Universität Hamburg - Fachbereich Informatik - Arbeitsbereich VSIS

Ausarbeitung im Seminar
Ausgewählte Technologien verteilter Systemsoftware

Multiagenten-Simulation und Werkzeuge

2. August 2009

Markus Alexander Kuppe
8kuppe@informatik.uni-hamburg.de
1. Fachsemester M.Sc. Informatik
Matr.-Nr. 6095945

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Aufbau der Arbeit 1

2 Grundlagen **2**
2.1 Der Agent 2
2.2 Die Multiagentensimulation 3
2.2.1 Vorteile der Multiagentensimulation 3
2.2.2 Nachteile der Multiagentensimulation 4
2.2.3 Einsatzgebiete von Multiagentensimulation 4
2.3 Definition von ABMS-Werkzeuge 5
2.3.1 Abgrenzung zu Agenten-Plattformen 5
2.3.2 Bestandteile von ABMS-Werkzeugen 6
2.3.3 Klassifikation anhand von Gruppenzugehörigkeit 7

3 Vorstellung eines Werkzeugs zur agentenbasierten Modellierung und Simulation **9**
3.1 REpast Simphony 9

4 Vergleiche von ABMS-Werkzeugen **10**
4.1 Vorstellung von Vergleichsstudien 11
4.2 Akkumulierte Werkzeug-Betrachtung 14

5 Zusammenfassung und Ausblick **15**

1 Einleitung

Multiagentensimulation (MAS) ist eine neue Simulationstechnik, die erst durch die weite Verbreitung von Desktop-Computern populär und nutzbar wurde (Macal und North (2008)). Ihre Verwendung findet sie jedoch nicht nur bei Informatikern, sondern kommt in diversen wissenschaftlichen Bereichen zum Einsatz . Sie erlaubt es, hochgradig komplexe Modelle der Makroebene zu simulieren, die auf der Mikroebene modelliert werden können (Klügl (2006)). Es adressiert damit nach Tobias und Hofmann (2004) das Wechsel-Problem zwischen der Makro- und der Mikroebene.
Axelrod (1997) geht so weit, Multiagentensimulation als dritte Art des wissenschaftlichen Arbeitens zu bezeichnen, die sich von reiner Deduktion und Induktion unterscheidet. Castle und Crooks (2006) führt dazu aus, dass das in der Multiagentensimulation der Simulation zugrunde liegende Modell zwar auf deduktiven Wege konstruiert sein kann, die aus der Simulation gewonnenen Daten jedoch nicht zum induktiven Schließen genutzt werden können. Dies begründet sich in der Tatsache, dass die Daten nicht Messungen der tatsächlichen Welt sind, sondern aus einer Simulation der Real-Welt entstammen.

Um die oben erwähnten Eigenschaften der Multiagentensimulation zu nutzen, bedarf es passender Werkzeuge. Sie müssen zum einen die benötigte Funktionalität bieten, sollen aber gleichzeitig einfach in der Handhabung sein[1] (Castle und Crooks (2006)). Die große Anzahl an verfügbaren Lösungen erleichtert die Auswahl nicht. Jedoch können verschiedene Vergleiche helfen, Agentenbasierte Modellierungs- und Simulations-Werkzeuge (ABMS-Werkzeug) zu bewerten und das Richtige auszuwählen.

1.1 Aufbau der Arbeit

Die Ausarbeitung gliedert sich in drei wesentliche Abschnitte. In den Grundlagen (siehe Abschnitt 2) werden die Begriffe Multiagentensimulation und ABMS-Werkzeuge eingeführt und erläutert. In Abschnitt 3 wird ein ABMS-Werkzeug kurz vorgestellt, welches im dritten Teil (siehe Abschnitt 4) in die Vergleiche von ABMS-Werkzeugen eingeht. Am Schluss steht die obligatorische Zusammenfassung sowie ein Ausblick auf die mögliche Zukunft von ABMS-Werkzeugen.

[1] Insbesondere Programmier-Wissen sollte zur Modell-Erstellung nicht nötig sein.

2 Grundlagen

Die Ansprüche an Simulationstechniken nehmen ständig zu. Dies begründet sich zum einen in den immer komplexeren und umfangreicheren Sachverhalten, die es zu simulieren gilt. Zum anderen in der zunehmenden Verfügbarkeit und Detailliertheit an Rohdaten, die den Simulationen zugrunde liegen (North und Macal (2007)). Parallel dazu nehmen die Kosten für Speicherplatz und Rechenleistung stetig ab. Daher rücken neue Simulationstechniken in den Vordergund, die die sonst zu treffenden a priori Annahmen unnötig machen und ein stabiles Optimum liefern (North und Macal (2007)).
Eine dieser Simulationstechniken ist die Multiagentensimulation, auf die über den Begriff "Agent" in den nächsten Abschnitten eingegangen werden soll. Anschließend wird im zweiten Teil, zur Vorbereitung auf die Abschnitte über ABMS-Werkzeuge, eine Definition von Werkzeugen zur Multiagentensimulation geliefert.

2.1 Der Agent

Bevor im Folgenden auf die Multiagentensimulation eingegangen werden kann, muss vorher ein klares Verständnis über Agenten erlangt werden. Die folgende Beschreibung bezieht sich nur auf die in Simulationen verwendeten Agenten und erhebt nicht den Anspruch, Agenten allgemein zu definieren. Für eine vollständige Definition des Begriffs Agent sei daher auf Wooldridge (2001) verwiesen.
Ein Agent ist in einer Simulation der Stellvertreter für den in der Realität existierenden Akteur. Demnach wird er, dem realen Akteur folgend, modelliert. Der Agent handelt autonom und verfolgt in der Simulation eigenen Zielen. Ein Agent zeichnet sich durch ein nicht triviales Verhaltensrepertoire aus. Dieses Verhalten klassifiziert man als proaktiv, d.h. der Agent führt unabhängig von externen Einflüssen verschiedene Verhaltensmuster aus, oder reaktiv. Reaktives Verhalten reagiert dagegen auf externe Einflüsse, die durch Sensoren wahrgenommen werden. Die externen Einflüsse manipuliert der Agent durch Effektoren.
Externe Einflüsse werden allgemein unter dem Begriff der Umwelt zusammengefasst. Die Umwelt setzt sich nicht nur aus weiteren Agenten zusammen, sondern auch aus (dynamischen) Ressourcen. Im Gegensatz zu Agenten haben Ressourcen jedoch keine aktive Komponente und sind daher passiv. Ein Agent interagiert und kommuniziert mit seiner Umwelt direkt oder indirekt (Castle und Crooks (2006)). Eine mögliche Realisierung der Umwelt wird über Zelluläre Automaten erreicht. Sie unterteilt sich in (diskrete) Zellen mit einem jeweiligen Zustand. Das Wahrnehmungsvermögen eines Agentens ist lokal begrenzt. Er nimmt somit nicht seine gesamte Umwelt wahr (alle Agenten und Ressourcen),

sondern lediglich benachbarte Bereiche. Trotzdem lassen sich auch globale Systemzustände modellieren, die auf den Agenten einwirken.
Innerhalb der Umwelt ist der Agent mobil und kann sich frei bewegen. Der Bewegungsradius richtet sich dabei nach den Eigenschaften des Agentens, sowie der Zeit und Verfügbarkeit von Nachbarzellen[2] (Klügl (2006)).

2.2 Die Multiagentensimulation

Multiagentensimulation (MAS) nutzt das Agentenparadigma, um in einer simulierten Umwelt und unter Ablauf von virtueller Zeit (diskret oder kontinuierlich) das Verhalten einzelner Agenten innerhalb der Gruppe oder das Verhalten einer heterogenen Gesamtpopulation zu untersuchen (da Silva und de Melo (2008)). Die Modellierung erfolgt dabei, wie in 2.1 beschrieben, auf der Ebene des einzelnen Agentens und somit auf der Mikroebene. Dies sieht Klügl (2006) als entscheidenden Unterschied zu traditionellen Simulationstechniken, deren Modellierung auf der Makroebene erfolgt.
Die Makroebene bezeichnet hierbei die Ebene, auf der das zu untersuchende Phänomen beobachtet werden kann. Die Mikroebene umfasst dagegen die Agenten und deren Umwelt. Weiterhin nennen North und Macal (2007) iteraktives Vorgehen bei der Modellerstellung von MAS als charakteristisch.

2.2.1 Vorteile der Multiagentensimulation

Die Modellierung auf der Mikroebene bietet laut North und Macal (2007); Klügl (2006) eine Reihe von Vorteilen gegenüber Simulationstechniken (Differentialgleichungsmodelle, System Dynamics), die auf der Makroebene modelliert werden. Insbesondere die den traditionellen Simulationen zugrunde liegenden Annahmen über Homogenität, wie über die zu modellierenden Akteure und deren Umwelt, ist bei MAS nicht notwendig. Vielmehr lässt sich das Verhalten der Agenten und die Interaktion untereinander direkt modellieren und muss nicht über globale Systemvariablen abgebildet werden. Diese Modell-Beschreibungen können dann parallel zur Simulation angepasst bzw. modifiziert werden, bis alle nötigen Faktoren abgebildet sind und das erwartete (emergente) Verhalten auf der Makroebene auftritt. Der Detailierungsgrad des Modells ist dabei unbegrenzt. Dieser potentiell sehr weitgehende Detaillierungsgrad wird von Klügl (2006) dann auch als ein entscheidender Nachteil von MAS gewertet.

[2] Eine Zelle kann durch andere Agent oder Ressourcen blockiert sein.

2.2.2 Nachteile der Multiagentensimulation

Mit jeder Steigerung des Detaillierungsgrad wird das Modell komplexer. Es bekommt weitere Modell-Variablen. Dies lässt zum einen die Anforderungen an Rechenkapazitäten wachsen. Zum anderen bedingt ein höherer Detaillierungsgrad weitere Modell-Parameter, deren optimale Einstellungen gefunden werden müssen. Als weiteren Nachteil ist die Schwierigkeit zu nennen, während der Modell-Entwicklung von der Makroebene auf die Mikroebene über zu leiten. Die der Simulation zugrunde liegenden empirischen Daten oder das eigentliche Phänomen beschreiben lediglich die Makroebene. Daten oder Beobachtungen helfen jedoch nicht bei der Modellierung auf der Mikroebene. Daher ist die direkte Herleitung der Mikroebene aus der Makroebene im Top-Down Verfahren auf Grund der nicht-linearen Zusammenhänge nicht möglich. Die Modellierung muss daher im Bottom-Up Verfahren im Try-And-Error Modus solange angepasst werden, bis die Makroebene das gewünschte Verhalten zeigt. Trotz der genannten Nachteile findet MAS in diversen Bereichen einen Einsatz.

2.2.3 Einsatzgebiete von Multiagentensimulation

Nikolai und Madey (2009b) sehen Informatiker nicht als einzige Nutzer von MAS. Insbesondere Forscher Informatik-fremder Fachgebiete werden als Nutzer genannt. Eine zweite Gruppe bilden Lehrende, die MAS zu Lehrzwecken einsetzen (Serenko und Detlor (2002)). Diesen beiden Gruppen ist gemein, daß nur ein begrenztes Programmier-Wissen vorhanden ist. Die Hauptanwendungsgebiete für Multiagentensimulation benennt Klügl (2006). Demnach wird MAS in der Soziologie zur Untersuchung von gesellschaftlichen Phänomenen eingesetzt, in der Biologie und Ökologie zur Verhaltensstudie oder in marktwirtschaftlichen Bereichen. Hier dient MAS vorrangig zur Marktsimulation. North und Macal (2007) widmen der Marktsimulation und insbesondere der strategischen Planung sogar eine umfassende Betrachtung. Aus dieser entstammt auch eine Anwendungsgebietsunabhängige Liste, wann sich MAS eignet:

- Wenn ein Hauptteil der zu modellierenden Realität die Interaktion von Akteuren umfasst.
- Wenn es Entscheidungen und Verhalten gibt, die diskretisiert werden können.
- Wenn Akteure ihr Verhalten an die Umwelt anpassen.
- Wenn Lernen und strategisches Verhalten der Akteure eine Rolle spielt.
- Wenn dynamische Gruppen zwischen einzelnen Akteuren entstehen.

- Wenn Lernen und Adaption auch auf der Gruppenebene wichtig sind.
- Wenn Raum einen Einfluss auf das Verhalten und die Interaktion von Akteuren hat.
- Wenn aus der Vergangenheit keine Schlüsse auf die Zukunft geschlossen werden können.
- Wenn die Skalierbarkeit des Modells entscheidend ist und die Skalierbarkeit die Anzahl von Akteuren und Interaktion zwischen Akteuren meint.
- Wenn emergentes Verhalten das Ziel der Simulation ist, anstatt ein Eingabeparameter des Modells zu sein.

2.3 Definition von ABMS-Werkzeuge

Nachdem der vorherige Grundlagenabschnitt ein Verständnis über die Multiagentensimulation gegeben hat, sollen im kommenden Teil die Werkzeuge zur Multiagentensimulation definiert werden.
Die Literatur hält jedoch keine allgemeingültige Definition für Werkzeuge zur Agentenbasierten Modellierung und Simulation (ABMS-Werkzeugen) bereit. Eine Ursache hierfür ist die Betrachtung der Werkzeuge durch die verschiedenen Nutzer- bzw. Anwendergruppen (Castle und Crooks (2006)). Daher werden in dieser Seminararbeit drei unterschiedliche Definitionen vorgestellt, mit deren Hilfe eine eindeutige Bestimmung von ABMS-Werkzeugen möglich wird.

2.3.1 Abgrenzung zu Agenten-Plattformen

Im Gegensatz zu Agenten-Plattformen (vgl. Braubach (2007)) sind ABMS-Werkzeuge nicht zur Entwicklung von Agenten-Anwendungen geeignet. Dies äußert sich in mehreren Eigenschaften der ABMS-Werkzeuge. ABMS-Werkzeuge bieten keinen Zugriff auf einzelne Agenten-Instanzen (Gilbert und Bankes (2002)). Die Agenten existieren lediglich innerhalb der Simulation und haben keine Mobilität, um in eine andere Simulation zu migrieren. Technisch sind Agenten-Instanzen in ABMS-Werkzeugen nicht durch einzelne Prozesse oder Threads realisiert. Aktionen werden hingegen pro Zeiteinheit, sequentiell über alle Agenten vom Simulator abgearbeitet. Durch diese Eigenschaften kann ein ABMS-Werkzeug zu jedem Zeitpunkt den Gesamt-Systemzustand über explizite Daten-Schnittstellen liefern. Dies ist vor dem Simulations-Hintergrund zwingend erforderlich. Über die Schnittstellen lassen sich Ergebnisse und Eingabeparameter leicht

in Auswertungs-Werkzeuge, wie zum Beispiel statistische Werkzeuge, übernehmen, sofern die integrierten Auswertungs- und Visualisierungs-Funktionen nicht ausreichen. Im Regelfall bieten ABMS-Werkzeuge weiterhin Unterstützung zum kontrollierten Probieren der Eingabewerte ("parameter swapping"). Abschließend läßt sich festhalten, dass ABMS-Werkzeuge, im Gegensatz zu den universellen Agenten-Plattformen, domänenspezifisches Wissen, wie beispielsweise aus der Soziologie, Biologie oder Medizin enthalten (Klügl (2001)).
Vor dem Hintergrund hybrider Ansätze wie JADEX (vgl. Braubach u. a. (2005)) hält diese Abgrenzung jedoch nicht stand. Dort finden sich sowohl Eigenschaften von Agenten-Plattformen, als auch von ABMS-Werkzeugen.

2.3.2 Bestandteile von ABMS-Werkzeugen

Klügl (2001) nennt die wesentlichen Komponenten eines ABMS-Werkzeugs am Beispiel von SeSAm (Universität Würzburg (2009)). Demnach setzt sich ein ABMS-Werkzeug aus einer (grafischen) Modellierungs-Oberfläche, einem Simulator und sowohl einer Einheit zur Visualisierung des Experiments zur Laufzeit als auch einer Auswertungsroutine zusammen (siehe Abbildung 1). Die *Modellierungs-Oberfläche* dient der Modellierung der Agenten als auch der Umwelt. Der Modellierer kann dabei auf generische, aber auch domänenabhängige Komponenten zurück greifen, um das Modell zu beschreiben. Allgemein stehen dem Modellierer entweder eine grafische Oberfläche zur Erstellung des Modells zur Verfügung, in der alle Bausteine der visuellen Programmierung zum Einsatz kommen können, oder eine klassische Programmierumgebung. Der *Simulator* ist die Ausführungeskomponente des Modells. Er führt das Experiment (schrittweise) durch und protokolliert dabei die *Experimentierdaten*. Währenddessen dient die *Experimentvisualisierung* der Beobachtung des laufenden Experiments. Sie bietet dazu verschiedene Darstellungen wie zwei- oder dreidimensionale Ansichten der Umwelt und der in ihr interagierenden Agenten. Im Gegensatz zur Experimentvisualisierung werden die *Auswertungsroutinen* zur späteren Analyse der Experimentierdaten genutzt. Dabei kommen beispielsweise statistische Werkzeuge zum Einsatz, denen die Experimentdaten durch Ausgabe-Schnittstellen übergeben werden. Die Auswertungsroutinen bilden häufig eine "Toolchain", in der die Ausgabe-Schnittstelle eines Auswertungs-Werkzeugs mit der Eingabe-Schnittstelle eines weiteren Auswertungs-Werkzeugs verbunden ist.

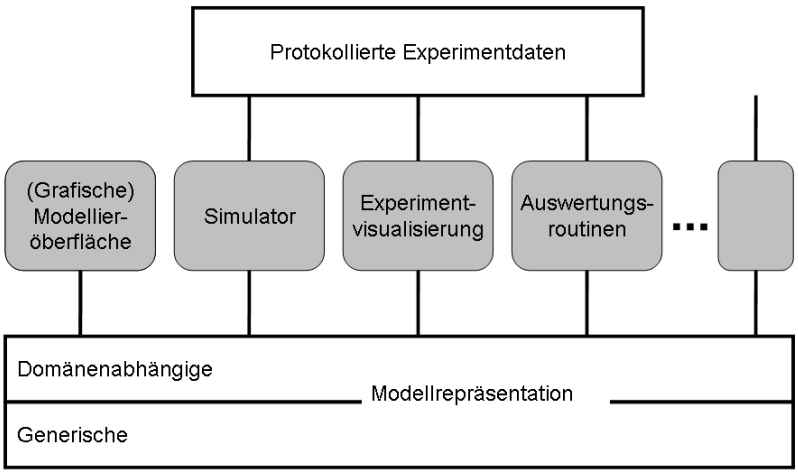

Abbildung 1: Bestandteile eines ABMS-Werkzeug nach Klügl (2001).

2.3.3 Klassifikation anhand von Gruppenzugehörigkeit

Macal und North (2008) ordnen ABMS-Werkzeuge ausgehend von ihren Charakteristika drei Klassen zu. Die Klassen unterteilen die Werkzeuge dabei anhand ihrer Mächtigkeit bzw. Universalität in:

Desktop Computing Werkzeuge die der Klasse des Desktop Computing zugeordnet werden, zeichnen sich durch die Möglichkeit zum Rapid Prototyping, also der schnellen und inkrementellen Entwicklung von Modellen aus. Ferner dienen sie häufig als Lernwerkzeug und benötigen daher einen minimalen Einarbeitungsaufwand. Jedoch sind sie in der Leistungsfähigkeit begrenzt, was sich direkt auf die mögliche Größe einer Simulation[3] auswirkt. Darüber hinaus ist ihr Funktionsumfang eingeschränkt. Sie eignen sich daher nicht für allen Arten von MAS (vgl. 2.2.3).

Large Scale ABMS Im Gegensatz zu Desktop-basierten ABMS-Werkzeugen steht bei Large Scale ABMS-Werkzeugen die Skalierbarkeit im Vordergrund. Die Stellvertreter dieser Gruppe unterstützen große Experimente mit einer Vielzahl von Agenten. Dies wird nicht nur durch die Möglichkeit zur Verteilung auf mehrere Rechner erreicht, sondern auch durch eine optimierte Implementierung der Werkzeuge in Bezug auf Performanz. Ein weiterer wesentlicher Klassifikator ist die Kombinierbarkeit der Werkzeuge. Die Experimentdaten lassen sich zwischen verschiedenen Werkzeugen austauschen und verarbeiten. Es entsteht eine Toolchain. Daneben

[3] Merkmale für die Größe einer Simulation sind die Größe der Umgebung und die Anzahl der Agenten.

steht die Erweiterbarkeit der Werkzeuge, die durch dafür vorgesehene Schnittstellen möglich ist. Im Gegensatz zur Kombinierbarkeit erfordert diese Möglichkeit Programmierkenntnisse, womit es nicht für alle Nutzer eines ABMS-Werkzeugs relevant ist. Ein offensichtlicher Unterschied zu Desktop-basierten Werkzeugen ist die Unterstützung für grafische Modellierung. Das Simulations-Modell muss nicht in einer speziellen Modellierungssprache oder einer allgemeinen Programmiersprache beschrieben werden, sondern kann anhand einer grafischen Oberfläche konstruiert werden. Ein Nachteil der Large Scale ABMS-Werkzeuge ist die durch den größeren Funktionsumfang entstehende Lernkurve. Die Werkzeuge lassen sich nicht mehr intuitiv beherrschen, sondern erfordern eine Einarbeitung.

General Puropse Programming Languages Die letzte Klasse bilden die allgemeinen Programmiersprachen. Allgemeine Programmiersprachen bieten, verglichen mit den beiden vorherigen Klassen, die größte Flexibilität in Bezug auf Modellierungs-Möglichkeiten. Diese Flexibilität kommt jedoch nicht ohne Preis daher. Die Aufwände und Kosten für die Modellierung steigen (Gilbert und Bankes (2002)). Dies begründet sich in den zur Modellierung zur Verfügung stehenden Sprachkonstrukten, die analog zu Assembler-Sprachen verglichen mit Hochsprachen, rudimentär sind. Darüber hinaus sind die erstellten Modelle wegen ihres Umfangs anfälliger für Fehler, was im schlechtesten Fall direkte Auswirkungen auf die Ergebnisse der Modellierung hat (Castle und Crooks (2006)). Ebenso eignen sie sich nur für Anwender mit Programmierkenntnissen. Diese sind vor dem Hintergrund der weiten Verbreitung von MAS (vgl. 2.2.3) jedoch eine Minderheit.

Zusammenfassend zeigt Abbildung 2 den Zusammenhang zwischen der gebotenen Flexibilität und Mächtigkeit bei den Modellierungs-Möglichkeiten der jeweiligen Klassen und der damit verbundenen Komplexität, die zur Erstellung eines Simulations-Modells notwendig ist. Je weniger Funktionalität vom Werkzeug bereit gestellt wird, desto leichter und schneller lässt sich eine Simulation durchführen. Jedoch stößt man bei der Modellierung mit Desktop-basierten Werkzeugen schnell an ihre Grenzen. Dies geschieht sowohl bei den Modellierungs-Möglichkeiten, als auch bei der Rechenleistung und der damit verbundenen Experiment-Größe Macal und North (2008). Tabelle 1 gibt einen Überblick über bekannte Stellvertreter der jeweiligen Klasse.

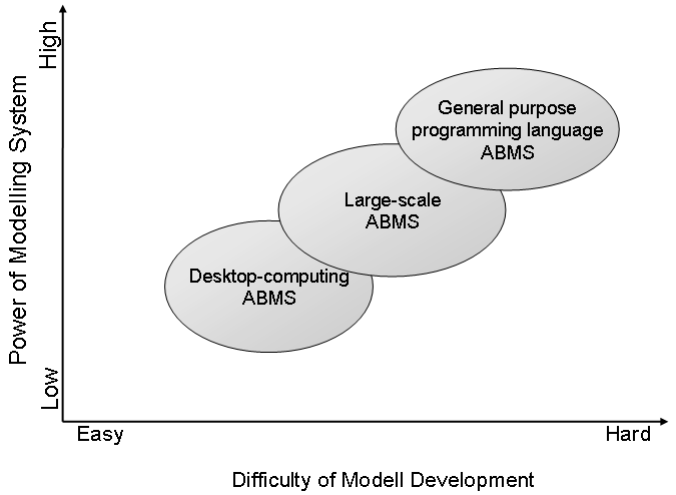

Abbildung 2: Die Mächtigkeit eines ABMS steigt mit der Klassenzugehörigkeit nach Macal und North (2008). Ebenso nimmt die Komplexität der Modellentwicklung zu (Samuelson und Macal (2006)).

3 Vorstellung eines Werkzeugs zur agentenbasierten Modellierung und Simulation

In diesem Kapitel soll das ABMS-Werkzeug REPAST SIMPHONY (Recursive Porous Agent Simulation Toolkit) in Version 1.2 kurz vorgestellt werden. Dabei wird den genannten Bestandteilen eines ABMS-Werkzeugs aus Abschnitt 2.3.2 gefolgt. Vorgreifend sei gesagt, dass REPAST SIMPHONY in Abschnitt 4 als eines der universellsten ABMS-Werkzeuge genannt wird.

3.1 REpast Simphony

REPAST SIMPHONY wird als Free and Open-Source Software (vgl. Wikipedia (2009)) entwickelt. Dort ist das Werkzeug weit verbreitet (Macal und North (2008)). Es basiert auf Java ab der Version 1.4 und ist daher Plattform-unabhängig. Es unterliegt einer liberalen BSD-ähnlichen Lizenz und eignet sich deshalb auch zum Einsatz in kommerziellen Umfeldern. REPAST SIMPHONY wird seit 2007 als Weiterentwicklung von REPAST fortgeführt. REPAST entstammt dem Deparment "Social Science Research Computing" der Universität von Chicago und wurde im Jahre 2000 erstmalig der Öffentlichkeit vorgestellt. Seit Beginn der Entwicklung können Agenten und Umwelt frei in Java modelliert werden. Neben Java bietet REPAST SIMPHONY zusätzlich Groovy als Programmiersprache an.

Desktop Computing	Large Scale ABMS	Programming Languages
Tabellenkalkulationen unter Verwendung von VBA bspw. Excel, (REpast Simphony), NetLogo, StarLogo, MATLAB, Mathematica	REpast, Swarm, MASON, AnyLogic, SeSAm	Java, C++, Python...

Tabelle 1: Bekannte ABMS-Werkzeuge zugeordnet zu den drei Klassifikatoren nach Macal und North (2008). Die Liste umfasst nicht alle existierenden ABMS-Werkzeuge. REpast Simphony ordnet Macal und North (2008) zu Desktop-basierten Werkzeugen. Dieser Zuordnung folgt diese Ausarbeitung nicht.

Darüber hinaus wird eine Möglichkeit zur visuellen Programmierung angeboten, wodurch eine deklarative Modellrepräsentation möglich wird. Abbildung 3 zeigt einen grafischen Editor, mit dem das Verhalten von Agenten beschrieben wird[4]. Alle Bestandteile der Modellierungsoberfläche sind in die Eclipse IDE (Eclipse Foundation (2009)) eingebettet. Somit steht dem programmier-erfahrenen Modellierer eine mächtige Entwicklungsumgebung zur Verfügung.
Der Simulator und die Experimentvisualisierung sind hingegen nicht in Eclipse integriert, sondern sind eigenständige Anwendungen (siehe Abbildung 4). Diese Anwendung dient zur Parametrisierung sämtlicher Simulationen. Weiterhin unterstützt sie "parameter swapping". Die Experimentvisualisierung kann entweder mit zwei- oder dreidimensionalen Räumen dargestellt werden. Sollten die vorhandenen Visualisierungs-Möglichkeiten nicht ausreichen, können eigene programmiert werden.
Auswertungsroutinen werden in REpast Simphony über Datenschnittstellen angebunden, wodurch eine Verbindung zu verschiedenen Auswertungs-Werkzeugen möglich ist.

4 Vergleiche von ABMS-Werkzeugen

Die Wahl eines passenden ABMS-Werkzeugs ist vor dem Hintergrund der großen Anzahl an möglichen Kandidaten schwierig. Samuelson und Macal (2006) argumentieren ferner, dass ein einziges ABMS-Werkzeug nicht alle Anforderungen erfüllen kann. Stattdessen schlagen sie vor, die Modell-Entwicklung in mehrere Phasen zu gliedern. In den ersten Phasen kommt ein einfaches ABMS-Werkzeug aus der Kategorie der Desktop-basierten

[4] Vor der Ausführung einer Simulation wird das deklarative Modell mit Code-Generatoren in Java umgewandelt. Dort kann es im Forward-Engineering weiter editiert werden.

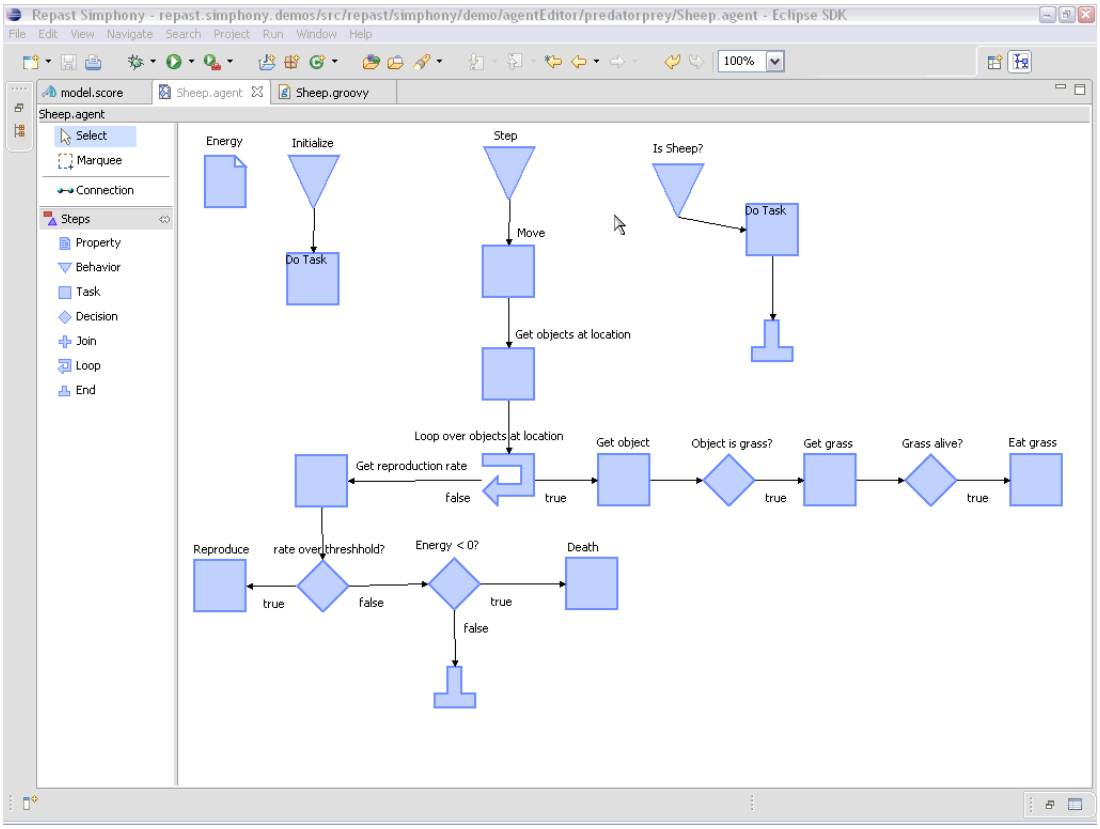

Abbildung 3: Visuelle Programmierung mit REPAST SIMPHONY. Die Abbildung zeigt den Verhaltens-Editor eines Agenten, der innerhalb der ECLIPSE IDE eingebettet ist.

ABMS zum Einsatz, in weiteren Phasen dann Large Scale ABMS (siehe 2.3.3). Dadurch lassen sich die Vorteile beider Kategorien ausnutzen.
Jedoch schränkt diese Vorgehensweise die Anzahl an Kandidaten nicht ein. Daher sollen an dieser Stelle vier Studien über ABMS-Werkzeuge vorgestellt werden, um anschließend eine zusammenfassende Bewertung zu geben. Eine eigene Studie würde den Rahmen dieser Ausarbeitung sprengen und bedarf darüber hinaus mehrjährige Kenntnis der zu untersuchenden Werkzeuge (Tobias und Hofmann (2004)).

4.1 Vorstellung von Vergleichsstudien

Im Bezug auf ABMS-Werkzeuge hält die Literatur eine Reihe von Vergleichen bereit. Ein solcher Vergleich versucht die zu untersuchenden ABMS-Werkzeuge unter Zuhilfenahme von bestimmten Merkmalen objektiv zu bewerten oder zumindest zu klassifizieren. Dies

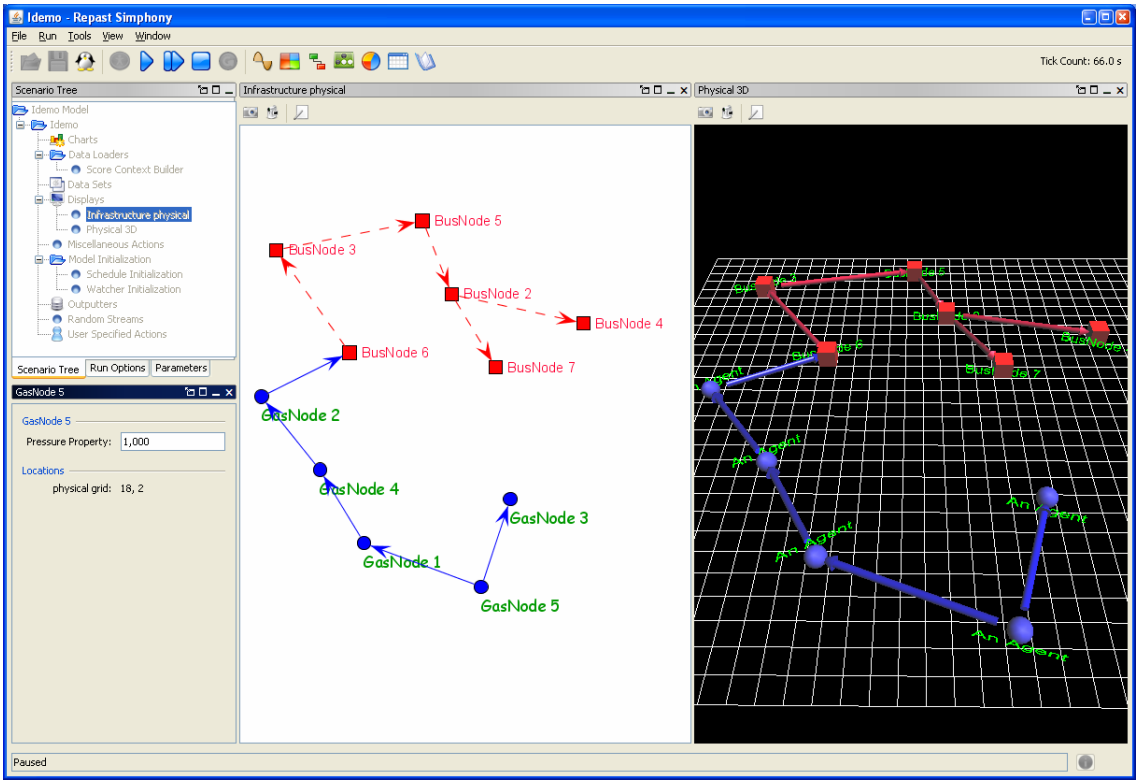

Abbildung 4: Simulator und Experimentvisualisierung sind bei REPAST eine eigenständige Anwendung. Diese Anwendung ermöglicht die Parametrisierung sämtlicher Simulation-Parameter. Die Experimentvisualisierung kann zwei- und dreidimensional dargestellt werden.

erfolgt im optimalen Fall über die Gesamtheit aller ABMS-Werkzeuge und viele Merkmale. Auf Grund von begrenzten Ressourcen ist ein solch idealisierter Vergleich jedoch selten möglich. Vielmehr muss ein Kompromiss zwischen Umfang und Merkmalen getroffen werden.

Tobias und Hofmann (2004) betrachten vier freie ABMS-Werkzeuge: REPAST (University of Chicago (2009)), SWARM (Swarm Development Group (2009)), QUICKSILVER (Bruse (2009)) und VSEIT (Brassel (2009)). Frei beschränkt sich in diesem Zusammenhang auf die Lizenz-Kosten der Werkzeuge und nicht, auf die Öffentlichkeit des Quellcodes. Dies erfolgt unter der Argumentation, dass nur freie Werkzeuge einen Austausch von Modellen unter Wissenschaftlern erlaubt. Neben der Lizenz begrenzen sie die Auswahl auf ausschließlich JAVA-basierte ABMS-Werkzeuge. Die Begründung liegt in der Standardisie-

rung der JAVA Sprache unter der Annahme, dass Modelle vom Modellierer programmiert werden müssen und dies durch eine standardisierte Sprache leichter wird. Die Anwendungsdomäne, anhand dessen die Werkzeuge verglichen werden, ist soziale Simulation. Es werden 17 verschiedene Merkmale in den Vergleich einbezogen. Die Studie nennt abschließend "REPAST [...] to be the clear winner".

In einer weiteren Studie betrachten Castle und Crooks (2006) acht ABMS-Werkzeuge: SWARM, MASON (George Mason University (2009)), REPAST/REPAST SIMPHONY, STARLOGO (Massachusetts Institute of Technology (2009)), NETLOGO (Northwestern University (2009)), OBEUS (Tel Aviv University (2009)), AGENTSHEETS (AgentSheets Inc. (2009)) und ANYLOGIC (XJ Technologies (2009)). Im Gegensatz zu Tobias und Hofmann (2004) beschränken sie die Vorauswahl nicht auf freie Lizenzen oder JAVA-basierte ABMS-Werkzeuge. Die Entstehung der Vorauswahl wird nicht näher erläutert mit der Ausnahme, dass die zum Zeitpunkt der Studie populärsten Werkzeuge ausgewählt wurden. Die Anwendungsdomäne ist die Simulation geographischer Informations-Systeme. Aus dieser Perspektive werden sechs Merkmale untersucht. Die Autoren küren keinen Gewinner, heben aber die Vorteile Quell-offener Werkzeuge hervor. Nur wenn der Quellcode eines Werkzeugs offen liegt, lassen sich Modelle validieren.

Aus dem selben Jahr stammt die Studie von Railsback u. a. (2006), die die Implementierung eines Beispiel-Modells mit fünf verschiedenen ABMS-Werkzeugen durchführt. Dabei wird die Komplexität des Modells sukzessive erhöht, um das jeweilige ABMS-Werkzeug zu prüfen. Die Werkzeug-Auswahl umfasst wieder REPAST, MASON, NETLOGO, SWARM. Lediglich SWARM J, eine Anbindung von Swarm an JAVA mit dem Java Native Interface, ist ein vorher nicht genanntes Werkzeug. Auch in diesem Fall wird keine Aussage zum Entstehen dieser Auswahl getroffen. Sieht man die Anzahl der Iterationen als Kriterien, in der die Komplexität des Ausgangs-Modells mit jedem Schritt stieg, wurden 15 Merkmale verglichen. Ein Hauptaugenmerk der Autoren lag insbesondere auf der Ausführungsgeschwindigkeit der Simulation. Zusammenfassend treffen die Autoren die Aussage, dass "REPAST [...] a good choice for many [...]" ist. REPAST eignet sich grundlegend für viele Anwendungsfälle. Die Autoren monieren lediglich den Einarbeitungsaufwand, der für REpast erforderlich ist. Diese Kritik wurde mit REPAST SIMPHONY teilweise adressiert, da bei der Entwicklung dieses Werkzeugs das Zusammenbringen von Modellierer und Entwickler im Vordergrund stand (North u. a. (2005b)).

Die umfassendste Betrachtung stammt von Nikolai und Madey (2009b). In dieser Stu-

die wurden 53 ABMS-Werkzeuge untersucht. Die Autoren beanspruchen, dass dies die Gesamtheit aller ABMS-Werkzeuge zum Zeitpunkt der Studie war. Unter dieser großen Anzahl an Werkzeugen leidet die Tiefe, anhand derer die Kandidaten verglichen werden. Lediglich fünf Merkmale werden betrachtet, die ohne Einarbeitung in das jeweilige Werkzeug bestimmt werden können. Daher trifft diese Studie keine Gewichtung oder Aussage über Gewinner, sondern gibt lediglich eine Orientierungshilfe für ABMS-Nutzer.
Auf dieser Arbeit basierend haben die Autoren eine Web-Suche für ABMS-Werkzeug aufgebaut (Nikolai und Madey (2009a)), die anhand von Kriterien passende Werkzeuge nennt. In ihrer Arbeit planen die Autoren, die der Suchmaschine zugrunde liegende Datenbank kontinuierlich auszubauen.

4.2 Akkumulierte Werkzeug-Betrachtung

Basierend auf den in 4.1 genannten Studien wurden im Rahmen dieser Ausarbeitung eine akkumulierte Betrachtung über vier ABMS-Werkzeuge erstellt. NetLogo und REpast Simphony zählen Macal und North (2008) zu den Desktop-basierten, REpast, Swarm und MASON zu den Large Scale Werkzeugen. Wie bereits erwähnt (siehe Tabelle 1), folgt diese Ausarbeitung nicht der Zuordnung von REpast Simphony zu Desktop-basierten ABMS. Daher können REpast und REpast Simphony vereinfacht als unterschiedliche Versionen des selben Werkzeugs betrachtet und somit als ein Werkzeug zusammen gefasst werden. Im Folgenden wird daher lediglich von REpast Simphony gesprochen.

Die Aussagekraft der akkumulierten Betrachtung ist durch eine Reihe von Einschränkungen abgeschwächt. So unterscheiden sich die Werkzeug-Versionen, auf die die eingangs erwähnten Studien basieren. Teilweise wurden Kritiken von den Werkzeug-Entwicklern bereits in neueren Versionen aufgegriffen. Die Tabelle 2 trifft hingegen nur Aussagen über ältere Werkzeug-Versionen. Ferner treffen nicht alle Studien quantifizierbare Aussagen (Noten). In diesem Fall wurde die Tendenz, die die jeweiligen Autoren nennen, subjektiv quantifiziert. Alle Betrachtungen gehen anteilig in die akkumulierte Betrachtung ein.

Aus der Tabelle 2 lässt sich ablesen, dass NetLogo Stärken in den Bereichen zeigt, die für Desktop-basierte ABMS-Werkzeuge typisch sind. Diese werden auch in Abschnitt 2.3.3 genannt. Genau diese Merkmale zeichnen sich bei den drei verbleibenden Werkzeugen als Schwachpunkte aus. Bezieht man die subjektive Betrachtung des Autors dieser Arbeit ein, holt REpast Simphony aber in diesen Bereichen auf. Dies ist unter Umständen eine Ursache, weshalb Macal und North (2008) das Werkzeug zur Gruppe der

	NetLogo	REpast	Swarm	MASON
Rapid Prototyping	+		-	
Werkzeug-Umfang		-/(+)	+	-
Lernkurve	+			–
Programmierkenntnisse	+	–/(-)	–	–
Dokumentation	+	-/(0)	-	-
Performanz		+	0	++
Speicherverbrauch/Anz. Agenten		+	0	++
Skalierbarkeit	-	+		
Mächtigkeit	-	(+)		
Quelloffen	-	+	+	+

Tabelle 2: Akkumulierte Werkzeug-Bewertungen nach Tobias und Hofmann (2004); Castle und Crooks (2006); Railsback u. a. (2006); Nikolai und Madey (2009b). Die Klammern kennzeichnen subjektive Wertungen des Autores dieser Ausarbeitung. Leere Zellen resultieren aus fehlenden Aussagen der zugrunde liegenden Studien.

Desktop-basierten ABMS-Werkzeuge zählt. Verglichen mit Swarm und MASON zeigt REpast Simphony das ausgewogenste Verhältnis bei den Merkmalen der Large Scale ABMS. Aus diesem Grund folgt diese Ausarbeitung der Empfehlung von Railsback u. a. (2006) und Tobias und Hofmann (2004) und sieht REpast Simphony grundsätzlich als richtige Wahl. Insbesondere wenn die Anforderungen nicht hinlänglich bekannt sind und keine Zeit zur umfangreichen Betrachtung der einzelnen Vergleichsstudien vorhanden ist. Darüber hinaus deckt REpast Simphony die Vorzüge von Desktop-basierten und Large Scale ABMS am Stärksten ab, was Samuelson und Macal (2006) genannte Empfehlung in Abschnitt 4 teilweise relativiert.

5 Zusammenfassung und Ausblick

In dieser Ausarbeitung zum Seminar "Ausgewählte Technologien verteilter Systemsoftware" wurde die Multiagentensimulation vorgestellt. Ihre Vor- und Nachteile wurden erläutert und Einsatzgebiete sowie ihre Eignung genannt. Anschließend wurde der Begriff des Agentenbasierten Modellierungs- und Simulations-Werkzeugs eingeführt und in einen Kontext zu Agenten-Plattformen gesetzt. Außerdem wurden typische Bestandteile von ABMS-Werkzeugen genannt. Ferner wurde eine Einteilung für ABMS-Werkzeuge erläutert, die sie anhand von Universalität und Mächtigkeit gegenüber Komplexität klassifiziert. Es ist anzumerken, dass durch hybride Ansätze, die sowohl Agenten-Plattformen

als auch ABMS-Werkzeuge abdecken, der Begriff ABMS-Werkzeug an Schärfe verliert. Der Nutzen dieser Ansätze ist insbesondere für die Software-Entwicklung interessant, da hierdurch die Implementierung einer Agenten-Anwendung auf einer Agenten-Plattform in einer Simulation getestet werden kann, ohne dass das zugrunde liegende Modell zweifach implementiert werden muss.

Im zweiten Teil wurde das ABMS-Werkzeug REPAST SIMPHONY kurz vorgestellt, da es in der akkumulierten Vergleichsstudie als grundsätzlich gute Wahl hervor geht. Sobald jedoch genaue Anforderungen an das zu simulierende System bekannt sind, oder das Werkzeug beispielsweise lediglich zu Lehrzwecken eingesetzt werden soll, sollten anhand der vorgestellten und umfangreicheren Vergleichsstudien Alternativen gesucht werden. Wünschenswert wäre vor dem Hintergrund der großen Anzahl an verschiedenen ABMS-Werkzeugen, eine Standardisierung der ABMS-Werkzeuge, wie sie in Tobias und Hofmann (2004) gefordert wird. Dadurch wäre die Portabilität der Modelle zwischen verschiedenen ABMS-Werkzeugen sichergestellt und ein einfache Migration bei wechselnden Anfoderungen möglich. Eine solche Standardisierungs-Bestrebung verfolgt das METAABM Projekt (Miles Parker (2009b)), das bereits eine Abstraktion von ASCAPE (Miles Parker (2009a)) und REPAST SIMPHONY bietet.

Literatur

[Axelrod 1997] Axelrod, Robert: *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton University Press, August 1997. – URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0691015678. – ISBN 0691015678

[Braubach 2007] Braubach, Lars: *Architekturen und Methoden zur Entwicklung verteilter agentenorientierter Softwaresysteme*, Universität Hamburg, Fachbereich Informatik, Verteilte Systeme und Informationssysteme, Dissertation, 1 2007

[Braubach u. a. 2005] Braubach, Lars ; Pokahr, Alexander ; Lamersdorf, Winfried: Jadex: A BDI Agent System Combining Middleware and Reasoning. In: R. Unland, M. K. (Hrsg.): *Software Agent-Based Applications, Platforms and Development Kits*, Birkhäuser-Verlag, 9 2005, S. 143–168. – Book chapter

[Castle und Crooks 2006] Castle, Christian ; Crooks, Andrew: Principles and Concepts of Agent-Based Modelling for Developing Geospatial Simulations, 09 2006

[Gilbert und Bankes 2002] Gilbert, N. ; Bankes, S.: Platforms and Methods for Agent-based Modeling. In: *National Academy of Sciences* Bd. 99 National Academy of Sciences (Veranst.), 2002. – Comparison of multi-agent simulation platforms

[Klügl 2001] Klügl, Franziska: *Multiagenten-Simulation: Konzepte, Werkzeuge, Anwendung*. Addision-Wesley, 2001

[Klügl 2006] Klügl, Franziska: Multiagentensimulation. In: *Informatik-Spektrum* 29 (2006), 12, Nr. 6, S. 412–415. – URL http://www.springerlink.com/content/f07v1444k8218955

[Macal und North 2008] Macal, Charles M. ; North, Michael J.: Agent-based modeling and simulation: ABMS examples. In: *WSC '08: Proceedings of the 40th Conference on Winter Simulation*, Winter Simulation Conference, 2008, S. 101–112. – ISBN 978-1-4244-2708-6

[Nikolai und Madey 2009a] Nikolai, Cynthia ; Madey, Gregory: *Agent Based Modeling Toolkit Search Engine*. 2009. – URL http://agent.cse.nd.edu/abmsearchengine.php. – [Online; Stand 31. Juli 2009]

[Nikolai und Madey 2009b] Nikolai, Cynthia ; Madey, Gregory: Tools of the Trade: A Survey of Various Agent Based Modeling Platforms. In: *Journal of Artificial Societies and Social Simulation* 12 (2009), Nr. 2, S. 2. – ISSN 1460-7425

[North und Macal 2007] North, Michael J. ; Macal, Charles M.: *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation.* New York, NY, USA : Oxford University Press, Inc., 2007. – ISBN 0195172116

[North u. a. 2005b] North, M.J. ; Howe, T.R. ; Collier, N.T. ; Vos, J.R.: The Repast Simphony Runtime System. In: Macal, C.M. (Hrsg.) ; North, M.J. (Hrsg.) ; Sallach, D. (Hrsg.): *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms, ANL/DIS-06-5, ISBN 0-9679168-6-0,*, URL http://agent2007.anl.gov/2005procpdf/Agent_2005_North_Runtime.pdf, 2005

[Railsback u. a. 2006] Railsback, Steven F. ; Lytinen, Steven L. ; Jackson, Stephen K.: Agent-based Simulation Platforms: Review and Development Recommendations. In: *Simulation* 82 (2006), Nr. 9, S. 609–623. – ISSN 0037-5497

[Samuelson und Macal 2006] Samuelson, Douglas ; Macal, Charles: Agent-based Simulation Comes of Age. In: *OR/MS Today* 33 (2006), 8, Nr. 4, S. 34–38. – URL http://www.lionhrtpub.com/orms/orms-8-06/fragent.html

[Serenko und Detlor 2002] Serenko, A. ; Detlor, B.: *Agent Toolkits: General Overview of the Market and an Assessment of Instructor Satisfaction with Utilizing Toolkits in the Classroom.* 2002. – Working Paper 455, McMaster University, Hamilton, Ontario, Canada

[da Silva und de Melo 2008] Silva, Paulo S. da ; Melo, Ana C. V. de: Reusing models in multi-agent simulation with software components. In: *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems.* Richland, SC : International Foundation for Autonomous Agents and Multiagent Systems, 2008, S. 1137–1144. – ISBN 978-0-9817381-1-6

[Tobias und Hofmann 2004] Tobias, R. ; Hofmann, C.: Evaluation of free Java-libraries for social-scientific agent based simulation. In: *Journal of Artificial Societies and Social Stimulation* 7 (2004), Nr. 1. – URL http://jasss.soc.surrey.ac.uk/7/1/6.html

[Wikipedia 2009] Wikipedia: *Free/Libre Open Source Software — Wikipedia, Die freie Enzyklopädie.* 2009. – URL http://de.wikipedia.org/w/index.php?title=

Free/Libre_Open_Source_Software&oldid=59351895. – [Online; Stand 1. August 2009]

[Wooldridge 2001] Wooldridge, Michael J.: *Multi-agent systems : an introduction*. Chichester : Wiley, 2001. – GBA1-Z6596 Michael Woolridge.

Verweise

[AgentSheets Inc. 2009] AgentSheets Inc.: *AgentSheets*. 2009. – URL http://www.agentsheets.com/. – [Online; Stand 31. Juli 2009]

[Brassel 2009] Brassel, Kai-H.: *VSEit*. 2009. – URL http://www.vseit.de. – [Online; Stand 31. Juli 2009]

[Bruse 2009] Bruse, Jan: *Quicksilver*. 2009. – URL http://sourceforge.net/projects/java4u. – [Online; Stand 31. Juli 2009]

[Eclipse Foundation 2009] Eclipse Foundation: *Eclipse*. 2009. – URL http://www.eclipse.org/. – [Online; Stand 31. Juli 2009]

[George Mason University 2009] George Mason University: *MASON*. 2009. – URL http://www.cs.gmu.edu/~eclab/projects/mason. – [Online; Stand 31. Juli 2009]

[Massachusetts Institute of Technology 2009] Massachusetts Institute of Technology: *StarLogo*. 2009. – URL http://education.mit.edu/starlogo. – [Online; Stand 31. Juli 2009]

[Miles Parker 2009a] Miles Parker: *Ascape*. 2009. – URL http://ascape.sourceforge.net. – [Online; Stand 31. Juli 2009]

[Miles Parker 2009b] Miles Parker: *MetaABM*. 2009. – URL http://metaabm.org/. – [Online; Stand 31. Juli 2009]

[Northwestern University 2009] Northwestern University: *NetLogo*. 2009. – URL http://ccl.northwestern.edu/netlogo. – [Online; Stand 31. Juli 2009]

[Swarm Development Group 2009] Swarm Development Group: *Swarm*. 2009. – URL http://www.swarm.org. – [Online; Stand 31. Juli 2009]

[Tel Aviv University 2009] Tel Aviv University: *OBEUS*. 2009. – URL http://eslab.tau.ac.il/Research/defaults.aspx. – [Online; Stand 31. Juli 2009]

[Universität Würzburg 2009] Universität Würzburg: *SeSAm*. 2009. – URL `http://www.simsesam.de/`. – [Online; Stand 31. Juli 2009]

[University of Chicago 2009] University of Chicago: *Repast*. 2009. – URL `http://repast.sf.net`. – [Online; Stand 31. Juli 2009]

[XJ Technologies 2009] XJ Technologies: *AnyLogic*. 2009. – URL `http://www.xjtek.com/`. – [Online; Stand 31. Juli 2009]