingaben von Parametern und die Dateiauswahl erfolgen über Pop-up-Fenster. Prüfen Sie diese sorgfältig, um Fehler zu vermeiden.

Schritt 1: check_installation.py ausführen

Informationen zum Skript:

Das Python-Skript check_installation.py prüft, ob die nötige Software sowie die nötigen Pakete auf Ihrem System installiert sind.

Vorgehen:

1. Führen Sie check_installation.py aus

   ➔ Ein Fenster öffnet sich. Es zeigt installierte und fehlende Software sowie Pakete an.

2. Prüfen Sie den Status der Installation:
   - Wenn alle Einträge grün hinterlegt sind, ist alles installiert.
   - Wenn Einträge rot hinterlegt sind, fehlen Software oder Pakete. installieren Sie die fehlende Software oder Pakete:
     - Installieren Sie fehlende Software manuell

Problem	Lösung
Java Runtime Environment ist nicht installiert	Installieren Sie ein Java Runtime Environment ihrer Wahl, beispielsweise Oracle Java https://www.oracle.com/java/technologies/downloads/.
Docker ist nicht installiert	Installieren Sie Docker von der offiziellen Website https://www.docker.com.

     - Fehlende Pakete werden von check_installation.py auf Ihren Befehl hin installiert:
       - Klicken Sie auf den Button „Fehlende Pakete installieren“ unten rechts im geöffneten Fenster.
       - Es öffnet sich ein Fenster. Bestätigen Sie dort, dass die Pakete installiert werden sollen. Klicken Sie dazu auf den Button „Yes“

         ➔ im Terminal finden Sie die Ausgabe „`Die fehlenden Pakete wurden erfolgreich installiert.`“
     - Klicken Sie auf den Button „Status aktualisieren“ unten links im geöffneten Fenster.

       ✔ Wenn alle Einträge grün hinterlegt sind, ist alles installiert.

Schritt 2: docker_creation.py ausführen

Informationen zum Skript:

Das Python-Skript docker_creation.py erstellt drei Docker-Container im selben Netzwerk:

Tabelle 2: Beschreibung der Docker-Container

Container-Name	Container-Funktion
projecttool_postgis_osm	Beinhaltet nach dem Import die OpenStreetMap-Daten. Der Container basiert auf PostGIS (**Hinweis:** für mehr Informationen, siehe postgis.net).
projecttool_citydb	Beinhaltet nach dem Import die CityGML-Daten. Der Container basiert auf 3DCityDB (**Hinweis:** für mehr Informationen, siehe https://www.3dcitydb.org) und enthält außerdem PostGIS.
projecttool_postgis_intersection	Beinhaltet nach dem Import die OpenStreetMap-Daten, GTFS-Daten und die CityGML-Daten. Innerhalb des Containers werden die Daten später verknüpft.

Nach Erstellung der Container importiert das Python-Skript die von Ihnen angegebene CityGML-Datei über den 3DCityDB Importer in den Container projecttool_citydb.

Vorgehen:

1. Starten Sie die Anwendung Docker
2. Führen Sie docker_creation.py aus

   ➔ Ein Fenster öffnet sich. Wählen Sie dort Ihre CityGML-Datei aus.

   Das Skript erstellt nun die Docker-Container. Dies kann etwas Zeit in Anspruch nehmen.
3. Prüfen Sie das Terminal auf Fehler
   - Bei Fehlermeldungen (`ERROR`) starten Sie das Programm erneut oder analysieren die Meldungen genauer. Nutzen Sie dazu die Logs docker_creation_log<Datum><Uhrzeit>.log.

✓ Wenn Sie im Terminal die Ausgabe „`docker_creation.py erfolgreich abgeschlossen.`“ sehen, wurde das Skript erfolgreich ausgeführt. Sie können die Docker-Container nun auch in der grafischen Oberfläche der Docker-Anwendung sehen.

Schritt 3: overpass_api_call.py ausführen

Informationen zum Skript:

Das Python-Skript overpass_api_call.py ermöglicht zwei Optionen zur Festlegung der Koordinaten für den Overpass-API-Abruf:

- **GeoJSON-Datei:** Laden Sie eine GeoJSON-Datei hoch, um die Koordinaten direkt zu verwenden.
- **GeoNames-Datei:** Laden Sie eine GeoNames-Datei hoch, geben Sie die gewünschten GeoNames ein und legen Sie einen Radius um die Punkte fest (**Hinweis:** für mehr Informationen, siehe https://www.geonames.org).

Das Skript führt anschließend die in overpass_request_config.txt hinterlegte Overpass-API-Abfrage mit den festgelegten Koordinaten aus und ruft die entsprechenden Daten ab.

Sie können die zur Overpass-API-Abfrage genutzte Query anpassen. Siehe dazu *Overpass-API-Query anpassen*.

Vorgehen:

1. Führen Sie overpass_api_call.py aus

   ➔ Ein Fenster öffnet sich
2. Entscheiden Sie sich, ob Sie GeoJSON oder GeoNames nutzen wollen
   - Wenn Sie sich für GeoJSON entschieden haben:
     - Klicken Sie auf den Button „GeoJSON-Datei auswählen“.

       ➔ Ein Dateiauswahl-Dialog öffnet sich
     - Wählen Sie eine GeoJSON-Datei aus
     - Klicken Sie auf den Button „Bestätigen“ unten mittig im geöffneten Fenster
   - Wenn Sie sich für GeoNames entschieden haben:
     - Wechseln Sie in die Ansicht für GeoNames, indem Sie oben mittig im Fenster den Tab „GeoNames“ anklicken
     - Klicken Sie auf den Button „GeoNames-Datei laden“

       ➔ Ein Dateiauswahl-Dialog öffnet sich
     - Wählen Sie eine GeoNames-Datei aus
     - Wählen Sie die gewünschten GeoNames aus
       - Klicken Sie in das Suchfeld im geöffneten Fenster

- Geben Sie dort ihren Suchtext ein. Dieser muss als GeoName existieren.
- Bestätigen Sie ihren Suchbegriff mit Enter

  ➔ Sie sehen nun in der darunterliegenden Liste die Suchergebnisse
- Wählen Sie die gewünschten Zeilen durch Mausklick an

  ➔ Die Zeile ist hervorgehoben
- Klicken Sie auf den Button „Auswahl hinzufügen“

  ➔ die gewählten Zeilen erscheinen in der zweiten Liste
- Wählen Sie so viele GeoNames aus wie gewünscht
- Geben Sie den Radius in Metern in das zugehörige Textfeld ein
- Klicken Sie auf den Button „Bestätigen“ unten mittig im Fenster

➔ das Fenster schließt sich.

3. Prüfen Sie das Terminal auf Fehler
   - Bei Fehlermeldungen (ERROR) starten Sie das Programm erneut oder analysieren die Meldungen genauer. Nutzen Sie dazu die Logs overpass_api_call<Datum><Uhrzeit>.log.

✓ Wenn Sie im Terminal die Ausgabe „`overpass_api_call.py erfolgreich abgeschlossen.`“ sehen, wurde das Skript erfolgreich ausgeführt. Das Ergebnis des API-Aufrufs wurde in den Ordner UrbanLink3D/exports in die Datei „overpass_result.xml“ geschrieben.

Overpass-API-Query anpassen

Wenn Sie die konfigurierten Informationen, die aus OpenStreetMap abgerufen werden, ändern wollen, müssen Sie die Query anpassen.

1. Öffnen Sie die Datei „overpass_request_config.txt“ aus dem Ordner UrbanLink3D
2. Wenn Sie neue Inhalte zu der Query hinzufügen wollen:
   - Fügen Sie neue Zeilen hinter der ersten runden Klammer ein:

```
[out:xml][timeout:90];
(
<ihre neuen Overpass-Regeln>
```

13

- Achten Sie darauf, dass jede Zeile mit der Zeichenkette `({{bbox}});` endet

3. Wenn Sie Zeilen aus der Query entfernen wollen:
   - Suchen Sie nach der Zeile, die Sie löschen wollen
   - Löschen Sie die gesamte Zeile

Schritt 4: osm_import.py ausführen

Informationen zum Skript:

Das Python-Skript osm_import.py importiert die von Ihnen angegebene OSM-Datei in den Docker-Container projecttool_postgis_osm.

Vorgehen:

1. Führen Sie osm_import.py aus

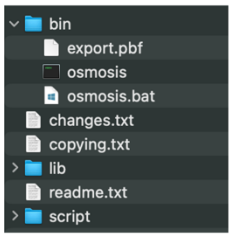
   ➔ Ein Fenster öffnet sich. Wählen Sie dort aus dem Ordner UrbanLink3D/exports die XML-Datei „overpass_result.xml“ aus

   Das Skript importiert nun die XML-Datei. Dies kann etwas Zeit in Anspruch nehmen.

2. Prüfen Sie das Terminal auf Fehler

   - Bei Fehlermeldungen (`ERROR`) starten Sie das Programm erneut oder analysieren die Meldungen genauer. Nutzen Sie dazu die Logs osm_import _log<Datum><Uhrzeit>.log.
   - Falls die Fehler bestehen bleiben, wandeln Sie die XML-Datei in eine PBF-Datei um.
     - Laden Sie das Tool Osmosis herunter: https://github.com/openstreetmap/osmosis/releases/tag/0.49.2
     - Entpacken Sie die Datei in einen Ordner

       Ihr Ordner sollte folgendermaßen aussehen:

     - Kopieren Sie die Datei „overpass_result.xml“ aus dem Ordner UrbanLink3D/exports in den Osmosis-Ordner
     - Öffnen Sie ein Terminal und navigieren Sie in Ihren Osmosis-Ordner
     - Führen Sie im Ordner folgenden Befehl aus:

       ```
       bin/osmosis --read-xml file=overpass_result.xml --write-pbf file=export.pbf
       ```

➔ die Datei export.pbf wird erstellt

- Wiederholen Sie den Prozess ab Schritt 1, wählen Sie jedoch die Datei „export.pbf“ aus Ihrem Osmosis-Ordner aus

✓ Wenn Sie im Terminal die Ausgabe „`osm_import.py erfolgreich abgeschlossen.`“ sehen, wurde das Skript erfolgreich ausgeführt. Prüfen Sie jedoch, ob alle Objekte importiert wurden.

- Im Terminal finden Sie eine Ausgabe nach folgendem Schema:

```
2025-01-04 16:01:56    Processed 1319 nodes in 0s - 1k/s
2025-01-04 16:01:56    Processed 747 ways in 0s - 747/s
2025-01-04 16:01:56    Processed 74 relations in 0s - 74/s
```

- Vergleichen Sie die Anzahl der verarbeiteten Objekte mit den Erkenntnissen aus der Analyse der Datenqualität. Wenn nicht alle Objekte vorhanden sind, ist dies ein Hinweis auf einen Fehler beim Import. Dies kann auch passieren, wenn der Import generell erfolgreich war.

Schritt 5: intersect_data.py ausführen

Informationen zum Skript:

Das Python-Skript intersect_data.py importiert die von Ihnen angegebene GTFS-Datei in die Datenbank im Docker-Container projecttool_postgis_intersection. Anschließend verknüpft es die OpenStreetMapDaten mit den CityGML-Daten und den GTFS-Daten. Dabei werden die von Ihnen angegebenen Schwellenwerte berücksichtigt.

Vorgehen:

1. Führen Sie intersect_data.py aus

   ➔ Ein Fenster öffnet sich. Geben Sie dort die Schwellenwerte ein:

   - Maximaler Abstand, den Punkte voneinander haben dürfen, um verknüpft zu werden
   - Maximaler Abstand, den GTFS-Haltestellenpunkte von OpenStreetMap-Punkten haben dürfen, um verknüpft zu werden
   - Minimaler Anteil, zu dem sich Flächen überschneiden müssen, um verknüpft zu werden

   • Klicken Sie auf den Button „Bestätigen“ um Ihre Eingabe zu bestätigen

   ➔ Ein Fenster öffnet sich. Wählen Sie dort eine GTFS-Datei mit dem Namen stops.txt aus

   Das Skript importiert nun die GTFS-Datei und führt die Verknüpfungsoperationen durch. Dies kann bis zu 5 Minuten dauern.

2. Prüfen Sie das Terminal auf Fehler

   • Bei Fehlermeldungen (`ERROR`) starten Sie das Programm erneut oder analysieren die Meldungen genauer. Nutzen Sie dazu die Logs intersect_data _log<Datum><Uhrzeit>.log.

   ✓ Wenn Sie im Terminal die Ausgabe „`osm_import.py erfolgreich abgeschlossen.`“ sehen, wurde das Skript erfolgreich ausgeführt. Prüfen Sie jedoch, ob alle Objekte importiert wurden.

✓ Sie haben alle Schritte ausgeführt ✓

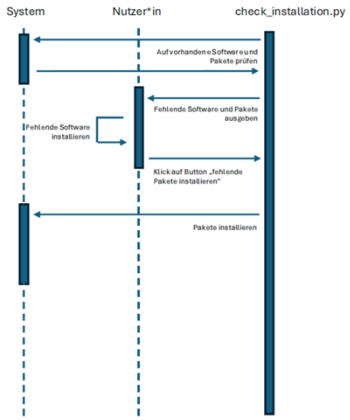
Anpassungsmöglichkeiten der Skripte

Sie können in allen Python-Skripten Anpassungen vornehmen. Um diese Anpassungen zu erleichtern, werden im Folgenden alle Skripte kurz erläutert und beispielhaft einfache Anpassungsmöglichkeiten genannt.

Wenn Sie tiefgreifende Änderungen vornehmen wollen, setzen Sie sich zuerst ausführlich mit dem Quellcode der Skripte auseinander. Dieser ist durch Quellcode-Kommentare verständlich gestaltet.

check_installation.py

Abbildung 3: check_installation.py

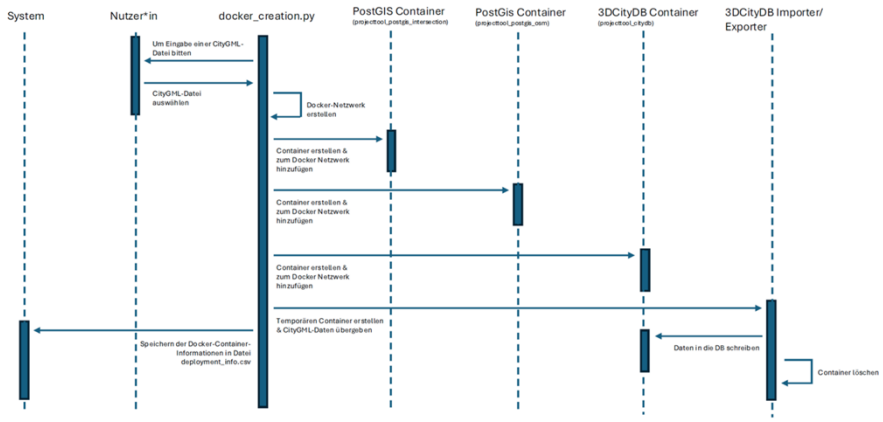


Check_installation.py prüft die installierte Software und installiert automatisch fehlende Python-Pakete.

Das Python-Skript kann um die Prüfung weiterer Software-Produkte oder Python-Pakete erweitert werden.

docker_creation.py

Abbildung 4: docker_creation.py

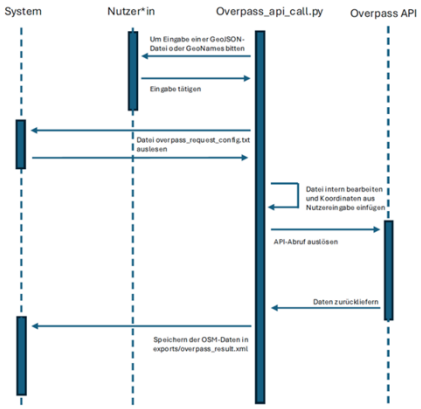


Docker_creation.py erstellt die Docker-Umgebung inklusive Docker-Containern und Docker-Netzwerk. Es speichert die Informationen zur Docker-Umgebung in logs/deployment_info.csv und importiert eine CityGML-Datei in die 3DCityDB.

Um die vom Python-Skript automatisch gesetzten Container-Namen, Nutzernamen und Passwörter zu ändern kann das Skript angepasst werden. Der 3DCityDB Importer – Befehl kann angepasst werden, wenn bestimmte Konfigurationen nötig sind. Name und Pfad der Datei für die Speicherung der Container-Informationen können geändert werden.

overpass_api_call.py

Abbildung 5: overpass_api_call.py

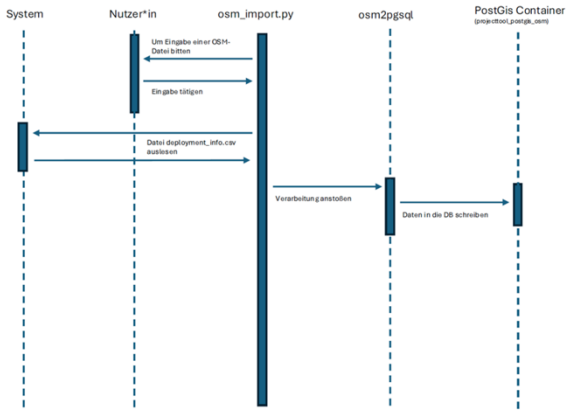


Das Python-Skript overpass_api_call.py liest Koordinaten aus einer GeoNames-Datei oder GeoJSON-Datei und ruft OpenStreetMap-Daten über die Overpass-API ab.

Sie können die von OpenStreetMap importierten Informationen anpassen. Lesen Sie dazu den Abschnitt *Overpass-API-Query anpassen*.

osm_import.py

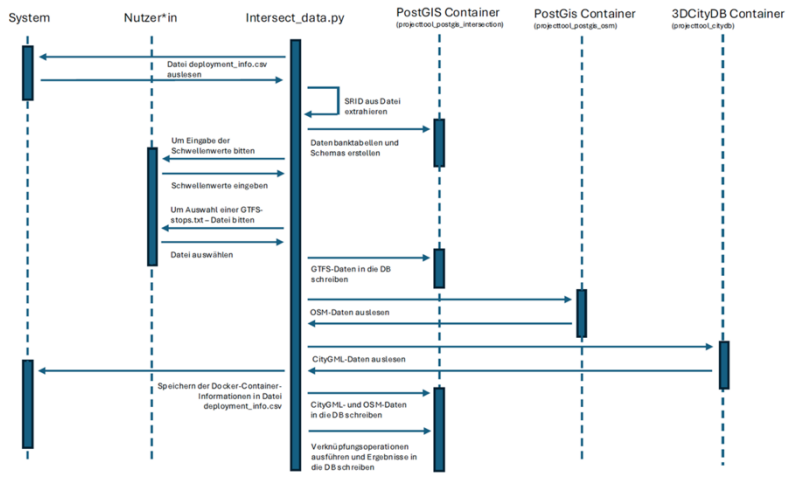
Abbildung 6: osm_import.py



Das Python-Skript osm_import.py importiert OpenStreetMap-Daten mithilfe von osm2pgsql in eine Datenbank.

Sie können den osm2pgsql-Befehl anpassen um weitere Datentypen für den Import zuzulassen.

intersect_data.py



Das Python-Skript intersect_data.py kombiniert CityGML-, OpenStreetMap- und GTFS-Daten in einer PostGIS-Datenbank mithilfe folgender Verknüpfungsregeln:

Tabelle 3: Verknüpfungsregeln

Fall	Gewählte Auflösung
OSM-Node liegt genau auf einem CityGML-Point	keine Auflösung nötig, Zuordnung aller OSM-Attribute zu CityGML-Objekt
OSM-Polygon liegt genau auf einer CityGML-Fläche	keine Auflösung nötig, Zuordnung aller OSM-Attribute zu CityGML-Objekt
OSM-Polygon liegt komplett innerhalb einer CityGML-Fläche	Anteil OSM-Polygon an CityGML-Fläche ≥ Schwellenwert: OSM-Attribute werden der CityGML-Fläche zugeordnet. Anteil OSM-Polygon an CityGML-Fläche < Schwellenwert: Keine Zuordnung findet statt
OSM-Polygon und CityGML-Fläche überschneiden sich teilweise	Anteil OSM-Polygon an CityGML-Fläche ≥ Schwellenwert: OSM-Attribute werden der CityGML-Fläche zugeordnet. Anteil OSM-Polygon an CityGML-Fläche < Schwellenwert: Keine Zuordnung findet statt
OSM-Node liegt nicht genau auf einem CityGML-Point	Abstand OSM-Punkt zu CityGML-Punkt ≤ Schwellenwert: OSM-Attribute werden CityGML-Punkt zugeordnet

	Abstand OSM-Punkt zu CityGML-Punkt > Schwellenwert: Keine Zuordnung findet statt
OSM-Way liegt innerhalb einer CityGML-Fläche	keine Zuordnung findet statt
CityGML-Fläche liegt komplett innerhalb eines OSM-Polygons	Zuordnung der OSM-Attribute zur CityGML-Fläche
GTFS-Punkt liegt genau auf einem OSM-Node	keine Auflösung nötig, Zuordnung der GTFS-ID zu OSM-Node
GTFS-Punkt liegt genau auf einem OSM-Node	Abstand GTFS-Punkt zu OSM-Node ≤ Schwellenwert: GTFS-ID wird OSM-Node zugeordnet Abstand GTFS-Punkt zu OSM-Node > Schwellenwert: Keine Zuordnung findet statt

Sie können die in die Verknüpfungsdatenbank importierten Daten anpassen und die implementierten Verknüpfungsregeln ändern.

Arbeitsschritte unter Nutzung des Terminals

Sie können das 3D-Stadtmodell und die OpenStreetMap-Daten auch manuell in Docker importieren. Ab dem Schritt der Datenverknüpfung ist die Nutzung einer Automatisierung jedoch empfehlenswert. Ab diesem Punkt werden mögliche Schritte nur noch beschrieben.

1. erstellen Sie die 3DCityDB

```
docker run -d --name projecttool_citydb --network projecttool_net \
    -e POSTGRES_PASSWORD=changeMe! \
    -e SRID=25833 \
    -p 5432:5432 \
    3dcitydb/3dcitydb-pg
```

2. importieren Sie die CityGML-Datei mit dem 3DCityDB Importer in die 3DCityDB

```
docker run -i -t --name impexp --rm --network projecttool_net \
    -v /Users/Pfad/zu/CityGML/Ordner:/data \
    3dcitydb/impexp:latest \
    import -H cdb -P 5432 -d postgres -u postgres -p changeMe! \
    /data/CityGML_Datei.gml
```

3. erstellen Sie die PostGIS Datenbank für die OpenStreetMap-Daten

```
docker run -d \
  --network projecttool_net \
  -v osm-data:/var/lib/postgresql/data \
  -e POSTGRES_USER=postgres \
  -e POSTGRES_PASSWORD=postgres \
  -e POSTGRES_DB=projecttool_postgis_osm \
  -p 5436:5432 \
  --name postgis_osm \
  postgis/postgis
```

4. fügen Sie der OpenStreetMap-Datenbank die hstore-Erweiterung hinzu

```
docker exec -it postgis_osm psql -U postgres -d
projecttool_postgis_osm -c "CREATE EXTENSION IF NOT EXISTS hstore;"
```

5. Exportieren Sie der OpenStreetMap-Daten, beispielsweise aus Overpass Turbo

6. Importieren Sie Ihre OpenStreetMap-Daten in die PostGIS-Datenbank

```
docker run --rm \
  --network projecttool_net \
  -v /Users/Pfad/zu/exports/Ordner:/osm-data \
  -e PGPASSWORD=postgres \
  iboates/osm2pgsql:latest \
  osm2pgsql \
  -H postgis_osm \
  -P 5432 \
  -d projecttool_postgis_osm \
  -U postgres \
```

```
--create \
--slim \
--hstore \
/osm-data/export.pbf
```

7. Verknüpfungsregeln festlegen

   Legen Sie fest, wie sich überschneidende Flächen und Punkte miteinander verknüpft werden sollen. Legen Sie Schwellenwerte fest, bis zu welchem Abstand und welchem Überschneidungsanteil die Objekte einander zugeordnet werden sollen.

8. Erstellen Sie eine weitere PostGIS-Datenbank, in welche die CityGML-Daten und die OpenStreetMap-Daten importiert werden können. Importieren Sie auch die GTFS-Daten in diese Datenbank.

9. Führen Sie die Verknüpfungsoperationen durch. Schreiben Sie die Ergebnisse in eine Verknüpfungstabelle, indem Sie die IDs der CityGML-Objekte, der OpenStreetMap-Objekte und der GTFS-Haltestellen einander zuordnen.

10. Exportieren Sie die Verknüpfungsergebnisse aus der Datenbank. Fügen Sie sie in die Tabelle `cityobject_genericattrib` der 3DCityDB ein.

11. Exportieren Sie die Daten über den 3DCityDB Exporter in ein Datenformat Ihrer Wahl. Es stehen die Datenformate CityGML, CityJSON, GZIP und Zip-Archiv zur Verfügung. Außerdem ist der Export der visuellen Daten möglich, in den Formaten KML und Collada/glTF.

    Der Export als glTF kann folgendermaßen gestartet werden:

    - Installieren Sie COLLADA2GLTF:
    - Führen Sie folgenden Terminal-Befehl aus:

```
docker run -i -t --name impexp --rm --network projecttool_net \
  -v /Users/Pfad/zu/exports/Ordner/3DCityDB_Exports:/output \
  3dcitydb/impexp:latest \
  export-vis -H cdb -P 5432 -d postgres -u postgres -p changeMe!
\
  -o /output/citydb_export \
  --display-form collada \
  --lod 2 \
  -G --remove-collada \
  --gltf-converter=/Pfad/zu/COLLADA2GLTF
```

Mögliche Fehlerquellen

Im Folgenden werden Fehlerquellen, die beim Test des Workflows aufgefallen sind, kurz beschrieben.

Unvollständiger Import der OSM-Daten

- Viele erwartete Polygone wurden durch osm2pgsql nicht in die PostGIS-Datenbank importiert oder lediglich als unvollständige Ways gespeichert.
- Ein Großteil der relevanten OSM-Punkte und -Flächen fehlte nach dem Import in die PostGIS-Datenbank, was die Grundlage für Überschneidungen mit CityGML-Objekten erheblich einschränkte.
- Wiederholte Importe mit verschiedenen Rohdaten brachten keine Verbesserung.

Fehlerhafte Polygonerkennung

- Manche geschlossene Ways, die Polygone repräsentieren sollten, wurden durch osm2pgsql entweder nicht erkannt oder nur teilweise in die PostGIS-Datenbank importiert.
- Trotz gezielter Abfragen zur Überprüfung geschlossener Geometrien konnten die fehlenden Polygone nicht identifiziert werden.

Aus den genannten Problemen entstanden Probleme beim Verknüpfen der Daten, da ein Großteil der Rohdaten fehlte oder fehlerhaft war. Aus diesem Grund konnte der Export der Verknüpften Daten nicht getestet werden und wurde nicht als konkrete Handlungsanweisung in dieses Dokument aufgenommen.

Anhang 6 Aufgabenstellung

Aufgabenstellung

- Verknüpfen Sie die bereitgestellte CityGML-Datei mit Daten aus OpenStreetMap und GTFS.
- Wenn Sie an Schritt 3 angelangt sind, führen Sie folgendes aus:
    - Passen Sie die Overpass-API-Query an, indem Sie folgende Zeile hinzufügen:

```
nwr["traffic_calming"="island"]({{bbox}});
```

Im Ordner „Rohdaten“ auf dem Desktop finden sie folgende Dateien, die Sie bei der Ausführung nutzen sollen:

Datei-/Ordnername	**Beschreibung**
GeoNamesDE.txt	GeoNames-Datei für Deutschland
Leipzig_Zentrum.geojson	GeoJSON-Datei mit Polygon für die relevante Fläche des OSM-Abrufs
GEB3D_Leipzig_LoD2_00_Zentrum.gml	3D-Stadtmodell der Leipziger Innenstadt
GTFS Ordner	Ordner mit GTFS-Daten zum ÖPNV in Deutschland

Informationen zur Datenqualität:

Teil des Workflows ist normalerweise die Analyse der Datenqualität der Rohdaten. Folgende Informationen können Ihnen dazu gegeben werden:

Die CityGML-Datei enthält ausschließlich Flächen und keine Punkte

Die OSM-Daten enthalten größtenteils Punkte und kaum Flächen

Aus diesem Grund werden keine Überschneidungen der Daten möglich sein. Der Importprozess kann trotzdem ausgeführt werden.

Anhang 7 Transkription Interview TP1

Testleiterin:

Hallo. Danke, dass du dir die Zeit nimmst und bei meinem Usability-Test teilzunehmen. Ich werde dir jetzt erst mal erklären, worum es in dem Test geht und welches Ziel wir jetzt verfolgen. Das Thema Geodaten ist für dich ja völlig neu, oder?

Testperson 1:

Ja

Testleiterin:

Ja ok, deswegen werde ich dir jetzt erstmal Grundlageninformationen geben. Wir testen heute einen Workflow, den ich im Rahmen meiner Masterarbeit entwickelt hab. Dabei möchte ich herausfinden, ob die Software-Skripte und deren Ausgaben verständlich sind und ob die Anleitung, die ich erstellt habe, verständlich und klar formuliert ist. Ziel des Workflows ist es, ein 3D-Stadtmodell mit weiteren mobilitätsrelevanten Daten zu verknüpfen. Ein 3D-Stadtmodell ist, wie der Name schon sagt, eine dreidimensionale Abbildung einer Stadt. Es zeigt beispielsweise Gebäude, Straßen, Flüsse oder markante Punkte. Diese Modelle werden im sogenannten CityGML-Format gespeichert, das auf XML basiert. Da die Daten stark miteinander verknüpft sind – beispielsweise gehören mehrere Wände zu einem Gebäude –, sind die Dateien oft recht komplex zu verarbeiten.

Unser Ziel der Evaluation ist es, beispielhaft das 3D-Stadtmodell der Leipziger Innenstadt mit Daten aus der General Transit Feed Specification (GTFS) und OpenStreetMap (OSM) zu kombinieren. GTFS liefert Informationen zum öffentlichen Nah- und Fernverkehr, wie Haltestellenpositionen, Linienführungen und Abfahrtszeiten, die in einzelnen Textdateien gespeichert sind. OSM ist eine frei zugängliche, weltweite Geodatenbank, über die du Punkte, Linien und Flächen abfragen kannst – von Einbahnstraßen und verkehrsberuhigten Bereichen bis hin zu Parks. Sogar spezifische Details wie das Material einer Parkbank können dort verzeichnet sein.

Zur Kombination dieser verschiedenen Datenquellen verwenden wir Docker-Container. Diese kannst du dir wie virtuelle Maschinen vorstellen, die auf Linux basieren und in denen spezielle Anwendungen laufen. Docker ermöglicht es, Software einfach zu verteilen und auszuführen.

Ich habe den Workflow mit Python-Skripten vereinfacht und eine Anwenderdokumentation erstellt, die ich dir gleich auf dem Laptop zeige. Deine Aufgabe wird es sein, die Anwenderdokumentation zu lesen und deren Schritte zu befolgen. Ich stelle dir alle notwendigen Rohdaten und Informationen zur Verfügung. Du wirst einmal den Prozess der Datenverknüpfung durchführen – der Export und die Beschaffung der Daten bleiben außen vor. Eine Übersicht deiner Aufgaben findest du zusätzlich auf einem Aufgabenblatt.

Während des Tests bitte ich dich laut zu denken. Erklär mir, was du gerade machst, was dein Ziel ist und ob dir etwas unklar erscheint oder Probleme bereitet. Ich bin hauptsächlich als Beobachterin hier, aber falls du gar nicht weiterkommst, kannst du mich jederzeit um Hilfe bitten.

Wie gesagt, hier ist deine Aufgabenstellung geöffnet.

Testperson 1:

Ja. [liest Aufgabenstellung laut vor]

Testleiterin:

Hier findest du die Anwenderdoku. Da kannst du einfach starten, wie du willst und ich hab dir hier einfach schon mal ein Terminal-Fenster aufgemacht, falls du irgendwie mit dem Terminal nicht ganz umgehen kannst, dann frag einfach.

Testperson 1:

Ja, am Mac ist das für mich nicht so ganz einfach, da ist ja alles etwas anders.

Testleiterin::

Deswegen.

Testperson 1:

So, dann kann ich anfangen mit der Aufgabenstellung.

Testleiterin:

Ja.

Testperson 1:

[liest die Anleitung laut, aber ungenau]

Rohdaten haben wir. Datenqualität stand in der Anleitung, genau. In der CityGML-Datei sind quasi ganz viele Flächen aber keine Punkte und in

Testleiterin:

Genau.

Testperson 1:

und in den OSM-Daten die man dann bekommt, sind ganz viele Punkte, aber quasi keine Flächen.

Testperson 1:

GTFS. Gehört das noch dazu? [Zum Abschnitt "Prüfen der Datenqualität"]

Testperson 1:

OK, Mhm Schritt 1. Und geht's jetzt hier los? Schritt 1. Geht's los. Mal gucken, was ja da noch kommt. [scrollt das gesamte Dokument durch]

Testperson 1:

Mhm.

Testperson 1:

Ok, dann machen wir das. Schritt für Schritt.

Testperson 1:

Okay. So. Schritt 1 Schritt. Check_installation_Python ausführen Informationen zum Schritt ... Okay dann wollen wir das tun.

[gibt falschen Befehl (check_installation.py statt python3 check_installation.py) ein, ohne vorher den Ordner zu wechseln]

Mhm. Error. Aber das ist doch richtig, oder? [liest noch mal, gibt nun den richtigen Befehl aber immer noch im falschen Ordner ein]

Testleiterin:

Also erst mal bist du noch im falschen Ordner. Du musst erst mal in den Projektordner.

Testperson 1:

Ah. Mit ls kann ich schauen, wo ich bin?

Testleiterin:

Genau. Damit kannst du gucken wo du bist und du musst auf Desktop und dann UrbanLink3D.

Testperson 1:

So okay, na dann. OK so. Ja, genau. [hat etwas Probleme mit dem Mac-Terminal]

Testperson 1:

Perfekt. So, dann wollen wir hier noch mal check_installation. Wieder mit python3 ja?

Testleiterin:

Richtig.

Testperson 1:

[überfliegt Anleitung laut] Software ist beides da. Pakete da fehlt pandas. Ok. Und jetzt? [längeres Denken und vor sich hin murmeln]

Hilf mir.

Testleiterin:

Software ist installiert, das hast du richtig erkannt, aber Pakete fehlen, hattest du gesagt.

Testperson 1:

[liest Anleitung noch Mal genauer]

Ja. OK.

Aktualisier Mal. Ach hat er. Perfekt. Und jetzt?

Testperson 1:

Führen Sie docker_creation.py aus? OK, dann. Das ist wieder mit Dings [der Befehl python3 ist gemeint] vorne oder?.

Testleiterin:

Mhm, genau das nutzt du immer, um halt so ein Programm auszuführen

Testperson 1:

Was ist jetzt die CityGML Datei? [sucht länger]

Testleiterin:

Die Dateinamen findest du auch noch mal eine Aufgabenstellung, sonst falls du dir da nicht sicher bist.

Testperson 1:

So, dann nehmen wir die.

Testperson 1:

So so. Das kann etwas Zeit. Anspruch nehmen, OK. [liest Anleitung laut vor]

Testleiterin:

Denkt dran, in Schritt 3 hattest du noch eine andere Aufgabe.

Testperson 1:

Mhm, das war aber erst am Ende, wenn ich das richtig in Erinnerung habe. Hallo, wenn es Schritt angelangt sind, führen Sie folgendes aus, passen Sie die Overpass Query an, indem sie folgende Zeile hinzufügen.

Testleiterin:

Das musst du tatsächlich vorher machen.

Testperson 1:

OK, also mach ich das jetzt erst ja. Ich dachte das kommt danach. Weil es drunter ist.

[versucht den overpass-Abruf der hinzugefügt werden soll ins Terminal zu schreiben]

Testperson 1:

Also öffnen Sie die Datei overpass. Die liegt im Projektordner drin?

Testleiterin:

Ja die liegt im Projektordner.

Testperson 1:

Der Projektordner ist der UrbanLink3D Ordner richtig?

Testleiterin:

Genau. Ja, ja.

Testperson 1:

[führt die Änderung durch]

Testperson 1:

Dann. Okay. Achten Sie darauf, dass Zeichenkette mit bbox endet. Ja, das tut sie, wenn die Zeilen entfernen wollen. Ok so, na dann.

Testperson 1:

Jetzt schließe ich das und dann kommt der Schritt. Overpass okay Vorgehen führen sie hier die Overpass okay sehr gut entscheiden Sie sich, ob Sie GeoJSON oder GeoNames nutzen wollen.

Testleiterin:

Das kannst du dir auch völlig frei aussuchen jetzt also da habe ich keine Vorgabe, was du da nehmen sollst.

Testperson 1:

Na, dann nehm ich GeoJSON ist gleich das erste. OK, wählen Sie die GeoJSON Datei aus, die liegt auch in den Rohdaten?

Testleiterin:

Auch in den Rohdaten genau.

Testperson 1:

OK, so klicken Sie auf den Button bestätigen. Gut Terminal dauert ne Weile.

Testperson 1:

OK OK. Hier erfolgreich abgeschlossen, sehr gut so. Prüfen Sie das Terminal auf Fehler bei Fehlermeldung Error haben wir nicht, zum Glück. [liest nächsten Schritt vor]

Testperson 1:

Es kann etwas Zeit in Anspruch nutzen. OK, sehr gut, scheint auch gut zu arbeiten.

Testperson 1:

OOK er scheint doch Probleme zu haben [aufgrund der XML im OSM-Import].

Testperson 1:

[liest Vorgehen bei Fehlermeldung]

Testleiterin:

Genau also die Fehler werden bestehen bleiben, das kann ich dir jetzt schon mal sagen.

Testperson 1:

Ok also nicht noch Mal probieren.

Testleiterin:

Nee.

Testperson 1:

[liest Vorgehen zum Umwandeln XML in Osmosis]

Testperson 1:

Das ist bestimmt schon runtergeladen.

Testleiterin:

Jap.

Testperson 1:

[liest Anleitung genauer]

So was haben wir denn hier? Der Ordner sollte wie folgt aussehen. Sieht sehr gut aus. Kopieren Sie die Datei in den Osmosis Ordner. Kann ich ja über den normalen Explorer machen, oder?

Testleiterin:

Genau das kannst du dann machen. Ja.

Testperson 1:

Führen Sie folgenden Befehl aus. Den kann ich bestimmt kopieren, richtig?

Testperson 1:

[Umwandeln hat geklappt] Sehr gut.

Testperson 1:

Die Datei Excel PBF wird erstellt. Wiederhole den Prozess. Schritt 1 jedoch die Datei okay, dann können wir die [ein weiteres Terminal-Fenster] wieder schließen. Gehen wir noch mal hoch.

Testperson 1:

Prüfen Sie die Importierten Daten auf Vollständigkeit. Ja ich gehe davon aus, dass das stimmt.

Testleiterin:

Nee also das stimmt tatsächlich nicht. Objekte werden nicht ordentlich importiert, da fehlen viele. Das ist einfach so. Aber wir arbeiten jetzt einfach trotzdem mit den Ergebnissen weiter.

Testperson 1:

[liest die Aufgabenstellung laut weiter]

Wählen sie GTFS Datei mit dem Namen stops.txt. Ok.

Testperson 1:

Wo liegt das. [sucht] Die Textdatei Stopp, Stopp, Stopp, genau stops.txt nehmen wir.

Testperson 1:

Das dauert. Oh krass.

Testperson 1:

So, das ist jetzt fertig. Intersect_data.py erfolgreich abgeschlossen sehr gut. Wo waren wir denn hier? Das wars.

Testleiterin:

Gut okay jetzt wo du fertig bist, will ich dir noch ein paar Fragen stellen. Dabei gibt es keine richtigen oder falschen Antworten. Teil mir einfach deine Erfahrung mit.

Testperson 1:

Ja.

Testleiterin:

Also erst mal als ganz allgemein. Wie hast du die Bearbeitung der Schritte im Workflow insgesamt empfunden.

Testperson 1:

Also die Schritte waren eigentlich sehr gut erklärt. Also Probleme hatte ich nicht, es war ja einmal das ich das lieber vertauscht gehabt hätte [Info zum Anpassen der Overpass-Query lieber über den Anleitungsschriten als darunter], aber ansonsten war super erklärt. Ließ sich auch leicht bearbeiten.

Testleiterin:

Okay okay super. Und empfandest du die Ausführung der Arbeitsschritte als einfach oder als kompliziert?

Testperson 1:

Also mit der Anleitung an sich war es sehr leicht. Aber ohne wäre es glaube ich, recht kompliziert geworden.

Testleiterin:

So, dann komm ich jetzt auch schon zu der Software an sich. Wie fandest du die Gestaltung der Software insbesondere mit der Kombination daraus, dass die Ausgaben im Terminal passieren und die Eingabe über grafische Benutzeroberflächen passiert sind.

Testperson 1:

Ja, also es hat mir sehr gefallen, dass man aus dem Terminal rausgekommen ist, dass man da die Eingaben machen konnte, das war schon gut. Sehr positiv würde ich sagen, dass man auch gesehen hat, okay, ich trage das jetzt hier ein und es passiert was und es ist richtig was gemacht habe gut und stand auch im Terminal. Aber ich.

Testleiterin:

Dann würdest du auch sagen, dass es dadurch quasi einfacher ist, als wenn du das im Terminal hättest eingeben müssen.

Testperson 1:

Ja, ich denke schon.

Testleiterin:

Hast du die Software als einheitlich und konsistent in der Bedienung wahrgenommen oder ist dir irgendwas aufgefallen, wo du sagen würdest, das fand ich jetzt irgendwie ein bisschen uneinheitlich?

Testperson 1:

Na, das einzige ist, wenn man mal zwischen so einer grafischen Oberfläche und dem Terminal hergesprungen ist, aber das fand ich gut. Aber ansonsten einheitlich war es ja mussten ja nur Daten eingetragen werden.

Testleiterin:

Gab es Situationen, in denen dir unklar war, welche Daten erforderlich sind und wenn ja, wann war das?

Testperson 1:

Mhm, ich meine, wenn man das selber macht, weiß man wo seine Daten abgelegt sind. Ich meine, hier musste ich immer mal nachgucken, wo die Daten jetzt drin liegen, welchen Ordner aber ansonsten alles direkt gefunden.

Testleiterin:

Und du wusstest auch, welche Dateien quasi von dir verlangt werden einzugeben.

Testperson 1:

Wie in der Anleitung geschrieben war. ja.

Testleiterin:

Musstest du dir während der Nutzung viele Details merken? Und wenn ja, hat die Anleitung dir dabei geholfen, dass du dir weniger merken musst?

Testperson 1:

Also ich glaub ich muss mir relativ wenig merken, weil es sehr kurze Schritte waren und einfach erklärt.

Testleiterin:

Ok. Wäre die Software auch ohne die Anleitung nutzbar gewesen?

Testperson 1:

Also von Leuten die sich auskennen, ja da vielleicht schon. Aber für mich, nein. Ohne die Anleitung hätte mir da zu viel gefehlt.

Testleiterin:

Und dann komme ich jetzt zu den Ausgaben, die wir erhalten hast. Wusstest du während der Nutzung immer, was die Skripte gerade machen? Also war dir das immer bewusst?

Testperson 1:

Ich habe immer den Text davor gelesen, was die Skripte machen sollen und dadurch wusste ich, was die machen.

Testleiterin:

Und waren Fehlermeldungen verständlich und hilfreich? Falls du auf welche gestoßen bist?

Testperson 1:

Ich bin auf eine gestoßen, aber da hat ja die Anleitung gesagt, was ich machen musste von daher. War sehr einfach herauszufinden.

Testleiterin:

Hältst du die Software für leicht erweiterbar, falls neue Anforderungen oder Aufgaben entstehen würden?

Testperson 1:

Gut, das kann ich schlecht sagen. Ich weiß ja nicht, wie die Software so richtig aufgebaut ist.

Testleiterin:

Und jetzt komm ich noch mal speziell zur Anleitung, weil die Anleitung einheitlich und verständlich?

Testperson 1:

Ja. Definitiv.

Testleiterin:

Und gab es Stellen in der Anleitung, die dir irgendwie unklar oder zu wenig detailliert erschienen?

Testperson 1:

Würde ich nicht sagen.

Testleiterin:

Also wir hatten ja den einen Punkt, dass die Erklärung für die Overpass-Anpassung darunter sein könnte.

Testperson 1:

Ja. Genau.

Testleiterin:

Und mir ist noch aufgefallen, dass du am Anfang hast du die Anleitung ja sage ich mal so ein bisschen überflogen und dann wusstest du am Anfang halt nicht, dass du da dieses python3 eingeben musst, also das Stand ja in dem Text. lag das jetzt einfach daran, dass du jetzt überflogen hast, oder woran?

Testperson 1:

Ja, ich denke, ich denke wenn ich jetzt das erste Mal hätte selber machen müssen. Ohne diese Untersuchung, wo ich jemanden dabei hab, dann hätte ich mir das definitiv noch genauer durchgelesen.

Testleiterin:

OK, und dann wäre der Fehler wahrscheinlich nicht aufgetreten. OK, gut.

Testperson 1:

Genau.

Testleiterin:

Hast du sonst noch etwas, was du anbringen möchtest?

Testperson 1:

Nein, das wars.

Testleiterin:

Dann danke ich dir für die Teilnahme und die Zeit.

Testperson 1:

Sehr gerne, sehr gerne.

Anhang 8 Transkription Interview TP2

Testleiterin:

Hallo Hallo. Vielen Dank, dass du dir die Zeit nimmst, an meinem Usability Test teilzunehmen. Ich möchte dir zunächst kurz erklären, worum es in diesem Test geht und welches Ziel wir verfolgen. Da das Thema Geodaten für dich wahrscheinlich neu ist, geb ich dir vorab einige grundlegende Informationen. Heute testen wir einen Workflow, den ich im Rahmen meiner Masterarbeit entwickelt habe. Dabei möchte ich herausfinden, ob die Software-Skripte und deren Ausgaben verständlich sind und ob die Anleitung klar formuliert ist. Das Ziel ist ess, das ein ein 3D-Stadtmodell mit weiteren für die nachhaltige Mobilität relevanten Daten zu verknüpfen? Ein 3D-Stadtmodell ist, wie der Name schon sagt, eine dreidimensionale Abbildung einer Stadt. Es zeigt beispielsweise Gebäude, Straßen, Flüsse oder markante Punkte. Diese Modelle werden im sogenannten City GML Format gespeichert, das auf XML basiert. Da die Daten stark miteinander verknüpft sind, beispielsweise gehören mehrere Wände zu einem Gebäude, sind die Dateien oft recht komplex zu verarbeiten. Unser Ziel der Evaluation ist es, beispielhaft das 3D-Stadtmodell der Leipziger Innenstadt mit Daten aus der General Transit Feets Specification, das wird ab jetzt immer mit GTFS abgekürzt, und openstreetmap zu kombinieren.OpenStreetMap wird mit OSM abgekürzt. GTFS liefert Informationen zum öffentlichen Nah- und Fernverkehr wie Haltestellen, Positionen, Linienführungen und Abfahrtszeiten, die in einzelnen Textdateien gespeichert sind. OSM ist eine frei zugängliche weltweite Geodatenbank, über die du Punkte, Linien und Flächen abgefragt wewrden können. Von Einbahnstraßen und verkehrsberuhigten Bereichen bis hin zu Parks sogar spezifische Details wie das Material einer Parkbank können dort verzeichnet werden. Zur Kombination dieser verschiedenen Datenquellen verwenden wir Docker-Container. Die kannst du dir wie virtuelle Maschinen vorstellen, die auf Linux basieren und in denen spezielle Anwendungen laufen. Docker ermöglicht es Software einfach zu verteilen und auszuführen. Ich habe den Workflow mit Python-Skripten vereinfacht und eine Anwenderdokumentation erstellt, die ich dir gleich auf Laptop zeige. Deine Aufgabe wird es sein, die Anwenderdokumentation zu lesen und deren Schritte zu befolgen. Ich stell dir alle notwendigen Rohdaten und Informationen zur Verfügung. Du wirst einmal den Prozess der Datenverknüpfung durchführen. Der Export und die Beschaffung der Daten bleiben außen vor. Eine Übersicht deiner Aufgaben findest du zusätzlich auf einem Aufgabenblatt. Während des Tests bitte ich dich laut zu denken, erklär mir, was du gerade machst, was dein Ziel ist und ob dir etwas unklar erscheint oder Probleme bereitet. Ich bin hauptsächlich als Beobachterin hier, aber falls du gar nicht weiterkommst, kannst du mich jederzeit bitten. Ich werd auch nebenbei so n bisschen mit draufhauen. Genau. Da ist deine Aufgabenstellung.

Testperson 2:

Hier sind die bereitgestellten Daten, genau das, was du mir gerade erzählt hast

Testleiterin:

Genau, und da ist in einem weiteren Tab auch schon die Anwenderdokumentation offen.

Testperson 2:

[liest die Anwenderdokumentation laut durch]

Testperson 2:

Also python3. Und dann kann ich das hier [check_installation.py] ausführen. Hm. Hier gibts nen Fehler.

Testleiterin:

Also du musst erstmal noch in dem Projektordner rein.

Testperson 2:

Ach ok, ja. Jetzt bin ich da drin und jetzt kann ich das ausführen.

Testleiterin:

Mhm.

Testperson 2:

Das geht aber immer noch nicht.

Testleiterin:

OK, und die spitzen Klammern sind Platzhalter.

Testperson 2:

Java Runtime Environment ist installiert. Docker ist auch installiert. Python-Pakete. [schaut in Dokumentation] hier bin ich. Alles klar. Dann installier ich die Pakete. Ist installiert. Schön, freut mich.

Testperson 2:

[liest laut weiter die Anleitung] die Dokumente brauche ich.

Testleiterin:

Genau, die Dateien sind wie gesagt alle in diesem Rohdatenordner.

Testperson 2:

Docker Container erfolgreich initialisiert.

Testleiterin:

Der lädt jetzt ein bisschen.

Testperson 2:

[schaut in die Docker-Anwendung] Da sind Sie. Alle da.

Testperson 2:

[liest die Anleitung weiter laut]

Testperson 2:

Da nehme ich GeoJSON [in Schritt 3].

Testleiterin:

Denk dran, was für Schritt 3 auch noch ne andere Aufgabe. In deinem Aufgabenblatt.

Testperson 2:

Ja, stimmt, stimmt, danke für die Erinnerung. [ohne in die Anleitung zu schauen] Da muss ich jetzt bestimmt in die Datei overpass_config.py navigieren.

Testleiterin:

War das eine Frage oder ne Aussage?

Testperson 2:

Das war ne Frage.

Testleiterin:

Das steht genau unter dem, was du gerade gelesen hast [dort ist Link zu Anleitung zum Anpassen der Query]

Testperson 2:

Ach. OK. Ich mach die Datei auf. Das [die neue Zeile] kann ich ja jetzt am Ende einfügen?

Testleiterin:

Guck mal, da steht, wo das hinsoll. Ja, grundsätzlich ist das egal. Die Anleitung sagt zwar was anderes, aber grundsätzlich ist es egal.

Testperson 2:

Ach das ist wahrscheinlich mein Problem. Also das ist ja ich lese, dann mach ich dann les ich. Ich sollte vielleicht erst lesen.

Testleiterin:

Alles gut.

Testperson 2:

OK. Gut, hab ich gemacht, geh ich weiter hoch hier? Wo war ich denn? [Sucht Anleitung zu Schritt 3]

Testperson 2:

Hier bin ich im richtigen Schritt. Toll.

Testperson 2:

So hab ich ausgeführt. Entscheiden Sie, ob Sie GeoJSON oder GeoNames nutzen wollen. Dabei klicken Sie auf den Button Datei auswählen. Und dann ein Datei Auswahldialog öffnet sich, ja JSON nehme ich.

Testperson 2:

Ja, warte mal. Ja, ich war gerade beim. Ja, alles gut. Ne. Fehler beim Versuch okay ja, jetzt hat es sich auch beendet, ja. Okay hab ich ausgeführt, wurde Fehler geworfen. Hier steht, Programm neu starten oder umwandeln.

Testleiterin:

Ja, du startest jetzt das Programm bitte nicht erneut. Ich kann dir jetzt mal einfach so sagen, dass das nicht funktionieren wird.

Testperson 2:

Dass es nicht funktioniert, ja. Stand irgendwo oben. Wo stand das noch mal? Stand das irgendwo? Das gelesen.

Testperson 2:

Ja, OK. Das heißt, ich soll den Fehler beheben.

Sprecher

Ja.

Testperson 2:

Wir wandeln die XML Datei in einer PBF Datei um wie mache ich das? [liest Anleitung]

Testperson 2:

Osmosis runterladen.

Testleiterin:

Das ist schon runtergeladen.

Testperson 2:

Das ist das hier?

Testleiterin:

Genau das ist dieser Ordner da.

Okay ich soll die [Datei] hier nehmen, hier reinkopieren, wo genau hin? In den Ordner. Einfach hier rein.

Testperson 2:

Ja, so. Jetzt schieb ich da rein und dann öffnen Sie ein Terminal. Navigieren Sie in den Ordner. Brauch ich das denn hier noch?

Testperson 2:

OK, dann führen Sie im Ordner folgende Befehl aus. Mache ich. Ich hoffe mal, das funktioniert einfach. Boah wo bin ich? Teilweise ist es in Konsolen besser, die so ein bisschen farblich sind, ne?

Testleiterin:

Ja.

Testperson 2:

Ging das? Das ging nicht. Ach hier ist ein Zeilenumbruch. OK. Ja, hier [in der Anleitung] wird nicht ganz erkenntlich, mit dem Zeilenumbruch.

Testleiterin:

Ok.

Testperson 2:

OK. Wiederholen Sie den Prozess ab Schritt 1. Schritt 1 des Schritt 4?

Testleiterin:

Ja.

Testperson 2:

Das heißt, das bedeutet? Ich geh wieder in den Projektordner. Führe Import aus. Und sollte sich jetzt das Dingens öffnen und jetzt gehe ich aber in den Ordner [in dem die PBF-Datei liegt] da.

Testleiterin:

Mhm ja.

Testperson 2:

Erfolgreich abgeschlossen. Perfekt.

Testperson 2:

Mhm OK, weitermachen gut, dann soll ich das nächste Skript ausführen.

Testleiterin:

Mhm, genau.

Testperson 2:

[liest die Aufgabenstellung laut vor] Ein Fenster öffnet sich. Ist richtig. Geben Sie Schwellenwerte ein. Da gibt es aber irgendwo n also eine Vorgabe? Gibt es da ein zu wenig und zu hoch, oder?

Testleiterin:

Das kommt halt immer auf die Anwendung drauf an und da kann man noch so n bisschen mit rum testen und drum spielen sag ich mal. Du kannst dir da einfach irgendwas überlegen. Nein.

Testleiterin:

Also das ist ja quasi dazu, da musst du ja so sehen, um bei der Überschneidung auftretende Ungenauigkeiten, sag ich mal auszugleichen, also zum Beispiel in CityGML ist n Punkt eingetragen für ne Haltestelle für n Bushaltestelle oder so und in OSM auch und die unterscheiden sich halt irgendwie n bisschen weil die irgendwie n bisschen ungenau eingetragen werden.

Testperson 2:

Mhm.

Testleiterin:

Und dann kannst du da jetzt halt im Radius einstellen, kannst sagen okay wenn die innerhalb von einem Meter voneinander sind zum Beispiel. Ist das der gleiche Punkt? Fertig.

Testperson 2:

Ah ok. Mit der Erklärung dazu, da passe ich die Werte noch Mal an.

Testleiterin:

Und würdest du sagen, so die Erklärung, würde dir quasi ein bisschen fehlen in der Anleitung?

Testperson 2:

Ich glaub ich fänd es jetzt nicht so schlecht, wenn sie drin wär. Also wenn man glaub ich bisschen mehr im Thema ist, vielleicht leuchtet das dann eher ein. Aber ja.

Testperson 2:

Intersect_data.py erfolgreich abgeschlossen. Jawohl.

Testleiterin:

Sehr schön.

Testperson 2:

Das heißt bedeutet? Keine Fehler. Prüfen Sie jedoch, jawoll, sie haben alle Schritte ausgeführt.

Testleiterin:

Toll. Fertig.

Sprecher

Mhm.

Testleiterin:

So, jetzt wo du fertig bist hab ich noch n paar Fragen an dich. Dabei gibt es keine richtigen oder falschen Antworten. Teil mir einfach deine Erfahrung mit und ich starte mit allgemeinen Fragen. Wie hast du die Bearbeitung der Schritte im Workshop insgesamt empfunden?

Testperson 2:

Hab ich verständlich empfunden bis auf diese Schwellenwertgeschichte da wir vielleicht noch ne kleine Ergänzung ganz gut ansonsten war das alles sehr verständlich und schön schrittweise erklärt worden.

Testleiterin:

Empfandest du die Ausführung der Arbeitsschritte als einfach oder als kompliziert?

Testperson 2:

Einfach

Testleiterin:

Wie fandest du die Gestaltung der Software, insbesondere die Kombination aus grafischer Eingabe und der Terminalausgabe?

Testperson 2:

Also die grafische Oberfläche fand ich sehr gut. Ob ich wenn ich das jetzt ganz allein gemacht hätte ob ich da jetzt noch mal in die Konsole geschaut hätte wüsste ich jetzt nicht wahrscheinlich schon wenn es wenn ich da so im Hintergrund irgendwie offen gehabt hätte und ich sehe da springt noch was rum dann hätte ich wahrscheinlich wieder reingeguckt aber wenn ich mir das Fenster jetzt davor geschoben hätte oder sowas weiß nicht ob mir das aufgefallen wäre. Groß gestört oder sowas hätte es mich jetzt nicht.

Testleiterin:

OK, aber wär es dir lieber, wenn dieser Terminalausgaben sozusagen auch mit in der grafischen Oberfläche ausgegeben werden würden?

Testperson 2:

Es muss nicht zwingend sein, aber vielleicht sollte in der grafischen Oberfläche einfach noch mal so n ganz kleiner Hinweis irgendwo beim bestätigen Knopf oder sowas oder in der ersten wo diese ganzen Pakete sind, ein Hinweis gezeigt werden, dass halt die ich sag mal Logs hier in der in der Konsole angezeigt werden.

Testleiterin:

Mhm. OK.

Testleiterin:

Hast du die Software jetzt einheitlich und konsistent in der Bedienung wahrgenommen?

Sprecher

Ja.

Testleiterin:

Gab es Situationen, in denen unklar war, welche Eingaben erforderlich sind? Wenn ja, welche waren das?

Testperson 2:

Da habe ich schon geagt.

Testleiterin:

Das meinst du mit den Schwellenwerten? Aber außerdem noch irgendwas?

Testperson 2:

Genau, sonst nichts.

Testleiterin:

Musstest du dir während der Nutzung viele Details merken und hat die Anleitung dabei geholfen, dass du dir weniger merken musst?

Testperson 2:

Ich musste mir wenig merken und die Anleitung hat geholfen, weil das ist ja eine ziemlich souveräne Stütze.

Testleiterin:

Was denkst du, wäre die Software auch ohne die Anwenderdokumentation nutzbar gewesen?

Testperson 2:

Puh. Puh, wahrscheinlich nicht.

Sprecher

Mhm, OK.

Testperson 2:

Also mit super viel Vorkenntnissen in den ganzen Geschichten hier. Also ich hätte jetzt auch noch nie was mit Docker zu tun, ne, also das nee, wahrscheinlich nicht.

Testleiterin:

OK. Dann hab ich jetzt n paar Fragen zu den Ausgaben, die da die Programme gegeben haben. Wusstest du während der Nutzung immer, was die Skripte gerade machen?

Testperson 2:

Die Erklärung dazu war immer da, also aus der Anleitung. Ja.

Testleiterin:

Waren die Fehlermeldungen, die du bekommen hast, verständlich und hilfreich? Und falls Nein, wo lag das Problem?

Testperson 2:

Also wenn ich sie gefunden habe in in der Konsole, also in der Mac Konsole hier in der in der in der Shell Konsole, da gibt es bessere Farbschemas, aber sonst ja.

Testleiterin:

Mhm, also nichts, was vom Programm aus jetzt kommt.

Testperson 2:

Ja, nee, nee, nee, nee. Nee, die Ausgaben sind in Ordnung, ja.

Sprecher

Okay.

Testleiterin:

Hältst du die Software für leicht erweiterbar, falls neue Aufgaben oder Anforderungen entstehen?

Testperson 2:

Ich geh mal davon aus, man kann dann einfach. Also man müsste sich wahrscheinlich erstmal in die Software reindenken, in die existenten Python Programme und ja, wahrscheinlich kann man dann einfach da noch weitere Programme hinzufügen, wenn weitere Schritte hinzukommen, oder, oder, oder. Also ich denke das ist schon möglich.

Testleiterin:

Also würdest du jetzt sagen, aufgrund dessen, dass das Halt aus mehreren einzelnen Skripten besteht, ist das ganz gut erweiterbar.

Testperson 2:

Ja, ich denke ja.

Testleiterin:

Dann hab ich jetzt noch 2 Fragen zur Anleitung an sich. Fandest du die Anleitung einheitlich und leicht verständlich?

Testperson 2:

Ja. Also eine Sache, mit der Anpassung der Overpass Anfrage. Sekunde. Da kann man zwar draufklickern, was ja auch erklärt wurde, dass das als Link gilt, aber.

Testleiterin:

Ja. Du hast da n bisschen die Orientierung verloren gehabt oder?

Testperson 2:

Ja, ich hab n bisschen Orientierung verloren. Vielleicht also ich weiß nicht, ob es jetzt optimal ist, den Punkt einfach hier drunter [direkt unter der Erklärung des Skripts vor den eigentlichen Anleitungsschritten].

Testleiterin:

Ja, Mhm OK.

Testleiterin:

Und dann hab ich noch die Frage, ob es stellen in der Anleitung gab, die dir unklar oder zu wenig detailliert erschienen.

Testperson 2:

Nee.

Testleiterin:

Hast du sonst noch irgendwas, was du sagen willst?

Testperson 2:

Nein. Danke, dass ich teilnehmen durfte.

Testleiterin:

Danke. Das du teilgenommen hast.