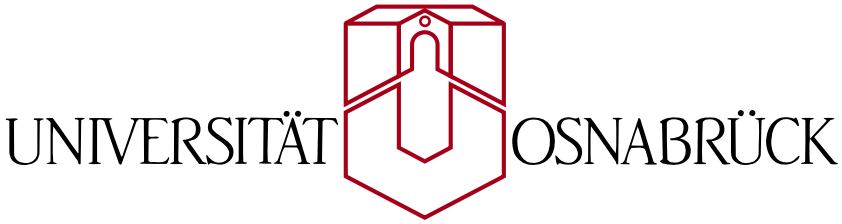


# Projekt OScillate: Ein Modell der Osnabrücker Buslinien 11 und 21

Jan Philipp Vogtherr & Timmy Schüller

Universität Osnabrück

Im Rahmen der Veranstaltung „Regelbasierte Modellierung“

Wintersemester 2013/2014

27. Februar 2014

# Inhaltsverzeichnis

# 1 Motivation

Als Student der Universität oder Hochschule in Osnabrück steht man täglich vor einer schwierigen Entscheidung, wenn man den Campus am Westerberg erreichen möchte. Der Weg dorthin beinhaltet für viele Studenten eine Busfahrt vom zentralen Verkehrsknotenpunkt, dem Neumarkt, zum Zielort, wobei es dort zwei verschiedene Buslinien gibt, die jeweils Vor- und Nachteile mit sich bringen. Die Buslinie 21 fährt einen kleinen Umweg, sodass sie langsamer ist, als die konkurrierende Buslinie 11. Außerdem fährt die 21 mit einem 15-Minuten Takt weniger oft als die 11 mit einem 10-Minuten Takt. Allerdings hält die 11 nicht direkt an der Haltestelle vor der Uni, sondern zieht immer einen Fußweg von bis zu fünf Minuten mit sich. Die zu Beginn angesprochene Entscheidung ist also nun: „Fahre ich heute mit der 11, oder der 21?“

Das im folgende vorgestellte Projekt OScillate beschäftigt sich mit ebendieser Problematik. Im Rahmen der Veranstaltung „Regelbasierte Modellierung“wurde ein agentenbasiertes Modell erstellt, das die Entscheidungsdynamik dieser Fragestellung nachstellen soll.

In Abschnitt 2 werden zunächst grundlegende Designentscheidungen erläutert. Darauf basierend geht es in Abschnitt 3 um die Umsetzung der Ideen, direkt gefolgt von einer Auswertung der daraus resultierenden Ergebnisse in Abschnitt 4. Abschließend diskutiert Abschnitt 5 den Verlauf des Projektes, sowie die erhaltenen Ergebnisse.

# 2 Design

Der Modellzweck ist explorativ und lässt sich in folgender Leitfrage zusammenfassen: „Welche Faktoren haben einen Einfluss auf die Wahl der Buslinie?“

Das Modell ist insgesamt sehr auf die Situation in der realen Welt zugeschnitten und lässt sich kaum auf andere übertragen. Außerdem verwendet das Modell sowohl eine sehr hohe zeitliche, als auch räumliche Auflösung, daher kann der Abstraktionsgrad als realistisch eingestuft werden. Die komplexe Entscheidung der homogenen Agenten, der Studenten, trägt ebenfalls dazu bei und führt zu einer hohen individuellen Komplexität. Funktionale Heterogenität gibt es in direktem Sinne nicht, weil es außer den Studenten keine weiteren Akteure gibt. Die Interaktionen der Studenten untereinander beschränken sich auf simple Beobachtung der Handlungen der anderen Akteure, echte Interaktionen gibt es nicht.

# 3 Implementierung

Die zentralen Elemente des Modells sind Studenten als Agenten des Systems und die Busverbindung, die die Umwelt darstellt, in der die Agenten selbstständig über ihr Handeln bestimmen können. Zunächst werden die verwendeten Mechanismen beider Elemente respektiv erklärt. Anschließend wird die Implementierung im Detail besprochen.

## 3.1 Agent: Student

Ein Student soll sich täglich selbstständig von seiner Wohnung zum Neumarkt bewegen und sich dort für eine Buslinie entscheiden, mit der er zur Universität fährt. Dort angekommen soll er für die Dauer einiger Vorlesungen auf dem Campus bleiben und anschließend von der Bushaltestelle an der Universität zurück zum Neumarkt fahren, wobei er auch in diesem Fall eine Buslinie auswählen muss. Zur Vereinfachung wurde hier die zwei zuvor angesprochenen Haltestellen der beiden Linien zusammengefasst. Das Ziel des Studenten ist die Auswahl einer Buslinie unter Berücksichtigung seiner persönlichen Eigenschaften, mit der er eine möglichst kurze Reisedauer und wenig Wartezeit an den Haltestellen erreicht.

Die Studenten in dem Modell unterscheiden sich hinsichtlich des Zeitpunktes ihres Erscheinens und ihrer Aufenthaltsdauer in der Universität. So kommen einige Studenten besonders früh und hören viele Vorlesungen, während andere Studenten erst spät kommen und nur wenige Vorlesungen hören, bevor sie wieder nach Hause fahren.

Jeder Student hat eine bevorzugte Buslinie, die sich über die Zeit ändern kann. Nach jeder Busfahrt evaluiert der Student seine Entscheidung und merkt sich das Ergebnis seiner Überlegungen, sodass er die Zufriedenheit des Vortages bei jeder Entscheidung aufs neue mit einbeziehen kann. Im Modell wird die bevorzugte Buslinie als Parameter des Studenten dargestellt. Zu Beginn sind die bevorzugten Buslinien unter allen Studenten zufällig verteilt - mit Ausnahme der Erstsemesterstudenten, deren bevorzugte Buslinie zu Beginn stets die Linie 21 ist. Die Anzahl der Erstsemesterstudenten kann als Simulationsparameter variiert werden.

Bei der Entscheidungsfindung der Buslinie kommt (bisher nur) ein Parameter zum Tragen, der im Folgenden als der „Sozialfaktor“ bezeichnet wird. Dieser Parameter ist unter den Studenten zufällig verteilt und modelliert die Bereitschaft eines Studenten, in einen bereits sehr vollen Bus einzusteigen. Dies kann verschiedene Situationen der echten Welt darstellen. Beispielsweise könnte ein niedriger Sozialfaktor zeigen, dass der Student in einer größeren Gruppe Studenten unterwegs ist und diese Gruppe nur dann in einen Bus einsteigt, wenn alle Mitglieder der Gruppe einen Sitzplatz bekommen können. Im Gegensatz dazu würde ein hoher Sozialfaktor zum Beispiel einen allein fahrenden Studenten darstellen, der auch bereit ist, in einen stark überfüllten Bus einzusteigen.

## 3.2 Umwelt: Busverbindung

Die Busverbindung verwaltet die Haltestellen am Neumarkt und an der Universität, sowie alle Busse, die diese Haltestellen anfahren. Den Studenten wird somit die Möglichkeit gegeben sich an einer Haltestelle anzustellen, auf einen beliebigen Bus zu warten und mit diesem Bus zum Ziel zu fahren. Nach Ablauf der Fahrtzeit muss der Student wieder aus dem Bus aussteigen und von der Haltestelle nach Hause beziehungsweise in die Universität gehen.

Um der Realität etwas näher zu kommen gibt es nicht nur Studenten die Bus fahren, sondern es wird mithilfe einer Polynomfunktion eine „Hintergrundlast“ die andere

Fahrgäste darstellen soll. Die verwendete Funktion besitzt ein Maximum morgens zum Schul/Arbeitsbeginn und eines zum Feierabendverkehr.

## 3.3 Implementierung mithilfe von Repast Symphony

Das Modell wird mit dem Modellierungswerkzeug Repast Simphony in Java implementiert und ausgewertet. Zunächst müssen Umwelt und Agenten als Java-Klassen programmiert werden. Jede Simulation muss eine Klasse beinhalten, die das Interface *ContextBuilder* implementiert. In diesem Fall ist dies die Klasse *OScillateBuilder*. Die Methode *build* soll nun die Umwelt initialisieren und die Agenten in der Umwelt ansiedeln, um die Simulation starten zu können. Zunächst werden die Simulationsparameter, die der Nutzer in der graphischen Oberfläche spezifizieren kann, ausgelesen und für die spätere Verwendung vorbereitet. Anschließend werden sowohl eine neue Busverbindung als auch die gewünschte Anzahl an Studenten erzeugt.

Die Busverbindung enthält drei Listen und eine Sammlung von Schlüssel/Wert-Paaren (im Folgenden Map). Die Listen enthalten alle Studenten die sich Zuhause, am Neumarkt oder an der Haltestelle an der Universität aufhalten. Die Map enthält alle Studenten, die sich zur Zeit in der Universität befinden, zusammen mit der entsprechenden restlichen Aufenthaltsdauer.

Außerdem enthält die Busverbindung zwei Instanzen der Klasse *Bus*, von der jede eine der beiden Buslinien 11 und 21 darstellt. Ein *Bus*-Objekt modelliert alle Busse, die derzeit auf der entsprechenden Linie fahren, mit allen Fahrgästen, die in einem der Busse sitzen. Konkret bedeutet das, dass die Klasse eine Kapselung einer Map ist. In dieser Map werden, ähnlich wie bei der Universität, die Studenten im Bus mit ihrer restlichen Fahrtzeit und ihrer Fahrtrichtung gespeichert. Somit kann man mit den beiden Objekten alle Studenten darstellen, die zur Zeit in irgendeinem Bus befinden.

Wenn ein Student also vom Neumarkt zur Uni fährt, so wird er zunächst in der „Neumarkt-Liste“ der Busverbindung eingeordnet, bis er in einen Bus einsteigt. Dann wird er in die Map des Busses eingetragen und verbleibt dort, bis seine Fahrtzeit abgelaufen ist. Nun verlässt er den Bus an der Universität und wird mit seiner Aufenthaltszeit in die Map der Universität eingetragen. Nachdem auch hier seine Aufenthaltszeit irgendwann abgelaufen ist, wird er in die der Haltestelle an der Universität entsprechenden Liste eingetragen, bis er dort in einen Bus steigt. In diesem Bus wird er wiederum mit seiner verbleibenden Fahrzeit eingetragen und verweilt im Bus, bis diese abgelaufen ist und er somit am Ende des Tages am Neumarkt angekommt und nach Hause geht, wo er in die entsprechende „Zuhause-Liste“ eingetragen wird, bis er am nächsten Morgen wieder zum Neumarkt geht. Zusammenfassend kann man sagen, dass alle Mengen von Studenten mit unbestimmter Aufenthaltszeit an ihrem aktuellen Ort als Liste und alle Mengen von Studenten mit begrenzter Aufenthaltszeit an ihrem aktuellen Ort als Map gespeichert werden.

Die Klasse *Student* speichert und abstrahiert die jeweiligen Eigenschaften des Studenten, die bei seiner Erstellung angegeben werden und sich im Laufe des Modells entwickeln. So kann beispielsweise der Sozialfaktor zu Beginn festgelegt werden. Die wichtigste Fähigkeit des Studenten ist die Entscheidung, die er vor jeder Busfahrt treffen muss. Dar-

gestellt wird diese mit der Methode *entscheide*. Übergeben werden die Informationen, welche Buslinie gerade an der Haltestelle steht und wie voll die jeweiligen Busse sind, wobei letzteres durch die Asynchronität des Modells möglich wird. So kann der Student in einen Bus einsteigen oder warten und vorerst gar nichts tun. Alle Möglichkeiten der Entscheidung werden überprüft und die Möglichkeit mit der höchsten Priorität wird ausgewählt, sodass der Student seine Lage nach seinem Wissen optimal eingeschätzt und effizient handelt. Eine wesentliche Rolle spielt hierbei der Sozialfaktor. Ist ein Bus voller als es der Sozialfaktor eines Studenten erlaubt, so wird er nicht einsteigen, wobei durch den maximalen Sozialfaktor indirekt der Parameter der Buskapazität eingeführt wird.

Während „normale“ Studenten dazu in der Lage sind mit einem Bus zu fahren, die nicht ihrem Bevorzugten entspricht, beispielsweise wenn dieser zu voll oder einfach gerade nicht da ist, dürfen die zuvor bereits erwähnten *Ersties* dies nicht. Ein Erstie wird zu einem „normalen“ Studenten, wenn er mit der 21 so unzufrieden ist, dass der bevorzugte Bus wechselt. Er wird, praktisch gesprochen, irgendwann experimentierfreudig und findet heraus, dass es eine weitere Linie gibt.

Eine Besonderheit der Modellierung mit Repast Simphony sind die sogenannten *ScheduledMethods*. Durch eine entsprechende *Annotation* im Java-Code kann man eine Methode zu einem definierten Zeitpunkt im Programmablauf ausführen lassen. Repast Simphony rechnet in *Ticks* als Zeitschritt. Im Modell entspricht ein Tick fünf Minuten. Somit ergeben sich für die Buslinie 11 zum Beispiel eine Fahrtzeit von zwei Ticks, während die Buslinie 21 eine Fahrtzeit von drei Ticks hat, um grobe Anlehnungen an die Realität zu erhalten. Die Busverbindung ist das einzige Objekt in der Simulation, das solche Methoden enthält und fungiert somit als eine Art Managerklasse, die den anderen Objekten zu jedem Zeitpunkt Anordnungen gibt. Das beinhaltet zum Beispiel das Aktualisieren der verbleibenden Aufenthalts- und Fahrtzeiten der Studenten in der Universität und in den Bussen, sowie das Ein- und Aussteigen an den Haltestellen. Außerdem bietet die Busverbindung die nötigen Schnittstellen zur graphischen Auswertung der Simulation, in dem die Klasse verschiedene Methoden für statistische Angaben zum Modell bietet.

# 4 Auswertung

# 5 Fazit

Das Projekt OScillate befasste sich mit der Entscheidung von Studenten aus Osnabrück, bezüglich der Buslinie, mit der sie zur Uni und wieder zurück fahren. Realisiert wurde dies durch ein agentenbasiertes Modell, welches auf die Plattform Repast Symphony aufbaut und den Tagesablauf eines Studenten realistisch darzustellen versucht.

Rückblickend ist das Modell nicht wie zunächst geplant explorativ, sondern eher zu einem Verständnismodell geworden. Im Verlaufe der Programmierung sind immer mehr Ideen und Zusatzfunktionen eingeflossen, bis schließlich ein annähernd realistisches Verhalten erreicht wurde und die zeitliche Grenze den Umfang des Projektes festhielt. Mit ein paar echten Daten und Fakten ist es darüberhinaus vermutlich sehr gut als Vorhersa-

gemodell unter einer Fragestellung wie zum Beispiel „Wie viele/große Busse muss ich in welchem Rhythmus einsetzen, ohne dass die Studenten lange warten müssen?“ geeignet.

Die eingebaute Lernfähigkeit der Studenten bezüglich ihrer bevorzugten Buslinie ist nicht sehr gut erkennbar. Zum einen liegt dies an der sehr hohen zeitlichen Auflösung, die das Modell nach wenigen Tagen zum erliegen bringt, zum andere könnte es daran liegen, dass Fehler in Code oder schon im Konzept vorliegen, was wegen dem ersten Punkt schwierig zu sagen ist. Dies verleitet direkt zur Überlegung von möglichen Erweiterungen und Verbesserungen, die den Umfang dieses Projektes gesprengt hätten. Zum einen könnte man das Modell in einer anderen, leistungsfähigeren Umgebung implementieren, um Entscheidungsdynamiken über längere Zeit zu betrachten.

Eine andere, besser in exakt dieses Modell integrierbare Idee wäre die Anordnung der Studenten in einem Netzwerk, dass Freundschaften darstellt und eine Berücksichtigung von darauf aufbauenden Regeln wie zum Beispiel „Ich fahre lieber mit einem Bus, wenn mindestens ein Freund mitfährt!“ . Weiterhin könnte man sonstige Fahrgäste genauer betrachten und die Funktion für die beiden Linien individuell variieren.