

Ausbau agentenbasierter Simulation in Simio

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

[Frühjahrssemester 2016]

Autor(en): Martina Staub, Ernst Füllemann
Betreuer: Prof. Dr. Andreas. Rinkel

ABSTRACT

Ausgangslage

Simulation ist eine leistungsfähige Methode zur Analyse und Optimierung komplexer Systemen. Die Diskrete Ereignissimulation eignet sich für die Abbildung von Entity-Flows (Einheiten-Fluss). Der agentenbasierte Ansatz wird zur Beschreibung kybernetischer Systeme verwendet. Mit Simio steht ein umfangreiches Simulationstool zur Verfügung, das die diskrete Ereignis Simulation unterstützt. Es ist eine Erweiterung für Simio zu entwickeln um den agentenbasierten Ansatz zu unterstützen.

Vorgehen/Technologie

Die agentenbasierte Modellmethodik wird analysiert und die Eigenschaften eines Agenten charakterisiert. Simio stellt zur Entwicklung von benutzerdefinierten Erweiterungen ein .NET API zur Verfügung. Die Programmierschnittstelle wird untersucht und eruiert wie einzelne Eigenschaften und Verhalten eines Agenten in Simio umgesetzt werden können.

Ergebnis

In dieser Arbeit werden Konzepte zur agentenbasierten Simulation analysiert. Die Programmierschnittstelle von Simio wird untersucht und analysiert und ein agentenbasierter Ansatz wird prototypisch umgesetzt. Die Evaluation des Prototypen zeigt welche Anforderungen an Simio gestellt werden um einen agentenbasierten Ansatz zu entwickeln.

MANAGEMENT SUMMARY

AUSGANGSLAGE

Simulation ist eine experimentelle Methode zur Analyse und Optimierung von komplexen Systemen. Dies kann einen Nutzen in den verschiedensten Bereichen erbringen wie zum Beispiel bei Finanzmärkten, Produktion, Ökonomie oder zur Vorbereitung auf Notfall- bzw. Katastrophenszenarien. Eine Simulation ist kosteneffizienter und risikoärmer als das Experimentieren am realen System, falls dies überhaupt Experimente zulässt. Die Simulation ist daher ein sehr wichtiges Instrument für die Entscheidungsfindung, Analyse und Optimierung von Systemen. Durch das Experimentieren mit Simulationsmodellen können fundierte Prognosen und Risikoeinschätzungen erstellt werden.

Simio ist ein umfangreiches Simulationstool. Bis anhin werden die diskrete Ereignissimulation und System dynamics unterstützt. Eine weitere Simulationsmethodik ist die agentenbasierte Simulation, welche zur Beschreibung kybernetischer Systeme verwendet wird. Mittels dieser Simulationsmethodik lassen sich Prozesse intelligenter Objekte simulieren und daraus emergente Verhalten analysieren. Es wurde untersucht ob eine Erweiterung für Simio entwickelt werden kann um einen agentenbasierten Ansatz zu unterstützen.

TECHNOLOGIEN

In Simio können, im Verhältnis zu anderen Programmen, auf einfache Weise eigene Komponenten und Bibliotheken implementiert werden. Es stehen viele Möglichkeiten zur Definition von Prozessen zur Verfügung. Zudem hat es ein .Net-API für die Entwicklung eigener Erweiterungen.

Es wurde sich vertieft mit den verschiedenen Simulationsmethodiken und Konzepten auseinandergesetzt. Anhand der Erkenntnisse aus der Studie der agentenbasierten Simulation wurde ein Prototyp für eine Erweiterung umgesetzt. Der Prototyp unterstützt einen Anwendungsfall der agentenbasierten Simulation.

ERGEBNISSE

Die Programmierschnittstelle zu Simio wurde analysiert und ein Prototyp für eine agentenbasierte Simulation erstellt. Anhand der Erkenntnisse aus der Analyse und der Umsetzung des Prototyps wurden Anforderungen an die Simio API definiert. Die Anforderungen definieren was erfüllt werden muss, damit eine Erweiterung, die das Erstellen von generischen Agenten unterstützt, entwickelt werden kann.

AUFGABENSTELLUNG

ZIEL DER ARBEIT

Simio ist ein System zur diskreten Ereignissimulation und unterstützt die Modellierung von Prozessabläufen. Die Agentenbasierte Modellierung stellt eine neue Modellierungsmethodik dar, die bisher nur ansatzweise in Simio umgesetzt ist.

Ziel der Arbeit ist es, ausgehend von einer umfangreichen Analyse ein Konzept zur agentenbasierten Simulationsmethodik in Simio zu entwerfen und prototypisch umzusetzen. Insbesondere gehören hierzu folgende Schritte:

1. Literaturstudie und Analyse Agentenbasierter Simulation
2. Analyse von Simio und der Programmierschnittstelle
3. Erstellung eines Prototyps einer Agentenbasierten Simulation in Simio
4. Auflistung von Anforderungen an Simio um eine Agentenbasierte Simulation zu ermöglichen



INHALTSVERZEICHNIS

I	TECHNISCHER BERICHT	3
1	EINLEITUNG	5
2	GRUNDLAGEN	7
2.1	Simulation	7
2.1.1	System	8
2.1.2	Modell	8
2.1.3	Prozess	9
2.2	System dynamics	9
2.3	Diskrete Eventsimulation	10
2.4	Agentenbasierte Simulation	10
2.4.1	Vergleich ABS mit SD	10
2.4.2	Vergleich ABS mit DES	11
2.4.3	Definition Agenten	12
2.4.4	Unterschied zwischen Agenten und Intelligen- ten Agenten	12
2.4.5	Agententypen	13
2.5	Simio	14
2.5.1	Agentenbasierter Ansatz in Simio	14
2.5.2	Fähigkeiten von Entity	16
2.5.3	Prozesse in Simio	16
2.6	Existierende Lösungen	17
2.6.1	Vergleich zwischen Anylogic und Simio	17
3	KONZEPTE	19
3.1	Kommunikation	19
3.1.1	Aktion und Reaktion	20
3.1.2	Interaktion	20
3.2	Eigenschaften	20
3.3	Wahrnehmung	20
3.3.1	Begrenzung der Wahrnehmung	21
3.3.2	Verarbeitung der Wahrnehmung	21
3.4	Wissen	23
II	UMSETZUNG	25
4	ANALYSE DER SIMIO API	27
4.1	Grundlegende Erweiterungen	27
4.2	Erstellen eines neuen Objektes	27
4.3	Einfluss auf die Benutzeroberfläche	28
4.4	Manipulation von Daten	29
5	ANFORDERUNGSKATALOG	31
6	PROTOTYP	33
6.1	Beschreibung Prototyp	33

6.2	Tutorial Prototyp	33
6.3	Tutorial Erstellung Steering Behaviors	37
6.4	Umsetzung des Bewegungsverhalten	39
6.5	Umsetzung der Erkennung	40
6.6	Umsetzung des Ausweichen	41
6.7	Umsetzung der Reaktion	42
6.8	Umsetzung Hindernis	42
III	ANHANG	43
A	KATALOG EXISTIERENDER LÖSUNGEN	49
A.1	Allgemeine agentenbasierte Applikationen	49
A.2	Applikationen für verteilte Simulationen	49
A.3	Applikationen mit pädagogischem Fokus	50
A.4	Applikationen für Multi-Agent Systeme	51
A.5	Applikationen für Artificial Intelligence	51
A.6	Applikationen für soziale Wissenschaften	52
A.7	Applikationen für Naturwissenschaften	52
A.8	Applikationen für sehr spezifische Anwendungen . . .	53
B	TUTORIAL	55
B.1	Templates installieren	55
B.2	Anpassungen und Starthilfen	56
B.2.1	Build	57
B.2.2	Debug	57
B.2.3	XAML-Viewer Task beenden	58
B.2.4	Heruntergeladene DLL's	58
C	ENTSCHEIDUNGSMANAGEMENT	61
	LITERATURVERZEICHNIS	65

Teil I

TECHNISCHER BERICHT

EINLEITUNG

Simulation wird zur Analyse und Optimierung von komplexen Systemen eingesetzt. Oft erlaubt das reale System das Experimentieren nicht, da es zu kostenaufwändig, risikoreich oder nicht möglich wäre. Dies ist der Fall bei der Modellierung von seltenen oder nicht steuerbaren Ereignissen wie zum Beispiel einer Naturkatastrophe.

Durch das Nachbilden des Systems können verschiedene Szenarien erstellt werden. Dadurch lassen sich verschiedene mögliche Lösungsansätze auswerten und vergleichen. Simulation hat nicht den Anspruch so genau wie das reale System zu sein. Die Charakteristiken des realen Systems werden so akkurat wie nötig modelliert. Simulation kann in verschiedenen Bereichen angewendet werden, wie bei der Verbesserung eines Fertigungsprozesses, Analyse einer Supply Chain oder Auswertung eines Katastrophenszenarios.

Es gibt verschiedene Simulationsmethodiken. System dynamics wird bei dynamischen Modellen eingesetzt welche aus einer Serie von Beständen und Flüssen bestehen. Die Diskrete Ereignis Simulation repräsentiert die reale Welt mengenmässig und simuliert ihre Dynamik Ereignis um Ereignis. Die agentenbasierte Simulation wird für kybernetische Systeme verwendet. Der Unterschied zu System dynamics und der Diskreten Ereignis Simulation besteht darin, dass die Entity-Flows (Flüssen von Einheiten) nicht passiv, sondern ein eigenes Verhalten haben. Die agentenbasierte Simulation eignet sich für die Modellierung der Prozesse von intelligenten Objekten wie das Verhalten von Menschen oder anderen Organismen.

Simio ist eine Simulationsapplikation, die System dynamics und die diskrete Ereignis Simulation unterstützt. Die Funktionalität von Simio für die Auswertung und Erstellung von Modellen ist sehr umfangreich. Simio unterstützt den Benutzer mit einer guten Hilfestellung und einem Forum. Im Simio Forum findet ein reger Austausch zwischen Benutzern und Simio-Entwicklern statt. Durch die Registrierung als SimioInsider lassen sich Anregungen für zukünftige Funktionen diskutieren und eigene Erweiterungen austauschen.

Es wird die technische machbarkeit überprüft ob eine Erweiterung für Simio entwickelt werden kann, die das Erstellen von agentenbasierten Simulationen unterstützt. Benutzerdefinierte Erweiterungen lassen sich durch das .Net-API implementieren. Das Simio API ist flüchtig dokumentiert. Im Forum wird der agentenbasierte Ansatz diskutiert, es ist aber noch keine Erweiterung in diesem Bereich entstanden.

2.1 SIMULATION

Simulation ist eine experimentelle Methode zur Analyse und Optimierung von komplexen Systemen. Dabei werden ein oder mehrere Prozesse eines Systems auf ein Modell abgebildet und imitiert.

Dies kann einen Nutzen in den verschiedensten Bereichen erbringen wie zum Beispiel bei Finanzmärkten, Produktion, Militär, Ökonomie oder zur Vorbereitung auf Notfall- bzw. Katastrophenszenarien. Eine Simulation ist kosteneffizienter und risikoärmer als das Experimentieren am realen System, falls dies überhaupt Experimente zulässt. Die Simulation ist daher ein sehr wichtiges Instrument für die Entscheidungsfindung, Analyse und Optimierung von Systemen. Durch das Experimentieren mit Simulationsmodellen können fundierte Prognosen und Risikoeinschätzungen erstellt werden.[9]

Es gibt unterschiedliche Ansätze für Simulation. Beispiele sind System dynamics [Abschnitt 2.2](#) und die diskrete Ereignissimulation [Abschnitt 2.3](#). Die agentenbasierte Simulation [Abschnitt 2.4](#) ist ein relativ neuer Ansatz. Die Wahl des Ansatzes ist abhängig von den Eigenschaften des zu simulierenden Systems und des Abstraktionsgrades des Problems.

System dynamics eignen sich vor allem um Trends und Wirkungen zu simulieren. Einzelne Objekte wie zum Beispiel Menschen, Produkte etc. spielen dabei keine Rolle. Wobei diskrete Eventsimulation weniger abstrakt ist und ein reales physisches System abbildet wie zum Beispiel eine Fertigungsstrasse. Der agentenbasierte Ansatz liegt am nächsten an der Realität und kann einzelnen Individuen ein eigenes Verhalten geben.[3]

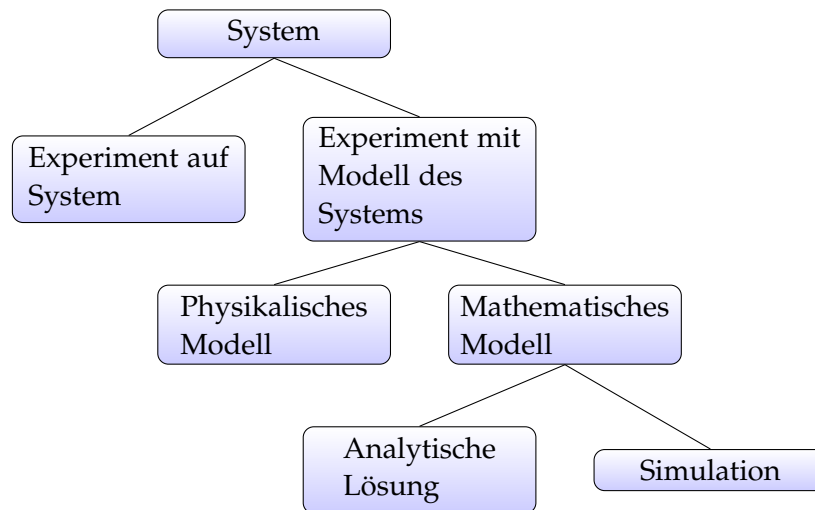


Abbildung 1: Arten ein System zu untersuchen.[4]

2.1.1 System

Ein System ist eine Einrichtung oder ein Prozess und besteht aus Systemkomponenten. Diese Komponenten sind miteinander verbunden oder aufeinander bezogen und interagieren miteinander. Systemkomponenten können wiederum selbst Systeme sein. Dabei wird zwischen diskreten und kontinuierlichen Systemen unterschieden. Ein Diskretes System hat eine bestimmte Anzahl definierter Zustände und basiert auf festen Ereigniszeitpunkten. Hingegen bei einem kontinuierlichen System ändern sich die Zustände über der Zeit stetig. Die Änderungen geschehen nicht zu einem bestimmten Zeitpunkt sondern zu jeder Zeit. Als Beispiel kann das Füllen einer Badewanne betrachtet werden. Fortwährend fließt Wasser in die Wanne. Der Zustand der Badewanne ändert sich laufend. In der Realität sind alle Systeme stetig. Auf einem Rechner simuliert ist alles getaktet-diskret.

2.1.2 Modell

Als Modell wird die Abstraktion eines realen oder eines noch nicht bestehenden Systems bezeichnet. Es fasst die relevanten Bereiche zusammen und beinhaltet die Hauptfunktionen und Charakteristiken des realen Systems damit aussagekräftige Experimente darauf ausgeführt werden können. Je besser ein Modell das System bzw. das zu untersuchende Verhalten des Systems reproduzieren vermag, desto genauere Ergebnisse können untersucht werden. Alles was vom Modell nicht beschrieben wird, wird als Rauschen bezeichnet. Das sogenannte Rauschen kann einen Einfluss auf das System haben, wird aber bewusst nicht berücksichtigt.

2.1.3 Prozess

Prozesse sind zum Beispiel triviale Abläufe wie die Zubereitung des Abendessens oder komplexe wie der Bestellungseingang eines Unternehmens ([Abbildung 2](#)). Ein Prozess hat einen Start-, sowie einen Endpunkt und besteht aus Schritten und Entscheidungen. Die einzelnen Schritte können wiederum Subprozesse sein. Ein Prozess kann von anderen Prozessen abhängig sein, oder von einem anderen Prozess gestartet werden. Während des Ablaufes kann dieser auf externe Inputs angewiesen sein. Ein Prozess kann ein Produkt oder Ergebnis hervorbringen. Störfaktoren können den Ablauf des Prozesses hindern. Der Prozess kann für den Ablauf eine oder mehrere Ressourcen benötigen. Ressourcen sind beispielsweise geforderte Materialien für die Abarbeitung des Prozesses oder Arbeitskräfte für die Ausführung der Arbeitsschritte.

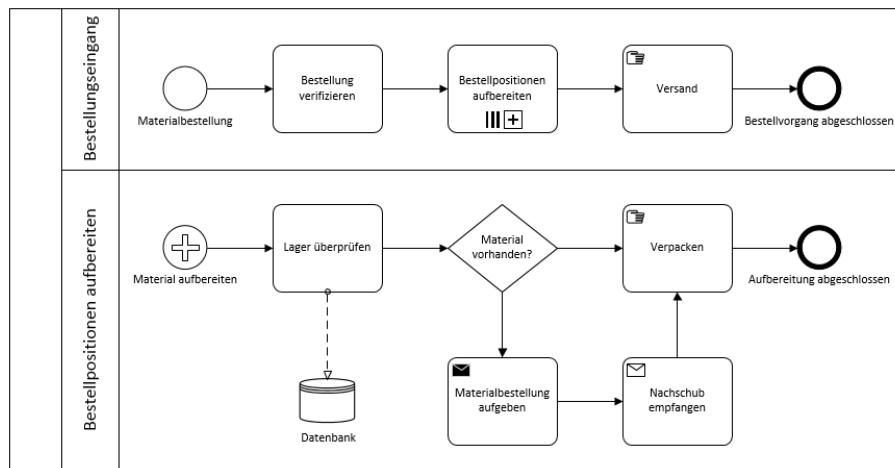


Abbildung 2: [BPMN](#) Flussdiagramm eines Prozesses mit parallelen Subprozessen

2.2 SYSTEM DYNAMICS

System dynamics ([SD](#)) wird bei kontinuierlichen Systemen eingesetzt und ist ein dynamisches Modell welches aus einer Serie von Beständen und Flüssen besteht. Die Einheiten der Flüsse sind meistens nicht klar definierbar. Es handelt sich dabei um kontinuierliche Einheiten wie zum Beispiel Wasser. Es kann sich dabei aber auch um nicht physische Dinge wie Wissen oder Meinungen handeln. Die Zu- und Abflüsse werden jeweils in Raten beschrieben. Der Zustand ändert sich somit kontinuierlich. SD wird vorallem bei der Analyse von sozioökonomischen Systemen eingesetzt wie zum Beispiel um das Entwicklungspotenzial eines Marktes ([Abbildung 3](#)) zu analysieren.

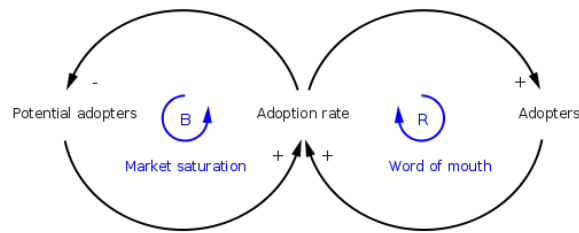


Abbildung 3: Loop Diagramm für die Einführung eines neuen Produktes

Ein Produkt wird eingeführt. Je mehr Menschen dieses Produkt besitzen, desto mehr Menschen gibt es die für das Produkt werben. Mehr Menschen erfahren von dem neuen Produkt und werden zu potenziellen Käufern.

2.3 DISKRETE EVENTSIMULATION

Diskrete Eventsimulation (DES) repräsentiert die reale Welt mengenmäßig, simuliert ihre Dynamik Ereignis um Ereignis und generiert detaillierte Reports. DES setzt sich mit Systemen auseinander welche beispielsweise den Fertigungsablauf einer Maschine beinhalten. Die DES besitzt Einheiten welche einen vorgegebenen Prozess durchlaufen. Bei auslösenden Ereignissen können sich die Zustände der Einheiten und der Umgebung verändern.

Anwendung findet diese Art von Simulation bei der Analyse von Geschäfts- und Fertigungsprozessen.[1]

2.4 AGENTENBASIERTE SIMULATION

Agentenbasierte Simulation (ABS) wird für die Simulation von komplexen dynamischen Systemen verwendet. Es werden dabei Agenten modelliert, die eigene Entscheidungs- und Handlungsmöglichkeiten haben (Tabelle 1). Die Agenten können mit anderen Agenten und mit ihrer Umwelt interagieren. Dadurch kann das dynamische Verhalten von Schwärmen, Ameisenkolonien, Wettbewerbsmärkten oder Katastrophenszenarien analysiert werden. Durch die Experimente können emergente Verhaltensmuster erkannt werden. Agentenbasierte Modelle können sich durch ihre komplexen Verhaltensweisen sehr nah an reale Systeme annähern.

2.4.1 Vergleich ABS mit SD

ABS kann auch in System dynamics angewendet werden und noch bessere Ergebnisse liefern. Voraussetzung dafür ist jedoch, dass das System die nötige dynamische Komplexität mitbringt und eine ABS

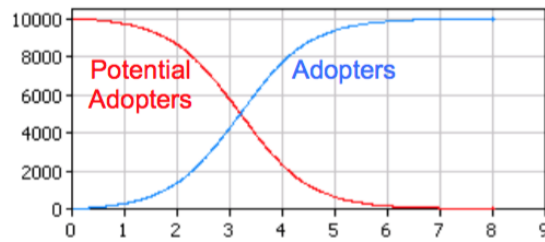


Abbildung 4: SD Simulationsresultat [3]

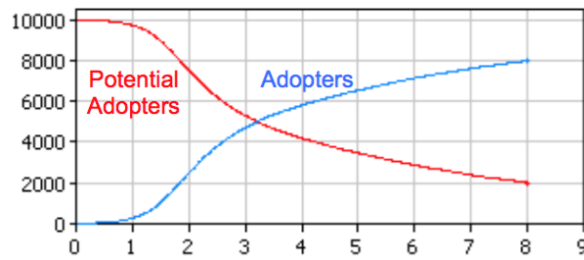


Abbildung 5: ABS Simulationsresultat [3]

rechtfertigt. Beim erwähnten Beispiel aus [Abbildung 3](#) von der Einführung eines neuen Produktes wird bei der [SD](#) die mündliche Werbung ein Durchschnittswert genommen.

In der realen Welt ist jeder Mensch einzigartig. Jeder kennt unterschiedlich viele Mitmenschen und macht mehr oder weniger intensiv Werbung. Modelliert man somit jeden Menschen als einen Agenten mit eigenem Wissen und Verhalten werden die Resultate präziser (Vergleich [Abbildung 4](#) und [Abbildung 5](#)).

Nicht in jeder Situation bringt eine [ABS](#) einen Vorteil. Bei der Simulation von Geldflüssen, unterscheidet sich ein Franken nicht vom anderen. Eine [ABS](#) würde keinen Nutzen bringen [3].

2.4.2 Vergleich [ABS](#) mit [DES](#)

[DES](#) und [ABS](#) haben viel gemeinsam. Die [DES](#) arbeitet bereits mit Einheiten welche durch Erweiterung von Autonomie als Agenten angesehen werden können. Die Einheiten des [DES](#) sind passiv und haben kein eigenes Verhalten. Mit [ABS](#) lässt sich das Verhalten intelligenter Objekte simulieren und somit ein genaueres Abbild des realen Systems. [6] [7]

DES MODELLIERUNGSANSATZ	ABS MODELLIERUNGSANSATZ
Prozessorientiert (top-down Modellierungsansatz); Fokus liegt bei der modellierung des Systems, nicht in den Entitäten	Individuum basiert (bottom-up Modellierungsansatz); Fokus liegt bei den Entitäten und deren Interaktionen
Ein Kontrollfluss (zentralisiert)	Jeder Agent hat seinen eigenen Kontrollfluss (dezentralisiert)
Passive Entitäten; Es geschieht etwas mit den Entitäten während sie sich durch das System bewegen; Intelligenz (Entscheidungsfassung) wird als Teil des Systems modelliert	Aktive Entitäten; Die Entitäten können selbst die Initiative ergreifen eine Aktion auszuführen; Intelligenz wird durch die individuelle Entität repräsentiert
Warteschlangen sind ein Schlüsselement	Kein Konzept von Warteschlangen
Fluss von Entitäten durch das System; Verhalten wird auf Makroebene modelliert	Kein Konzept von Flussverhalten; Verhalten resultiert aus Entscheidungen welche auf der Mikroebene der einzelnen Agenten beschrieben werden
Eingangsverteilungen basieren meist auf gesammelten/gemessenen (objektiven) Daten	Eingangsverteilungen basieren meist auf Theorien (subjektiv)

Tabelle 1: Attribute welche den Modelltyp definieren[8]

2.4.3 Definition Agenten

Allgemein sind Agenten Einheiten, welche über eigene Attribute und Verhalten verfügen. Sie haben eigene Ziele und reagieren autonom auf ihr Umfeld um ihre Ziele zu verfolgen. Eine allgemein eindeutige Definition für Agenten gibt es nicht [10]. Je nach Einsatz und Umfeld eines Agenten, werden seine Fähigkeiten unterschiedlich gewichtet. Keine der Fähigkeiten ist essentiell um einen Agenten auszumachen. Die Kommunikation zwischen zwei Agenten kann in einem Szenario von grosser Bedeutung sein, während sie in anderen Simulationen unerwünscht ist.

2.4.4 Unterschied zwischen Agenten und Intelligenten Agenten

Ein einfacher Agent hat für jeden definierten Eingang von Information eine mögliche Aktion als Antwort. Es handelt sich dabei um einen

einfachen wenn-dann-Ablauf. Ein intelligenter Agent zeichnet sich dadurch aus, dass er flexibel autonome Reaktionen auswählen kann um seine Ziele zu erreichen. Seine Fähigkeiten sind in drei Kategorien gegliedert [10].

1. *Reaktivität*: Agenten können Informationen ihres Umfelds aufnehmen, speichern und darauf reagieren um ihre Ziele zu erreichen.
2. *Aktivität*: Sie haben ein zielgerichtetes Verhalten und können auch die Initiative ergreifen um ihre Ziele zu erreichen.
3. *Soziale-Fähigkeit*: Interaktion und Kommunikation mit anderen Agenten und Systembenutzer ist möglich um die Ziele zu erreichen. Die Agenten können unterschiedliche Kommunikationsarten verwenden. z.B. Point-to-Point¹ oder Point-to-Multipoint².

2.4.5 Agententypen

Durch unterschiedliche Fähigkeiten oder Ausprägungen davon, bilden sich folgende Typen von Agenten.

REAKTIVE AGENTEN: (Simple Reflex Agents) Der Agent verfügt nicht über eigenes Wissen. Er erhält von seinem Umfeld Informationen und reagiert entsprechend darauf.

INTERFACE AGENTEN: Durch Kommunikation mit dem Systembenutzer, werden die Ziele erreicht.

ADAPTIVE AGENTEN: Der Agent kann aufgrund seines eigenen Status und dem Status seiner Umwelt eine möglichst optimale, den Bedingungen angepasste, Entscheidung treffen.

BEOBACHTENDE AGENTEN: (Model Based reflex Agent) Im Gegensatz zum reaktiven Agenten hat dieser Agent ein eigenes Gedächtnis und sammelt Informationen über seine Umwelt. Seine Entscheidungen trifft er basierend auf den gesammelten Informationen. Er weiss wie sich seine Umwelt in der Vergangenheit verhalten hat und kann somit eine möglichst optimale Entscheidung treffen.

LERNENDE AGENTEN: Durch ein Feedback-Element erhält der Agent eine Kritik wie gut er sich verhält. Der Agent kann in einem für ihn unbekannten Umfeld bewegen und Informationen sammeln. Basierend auf seinem aufgebauten Wissen, trifft er seine

¹ Point-to-Point oder auch Direktverbindung bezeichnet eine direkte, unmittelbare Verbindung zwischen zwei Punkten.

² Point-to-Multipoint bezeichnet die Verbindung zwischen einem bestimmten Punkt zu mehreren verschiedenen Punkten wie z.B. Wirelessrouter oder Radiostationen

Entscheidungen. Das Feedback-Element, welches auch der Systembenutzer sein kann, gibt dem Agenten eine Rückmeldung wie gut seine jeweiligen Entscheidungen waren. So lernt der Agent immer bessere Entscheidungen zu treffen.

KOLLABORATIVE AGENTEN: Agenten können durch Zusammenarbeit und durch Kooperation mit anderen Agenten ihre Ziele erreichen. Sie sind autonom und können mit anderen Agenten kommunizieren. Meistens sind sie auch selbstlernend.

KOGNITIVE AGENTEN: (Goal Based, Utility based Agents) Dieser Agent kennt die Folgen seiner Aktionen. Er plant die Schritte, welche ihn zu seinem Ziel führen. Es kann auch Erweiterungen von diesem Agenten geben, bei denen der Agent den Nutzen seiner Ziele ermitteln kann und eine Risikoeinschätzung seiner Ziele durchführen. Durch die Analyse der Nutzen seiner Ziele, kann er entscheiden, welches Ziel das Erstrebenswerteste ist. Er weiss, wie seine Schritte sein Umfeld beeinflussen und ob sich dies positiv oder negativ auf seine weiteren Ziele auswirkt.

SMARTE AGENTEN: Besitzen alle bereits aufgeführten Eigenschaften. [10]

2.5 SIMIO

Simio ist eine Simulationssoftware, welche die Modellierung von diskreten und kontinuierlichen Systemen ermöglicht. Es baut auf einem objektorientierten Ansatz auf und unterstützt folgende Modellierungsparadigmen: Event-, Prozess-, Objekt- und Agentenbasiert. Simio versucht durch die umfangreiche grafische Oberfläche auch die Modellierung von komplexen Systemen möglichst einfach zu halten und setzt keine Programmierkenntnisse voraus. Trotzdem können in .Net komplexere benutzerdefinierte Erweiterungen programmiert werden.

Simio bietet mit dem Simio Software Discussion Forum³ ein öffentliches Forum für Austausch und Hilfe an. Durch die kostenlose Anmeldung bei SimioInsider sind weitere Topics im Forum zugänglich, welche nur für SimioInsider zur Verfügung stehen. Jede Person kann SimioInsider werden, es wird lediglich eine Registration benötigt. Die SimioInsider tauschen im Forum ihre selbst entwickelte Erweiterungen aus und geben Anregungen für Verbesserungen an Simio.

2.5.1 Agentenbasierter Ansatz in Simio

Simio wirbt damit, ABS zu unterstützen. Wenn man die [Tabelle 1](#) betrachtet, ist zu erkennen, dass zumindest einige ABS Modellierungsansätze implementiert sind. Beispielsweise können Prozessabläufe in

³ Simio Software Discussion Forum: www.simio.com/forums

Objekten definiert werden (**bottom-up**, dezentralisiert) und beschränken sich nicht auf die Systemebene. Da zusätzlich **DES** spezifische Elemente wie Warteschlangen implementiert werden, kann Simio als hybrides Simulationstool bezeichnet werden.

In der Objekthierarchie von Simio ist der Agent eine Vorlage der implementierten Elementen **Entity** und **Transporter**. Der Agent selbst kann jedoch nicht instanziiert werden (**Abbildung 6**). Die Objekte in Simio weisen Verhalten von komplexeren **reaktiven Agenten** und eingeschränkten **adaptiven Agenten** auf. Sie zeigen jedoch kein intrinsisch autonomes oder intelligentes Verhalten auf und haben auch keine Vorstellung von ihrer Umwelt.

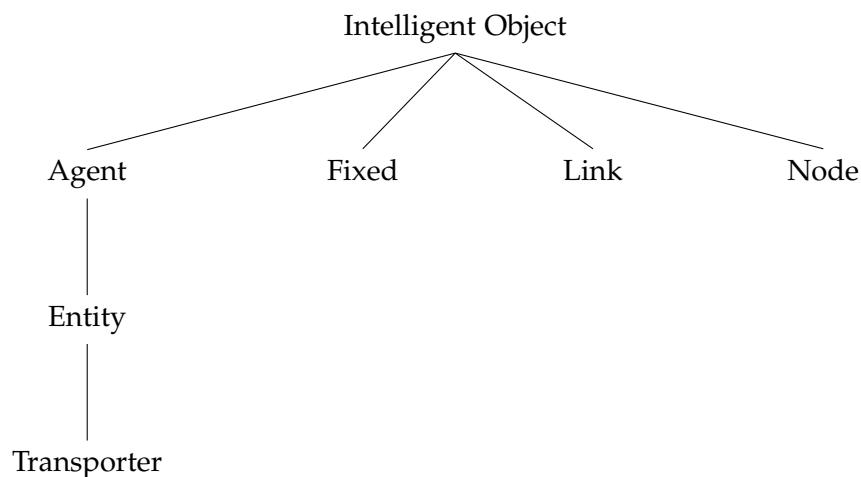


Abbildung 6: Simio Objekthierarchie

2.5.2 Fähigkeiten von Entity

1. *Bewegung*: Objekte können sich frei auf einem Pfad bewegen. Ohne Pfad können nur direkte stationäre Ziele angesteuert werden. Pfade können uni- oder bidirektional sein. Die Objekte bewegen sich stets vorwärts. Falls der Pfad eine Breite hat und sich nicht nur auf eine Linie beschränkt, können sich die Entities in Kurven fortbewegen. Die Entities können eine Kollisionserkennung aktiviert haben. Somit wird verhindert, dass sie sich auf dem Pfad kreuzen. Es gibt keine Möglichkeit die Kollisionsdetektion zu konfigurieren (z.B. Distanz).
2. *Aktivität*: Die Entities haben ein Ziel. Sie wissen nichts über das Modell in dem sie sich bewegen. Sie folgen dem vorgegebenen Pfad und passieren z.B. [Servers](#) oder [Workstations](#) bis sie zum gewünschten [Node](#) gelangen. Die Entities können über [Properties](#) und Prozesse verfügen.
3. *Soziales Verhalten*: Die Entities können nicht untereinander kommunizieren.

2.5.3 Prozesse in Simio

Simio bietet über ein Benutzerfenster die Möglichkeit Prozessabläufe zu definieren. Prozesse werden zur Modellierungszeit auf einem Modell oder direkt auf einem Objekt (mit Ausnahme von [Entity](#) und [Link](#)) erstellt. Prozesse können einerseits durch einen anderen Prozess mit einem Execute-[Step](#) angestoßen werden oder indem ein [Step](#) als Auslöser referenziert wird. Im Falle eines Prozesses welcher im Modell eines [Entities](#) definiert ist besitzt zur Laufzeit jede Instanz eine eigene Kopie. Der Prozess besitzt den gleichen [Scope](#) wie das zugehörige Objekt und hat somit auch nur Zugriff auf Daten welche für das Objekt sichtbar sind. Prozesse werden in Simio hauptsächlich dazu verwendet das Verhalten eines existierenden Objektes anzupassen oder um neue Objektdefinitionen zu definieren beim Erzeugen einer Unterklasse eines existierenden Objektes. Ein Prozess besteht aus [Steps](#), [Elements](#) und [Tokens](#).

STEP: [Steps](#) repräsentieren Aktivitäten in einem Prozessablauf und werden benutzt um Logik zu definieren. [Steps](#) sind Zustandslos können aber die Zustände auf Objekten, [Tokens](#), [Elements](#) und [Entities](#) verändern. Im Prozessfenster können [Steps](#), nach eigenem Ermessen angeordnet und miteinander verbunden werden.

ELEMENT: Ein [Element](#) repräsentiert Objekte in einem Prozess welche ihren Zustand über die Zeit verändern. Sie werden in einer

Liste gespeichert und von einem oder mehreren Prozessen referenziert.

TOKEN: **Token** durchlaufen den Prozessfluss und führen die darin enthaltenen **Steps** aus. **Tokens** können benutzerdefinierte Zustände enthalten um Informationen von **Step** zu **Step** zu transportieren.

2.6 EXISTIERENDE LÖSUNGEN

Es sind sehr viele unterschiedliche Simulationsprogramme für Agentenverhalten auf dem Markt. Viele wurden für einen Anwendungsbereich entwickelt und sind sehr gut darin, z.B. Fussgängerbewegungen zu simulieren. Daher ist es ratsam die Simulationsapplikation nach dem Anwendungszweck auszuwählen. Einige Softwares sind sehr verbreitet und verfügen über Tutorials und Dokumentationen, wobei bei anderen kaum oder keine Dokumentation verfügbar ist.

Cynthia Nikolai und Madey Gregory haben eine Übersicht zusammen gestellt über bekannte agentenbasierte Simulationssoftware. Es wurden dabei 53 Softwares unter den folgenden fünf Gesichtspunkten kategorisiert: Vorausgesetzte Programmiersprache um ein Model zu bilden, Betriebssystem, Lizenz, Anwendungsgebiet, Support für den Anwender [5]. Bei vielen Software werden Programmierkenntnisse vorausgesetzt um ein Model zu erstellen. Ein wesentlicher Vorteil von Simio ist, dass alles im Userinterface getätigt werden kann. Simio stellt jedoch trotzdem eine Programmierschnittstelle für eigene Erweiterungen zur Verfügung.

Es ist schwierig die verschiedenen Applikationen für agentenbasierte Simulation zu vergleichen. Viele sind für Spezialgebiete entwickelt worden und verfolgen daher andere Ansprüche. Der Katalog im Anhang **Anhang A** kategorisiert die Applikationen nach ihrem primären Anwendungszweck.

Wir finden Anylogic von den aufgeführten Applikationen erwähnenswert, da es von seiner Funktionalität und Bedienung eine grosse Ähnlichkeit mit Simio hat, aber bereits agentenbasierte Simulation unterstützt.

2.6.1 Vergleich zwischen Anylogic und Simio

Wie Simio unterstützt Anylogic die Eventbasierte Simulation und System dynamics und zusätzlich die agentenbasierte Simulation. Anylogic wurde als eine Inspiration für einen agentenbasierten Ansatz in Simio verwendet. In der Bedienung für Eventbasierte Simulationen sind die beiden Applikationen sehr ähnlich. Anylogic verfügt wie Simio über eine sehr gute und intuitive Benutzeroberfläche und setzt für simple Simulationen beinahe keine Programmierkenntnisse

voraus. Mit wenigen Klicks kann eine Simulation erstellt werden (Abbildung 7). Da bei Anylogic sich die Agenten frei bewegen, können sehr einfach Dichtediagramme (Abbildung 8) erstellt werden um die Verhaltensmuster zu analysieren.

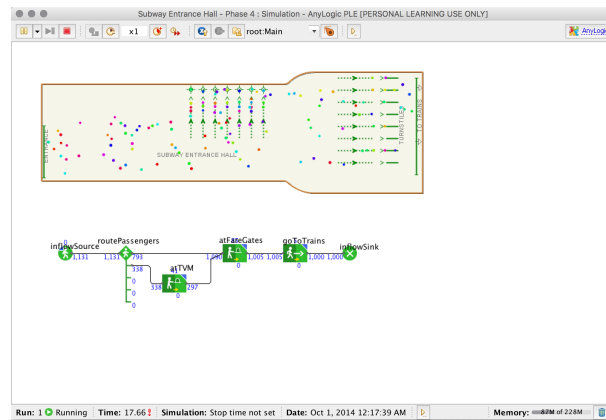


Abbildung 7: Beispiel-Simulation in Anylogic

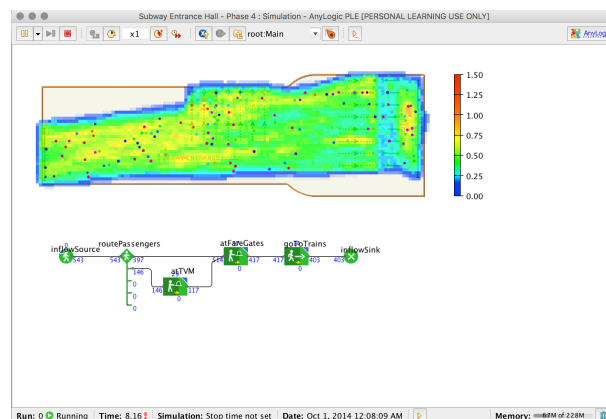


Abbildung 8: Dichtediagramm der Beispiel-Simulation in Anylogic

Bei Anylogic können komplexere Eigenschaften direkt im Property in Java implementiert werden und bietet daher gegenüber Simio grösseren Spielraum.

Interessante Eigenschaften von Anylogic gegenüber Simio sind [2]

- Agenten bewegen sich frei (nicht an Pfad gebunden)
- Erkennung von Nachbar-Agenten
- Messaging zwischen einzelnen Agenten und von Agenten zu Agenten-Gruppen
- Zugriff von Agent auf Variablen und Parametern von anderen Agenten
- Agenten nehmen ihre Umwelt wahr und reagieren darauf wie z.B. Agenten könne nicht durch Wände gehen

KONZEPTE

In agentenbasierter Modellierung steht das Verhalten einzelner Agenten im Mittelpunkt. Die Auswirkungen des Zusammenspiels mehreren evtl. verschiedenen Agenten ist Zentrum der Untersuchung von Prognosen oder Vermutungen und wird simuliert. Aufgrund der in [Abschnitt 2.4](#) beschriebenen Agententypen und unter Berücksichtigung der Funktionsweise von Simio wurden die folgenden Konzepte ausgearbeitet:

- Kommunikation
- Eigenschaften
- Wahrnehmung
- Wissen

Dies geschah zum einen durch Brainstorming im Team sowie auch im Dialog mit Herrn Prof. Dr. Andreas Rinkel und seiner Assistentin Frau Olivia Müller.

3.1 KOMMUNIKATION

Kommunikation als Konzept beschreibt welche Möglichkeiten zur Handlung einem Agenten zur Verfügung stehen. Dabei ist die Kommunikation als Informationsaustausch zu verstehen und kann somit je nach Implementation als physische Handlung, als Wortwechsel oder als beliebiges anderes Verhalten dargestellt werden. Kommunikationsarten sollen zur Modellierzeit bestimmt, aber auch zur Laufzeit gelernt bzw. verlernt werden können. Kommunikation kann eine Voraussetzung sein für die Aneignung von Wissen ([Abschnitt 3.4](#)) oder die Veränderungen von Eigenschaften ([Abschnitt 3.2](#)). Umgekehrt können aber auch Eigenschaften und Wissen benötigt werden um Kommunikationsarten zu erlernen. Kommunikation muss gegebenenfalls Parallel ausgeführt werden können.

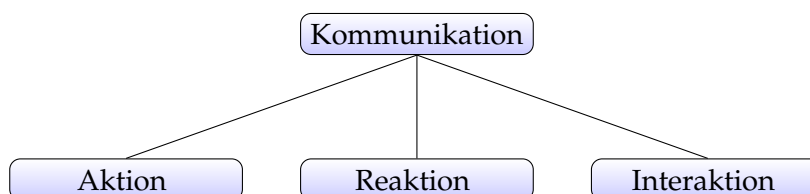


Abbildung 9: Kommunikation als Konzept für Agenten

3.1.1 *Aktion und Reaktion*

Reaktionen sind eine Untergruppe der Kommunikation. Eine Reaktion beschreibt eine Handlung eines Agenten die auf die Veränderung des Zustandes von Ausser- bzw. Innerhalb oder durch [Wahrnehmung](#) (siehe [Abschnitt 3.3](#)) erfolgt. Ein Agent kann eventuell nicht erkennen ob ein Ereignis durch einen anderen Agenten, den Systembenutzer oder per Zufall ausgelöst wird. Aus diesem Grund wird für die [Wahrnehmung](#) zwischen Aktion und Reaktion unterschieden. In einem Agenten jedoch laufen nur Reaktionen ab. Aktionen und Reaktionen können unter Umständen zu einer Interaktion ([Unterabschnitt 3.1.2](#)) führen und Bestandteil einer solchen sein, jedoch ist dies nicht die ursprüngliche Intention.

3.1.2 *Interaktion*

Eine Interaktion beschreibt eine Abfolge von Aktionen und/oder Reaktionen. Im Gegensatz zu Aktion und Reaktion muss jedoch im Vorhinein klar sein mit welchen Akteuren interagiert wird. An einer Interaktion müssen mindestens Zwei Akteure beteiligt sein. Das Konzept der Interaktion ist dazu angedacht, verteilte Algorithmen bzw. Protokolle umzusetzen. Im Rahmen der Interaktion sollen bekannte Konzepte wie Point to Point, Point to Multipoint, synchroner und asynchroner sowie verbindungsorientierter und verbindungsloser Informationsaustausch implementiert werden können.

3.2 EIGENSCHAFTEN

Eigenschaften bezeichnen die verschiedenen Zustände, die ein Agent annehmen kann. Dieses Konzept ist abgeleitet von Datenfeldern aus der Programmierung und kann von Simio übernommen werden. Zusätzlich sollen Eigenschaften zur Laufzeit hinzugefügt oder entfernt werden können.

3.3 WAHRNEHMUNG

Die Wahrnehmung bildet einen oder mehrere Filter ab über welche entschieden werden kann welche Informationen aus der Umwelt aufgenommen und welche verworfen werden. Die Arten der Wahrnehmung lassen sich in drei Unterkategorien aufteilen ([Abbildung 10](#)). Jede Art von Wahrnehmung verfügt über einen Kanal und einen Bezugsbereich. Der Kanal dient zur Implementation von Unterschiedlichen Wahrnehmungen wie z.B. Akkustische Signale, Infrarot, Ultraschall und deren Unterscheidung bzw. Abhörung. Der Bezugsbereich kann auch von einem Medium verändert oder geblockt werden (Luft, Wasser, Blei usw.).

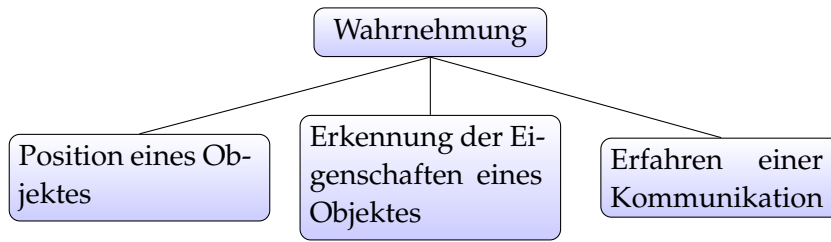


Abbildung 10: Arten der Wahrnehmung

3.3.1 Begrenzung der Wahrnehmung

POSITION EINES OBJEKTES: Über einen definierten Kanal können Agenten andere Objekte und deren Position wahrnehmen. Unter Umständen lassen sich andere Objekte oder Agenten nicht lokalisieren. Z.B. wenn das Objekt zu klein ist um es wahrzunehmen oder nicht über den nötigen Kanal verfügt wird. Über diese Art von Wahrnehmung kann auch eine Kollisionserkennung implementiert werden.

ERKENNUNG DER EIGENSCHAFTEN EINES OBJEKTES: Lässt ein Objekt oder ein Agent dies zu, kann die Position wahrgenommen werden und Eigenschaften gelesen werden. Dies kann z.B. die Dimension eines Objektes sein..

ERFAHREN EINER KOMMUNIKATION: Kommunizieren zwei Agenten miteinander kann dies evtl. von einem dritten Agenten wahrgenommen werden. Je nach Spezifikation des Filters der Wahrnehmung kann eine Kommunikation wahrgenommen werden, ohne dass es zwingend eine eigene Reaktion zur Folge haben. Eine weitere Möglichkeit ist, dass eine Kommunikation wahrgenommen wird ohne dass die beteiligten Objekte wahrgenommen wurden.

3.3.2 Verarbeitung der Wahrnehmung

Die Informationen die ein Objekt von seiner Umwelt erhält, können eine Reaktion hervorrufen oder im Wissen des Objektes gespeichert werden. Da nicht alle Informationen erfasst werden können, verändert sich das Wissen des Objektes im Laufe der Zeit ([Abschnitt 3.4](#)). Das Vergessen kann abhängig von der Art der Wahrnehmung sein.

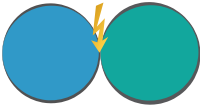
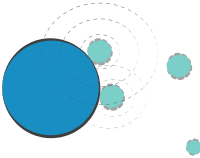
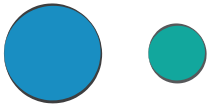
WAHRNEHMUNGSART		BESCHREIBUNG
Physischer	Kontakt	Objekte können sich berühren respektive kollidieren und sich durch den Kontakt wahrnehmen. Bei blinden Menschen oder von Sensoren gesteuerten Objekten eine grundlegende Art um die Umgebung wahrzunehmen.
		
Erfahren einer Kommunikation		Objekte können auch nur eine Kommunikation mit einem anderen Objekt auslösen. Objekte müssen sich nicht zwingend sehen um Kommunikation wahrzunehmen. Ein Ebola-Erreger kann nicht gesehen werden und trotzdem eine Auswirkung haben. Die Auswirkung, respektive die Fähigkeit des Erregers, wird erkannt.
		
Erkennen	einer Gestalt	Für die Wahrnehmung eines Objektes, reicht es aus, wenn sie sich in einem Sichtbarkeitsradius befinden. Dadurch nimmt ein Objekt seine Umwelt war.
		

Tabelle 2: Arten der Wahrnehmung

3.4 WISSEN

Das Konzept des Wissens beschreibt die Kenntnisse von Objekten, Abläufen und Sachverhalten. Wissen ist in dieser Arbeit als Wertungsneutral einzustufen. Somit kann Wissen auch Überzeugungen, Bedürfnisse oder falsche Tatsachen beinhalten. Wissen kann bereits zur Modellierungszeit zur Verfügung stehen oder zur Laufzeit angeeignet, verloren oder modifiziert werden. Um Wissen zur Laufzeit anzueignen, zuverlernen oder zumodifizieren, können Eigenschaften oder Kommunikation vorausgesetzt werden. Somit kann Wissen auch eine Voraussetzung zur Aneignung von Kommunikation oder zur Veränderung von Eigenschaften sein. Wissen soll in verschiedenen Komplexitäten vorliegen und modifizierbar sein. So kann z.B. ein Kenntnis vorhanden sein oder nicht (**Boolean**), die Präferenz für ein Produkt einen numerischen Wert annehmen (**Integer**, **Float**) oder die Ausführung einer Kommunikation verbessert oder verschlechtert werden (Objekte).

Ein Bestandteil des Konzepts Wissen ist das Vergessen. Das Vergessen soll für jedes Wissen konfigurierbar sein. Beispielsweise kann Wissen als Funktion über der Zeit reduziert werden. Je nach Komplexität des Wissens wird konfiguriert wie der Vorgang des Vergessens abläuft. Es können z.B. die älteste oder unwichtigsten Informationen zuerst vergessen werden oder neu Erlerntes.

Teil II

UMSETZUNG

ANALYSE DER SIMIO API

Das .NET Application Programming Interface ([API](#)) *Simio API Reference Guide.chm* steht jedermann zur Benutzung frei und kann im Simio Installationsverzeichnis gefunden werden. Das Dokument gibt eine Übersicht welche Daten dem Benutzer zur Verfügung stehen. Die Dokumentation ist jedoch minimal und Zusammenhänge sind nur in den wenigen Beispielen welche im Simio Insiders Forum und im *UserExtensions* Verzeichnis von Simio beschrieben sind verfügbar. Mit dem für diese Arbeit zur Verfügung stehenden Zeitbudget werden möglichst klare Aussagen und optimale Lösungen zusammengetragen.

4.1 GRUNDLEGENDE ERWEITERUNGEN

In Simio sind bereits Schaltflächen für Erweiterungen im Graphical User Interface ([GUI](#)) sichtbar. Diese sind für *Add-In* im Project Home Reiter, *User Defined Element* im Definitionsfenster und *User Defined Step* im Prozessfenster definiert [Abbildung 11](#).

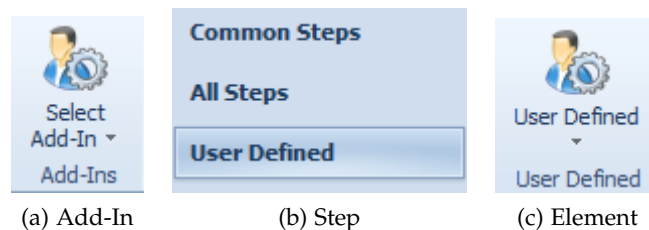


Abbildung 11: [GUI](#) Zugriff auf benutzerdefinierte Erweiterungen

Der Methodenrumpf der *Execute* Methode des Interfaces *IDesignAddIn* wird ausgeführt wenn auf das Add-In Ribbon geklickt wird. Dies ist die einzige Möglichkeit die zur Verfügung steht um direkt geschriebenen Code aktiv auszuführen. Beispielsweise können durch [WinForms](#) oder Windows Presentation Foundation ([WPF](#)) Daten repräsentiert bzw. durch den Benutzer angefordert oder manipuliert werden. Der Methode wird ein *IDesignContext* Objekt mitgeliefert über welches auf das aktive Model zugegriffen werden kann.

4.2 ERSTELLEN EINES NEUEN OBJEKTES

Das *Facility* Objekt welches durch das aktive Model bezogen werden kann stellt eine [FactoryMethod](#) zur Verfügung. Diese Methode ist die

einzigste Möglichkeit Objekte zur Modellierzeit in der [Facility](#) zu erstellen und in Simio verfügbar zu machen. Diese Methode beschränkt sich jedoch auf die bereits in Simio existierenden Objekte wie z.B. [Source](#), [Server](#) oder [Sink](#) und verhindert somit die benutzung selbst Implementierter Klassen. Durch einen [String](#)-Parameter wird der Methode mitgeteilt was für ein Objekt erstellt werden soll. Hier kann im gegensatz zu Simio auch ein Agent erstellt werden. Dieser besitzt jedoch sogut wie keine Funktionalität da er wie in [Abbildung 6](#) erwähnt eine Vorlage für die Objekte [Entity](#) und [Transporter](#) darstellt.

Listing 1: Erstellung von Objekten

```

1 //Folgende Zeile erstellt ein Source Objekt in der Facility
2 var source = intelligentObjects.CreateObject("Source", new
    FacilityLocation(0, 0, 1)) as
3 IIntelligentObject;
4 //Folgende Zeile resultiert in einer NULL-Referenz (TestAgent
    implementiert IIntelligentObject)
5 var agent = intelligentObjects.CreateObject("Agent", new
    FacilityLocation(0, 0, 1)) as TestAgent;

```

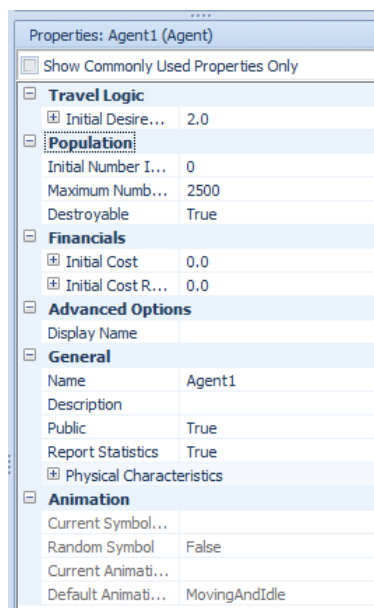


Abbildung 12: Properties des Agenten aus [Abbildung 6](#)

4.3 EINFLUSS AUF DIE BENUTZEROBERFLÄCHE

Der einzige Ansatz um Einfluss auf die Benutzeroberfläche zu nehmen wurde im *IDesignContext* Objekt mit der Methode *ExecuteUICommand* gefunden. Diese Methode kann laut [API](#) dazu benutzt werden Befehle durch die Benutzeroberfläche zu emulieren. Die konkrete Funktionsweise ist jedoch nicht beschrieben und konnte auch nicht durch Trial und Error während dieser Arbeit evaluiert werden. Das

Einbetten von benutzerspezifischen Ansichten in Simio wird nicht unterstützt.

4.4 MANIPULATION VON DATEN

Das definieren von Daten wie z.B. [States](#) und [Properties](#) kann zur Modellierungszeit über das *IDesignContext* Objekt vorgenommen werden. Zur Laufzeit können jedoch keine Definitionen von solchen Daten mehr hinzugefügt bzw. gelöscht werden. Solches Verhalten wäre z.B. für das erstellen eines lernenden Agenten wünschenswert.

Listing 2: Hinzufügen eines States zum aktiven Model

```
1 | _context.ActiveModel.StateDefinitions.AddRealState("TestState");
```


Anforderung	Beschreibung
Hinzufügen und Entfernen von States, Properties, usw. zur Laufzeit	Die gewünschten Events und States müssen mit dem exakten Namen erstellt werden. Im Prototyp wird bei der Erkennung genau der Event mit dem String EndingNode ausgelesen. Das Steering Behavior kann somit nur benutzt werden, wenn man die Dokumentation dazu hat und weiss, dass der folgende State erstellt werden muss.
Auswählen eines Models	States können mittels eines Addins zur Modellierungszeit hinzugefügt werden. Jedoch nur beim aktuell in der Simio Benutzeroberfläche angewählten Modell.
Einbettung im Simio GUI	Bis jetzt kann ein Addin nur mit zusätzlichem Fenster mit WinForms oder WPF erstellt werden
Wahrnehmung von Steering-behavior entkoppeln und Events feuern lassen	Die Wahrnehmung als Konzept implementieren
Verfügbare Objektreferenzen	Objekte können nur über String Abfragen referenziert werden

PROTOTYP

6.1 BESCHREIBUNG PROTOTYP

In einem ersten Prototyp wird ein Agent in Simio umgesetzt. Die Agenten haben einen Einkaufsladen als Ziel. Sie wählen den Weg zu ihrem gewünschten Ziel. Falls sie auf ihrem Weg zum Ziel eine Werbung der Konkurrenz sehen, ändern sie ihre Meinung und ändern das Ziel auf den Laden der Konkurrenz.

Folgende Aspekte eines Agenten werden realisiert:

1. *Bewegung*: Die Agenten bewegen sich frei und sind nicht an einen Pfad gebunden. Sie erkennen andere Agenten und Objekte und weichen diesen aus.
2. *Kommunikation*: Das Austauschen von Informationen ist auf eine bestimmte Distanz möglich.
3. *Reaktivität und Aktivität*: Anhand der erhaltenen Informationen können die Agenten ihr Ziel ändern.

6.2 TUTORIAL PROTOTYP

Der Prototyp ist ein eigenes Steering Behavior in einem Travelstep. Für die Ausführung des Prototyps muss das entwickelte Addin gestartet werden. ([Abbildung 13](#)).

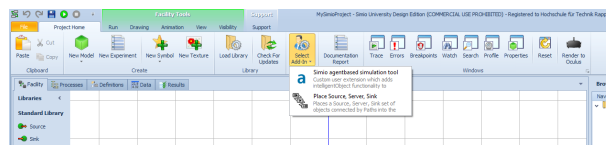


Abbildung 13: Starten des Addin in Simio

Der Travelstep mit dem eigenen Steering Behavior kann auch ohne die Ausführung des Addin ausgewählt werden. Jedoch beim Start der Simulation wird eine Fehlermeldung angezeigt.

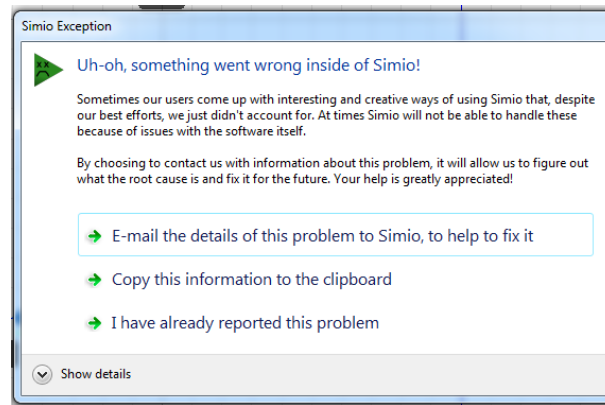


Abbildung 14: Fehlermeldung wenn vergessen wurde das Addin zu starten

Beim Öffnen des Addins wird ein Fenster angezeigt ([Abbildung 15](#)). Beim Knopf Add kann ein Obstacle (Hindernis) hinzugefügt werden. In Simio können Grafiken erstellt und Bilder eingefügt werden um eine möglichst reale Umgebung zu erschaffen. Da vom API nicht darauf zugegriffen werden kann, konnte keine Erkennung der Agenten für ihre Umgebung implementiert werden. Das Addin ermöglicht durch das Einfügen eines Hindernis ein Objekt hinzuzufügen, welches Agenten erkennen und darauf reagieren können.

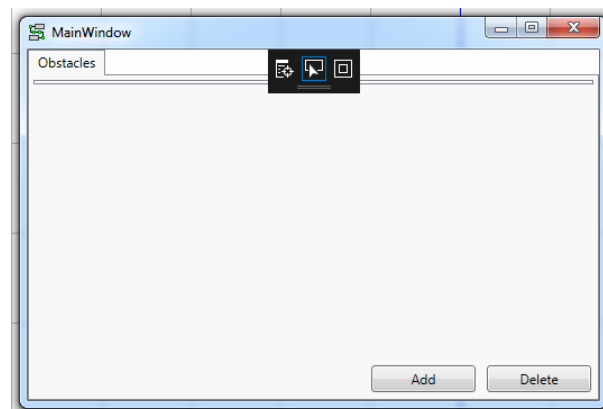


Abbildung 15: Fenster zum einfügen eines Hindernisses

Für die Ausführung des Prototypen muss minimal eine Source, zwei Sinks, ein Entity Model und ein weiteres Intelligent Object, das als Werbung dient, vorhanden sein ([Abbildung 16](#)). Das eingefügte Hindernis ist vom Typ Intelligent Object, da von Agenten nur Intelligent Objects erkannt werden.

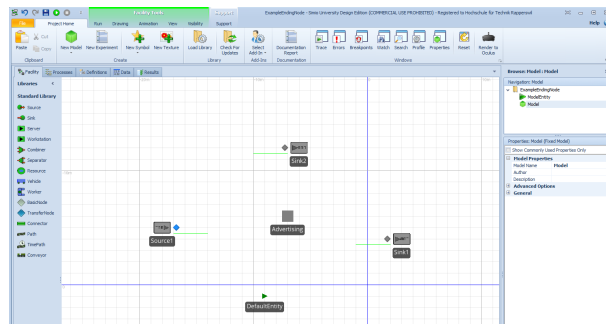


Abbildung 16: Aufbau des Prototyps

Das eingefügte Hindernis wird während der Simulation noch nicht angezeigt. Dem Hindernis wird ein Symbol hinzugefügt und unter dem Reiter Visibility das Feld Entity Instances angewählt.

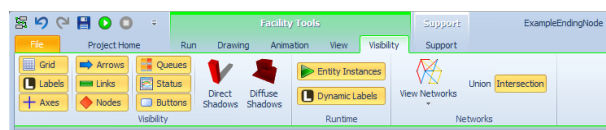


Abbildung 17: Der Visibility Reiter

Der Source wird ein Prozess beim Exit hinzugefügt (Abbildung 18). Der Prozess besteht aus einem Transfer Step. Der Transfer Step bewegt das Entity vom Node in den Freespace (freier Raum). Beim folgenden Travel Step kann in der Detailansicht (Abbildung 19) als Steering Behaviour (Lenkverhalten) Freespace Walking to Destination angegeben werden. Mit den folgenden Properties kann die Distanz angegeben werden in welcher ein Agent auf einen anderen Agenten reagiert und auszuweichen beginnt. Das Property Reaction Distance legt die Distanz in welcher ein Agent auf ein Objekt reagiert fest.

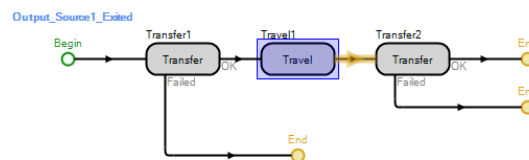


Abbildung 18: Der Prozess im Model

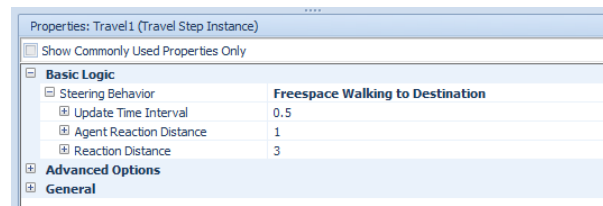


Abbildung 19: Detailansicht des Travel Step

Der letzte Transfer Step beendet die Bewegung der Entity, wenn diese ihr Ziel erreicht hat.

Dem Entity Model wird der Endknoten und Kommunikationsknoten mit zwei States angegeben. Der Kommunikationsknoten ist im Prototyp die Werbung.

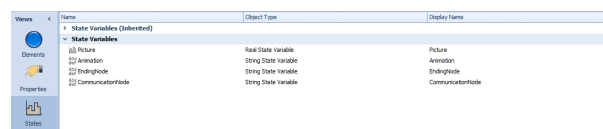


Abbildung 20: Ansicht der States des Entity Model

Beim Entity Model wird ein Prozess mit einem Assign Step hinzugefügt. Der Prozess wird durch das Event Information Update gefeuert. Dieses Event wird aufgerufen, wenn der Agent das Plakat sieht und seine Meinung ändert. Der Agent läuft nicht mehr zu seinem ursprünglichen Endpunkt. Der Assign Step fügt dem Endknoten State des Agenten den Namen des neuen Endknoten hinzu ([Abbildung 22](#)).

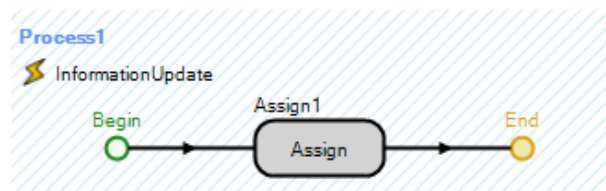


Abbildung 21: Prozess des Entity mit Assign Step

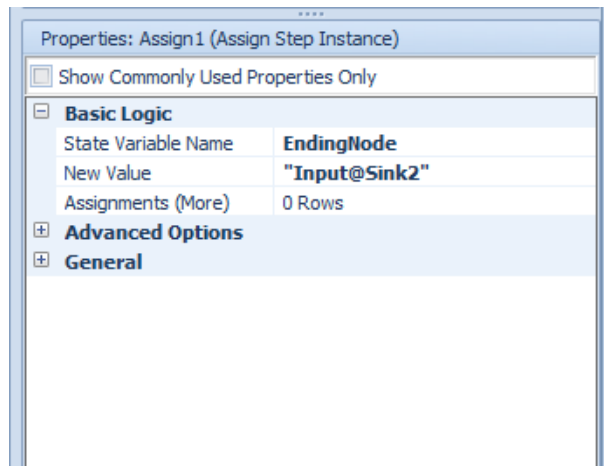


Abbildung 22: Detailansicht des Assign Step

Die Simulation wird gestartet (Abbildung 23). Die Entities bewegen sich auf ihren im State Endknoten definiertes Ziel zu. In dieser Anwendung auf die Sink1. Falls sie die Werbung passieren, ändern sie ihr Ziel und steuern auf die Sink2 zu. Die Objekte können während der Laufzeit verschoben werden. Dabei lässt sich beobachten, wie die Agenten selbstständig erkennen, dass sich die Position ihres Ziels verändert hat. Anhand des veränderten Wissens wird das Bewegungsverhalten angepasst.

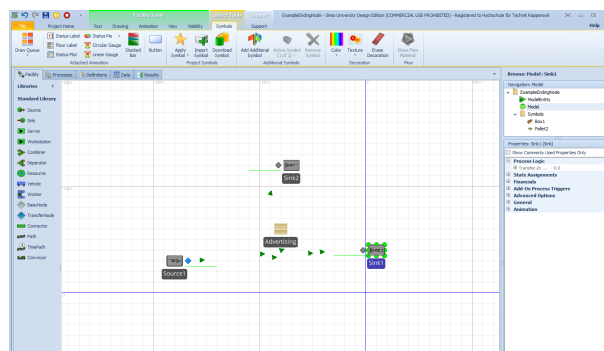


Abbildung 23: Der laufende Prototyp

6.3 TUTORIAL ERSTELLUNG STEERING BEHAVIORS

Das Simio API bietet einen Travelstep. Im Travelstep lässt sich ein Steering Behaviour auswählen (Abbildung 19). Das API bietet ein Interface *ITravelSteeringBehavior* an, mit welchem ein eigenes Bewegungsverhalten implementieren werden kann.

Die wichtigsten Klassen zur Umsetzung einer eigenen Bewegung sind:

ITravelSteeringBehaviorContext

ITravelSteeringBehavior

TravelSteeringMovement

ITravelSteeringBehavior Interface	
API Reference > SimioAPI.Extensions > ITravelSteeringBehavior	
An interface that is implemented by a user-code object to give functionality to an instance of a travel steering behavior.	
Members	
All Members	Methods
<input checked="" type="checkbox"/> Public	<input checked="" type="checkbox"/> Instance
<input checked="" type="checkbox"/> Protected	<input checked="" type="checkbox"/> Static
	<input checked="" type="checkbox"/> Declared
	<input checked="" type="checkbox"/> Inherited
Icon Member	Description
OnTravelCancelling(IExecutionContext)	Called when the travel of the entity is about to be canceled. For example, when other logic explicitly assigns the entity position or direction.
OnTravelResumed()	Called when the travel of the entity has been resumed from suspension. This will be called immediately before <code>Steer()</code> is called to setup the travel again.
OnTravelSuspended()	Called when the travel of the entity has been suspended.
Steer(ITravelSteeringBehaviorContext)	Called at the start or resumption of travel, to do any setup that needs to happen for the travel steering behavior. For example, scheduling events on the calendar. Note that any events scheduled via the Calendar object inside the context will automatically be canceled if the Travel is Suspended (via the Suspend step), or Canceled (if the movement state of the entity is assigned by other logic in the model).

Abbildung 24: API des Steering Behavior

Das Steering Behaviour Interface wird implementiert (Abbildung 24). In der Steer()-Methode wird das Bewegungsverhalten implementiert. Innerhalb der Steer-Methode wird der aktuelle Ort des Agenten ausgelesen. Die Überprüfung ob der Agent auf einen anderen Agent oder Objekt reagieren muss, findet statt. Der neue Bewegungsvektor wird berechnet. Innerhalb der Steer-Methode wird ein neues TravelSteeringMovement mit dem neuen Bewegungsvektor erstellt.

```

1 context.SetPreferredMovement(new TravelSteeringMovement()
2 {
3     Direction = direction,
4     Velocity = entity.velocity,
5     OrientInTheSameDirection = true,
6     Acceleration = acceleration,
7     Acceleration = accelerationVector,
8     AccelerationDuration = time
9 });

```

Listing 3: TravelSteeringMovement

Beim Prototyp wurde die Beschleunigung (Acceleration) nicht betrachtet. Dieses TravelSteeringMovement beschreibt den neuen Bewegungsvektor.

Das TravelSteeringMovement wird dem Kontext (ITravelSteeringBehaviorContext) mit der Methode SetPreferredMovement(TravelSteeringMovement) übergeben. Die Steer()-Methode wird dem Kalender des Kontexts als Callback mit einem Zeitintervall übergeben.

```

1 context.Calendar.ScheduleEvent(context.Calendar.TimeNow +
    timeInterval, null, _ => Steer(context));

```

Listing 4: ITravelSteeringBehavior Kalender

```

1 context.SetPreferredMovement(new TravelSteeringMovement()
2 {
3     Direction = direction,
4     Velocity = entity.velocity,
5     OrientInTheSameDirection = true,
6     Acceleration = acceleration,
7     Acceleration = accelerationVector,
8     AccelerationDuration = time
9 });

```

Listing 5: TravelSteeringMovement

Der Kontext implementiert das Interface `ITravelSteeringBehaviorContext` und ist der Ausführungsrahmen des SteeringBehaviors.

Simio API Reference Guide
ITravelSteeringBehaviorContext Interface
 API Reference ▶ SimioAPI.Extensions ▶ ITravelSteeringBehaviorContext

An interface used when the engine calls `SimioAPI.ITravelSteeringBehavior.Initialize`.

Members		Methods	Properties
<input checked="" type="checkbox"/> All	<input checked="" type="checkbox"/> Public	<input checked="" type="checkbox"/> Instance	<input checked="" type="checkbox"/> Declared
<input checked="" type="checkbox"/> Protected	<input checked="" type="checkbox"/> Static	<input checked="" type="checkbox"/> Inherited	

Icon	Member	Description
	Calendar	The object used to manipulate the simulation calendar (Inherited from IExecutionContext .)
	EntityData	The runtime data for the Entity
	ExecutionInformation	The object used to report runtime information (Inherited from IExecutionContext .)
	GlobalData	An object reference which is passed to all steering rules originating from the same step.
	MovementFinished()	Called by the steering behavior when movement has been finished. If a token is waiting at a Travel step it won't be released until this point. Calling this method tell Simio the steering behavior is complete.
	Random	The object used to query the random number generator (Inherited from IExecutionContext .)
	SetPreferredMovement(TravelSteeringMovement)	Assigns the preferred movement of the entity in terms of a directional velocity vector in the units meters per hour. The Simio engine may decide not to actually ultimately assign this movement based on other factors. Calling this method simply indicates what the steering behavior prefers the movement to be.
	WithCandidate(IIntelligentObjectRuntimeData)	Creates a new execution context with the provided object assigned as the Candidate object. (Inherited from IExecutionContext .)

Abbildung 25: API des Steering Behavior Context

Der Kalender plant die Methode nach jedem Intervall in der Simulationszeit aufzurufen, solange der Agent das Steering Behavior ausführt. Im Prototypen ist das Intervall bei Verwendung der Standardkonfiguration auf 0.5 Sekunden gesetzt. Dadurch wird alle 0.5 Sekunden einen neuen Bewegungvektor für den Agenten gesetzt. Das Intervall kann bei der Detailansicht vom TravelStep angepasst werden (Abbildung 19).

6.4 UMSETZUNG DES BEWEGUNGSVERHALTEN

Die Beschleunigung wurde bei diesem Prototyp nicht beachtet. Falls auf keine anderen Objekte oder Agenten reagiert werden muss, wird der Normalvektor zwischen dem Agenten und seinem Ziel berechnet. Der Richtungsvektor zusammen mit der Geschwindigkeit werden dem TravelSteeringMovement übergeben, welches dann den Bewegungsvektor berechnet.

```

1 FacilityDirection directionVector = (endingNode.Location - entity
  .Location).ToUnitLength();

```

Listing 6: TravelSteeringMovement

6.5 UMSETZUNG DER ERKENNUNG

In den Properties des Steering Behaviours werden die Distanzen zur Erkennung von anderen Objekten und Agenten in Meter angegeben ([Abbildung 19](#)).

Der Agent besitzt zwei Reaktionsvektoren ([Abbildung 26](#)). Der erste Vektor ist so lang wie im Property definiert wurde, der zweite ist halb so lang.

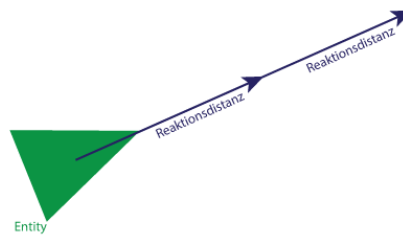


Abbildung 26: Reaktionsvektoren eines Agenten

Der kleine Reaktionsvektor dient dazu Objekte zu erkennen wenn der grosse Vektor bereits darüber hinaus geht ([Abbildung 27](#)). Der Reaktionsvektor hat drei Variablen X, Y, Z. Die Y Variable wird nicht beachtet, da sich die Bewegung in der Ebene abspielt. In Simio ist Y die Variable für die Dimension Höhe. Bei der Erkennung wird überprüft ob der X und Z Wert des Vektors innerhalb der Fläche des Objektes liegt.

Im Prototyp ist das Intervall für den Berechnungsvektor auf 0.5 Sekunden eingestellt. Vom Moment an, wenn der Agent seine Source verlässt, wird alle 0.5 Sekunden der neue Bewegungsvektor berechnet. Die Berechnung des Bewegungsvektors geschieht bei jedem Agenten zu einem anderen Zeitpunkt. Daher ist es möglich, dass obwohl vor der Bewegung überprüft wurde ob der Weg blockiert ist, ein anderer Agent sich dazwischen bewegt hat.

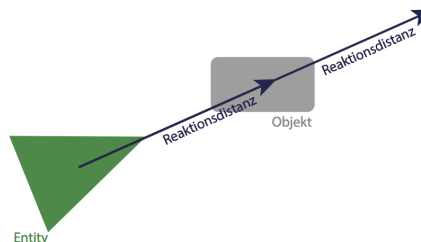


Abbildung 27: Erkennung durch Reaktionsvektoren

Der Agent kann auf Objekte, die in seiner Bewegungsrichtung sind, reagieren. Das Spektrum kann erweitert werden indem dem Agenten weitere Reaktionsvektoren in andere Richtungen hinzugefügt werden.

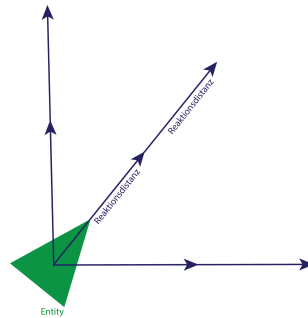


Abbildung 28: Mehrere Reaktionsvektoren

Dem Agenten können verschiedene Vektoren angegeben werden. Im Prototyp besitzt der Agent Reaktionsvektoren für Agenten und Reaktionsvektoren für Objekte seiner Umwelt. Es können unterschiedliche Distanzen eingegeben werden.

Der Agent reagiert auf einen anderen Agenten wenn er einen Meter von ihm entfernt ist. Bei Objekten seiner Umwelt ist die Distanz drei Meter. Die Unterscheidung ergibt ein natürlicheres Verhalten der Agenten. In der realen Welt nehmen wir Häuser etc. von Weitem wahr und wählen unseren Weg entsprechend.

6.6 UMSETZUNG DES AUSWEICHEN

Überschneidet ein Vektor ein anderes Objekt, wird die Abwendung berechnet (Abbildung 29). Die Abwendung kann auch durch einen fixen Wert multipliziert werden um eine stärkere Lenkung zu erzielen. Überschneidet der kürzere Vektor ein anderes Objekt ist es sinnvoll eine stärkere Abweichung einzuleiten.

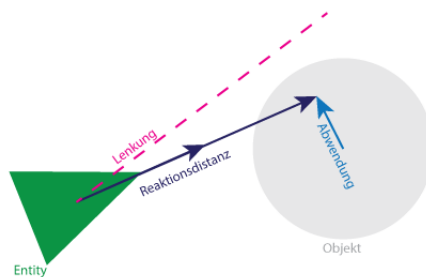


Abbildung 29: Ausweichen eines Objektes

Falls der Agent einen Bewegungsvektor in der exakten Richtung wie das kollidierende Objekt hat, würde der Abwendungsvektor in die Gegenrichtung zeigen. Da sich der Agent wieder auf sein Ziel aus-

richtet, wählt er wieder dieselbe Richtung. Somit blockiert der Vektor vor dem Objekt. In diesem Falle wird ein perpendikulärer Vektor zur Berechnung der Abwendung zur Hilfe genommen.

6.7 UMSETZUNG DER REAKTION

Die Reaktion auf die Werbung wird durch States und Events umgesetzt. In Simio wird dem Entity Model zwei States mit den Namen Communication Node und EndingNode hinzugefügt ([Abbildung 20](#)). Der CommunicationNode hat einen String mit dem Namen des Knoten mit welchem der Agent kommunizieren kann.

Das Entity Model hat einen Prozess mit einem Assign-Step, welcher beim Triggerevent InformationUpdate ausgeführt wird. Der Assign Step fügt dem EndingNode einen neuen Wert hinzu. Falls der Erkennungsvektor den Kommunikationspunkt detektiert, wird das Event InformationUpdate gefeuert.

```
1 | if(Intersect(perceptionVector, CommunicationNode)
2 | {
3 |     InformationUpdate.fire();
4 | }
```

Listing 7: Feuern des InformatonUpdate Events

6.8 UMSETZUNG HINDERNIS

Das Einfügen eines Hindernisses wurde umgesetzt da ein Agent sonst nicht auf seine Umwelt reagieren kann. Somit kann ein Agent sich frei bewegen und sich seinen Weg zu seinem Ziel bahnen. Es lassen sich vom API Objekte wie zum Beispiel Source, Sink etc. instanzieren jedoch nicht neue eigene Objekte erstellen. Die Obstacle sind von der Klasse Agent von Simio. Sie haben jedoch keine Funktionalität.

Teil III

ANHANG

EIGENSTÄNDIGKEITSERKLÄRUNG

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, die explizit in der Aufgabestellung erwähnt sind oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe,
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Rapperswil, 3. Juni 2016



Martina Staub

DANKSAGUNG

Ich möchte mich bei allen Personen bedanken, die mich bei der Durchführung dieser Arbeit unterstützt haben. Ein besonderer Dank gilt Herr Prof. Dr. Rinkel und Olivia Müller für die sehr gute Betreuung und für das angenehme Arbeitsklima. Zudem möchte ich mich bei Ernst Füllemann für die sehr gute Zusammenarbeit bedanken. Für die Erstellung dieser Arbeit wurde das *classicthesis* L^AT_EX-Template benutzt wofür ich mich herzlich bei Prof. Dr.-Ing. André Miede bedanke.

KATALOG EXISTIERENDER LÖSUNGEN

A.1 ALLGEMEINE AGENTENBASIERTE APPLIKATIONEN

Anwendung	Produkt
Allgemeine agentenbasierte Anwendung	AnyLogic
	Ascape
	DOMAR
	JAS
	MAGSY
	Mason
	SeSAm
	SimPack
Multi-agent Systeme mit agentenbasierter Simulations- schicht	Swarm
	Madkit
Allgemeine agentenbasierte Anwendung für parallele Applikationen	DeX

Tabelle 3: Software für allgemeine agentenbasierte Anwendung [5]

A.2 APPLIKATIONEN FÜR VERTEILTE SIMULATIONEN

Anwendung	Produkt
Allgemeine Anwendung für verteilte Simulationen	AnyLogic
	DOMAR
	Jade
	Madkit
	ADK
	Cougaar
	SimAGent
	oRIS

Tabelle 4: Software für verteilte Simulation [5]

A.3 APPLIKATIONEN MIT PÄDAGOGISCHEM FOKUS

Anwendung	Produkt
Allgemein für Bildung in Simulation	MIMOSE NetLogo StarLogo TNG VSEit
Artificial Intelligence	Breve
Für Studenten um das Verhalten von dezentralisier- ten Systemen zu modellieren	StarLogo TNG
Implementierung von Software Agenten	Brahms SimAgent
Lernverhalten	SOAR
Computer Simulation	FAMOJA Matlab oRIS SeSAm SimPack
Soziale Wissenschaften, Mathematik und Naturwis- senschaften	AgentSheets StarLogo
Objektorientierte Prinzipien	jECHO
Programmiertechniken für neue Anwender im Be- reich Simulation	Matlab NetLogo StarLogo Sugarscape
Wissenschaftliche und technische Mathematik, Da- tenanalyse, Exploration und Visualisierung	Matlab

Tabelle 5: Software mit pädagogischem Fokus [5]

A.4 APPLIKATIONEN FÜR MULTI-AGENT SYSTEME

Anwendung	Produkt
Allgemeine Anwendung für vMulti-agent Systeme	AgnetBuilder oRIS MAGSY
Verteilte Multi-agent Systeme	ADK Cougaar Jade
Komplexe Umwelt	SimAgent
Organisationsprozesse	Brahms

Tabelle 6: Software spezialisiert für Multi-Agent Systeme [5]

A.5 APPLIKATIONEN FÜR ARTIFICIAL INTELLIGENCE

Anwendung	Produkt
Allgemeine Anwendung für Artificial Intelligence	Breve iGen SOAR
Machine Learning und logisches Denken	ABLE Zeus
Soziale Wissenschaften	Omonia
3D Simulationen	Breve
Menschenähnliche intelligente Agenten	SimAgent

Tabelle 7: Software für Artificial Intelligence [5]

A.6 APPLIKATIONEN FÜR SOZIALE WISSENSCHAFTEN

Anwendung	Produkt
Allgemeine Anwendung für Soziale Wissenschaften	AgentSheets
	LSD
	FAMOJA
	MAML
	MAS-SOC
	MIMOSE
	NetLogo
	Repast
	SimBioSys
	StarLogo
	Sugarscape
	VSeit
Hilft Anfängern beim erstellen von Modellen	NetLogo
Soziale Systeme	Moduleco

Tabelle 8: Software für Soziale Wissenschaften [5]

A.7 APPLIKATIONEN FÜR NATURWISSENSCHAFTEN

Anwendung	Produkt
Allgemeine Anwendung für Naturwissenschaften	Matlab
	NetLogo
	OpenStarLogo
	StarLogo

Tabelle 9: Software für Naturwissenschaften [5]

A.8 APPLIKATIONEN FÜR SEHR SPEZIFISCHE ANWENDUNGEN

Anwendung	Produkt
Angewandte Simulation/Elektronisches CAD	Jade's Sim++
Biologie	SimBiosSys
Zelluläre Automaten	JCA-Sim
Ökonomie/Auktionsmechanismen	JASA
Ökologische Modellierung	Echo
	jEcho
	SME
Unternehmen	jES
Evolutionäre Algorithmen	ECJ
Menschliche Leistungen Modellieren	iGen
Natürliche Ressourcen Management	Cormas
Politische Phänomene	PS-I
Regel-Maschine und Scripting Umgebung	JESS
	Zeus
Organisatorische Prozesse simulieren	Brahms
Base24 Applikationen Testen	SimPlusPlus
Städte Simulationen	OBEUS

Tabelle 10: Software für sehr spezifische Anwendungsgebiete [5]

GLOSSAR

Boolean	Eine Schaltvariable mit nur zwei Zuständen. 25
bottom-up	Ist eine Vorgehensweise bei der Problemlösung. Zuerst werden Teilprobleme gelöst, mit deren Hilfe dann grössere, darüber liegende Probleme etc. Die einzelnen Teillösungen werden von unten nach oben zusammengesetzt, bis das Gesamtproblem gelöst ist. 17
Element	Datentyp welcher exklusiv für Prozesse in Simio bestimmt ist. 18
Entity	Ein Simio Objekt. 17 , 18 , 36
Facility	Simio Laufzeitumgebung eines Models //Korrekt?. 36
FactoryMethod	Design Pattern der Objektorientierten Programmierung welche das erstellen von Objekten kapselt. 35
Float	Datentyp der Fließkommazahlen speichert. 25
Integer	Datentyp der ganzzahlige Werte speichert. 25
Link	Verbindungselement in Simio. 18
Node	Ein Simio Objekt. 18
Property	Eigenschaften eines Simio Objektes welche zur Modellierungszeit definiert und festgelegt werden. Diese können zur Laufzeit nicht mehr verändert werden. 18 , 37
Scope	Rahmen in welchem Objekte, Daten und Funktionen (Prozesse in Simio) sichtbar sind. 18
Server	Ein Simio Objekt. 18 , 36
Sink	Ein Simio Objekt. 36
Source	Simio Objekt in welcher die Erstellung von Laufzeitobjekten konfiguriert und ausgeführt werden kann. 36

State	Eigenschaften eines Simio Objektes welche zur Modellierungszeit definiert werden. States können zur Laufzeit verändert werden. 37
Step	Objekt welches ein Ereignis in Simio repräsentiert welches gefeuert werden kann. Prozesse können auf das feuern eines Events in gang gesetzt werden.. 18
Step	Aktivität in einem Prozess in Simio. 18 , 19
String	Datentyp zur Repräsentation von Zeichenketten. 36
Token	Objekt welches für den Ablauf eines Prozesses in Simio benötigt wird. 18 , 19
Transporter	Ein Simio Objekt. 17 , 36
WinForms	.Net blibla. 35
Worker	Ein Simio Objekt. 65
Workstation	Ein Simio Objekt. 18

LITERATURVERZEICHNIS

- [1] Eduard Babulak und Ming Wang. *Discrete Event Simulation*. <http://cdn.intechopen.com/pdfs/11536.pdf>. [Online; letzter Zugriff 20.03.2016].
- [2] Andrei Borshchev. *The Big Book of Simulation Modeling: Multithread Modeling with AnyLogic 6*. AnyLogic North America, 2013.
- [3] Andrei Borshchev und Alexei Filippov. *From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools*. http://link.springer.com/chapter/10.1007/3-540-45982-0_1. [Online; letzter Zugriff 13.04.2016]. 2004.
- [4] George Fishman. *Discrete-Event Simulation*. Springer-Verlag New York, 2001.
- [5] Cynthia Nikolai und Gregory Madey. "Tools of the Trade: A Survey of Various Agent Based Modeling Platforms". In: *Journal of Artificial Societies and Social Simulation* 12.2 (2009), S. 2. ISSN: 1460-7425. URL: <http://jasss.soc.surrey.ac.uk/12/2/2.html>.
- [6] Dirk Pawlaszczyk. *Skalierbare agentenbasierte Simulation*. <http://link.springer.com/article/10.1007/s13218-010-0031-5/fulltext.html>. [Online; letzter Zugriff 15.03.2016]. 2009.
- [7] Wolf-Ulrich Raffel. *Agentenbasierte Simulation als Verfeinerung der Diskreten-Ereignis-Simulation*. <http://www.wuraffel.de/wissenschaft/publikationen/Dissertation.pdf>. [Online; letzter Zugriff 15.03.2016].
- [8] PO Siebers, CM Macal, J Garnett, D Buxton und M Pidd. "Discrete-Event Simulation is Dead, Long Live Agent-Based Simulation!" In: *Journal of Simulation*, 4(3) (2010).
- [9] Roger D. Smith. "Simulation Article". In: *Encyclopedia of Computer Science* (1988).
- [10] Michael Wooldridge. *Intelligent Agents: The Key Concepts*. Springer-Verlag Berlin Heidelberg, 2002.

ABBILDUNGSVERZEICHNIS

Abbildung 1	Arten ein System zu untersuchen.[4]	8
Abbildung 2	BPMN Flussdiagramm eines Prozesses mit parallelen Subprozessen	9
Abbildung 3	Loop Diagramm für die Einführung eines neuen Produktes	10
Abbildung 4	SD Simulationsresultat [3]	11
Abbildung 5	ABS Simulationsresultat [3]	11
Abbildung 6	Simio Objekthierarchie	15
Abbildung 7	Beispiel-Simulation in Anylogic	18
Abbildung 8	Dichtediagramm der Beispiel-Simulation in Anylogic	18
Abbildung 9	Kommunikation als Konzept für Agenten . . .	19
Abbildung 10	Arten der Wahrnehmung	21
Abbildung 11	Custom Stuff	27
Abbildung 12	Properties des Agenten aus Abbildung 6 . . .	28
Abbildung 13	Starten des Addin in Simio	33
Abbildung 14	Fehlermeldung wenn vergessen wurde das Addin zu starten	34
Abbildung 15	Fenster zum einfügen eines Hindernisses . . .	34
Abbildung 16	Aufbau des Prototyps	35
Abbildung 17	Der Visibility Reiter	35
Abbildung 18	Der Prozess im Model	35
Abbildung 19	Detailansicht des Travel Step	36
Abbildung 20	Ansicht der States des Entity Model	36
Abbildung 21	Prozess des Entity mit Assign Step	36
Abbildung 22	Detailansicht des Assign Step	37
Abbildung 23	Der laufende Prototyp	37
Abbildung 24	API des Steering Behavior	38
Abbildung 25	API des Steering Behavior Context	39
Abbildung 26	Reaktionsvektoren eines Agenten	40
Abbildung 27	Erkennung durch Reaktionsvektoren	40
Abbildung 28	Mehrere Reaktionsvektoren	41
Abbildung 29	Ausweichen eines Objektes	41
Abbildung 30	Installierte Simio Templates	56
Abbildung 31	Simio Libraries	56
Abbildung 32	Libraries in korrekten Zielordner kompilieren	57
Abbildung 33	Simio als externes Programm starten	58
Abbildung 34	XAML-Viewer beenden	58
Abbildung 35	Fremde Libraries zugänglich machen	59

TABELLENVERZEICHNIS

Tabelle 1	Attribute welche den Modelltyp definieren . . .	12
Tabelle 2	Arten der Wahrnehmung	22
Tabelle 3	Software für allgemeine agentenbasierte Anwendung [5]	49
Tabelle 4	Software für verteilte Simulation [5]	49
Tabelle 5	Software mit pädagogischem Fokus [5]	50
Tabelle 6	Software spezialisiert für Multi-Agent Systeme [5]	51
Tabelle 7	Software für Artificial Intelligence [5]	51
Tabelle 8	Software für Soziale Wissenschaften [5]	52
Tabelle 9	Software für Naturwissenschaften [5]	52
Tabelle 10	Software für sehr spezifische Anwendungsgebiete [5]	53

LISTINGS

Listing 1	Erstellung von Objekten	28
Listing 2	Hinzufügen eines States zum aktiven Model .	29
Listing 3	TravelSteeringMovement	38
Listing 4	ITravelSteeringBehavior Kalender	38
Listing 5	TravelSteeringMovement	39
Listing 6	TravelSteeringMovement	40
Listing 7	Feuern des InformatonUpdate Events	42

ACRONYMS

ABS	Agentenbasierte Simulation
API	Application Programming Interface
BPMN	Business Process Model and Notation
DES	Diskrete Eventsimulation
GUI	Graphical User Interface
SD	System dynamics
WPF	Windows Presentation Foundation