In Wittmann J (ed.):
Simulation in den Umwelt- und Geowissenschaften. Workshop 2020.
ASIM Mitteilung AM 173: 117–130. ISBN 978-3-8440-7579-3, Shaker, Düren 2020.

Co-Simulation zwischen AnyLogic und MATSim

Marten Meinhardt, Tim Meyran, Kevin Schoolmann,
Richard Pump, Volker Ahlers, Arne Koschel

Hochschule Hannover
Fakultät IV, Abteilung Informatik
Ricklinger Stadtweg 120, 30459 Hannover
{Richard.Pump | Volker.Ahlers | Arne.Koschel}@hs-hannover.de
{Marten.Meinhardt | Tim.Meyran | Kevin.Schoolmann}@stud.hs-hannover.de

Zusammenfassung:

Im Rahmen des BMBF-Forschungsprojekts USEfUL (Untersuchungs-, Simulations- und Evaluations-Tool für Urbane Logistik) wird in Kooperation mit der Abteilung Informatik der Fakultät IV der Hochschule Hannover eine Co-Simulation zwischen den Simulationstools AnyLogic und MATSim entwickelt. Die entwickelte Software ermöglicht die zentrale Steuerung der Co-Simulation sowie die Synchronisation und den Datenaustausch der beiden Simulationstools. MATSim (Horni et al. 2016) simuliert hierbei makroskopische Zusammenhänge in der Größenordnung einer Stadt. AnyLogic (Boshchev 2013) ermöglicht die Simulation mikroskopischer Zusammenhänge auf der Ebene von Stadtteilen. AnyLogic bietet hierbei die Möglichkeit, moderne Verkehrskonzepte abzubilden.

Zur Realisierung der Co-Simulation wurden eine Middleware sowie je eine Erweiterung der Simulationstools entwickelt. Die Kommunikation zwischen allen Modulen erfolgt über eine RPC-Schnittstelle. Da beide Simulationstools sehr unterschiedliche Ansätze verfolgen, bietet die Umsetzung der Co-Simulation einige Herausforderungen. Ein grundsätzliches Problem stellt die Synchronisation dar, da beide Tools einerseits möglichst wenig aufeinander warten sollen, die Simulationen sich aber maßgeblich gegenseitig beeinflussen. Im Beitrag wird die Umsetzung der Co-Simulation in Bezug auf Kommunikation, Synchronisation und Datenaustausch beschrieben.

1 Einleitung

Die Zunahme des innerstädtischen Lieferverkehrs stellt Städte und Kommunen vor immense Herausforderungen. Um diesen zu begegnen wurden in den letzten Jahren zahlreiche Logistikkonzepte entwickelt und oftmals in lokalen Pilotversuchen getestet. Aus der Projektinitiative Urbane Logistik Hannover (Urbane Logistik Hannover) heraus, in der Partner aus Verwaltung, Wissenschaft, Logistikdienstleistungen und Industrie zusammenarbeiten, wurde das Forschungsprojekt USEfUL gestartet (Untersuchungs-, Simulations- und Evaluationstool für Urbane Logistik). Ziel des Projekts ist die Entwicklung eines Tools zur Entscheidungsunterstützung für Kommunen, welches aufzeigt, welche Auswirkungen verschiedende Logistikkonzepte in einem gegebenen städtischen Rahmen haben.

Die Datengrundlage des Tools liefern Verkehrssimulationen, in denen die Logistikkonzepte in ausgewählten repräsentativen Stadtteilen Hannovers umgesetzt werden. Dabei kommen die Simulationstools MATSim und AnyLogic zum Einsatz. Während MATSim für die Simulation makroskopischer Zusammenhänge auf Ebene

der gesamten Stadt genutzt wird, bietet AnyLogic die Möglichkeit, neue Logistikkonzepte detailliert bis hin zur Belieferung einzelner Häuser zu simulieren. Aufgrund längerer Laufzeiten eignet sich AnyLogic im Gegensatz zu MATSim jedoch nur für die Simulation einzelner Stadtteile. Ein Grund für die längeren Laufzeiten ist die Möglichkeit, komplexe Ablaufpläne für die Agenten zu schreiben. Sowohl für die makroskopischen MATSim- als auch für die mikroskopischen AnyLogic-Modelle wird auf reale Daten aus einer Vielzahl von Quellen zurückgegriffen (Auf der Landwehr et al. 2019, Bienzeisler et al. 2020, Pump et al. 2019).

Das folgende Paper beschreibt die Ergebnisse eines laufenden Praxisprojekts im Studiengang Angewandte Informatik der Hochschule Hannover, dessen Aufgabe es ist, eine Co-Simulation zur effizienten Kombination der Simulationstools AnyLogic und MATSim zu entwickeln. Abbildung 1 zeigt den bisherigen Ablauf im Projekt USEfUL, beginnend mit der Simulation der ganzen Stadt in MATSim. Zunächst werden die Agentenpläne in MATSim durch einen genetischen Algorithmus in mehreren hundert Iterationen optimiert. Anschließend werden die Pläne konvertiert, um mit AnyLogic einen kleinen Teil der Agentenpläne für ausgewählte Stadtteile zu simulieren. Hier werden auch die Verkehrskonzepte berücksichtigt. Nach Abschluss müssen die Pläne wieder in das Ursprungsformat konvertiert werden, um mit MATSim im Anschluss erneut die ganze Stadt zu simulieren. Hier liegen die ursprünglichen Pläne nach der ersten MATSim-Simulation vor, ergänzt durch die veränderten Pläne der AnyLogic-Simulation. Das Ergebnis dieses letzten Schrittes soll ein realitätsnahes Bild der Stadt mit den Veränderungen durch die neuen Verkehrskonzepte abbilden.

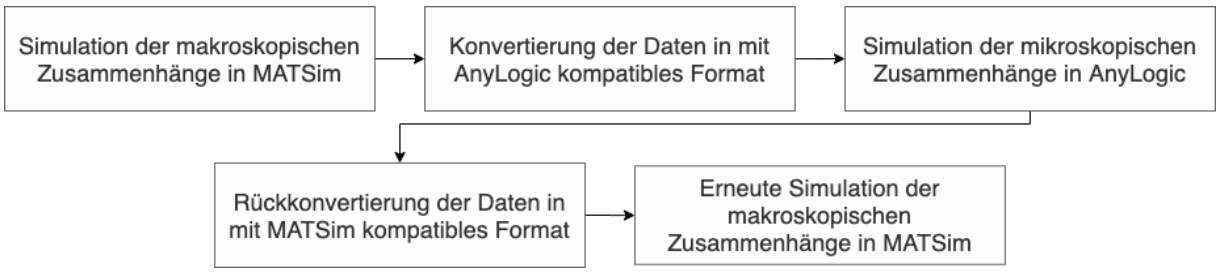

Abbildung 1: Bisheriger Ablauf der Simulation

Die Co-Simulation soll beide Anwendungen in Echtzeit Agenten austauschen lassen, um eine Ersparnis in der gesamten Simulationszeit zu erlangen und die Ergebnisse weiter an die Realität anzunähern.

2 Problemstellung: Co-Simulation mit MATSim und AnyLogic

In diesem Abschnitt wird beschrieben, wie die Simulationen innerhalb von MATSim und AnyLogic funktionieren, mit Blick auf die für die Co-Simulation relevanten Eigenschaften sowie den Aufbau und die allgemeine Funktionsweise der Co-Simulation.

2.1 MATSim

MATSim ist ein aktivitätsbasiertes, erweiterbares Multi-Agenten-Simulations-Framework (MATSim User Guide, Abschnitt 1.2; Horni et al. 2016). Es wurde zur Verkehrs-Simulation von großen Szenarien im Maßstab z.B. einer ganzen Stadt oder Region entwickelt. Verkehrsflüsse werden durch Agenten simuliert, welche einen vorgegebenen Tagesablaufplan haben. Diese Pläne werden durch eine wiederholte Simulation und einen ko-evolutionären Algorithmus vielfach optimiert, sodass die Simulation am Ende einen möglichst realistischen Verkehrsfluss abbildet. Zu Beginn der Simulation existieren die initialen Agentenpläne, auf deren Basis von der Mobsim (*mobility simulation*) ein ganzer Tag simuliert wird (Abb. 2). Jedes Ereignis, welches ein Agent generiert, wird gespeichert. Auf der Basis dieser Ereignisse werden die jeweiligen Agentenpläne in der Scoring-Phase bewertet und in der Replanning-Phase modifiziert. MATSim bietet dabei verschiedene Strategien zur Modifikation der Agenten an.

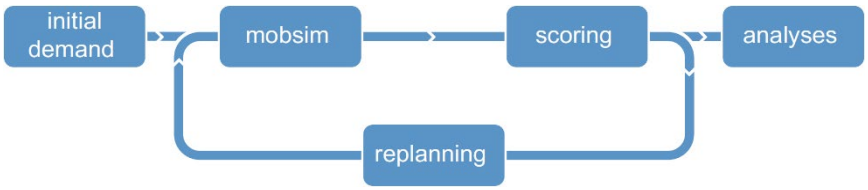

Abbildung 2: Interner Ablauf von MATSim (MATSim User Guide, Abschnitt 1.1)

MATSim ist in der Mobsim-Phase nicht an eine spezifische Simulation gebunden. Vielmehr bietet es die Auswahl aus einer Vielzahl von Simulationsmodulen unterschiedlicher Art. Zum Beispiel hat die ereignisbasierte JDEQSim gegenüber der nicht parallelisierten zeitschrittbasierten QSim eine bessere Performanz (Waraich et al. 2009), wohingegen sich die QSim besser parallelisieren lässt (Dobler & Axhausen 2011). Aus diesem Grund ist die parallelisierte QSim das derzeitige Standardsimulationsmodul von MATSim. Desweiteren gibt es die Möglichkeit, die Verkehrssimulation komplett von einem externen Tool durchführen zu lassen. Dabei muss die externe Simulation jedoch alle Anforderungen von MATSim bedienen können.

Events

Während der Mobilitätssimulation (Mobsim) werden für alle wichtigen Ereignisse interne Events erzeugt. Theoretisch lässt sich die gesamte Simulation im Nachhinein anhand dieser Events erneut abspielen. Es ist möglich, Event-Handler für bestimmte Events zu registrieren, um auf bestimmte Ereignisse zu horchen. Durch die Offenheit von MATSim hinsichtlich der Erweiterung können sogar neue Eventtypen erstellt werden, für welche sich dann ganz neue Event-Handler registrieren lassen.

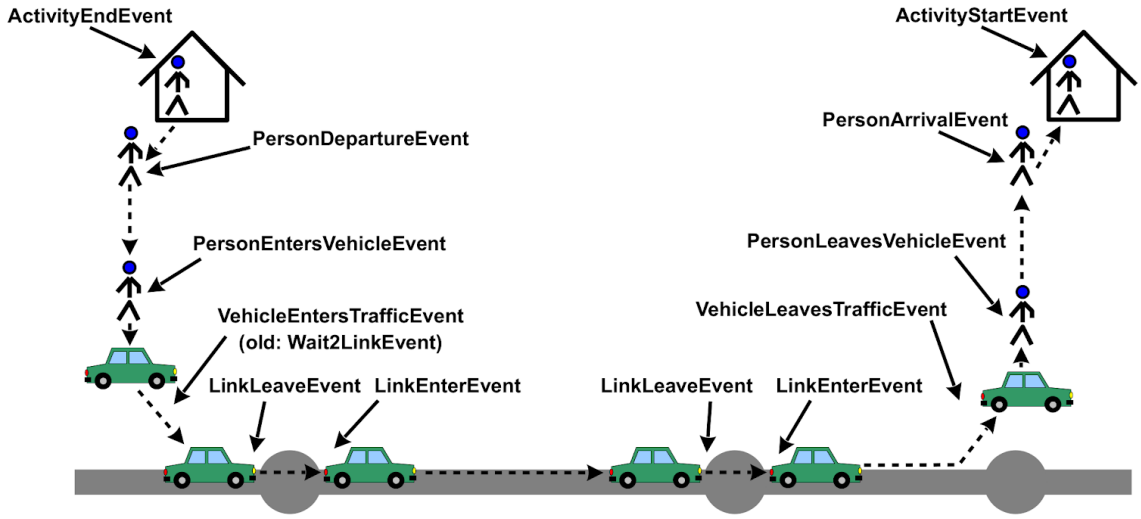

Abbildung 3: Eventübersicht MATSim (MATSim User Guide, Abschnitt 7.1.2)

QSim

Die QSim ist eine Zeitschritt- und Warteschlangen-basierte Implementierung der Mobilitätssimulation, welche die Bewegung von Agenten auf Basis ihrere Pläne simuliert. Das Straßennetz in der QSim wird durch Knoten und Verbindungen (Links) zwischen diesen dargestellt. Die QSim unterscheidet im Allgemeinen zwischen verschiedenen Arten der Fortbewegung und führt eine Verkehrsflusssimulation nur für Fahrzeuge durch. Fahrräder und Fußgänger werden unter Berücksichtigung der Distanz zweier Knoten lediglich zwischen ihnen teleportiert. Fahrzeuge befinden sich jedoch immer auf einem Link, welcher in der QSim eine Implementierung des QLinkI Interfaces ist. Links bestehen wiederum aus mindestens einer Lane (Fahrspur) und sind gerichtet. Eine Fahrspur ist als Warteschlange nach dem FIFO-Prinzip (*first in first out*) implementiert. Dies bedeutet, dass ein Fahrzeug, welches als erstes in eine Spur einfährt, diese auch als erstes wieder verlässt. In der QSim werden Fahrzeuge von den Knoten ausgehend weiterbewegt. Jeder Knoten prüft zu jedem Simulationsschritt seine eingehenden Fahrspuren auf Fahrzeuge, welche über den Knoten hinweg zur nächsten Spur bewegt werden müssen. Dabei gibt es eine Reihe von Kriterien, z.B. die Verweildauer auf der Spur, anhand welcher geprüft wird, ob ein Agent unter Berücksichtigung der erlaubten Geschwindigkeit die Spur passiert haben kann.

Scoring

Beim Scoring wird der aktuelle Plan eines Agenten hinsichtlich verschiedener Kriterien bewertet. Das Scoring selbst kann bei der Konfiguration der Simulation eingestellt werden. Da es eine subjektive Entscheidung ist, ob ein Plan tatsächlich gut ist, kann es keine universelle Funktion geben, welche nur gute Pläne mit einer hohen Punktzahl bewertet. Dieses Empfinden kann sich von Person zu Person unterscheiden: „Some may prefer a congested car trip, others may prefer a crowded, but affordable, trip by public transit, while others may prefer using the bicycle, even in bad weather“ (MATSim User Guide, Abschnitt 5.1).

Wichtig ist, dass das Scoring auf Informationen beruht, welche in Form von Events vermittelt werden. Somit ist es äußerst wichtig, dass alle Ereignisse, die ein Agent auslöst, auch in Form von Events an alle Event-Handler übermittelt werden.

Replanning

Im Gegensatz zu einem evolutionären Algorithmus basiert MATSim auf einem koevolutionärem Algorithmus, wobei statt einem Maximum für die ganze Simulation für jeden Agenten der optimale Plan gesucht wird (MATSim User Guide, Abschnitt 1.4). Da jeder Agent vom Verhalten der anderen Agenten abhängt, wird durch die Veränderung der Pläne aller Agenten im Laufe der Iterationen ein optimaler Plan für jeden Agenten gefunden. Hierdurch ist es MATSim möglich, einen realen Verkehrsfluss relativ präzise vorherzusagen, da die Suche nach einem optimalen Plan das ist, was jeder Verkehrsteilnehmer jeden Tag auf ein Neues in der Realität tut.

MATSim nutzt verschiedene Strategien, um den optimalen Plan eines Agenten zu finden. Dabei kann zwischen dem Erzeugen und Entfernen von Plänen und der Selektion alter Pläne unterschieden werden. Beim Erzeugen neuer Pläne werden entweder die Zeit, die Route oder das Transportmittel verändert. Bei der Selektion alter Pläne wird zufällig ein alter Plan ausgewählt. Die Wahrscheinlichkeit, mit welcher jede Strategie gewählt wird, kann konfiguriert werden.

Erweiterung

MATSim ist eine Open Source Java-Bibliothek, deren Quellcode auf GitHub eingesehen werden kann (MATSim GitHub Repository). MATSim nutzt das Dependency Injection Framework Guice. Auf dessen Basis gibt es vorgesehene Methoden zum Installieren eigener Implementierungen, wodurch MATSim einen einheitlichen Mechanismus zur Erweiterung bietet. So können leicht vorhandene Implementierungen verändert oder neue Funktionalitäten hinzugefügt werden.

2.2 AnyLogic

AnyLogic ermöglicht die detaillierte Simulation von Agenten in Straßenverkehrsnetzen welche auf GIS-Karten (Geografische Informationssysteme) basieren (Boshchev 2013). Als Datenquelle werden einerseits lokale Netzwerkdateien, aber auch eine Vielzahl an Geodaten-Servern unterstützt, welche unter anderem von AnyLogic selbst gehostet werden. Das Routing von Agenten kann entweder von Routing-Servern oder einem lokal erstellten Routing-Graphen übernommen werden. Dieser muss jedoch im Vorherein beim Erstellen des Modells einmalig für ein konkretes Straßennetz berechnet werden. Das Verhalten von Agenten kann durch Zustandsgraphen modelliert werden. So ist es möglich, Agenten wie z.B. Fahrradfahrer und Autofahrer mit verschiedenen Verhaltensmustern zu modellieren. AnyLogic unterstützt unter anderem die Simulation in diskreten Zeitschritten, wobei die Länge der Zeitschritte festgelegt werden kann.

Erweiterbarkeit

AnyLogic selbst sowie alle weiteren Bibliotheken sind in Java geschrieben. Die grafische Oberfläche von AnyLogic basiert auf der IDE Eclipse und erweitert diese um Dialoge zum Entwurf von Modellen, ohne selbst Code schreiben zu müssen. Der Java-Code wird letztendlich im Hintergrund automatisch geschrieben und beim Starten der Simulation wie jedes herkömmliche Java-Programm kompiliert und ausgeführt.

Für erfahrene Java-Entwickler bietet ein AnyLogic-Modell eine Menge von Hook-Methoden an, in welchen eigener Code eingefügt werden kann. Dieser kann entweder über die dafür vorgesehen Felder in der AnyLogic IDE oder in den erzeugten Java-Dateien selbst eingefügt werden. Außerdem können zusätzliche Java-Bibliotheken eingebunden werden. Modelle können als ausführbare JAR-Dateien exportiert werden.

Steuerungsdiagramme

Steuerungsdiagramme definieren das Verhalten der Agenten in der Simulation. Sie werden über eine grafische Oberfläche aufgebaut. Die Bausteine entstammen der Process Modeling Library von AnyLogic. Zudem hat jeder Block die Möglichkeit, Code auszuführen, wenn der Block während der Simulation betreten oder verlassen wird, sowie kurz vor dem Verlassen des Blockes. Alternativ können klassische Zustandsdiagramme zur Agentensteuerung verwendet werden, diese verfügen über eine Entry Action und eine Exit Action. Beide Varianten können zum Steuern verwendet werden; wir verwenden die Process Modeling Language.

Das Zustandsdiagramm der Co-Simulation (Abb. 4) startet mit einem Enterblock. Jeder Agent hat eine Route, der er folgen soll. Wenn noch weitere Knoten in der Route vorhanden sind, wird der Agent dorthin bewegt. Es wird geprüft, ob der nächste Knoten außerhalb des definierten AnyLogic-Bereiches liegt: Falls ja, dann wird der Agent an MATSim übertragen. Desweiteren wird berücksichtigt, ob der Agent seine Route innerhalb von AnyLogic beendet. Delay-Blöcke geben der Simulation die Möglichkeit einzelne Werte zu setzen. Ohne die Delay-Blöcke sind Phänomene aufgetreten, in denen AnyLogic die bedingten Verzweigungen nicht korrekt auswerten konnte.

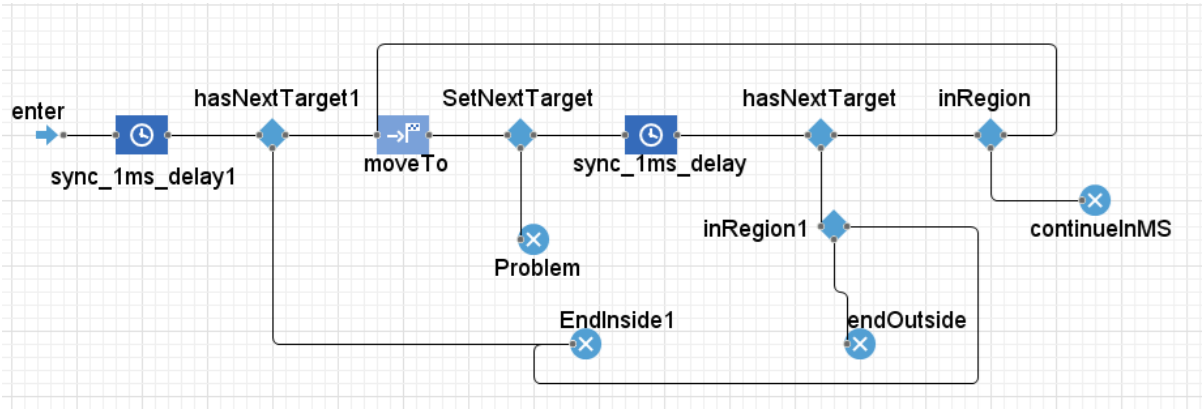

Abbildung 4: Steuerungsdiagramm der Co-Simulation

2.3 Aufbau der Co-Simulation

In diesem Abschnitt wird auf die Architektur der Co-Simulation, den allgemeinen Ablauf und die einzelnen Module der Anwendung eingegangen. Die Architektur der Co-Simulation besteht aus drei Modulen (Abb. 5).

Die Simulationstools sind jeweils gekapselt und durch eine Fassade an den Rest der Anwendung angebunden. Innerhalb des Moduls existieren Klassen zur Konvertierung der einkommenden Daten. Die Daten werden in einem möglichst einfachen Format übermittelt, die Simulationstools brauchen die Daten aber in verschiedenen Formaten. Dies wird in Abschnitt 4 genauer betrachtet. Das Eventhandling sieht für die Simulationstools unterschiedlich aus. Zu Gunsten der Übertragung erhält MATSim verschiedene neue Events, geerbt von existenten Event-Klassen. Desweiteren werden Event-Handler angelegt, die speziell auf CoSim-Events reagieren.

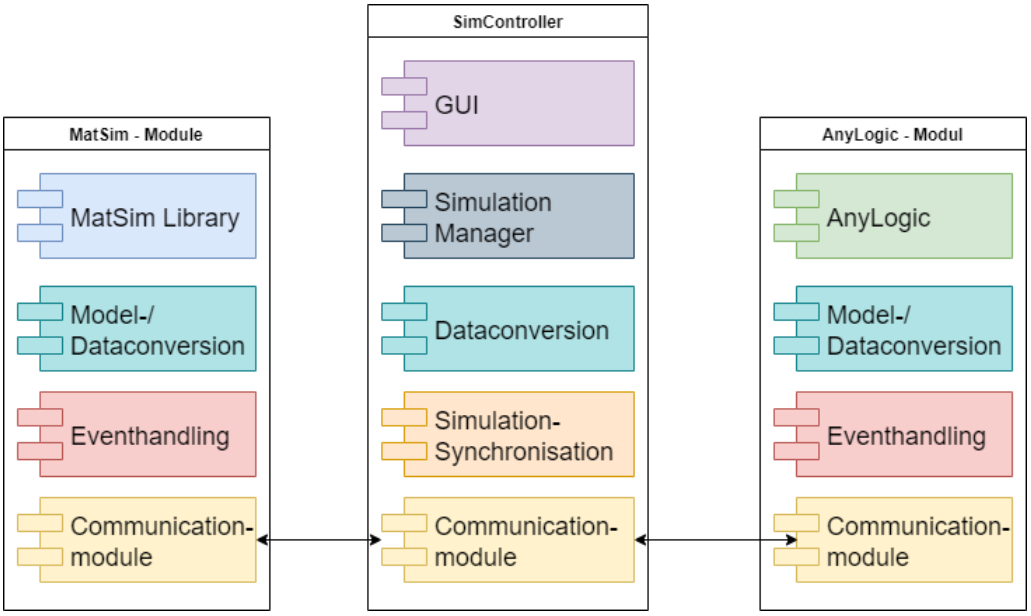

Abbildung 5: Architektur der Co-Simulation

Auf Seiten von AnyLogic passiert die Eventverarbeitung in einem Steuerungsdiagramm (Abb. 4), das fest codiert sein muss. Diese Diagramme bieten die Möglichkeit, neue Verkehrskonzepte zu implementieren (AnyLogic Help).

Alle Module sind über eine einheitliche Kommunikationsschnittstelle verbunden, für die verschiedene Technologien evaluiert wurden. Für die Implementierung wurde RMI genutzt. Details hierzu finden sich in Abschnitt 3.

Das zentrale Modul bildet die Middleware, die die Rolle des SimControllers übernimmt. Hier existiert eine einfache GUI zum manuellen Pausieren der Co-Simulation und für Debugging-Ausgaben. Die Hauptaufgabe der Middleware ist allerdings die Synchronisation der Co-Simulationen. Hier wurden im Laufe des Projekts mehrere Ansätze evaluiert. Der implementierte Ansatz wird im Abschnitt 5 beschrieben.

Der allgemeine Ablauf der Co-Simulation startet bei MATSim. Hier liegt die Grundpopulation der Simulation. Die Agenten folgen ihren Tagesplänen. Wenn die

Route in einen Bereich führt, der im Straßennetz als *external* makiert ist, dann sorgen Event-Handler für eine Übertragung zu AnyLogic. Hier wird der Agent in ein Vehikel gekapselt und an die Middleware übertragen. Der Agent erhält außerdem die Route, der innerhalb des AnyLogic-Bereichs gefolgt werden soll. Die Middleware puffert die übertragenen Daten und platziert die Agenten in der AnyLogic-Instanz. Auf der Seite von MATSim bleibt der Agent existent und seine Position wird bei jedem erreichten Wegpunkt in AnyLogic aktualisiert, damit die Auslastung der Straßen korrekt gespeichert wird, bis der Agent zurückübertragen wird. Innerhalb von AnyLogic ist das Verhalten des Agenten über ein Steuerungsdiagramm definiert. Die Rückrichtung funktioniert konzeptionell gleich. Die übertragenen Daten unterscheiden sich allerdings. Von MATSim nach AnyLogic wird die Route des Agenten übertragen. Von AnyLogic nach MATSim wird die Zeit übertragen, die der Agent in AnyLogic verbracht hat. Wenn der Agent zurückkehrt wird die Position aktualisiert und die Zeit, die der Agent in der fremden Instanz verbracht hat, für das Scoring des genetischen Algorithmus vorgemerkt.

Sollte eine Agent im durch AnyLogic simulierten externen Bereich enden, dann wird der Agent am Ende der Iteration zu MATSim übertragen, damit das Scoring stattfinden kann. Die Anzahl der Iterationen kann über die Konfiguration des Szenarios eingestellt werden.

3 Kommunikation

Um Agentendaten zwischen MATSim, der Middleware und AnyLogic zu übertragen, und für die zentrale Simulationssteuerung wird eine Kommunikationsschnittstelle benötigt. Diese muss eine bidirektionale Kommunikation ermöglichen, da Daten in beide Richtungen versendet werden müssen.

3.1 Evaluation der Kommunikationsschnittstellen

Die Anforderungen an die Schnittstelle beinhalten Zuverlässigkeit und Schnelligkeit. Das Risiko eines Datenverlustes beim Übertragen soll minimal sein. Da von AnyLogic und MATSim jeweils große zu übertragene Datenmengen zu erwarten sind, muss die Übertragung außerdem sehr schnell sein. Im Rahmen des Projekts wurden verschiedene Technologien untersucht und auf ihre Eignung geprüft. Dazu zählen Apache Kafka, Apache CXF, WebSocket und Java RMI. Im Folgenden werden die untersuchten Technologien kurz vorgestellt.

Apache Kafka ist eine Software, die der Erstellung und Verarbeitung von Datenströmen dient. Kafka kann genutzt werden, um Daten in Echtzeit über Pipelines zwischen verschiedenen Anwendungen auszutauschen. Kafka unterscheidet hierbei zwischen Produzenten (Producers) und Konsumenten (Consumers) der Datenströme. Kafka wird von der Apache Foundation entwickelt (Apache Kafka).

Apache CXF ist ein Web Service Framework. CXF unterstützt eine Reihe an Web Service Protokollen, wie zum Beispiel SOAP, REST und HTTP. CXF ist hierbei auf

eine einfache Benutzbarkeit ausgelegt. Die erstellten Webservices laufen anschließend in einem Webcontainer wie Apache Tomcat (Apache CXF).

WebSocket ist ein auf TCP basierendes Netzwerkprotokoll. Der Vorteil gegenüber HTTP ist, dass der Client den Server nicht ständig anfragen muss. Der Client muss nur die Verbindung zum Server eröffnen. Der Server kann dann über die Verbindung jederzeit Daten an den Client senden (WebSocket).

JavaRMI ist die Java-eigene Implementierung eines entfernten Methodenaufrufs. RMI erlaubt es Methoden von Java-Objekten auszuführen, welche sich in einer anderen JVM (Java Virtual Machine) befinden. RMI nutzt für die Kommunikation zwischen den JVM-Instanzen ein auf TCP/IP basierendes Protokoll. Da RMI in der Java-Standardbibliothek bereits enthalten ist, müssen keine weiteren Bibliotheken oder Frameworks für die Nutzung installiert werden. Außerdem muss der Empfänger bei RMI keine Nachrichten interpretieren, da die Methoden direkt ausgeführt werden (Java RMI).

Für die Realisierung der Co-Simulation wurde sich letztendlich für Java RMI entschieden. RMI ist bereits nativ in Java vorhanden und die Umsetzung ist simpler als bei den anderen möglichen Technologien.

3.2 Umsetzung

Umgesetzt wurde die Kommunikationsschnittstelle als System aus verschiedenen Klassen und Interfaces (Abb. 6). Jeder Teil der CoSim bestimmt über seine Interfaces die Methoden, welche er zum Ausführen anbieten möchte und welche er auf dem Server ausführen darf. Der Server implementiert sämtliche Servermethoden für AnyLogic als auch für MATSim. Die Methoden, die MATSim und AnyLogic auf dem Server aufrufen können, sind durch die zwei Interfaces MatSimRemoteServer und AnyLogicRemoteServer voneinander getrennt. Der Server kann über seine Referenzen auf die Objekte vom Typ AnyLogicRemoteClient und MatSimRemoteClient auf die beiden Simulationen zugreifen. Die tatsächliche Implementierung der Client-Interfaces befindet sich auf Seiten von AnyLogic und MATSim. Durch dieses modulare Klassenmodell ergibt sich eine lose Kopplung der Komponenten. Dies führt auch zu einer erleichterten Änderbarkeit, da eine Änderung in einer Klasse sich nicht auf Änderung in anderen Klassen auswirken kann. Allerdings entsteht durch die vielen Klassen und Interfaces eine hohe Unübersichtlichkeit. Ebenso entsteht so ein leichter Overhead in der Programmierung.

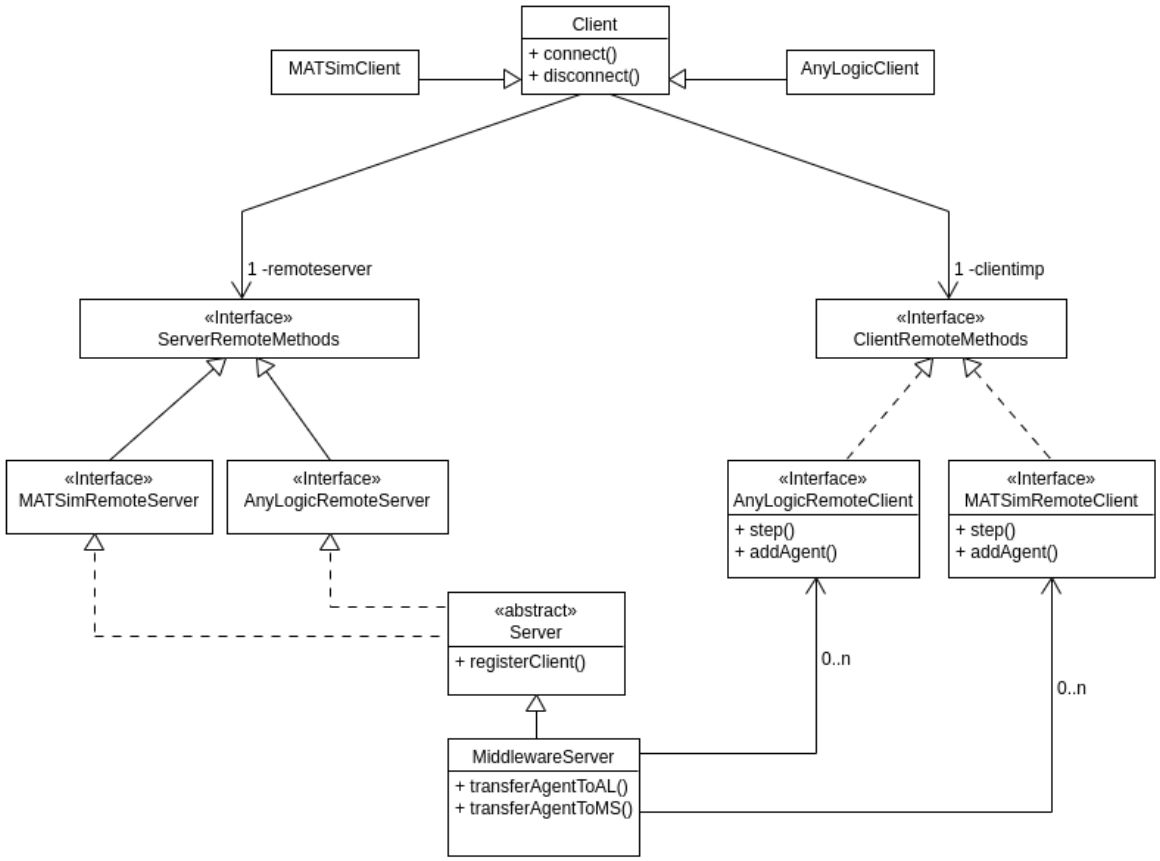

Abbildung 6: Klassendiagramm der Kommunikationsschnittstelle (vereinfacht)

4 Datentransfer

Beim Transfer eines Agenten von MATSim zu AnyLogic wird eine Anfrage für einen Transfer mit der ID, der aktuellen Position sowie der Zielposition des Agenten an AnyLogic übermittelt. Bevor der Agent tatsächlich von der Fahrbahn in MATSim entfernt wird, muss geprüft werden, ob die entsprechende Fahrbahn in AnyLogic überhaupt Platz für den Agenten bietet. Erst nachdem AnyLogic den Transfer akzeptiert hat, wird der Agent von der entsprechenden Fahrbahn in MATSim entfernt und in AnyLogic eingefügt. Beim Rücktransfer von AnyLogic zu MATSim muss die gleiche Prüfung der Fahrbahn in MATSim erfolgen.

Nachdem ein Agent zu AnyLogic übertragen wurde, wird eine Route zum Ziel berechnet. Daraufhin beginnt der Agent sich in Richtung Ziel zu bewegen. AnyLogic muss dabei jeden erreichten Knoten an MATSim melden, da auch die Aktivitäten innerhalb der AnyLogic-Region vom Scoring berücksichtigt werden müssen. Konkret müssen alle internen Events in MATSim erzeugt werden, welche normalerweise von der QSim erzeugt werden würden.

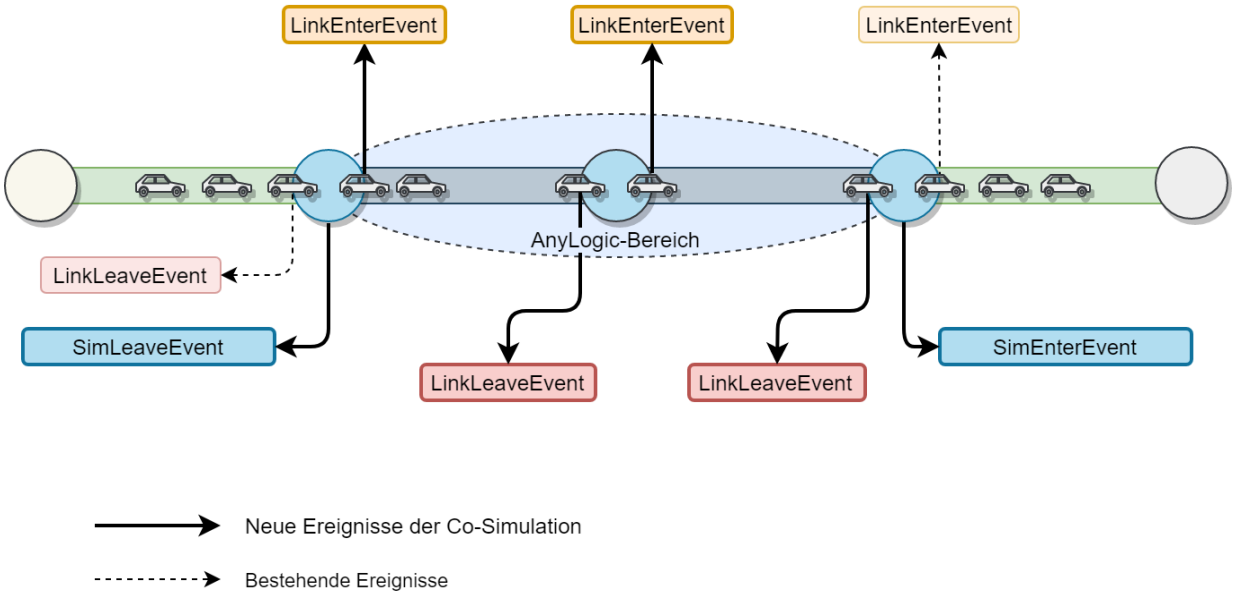

Abbildung 7: Events in der Co-Simulation

5 Synchronisation

Bei der Durchführung der Co-Simulation ist es wichtig, dass beide Simulationen dem gleichen Zeitstrahl folgen. Wäre dies nicht der Fall, wäre es zum Beispiel möglich, dass Agenten beim Transfer von einer Simulation zur anderen in die Vergangenheit geschickt werden. Dies würde zu einer Verfälschung der Simulationsergebnisse führen. Um die Synchronisation der beiden Simulationen zu gewährleisten, wurden zwei Ansätze ausgearbeitet. Bei beiden übernimmt die Middleware die zeitliche Kontrolle über beide Simulationen.

Shared Event-Queue

Bei diesem Ansatz verwaltet die Middleware eine gemeinsame Queue (Warteschlange) aus Events. MATSim und AnyLogic sind beide Event-basiert. Die Middleware hat bei diesem Ansatz die Aufgabe, die von MATSim und AnyLogic produzierten Events über die Kommunikationsschnittstelle abzufangen und intern in einer Queue zu speichern. Jedes Event besitzt hierbei einen Zeitstempel, der den Zeitpunkt der Erzeugung darstellt. Dem Zeitstempel nach übergibt die Middleware, wie in Abbildung 8 dargestellt, die in der Queue enthaltenen Events wieder an das zuständige Simulationstool zurück. Diese Simulation verarbeitet dann das Event. Durch diesen Ablauf wird ein synchroner Ablauf der Simulation in chronologischer Reihenfolge der Events sichergestellt. Ein Nachteil dieses Ansatzes ist, dass die gesamte Simulation verlangsamt wird, da stets nur ein Event verarbeitet wird. Falls die Verarbeitung eines Events länger dauert, muss die andere Simulation erst auf deren Beendigung warten.

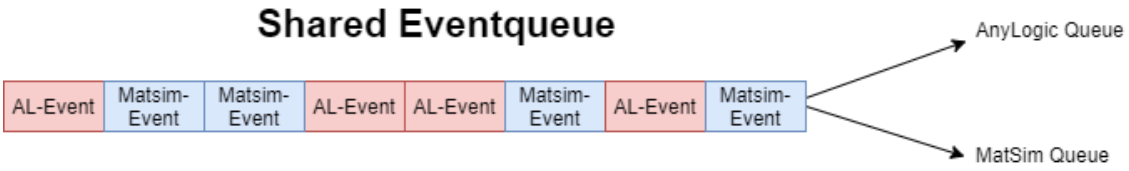

Abbildung 8: Schematische Darstellung der Shared Eventqueue

Tick-basierte Eventqueue

Der zweite Ansatz basiert darauf, dass die Middleware die beiden Simulationen zeitbasiert steuert. Hierbei bringt die Middleware die beiden Simulationen jeweils dazu, einen elementaren Simulationsschritt (Tick) auszuführen. Voraussetzung hierfür ist, dass beide Simulationen das gleiche Zeitintervall in einem Tick simulieren. Die Middleware erwartet anschließend eine Bestätigung, dass beide Simulationen ihren Simulationsschritt erfolgreich durchgeführt haben. Erst anschließend ruft sie erneut beide Simulationstools auf, um einen weiteren Schritt auszuführen (Abb. 9).

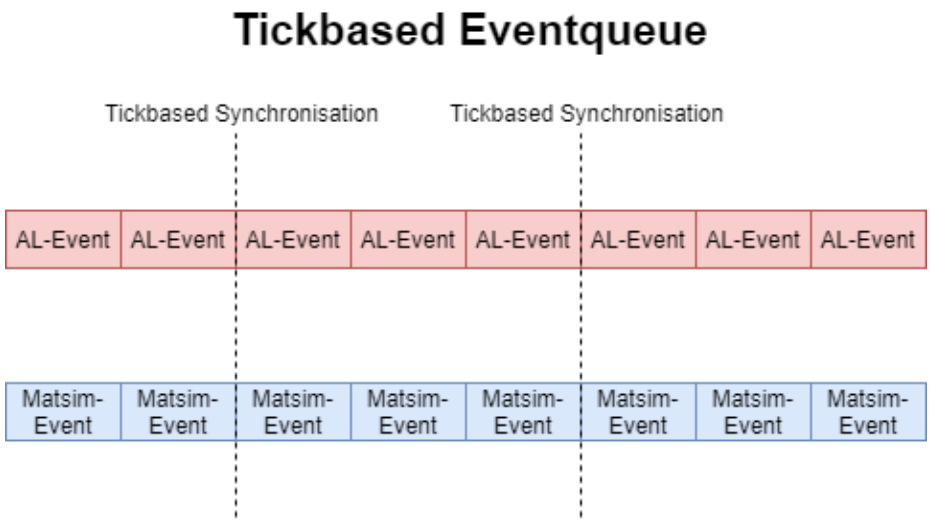

Abbildung 9: Schematische Darstellung der Tick-basierten Eventqueue

Umsetzung

Bei der Umsetzung wurde sich schließlich für den zweiten Ansatz einer Tick-basierten Synchronisation entschieden. Der Grund hierfür liegt bei MATSim: Im Rahmen des Projekts wurde kein Weg gefunden, Events aus MATSim vor der Verarbeitung zu extrahieren.

In Abbildung 10 ist der schematische Ablauf der Synchronisation aus Sicht der Middleware in einem Sequenzdiagramm dargestellt. Innerhalb einer Schleife ruft die Middleware über den eigenen Server die Methode zur Durchführung eines Schrittes auf den registrierten MATSim- und AnyLogic-Clients auf. Diese melden sich nach der erfolgreichen Ausführung am Server zurück, welcher sich anschließend bei der Middleware zurückmeldet. Die Middleware beginnt anschließend mit der nächsten Schleifeniteration.

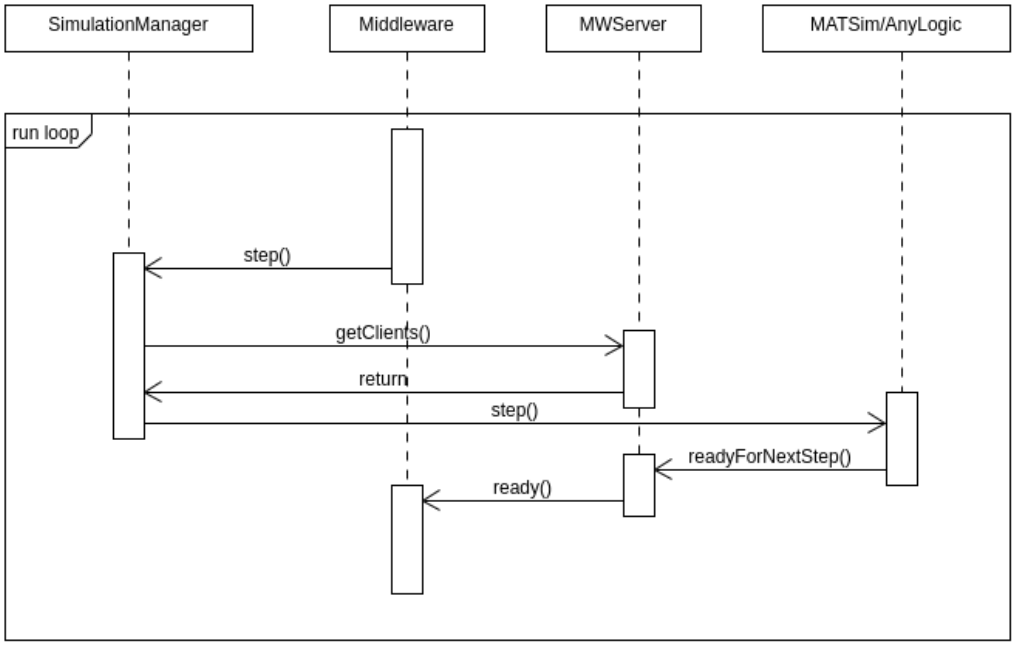

Abbildung 10: Ablauf der Synchronisation

6 Zusammenfassung

Die drei wesentlichen technischen Hürden beim Aufbau der Co-Simulation sind die Kommunikation, der Datentransfer und die Synchronisation. Die Kommunikation ist in RMI implementiert, da RMI nativ in Java zur Verfügung steht, Methoden direkt aufgerufen werden können und die allgemeine Implementierung relativ simpel ist. Auf der Basis von RMI wurden Client- und Server-Interfaces erstellt, um die Kommunikation soweit wie möglich zu entkoppeln. Der Datentransfer umfasst eine ID, die aktuelle Position und die Zielposition. Übertragene Agenten müssen aktiv angenommen werden, um Straßenkapazitäten der anderen Simulation zu berücksichtigen. Insgesamt werden zugunsten der Performanz nur die nötigsten Daten übertragen. Zur Synchronisation wurde eine Tick-basierte Synchronisation implementiert. Aufgrund von technischen Eigenheiten von MATSim war es nicht möglich, Events zu extrahieren, so dass nur eine Steuerung von außen genutzt werden konnte.

Zukünftig soll die Co-Simulation über eine webbasierte Schnittstelle angesteuert werden können, um Simulationen zu starten oder in eine Warteschlange einzureihen. Die wartenden Simulationen sollen über einen Jobmanager verwaltet werden, der Simulationen startet, sobald Rechenkapazitäten zur Verfügung stehen. Außerdem soll das Routing innerhalb von AnyLogic nicht mehr komplett von MATSim vorgegeben werden, um die Optionen für die Implementierung von Verkehrskonzepten zu erweitern.

Danksagung

Wir danken den Beteiligten des Praxisprojekts Torsten Bergmann, Nicolai Böttger, Dominic Dörries, Norman Göhl, Jannes Hachmer, Norbert Piotrkowicz, Kai Schauerte, Steven Streller, Florian Stronk, Benjamin Walter und Danny Zawadzki sowie den Mitarbeiterinnen und Mitarbeitern des Forschungsprojekts USEfUL für Ihre Unterstützung. Das Projekt USEfUL wird finanziell gefördert durch das Bundesministerium für Bildung und Forschung (BMBF, Fkz. 03SF0547D).

Literaturangaben

AnyLogic: Simulation Modeling Software Tools. https://www.anylogic.com/, letzter Zugriff 14.04.2020.

AnyLogic Help. https://help.anylogic.com/index.jsp, letzter Zugriff 14.04.2020.

Apache CXF. https://cxf.apache.org/, letzter Zugriff 20.04.2020.

Apache Kafka. https://kafka.apache.org/, letzter Zugriff 20.04.2020.

Auf der Landwehr, M., Trott, M., von Viebahn, C. (2019): E-Grocery in terms of sustainability – simulating the environmental impact of grocery shopping for an urban area in Hanover. In Putz, M., Schlegel, A. (Hg.): Simulation in Produktion und Logistik 2019, 87–96.Verlag Wissenschaftliche Scripten, Auerbach.

Bienzeisler, L., Lelke, T., Wage, O., Thiel, F., Friedrich, B. (2020): Development of an agent-based transport model for the city of Hanover using empirical mobility data and data fusion. Transportation Research Procedia 47, 99–106. https://doi.org/10.1016/j.trpro.2020.03.073, letzter Zugriff 15.05.2020.

Boshchev, A. (2013): The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6. AnyLogic North America, Chicago.

Dobler, C., Axhausen, K.W. (2011): Design and implementation of a parallel queue-based traffic flow simulation. Arbeitsberichte Verkehrs- und Raumplanung 732, IVT, ETH Zürich. https://pdfs.semanticscholar.org/4f67/f97ac57c47212a840a50dbb20e9a6c22fc3b.pdf, letzter Zugriff 27.04.2020.

Horni, A., Nagel, K. Axhausen, K.W. (Hg., 2016): The Multi-Agent Transport Simulation MATSim. Ubiquity Press, London. https://doi.org/10.5334/baw, letzter Zugriff 27.04.2020.

Java RMI. https://docs.oracle.com/javase/tutorial/rmi/, letzter Zugriff 20.04.2020.

MATSim: Multi-Agent Transport Simulation. https://www.matsim.org/, letzter Zugriff 05.05.2020.

MATSim GitHub Repository. https://github.com/matsim-org/matsim-libs, letzter Zugriff 05.05.2020.

MATSim User Guide (2020). https://www.matsim.org/docs/userguide/, letzter Zugriff 05.05.2020.

Pump, R., Baumann, M., Bellok, J.T., Ahlers, V., Koschel, A. (2019): Kombinierte Simulation logistikrelevanter Verkehrszusammenhänge. In Wittmann, J. (Hg.): Simulation in Umwelt- und Geowissenschaften. Workshop Kassel 2019. ASIM Mitteilung AM 171, 55–66. Shaker, Düren.

Urbane Logistik Hannover. https://www.hannover.de/Urbane-Logistik-Hannover/, letzter Zugriff 14.04.2020.

Waraich, R.A., Charypar, D., Balmer, M., Axhausen, K.W. (2009): Performance improvements for large scale traffic simulation. In: Proceedings of the 9th Swiss Transport Research Conference (STRC 2009). https://doi.org/10.3929/ethz-a-005864320, letzter Zugriff 27.04.2020.

WebSocket. https://tools.ietf.org/html/rfc6455, letzter Zugriff 20.04.2020.