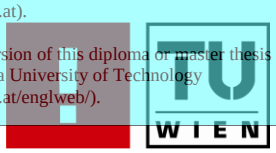

FAKULTÄT FÜR !NFORMATIK

Schwachstellenanalyse von Simulationsmodellen sozialer Netzwerke mittel Replikation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Magister

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Christoph Kaniak
Matrikelnummer 0025826

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Ao.Univ.Prof. Mag.rer.soc.oec. Dr.rer.oec. Gerhard Hanappi
Mitwirkung: Univ.Ass. Mag.rer.soc.oec. Dr.rer.soc.oec. Bernhard Rengs

Wien, 21.10.2008 _____ _____

(Unterschrift Verfasser) (Unterschrift Betreuer)

1 Inhaltsverzeichnis

2 Einleitung

2.1 Thematik

Die Idee der agentenbasierten Modellierung wurde bereits in einer simplen Form in den späten 1940er geboren, doch auf Grund der sehr rechenintensiven Prozeduren und der mangelnden Rechenleistung konnte sich diese Technik erst in den 1990ern profilieren und in vielen wissenschaftlichen Bereichen langsam Fuß fassen. Heute erfreut sich diese Methode, durch die immer leistungsfähigeren Prozessoren und die verhältnismäßig günstige Anschaffung dieser, sehr großer Beliebtheit um heterogene Systeme zu analysieren. Durch agentenbasierte Simulationen lässt sich das Verständnis gewinnen, wie sich der Einfluss von Mikrolevel-Prozessen auf das Makrolevel-Verhalten auswirkt und somit die Entstehung ökologischer, wirtschaftlicher oder sozialer Mustern bewirkt.

Die Vorgehensweise lässt sich wie folgt skizzieren: in einer beliebigen Programmiersprache werden Agenten in Form von Objekten erstellt, welche sich in einer künstlichen Umgebung bewegen können. Mit Hilfe ihrer eigenen Attribute und Methoden können sie miteinander interagieren und somit ihre eigenen Zustände ändern beziehungsweise die Zustände anderer Agenten beeinflussen. Betrachtet man schließlich das System im Ganzen, so können hiermit beispielsweise Strömungen oder Entstehungen von Verhaltensmustern beobachtet werden.

Die Umsetzung eines theoretischen Modells zu einer agentenbasierten Computersimulation bedarf allerdings einiger Schritte, welche jeweils Fehler mit sich bringen können. Bereits ein Missverstehen des Modells selbst, ein Missinterpretieren von Werten oder auch ein Logikfehler während der Implementierung in der jeweiligen Programmiersprache lassen eine gültige Abbildung nicht mehr zu. Sollten auch die Ergebnisse der Simulation in etwa mit den Erwartungswerten übereinstimmen, bedeutet dies nicht zwingend, dass es sich hierbei um eine korrekte Umsetzung handelt.

Genau an diesem Punkt setzt nun die Replikation von Simulationen an: durch wiederholtes Replizieren eines theoretischen Modells in unterschiedlichen Programmiersprachen, sollen Fehler bei der Umsetzung aufgezeigt und in späterer Folge natürlich auch korrigiert werden.

Diese Arbeit soll nun theoretische Einzelheiten dieser Schwachstellenanalyse erklären und schließlich anhand einer praktischen Umsetzung des SARS-Modells den Vorgang einer Replikation veranschaulichen. Durch Vergleich der Ergebnisse der erstellten Simulation mit deren der Vorgegebenen, sollen eventuelle Fehler erkannt sowie Erfahrungen und Erkenntnisse gewonnen werden.

2.2 Vorgehen

Vor der praktischen Durchführung der Replikation werden zunächst grundlegende Thematiken angeschnitten. Neben der anfänglichen allgemeinen Einführung in das Thema Simulation und dem theoretischen Ansatz ein Modell zu replizieren, wird vorerst das abzubildende Modell gewissenhaft erklärt.
Hierfür müssen allerdings noch angewandte Techniken beziehungsweise verwendete Modelle wie etwa das „Modell der kleinen Welt“ erörtert werden. Nach der Erläuterung aller notwendigen theoretischen Details wird schlussendlich die Replikation mit Hilfe des Simulationstoolkits Repast veranschaulicht. Gewonnene Erkenntnisse und Erfahrungen sollen mitgeteilt und eventuelle Schwachstellen aufgezeigt werden.

2.3 Gliederung

Kapitel 3 – Einführung in die Thematik

Zunächst werden grundlegende Aspekte von Simulationen angeführt, wobei unter anderem auf die Klassifikation dieser als auch auf die Verifikation eingegangen wird. Anhand einer bereits bestehenden Replikation soll aufgezeigt werden warum das Replizieren von Modellen eine so entscheidende Rolle spielt.

Kapitel 4 – Multi-Agenten Simulationen

Nach einem kurzen Abschnitt über den geschichtlichen Hintergrund von Simulationen wird die Notwendigkeit dieser in einzelnen Wissenschaftsbereichen begründet. Hierbei werden auch Anwendungsbeispiele vorgestellt, damit auch die Mächtigkeit dieser Methode verdeutlicht wird.
Anschließend wird das Thema Validierung einer Simulation durchleuchtet, welches unmittelbar auf die Hauptproblemstellung dieser Arbeit, nämlich das Replizieren von Simulationsmodellen, überleitet.

Kapitel 5 – Das SARS-Modell im Detail

Um die Funktionsweise des zu replizierenden Modells im Ganzen verstehen zu können, muss vorerst noch das Modell der kleinen Welt erklärt werden, da dieses oft für epidemische Simulationen herangezogen wird. Danach wird vorerst ein einfaches Modell einer Epidemie vorgestellt, damit man sich bereits den groben Ablauf vorstellen kann.
Nach einer kurzen Einführung über den SARS-Virus und dessen Eigenschaften wird schließlich das Modell im Detail erklärt, welches für die Replikation herangezogen wird.

Kapitel 6 – Replikation und Gegenüberstellung der Simulationsergebnisse

Nach einer kurzen Vorstellung des gewählten Simulationstoolkits Repast und der Übermittlung der gewonnen Erfahrungen wird die Vorgehensweise sowie die Struktur des entwickelten Programms unter anderem mit Klassendiagrammen veranschaulicht.
Anhand von Graphen sollen schlussendlich die Ergebnisse beider Simulationen miteinander verglichen und eventuelle Schwachstellen herausgefunden werden. Diese Analyse sowie ein Bericht über die Stärken und Schwächen einer Replikation in der Praxis bilden schließlich den Abschluss dieser Arbeit.

3 Einführung in die Thematik

3.1 Motivation

Sozialsimulationen konzentrieren sich stets auf einen sozialen Prozess und versuchen dessen Verhalten näher zu bringen. Zwischen Simulation und dem zu analysierenden Prozess muss eine Verbindung bestehen, welche in direkter oder indirekter Form auftreten kann. Von einer indirekten Verbindung spricht man dann, wenn ein soziales Ereignis behandelt wird, welches nur theoretisch von Sozialwissenschaftlern hergeleitet wurde. Man bezeichnet dies auch als ein konzeptuelles Modell, welches zwischen der Simulation und dem Prozess selbst vermittelt.
Es besteht jedoch die Möglichkeit die Grenzen des indirekten Ansatzes von Simulationen zu überschreiten, um einen direkteren Bezug zu gewährleisten. In diesem Fall werden die Ergebnisse der Simulation mit dem Datenmodell des sozialen Phänomens verglichen. Hierfür gibt es zwei verschiedene Ansätze:

- eine Post-Hoc-Überprüfung der Validität
- eine Kalibrierung, bis der gewünschte Übereinstimmungsgrad erreicht ist

Beide dieser Methoden vergleichen also die Simulationswerte mit den vorliegenden Daten des Ereignisses, wobei entschieden wird ob die Simulation den Anforderungen genügt und beibehalten werden kann. Bei Sozialsimulationen sind die verfügbaren Daten jedoch nie ausreichend um ein Modell so einzuschränken, dass es als einzigartig bzw. absolut korrekt bezeichnet werden kann.
Während die Erstellung der Modelle für soziale Ereignisse gewissenhaft durchgeführt wird, findet man meist eine mangelhafte Ausarbeitung der zugehörigen Simulationen vor. Da letztere mittels Verwendung der natürlichen Sprache entwickelt werden, kann dies leicht zu Mehrdeutigkeiten, ungenauen Definitionen oder sonstigen Fehlern führen, welche das Ergebnis verfälschen.

Genau diese Fehlerquellen sollen durch Replikation einer Simulation vermieden und somit die Qualität und die Gültigkeit drastisch gesteigert werden.

3.2 Replikation von Simulationen

Bruce Edmonds und David Hales [EDMONDS, 2002] haben bereits ihre Erfahrungen mit Replikationen von Simulationen anhand eines einfachen Beispielmodells gezeigt, indem sie zwei verschiedene Implementierungen gegenüber gestellt haben. Laut ihrer Meinung ist eine Replikation sehr schwierig und langwierig, jedoch auch sehr aufschlussreich über die Qualität und Gültigkeit eines Simulationsmodells. Diese Vorgehensweise deckt eine Menge kleiner Fehler und schlecht bestimmter Definitionen auf, welche sich beim Erstellen der Simulation eventuell eingeschlichen haben.
Da ein Implementieren einer Simulation zu einem gegebenen Modell und ein nachträgliches Vergleichen der Ergebnisse alleine nicht ausreichen um die Korrektheit nachzuweisen. Dies führt dann allerdings zu dem Schluss, dass eine große Menge an Sozialsimulationen, welche nicht durch Replikation überprüft worden sind, nicht unbedingt die gewünschten Ergebnisse der Autoren liefern. Dabei muss man jedoch auch erwähnen, dass in vielen Fällen diese Unterschiede keine bemerkenswerten Auswirkungen auf das Ergebnis und somit den Gesamtcharakter der Simulation haben, und dadurch die Gültigkeit erhalten bleibt. Doch manchmal können diese Differenzen so groß sein, so dass man mit neu gewählten Parametern ein inkonsistentes Ergebnis erhält.

3.3 Problem der Verifikation

Das Problem der Verifikation ist in der Computerwissenschaft gut bekannt und auch weit verbreitet. Förmliche und strukturierte Implementierungstechniken helfen den Softwareentwicklern die gegebene Spezifikation korrekt zu implementieren. Diese Techniken sind auch im Bereich der Sozialsimulationen sehr nützlich, jedoch ist der Grad der Komplexität meist so immens, dass ein Anwenden dieser Techniken nur begrenzt möglich ist.

Im Bereich der Computerwissenschaft geht man oft davon aus, dass man das Ergebnis eines Programms im Vorhinein abschätzen kann. Im Wesentlichen muss ein Programm nur den erforderlichen funktionellen Bedingungen genügen. Auf Grund der forschenden Natur der Simulationswissenschaft und dem Hauptaugenmerk auf die vorhandenen Eigenschaften, verschwimmen jedoch die funktionellen Bedingungen oder sind erst gar nicht vorhanden. Die Vorgehensweise beruht meist darauf, dass man lediglich die Werte nach einem Simulationslauf analysiert und auf Korrektheit abschätzt. Darum ist es allerdings auch für Entwickler schwer die Ausgabe einer Simulation vorherzusagen und die Richtigkeit gemäß der Spezifikation zu bestätigen.

3.4 Klassifikation von Simulationen

Laut Ken Kollman [KOLLMAN, 2003] lassen sich computerberechnete Modelle in drei Klassen unterteilen:

- Simulationen
- Computerberechnungen
- Agenten-basierte Modelle

Simulationen

Unter einer Simulation versteht man eine numerische Abhandlung. Gewöhnlich handelt es sich dabei um ein Gleichgewicht des Ergebnisses eines Spiels oder eines sozialen Systems. Diese numerischen Beispiele helfen die Logik eines Modells näher zu bringen, in manchen Fällen unterstützen sie sogar Modelle zu kalibrieren und zu verfeinern. Damit konkurrieren Simulationen nicht mit den Vorgaben der Hauptmodellierungsansätze der Sozialwissenschaft, sondern stellen ein sehr hilfreiches Werkzeug dar, um die Ergebnisse zu analysieren und zu veranschaulichen.

Computerberechnungen

Computerberechnungen sind numerische Annäherungen an ein Gleichgewicht, das mit analytischen Mitteln und Methoden nicht auffindbar ist. Während eine Simulation bei der Interpretation und des Verständnisses einer abgeleiteten Funktion hilft, ist es mittels Computerberechnungen möglich Verbindungen aufzudecken, mit welchen das Problem schließlich doch analytisch gelöst werden kann. Der Computer ist quasi ein Hilfsmittel um Gleichgewichte von Modellen zu finden, wenn diese über einen hohen Komplexitätsgrad verfügen.

Agenten-basierte Modelle

Bei der dritten Gruppe handelt es sich um agenten-basierte Modelle, welche einen komplett anderen Ansatz im Bereich der Modellierung bieten. Der Computer ist für agenten-basierte Modelle fast Grundvoraussetzung und wird für zwei Aufgaben benötigt. Erstens erleichtert der Computer das Ableiten und Lösen von vereinfachten Gleichungen, welche das Verhalten bzw. die Information der Agenten beeinflusst. Zweitens bietet der Rechner die Möglichkeit eine Plattform zur Analyse von künstlichen Welten zu schaffen, welche dazu verwendet werden Verhaltensmuster von Grund auf zu erforschen. Dabei berechnet der Computer die Zustände und das Verhalten der einzelnen Agenten, woraus sich durch die Interaktion der Akteure das Verhalten des gesamten Systems einstellt.
Agenten-basierte Simulationen bringen vier Erkennungsmerkmale mit sich:

- Agenten unterscheiden sich untereinander
- Agenten interagieren miteinander in einer dezentralisierten Art und Weise
- Agenten sind eingeschränkt lernfähig
- Ausgabemuster führen nicht zwangsweise zu einem Gleichgewicht

4 Multi-Agenten Simulationen

4.1 Anwendungsbereiche im Bereich der Ökonomie und Sozialwissenschaft

Multi-Agenten Systeme finden in der Wirtschaft immer größere Beliebtheit, da sie die Möglichkeit bieten, durch das Zusammenspiel vieler unabhängiger selbstständiger Agenten einen beliebigen Wirtschaftsraum zu simulieren und diesen auch zu studieren.
Hierbei wird auf zwei Schwerpunkte besonders Rücksicht genommen. Der Erste zielt darauf ab auftretendes globales Verhalten erklären und beschreiben zu können, wodurch Antworten auf bestehende Fragen gewonnen werden sollen. Warum haben sich bestimmte Gesetzmäßigkeiten entwickelt und werden diese in dezentralisierten Wirtschaftsmärkten beibehalten? Wie sind diese Gesetzmäßigkeiten von Grund auf, durch wiederholtes interagieren autonomer Agenten entstanden? Und warum gerade diese Gesetzmäßigkeiten und nicht andere?
Der zweite Schwerpunkt konzentriert sich auf den Aufbau der ökonomischen Maschinerie. Welche Auswirkungen hat ein bestimmter Mechanismus, tatsächlich existierend oder auch nur ausgedacht, auf die Leistung des ganzheitlichen wirtschaftlichen Raumes? Welche sozialen Folgen resultieren von den wiederholten Versuchen der Agenten das System zu ihren Gunsten auszunützen? Im nächsten Abschnitt werden einige Anwendungsbereiche genauer durchleuchtet, um das Potential von Multi-Agenten Systemen im wirtschaftlichen Sektor zu veranschaulichen.

Lernen und der verkörperte Verstand

Die Lernprozesse von computerberechneten Agenten treten durch eine große Anzahl von Ausprägungen auf. Zu ihnen zählen unter anderen Lernalgorithmen, neuronale Netze, genetische Algorithmen, genetische Programmierung und eine Reihe anderer Entwicklungsalgorithmen, welche versuchen das induktive Lernverhalten wiederzugeben. Die meisten der genannten Lernalgorithmen wurden entwickelt um globale Ziele zu

optimieren, daher muss man vorsichtig bei deren Umlegung auf Geschäftsprozesse sein. Einerseits ist es vernünftig ein globales Lernschema für Gemeinschaftsprobleme oder Geschäftsprozesse zu erstellen, wodurch alle Agenten zusammen auf spezifizierte Ziele hinarbeiten. Andererseits soll bei computerberechneten Modellen, welche ökonomische Prozesse repräsentieren und unterschiedliche menschliche Verhaltensweisen aufweisen, der jeweilige Lernalgorithmus der Agenten die Eigenschaften von Menschen widerspiegeln, welche für die jeweilige Situation eine unabhängige Entscheidung treffen. Daraus resultiert, dass ein lokales Lernschema vorhanden sein muss, welches einzelne Agenten oder auch Gruppen von Agenten dazu veranlasst ihre Strategie auf Basis ihres lokalen Nutzens auszubilden.

Gintis [GINTIS, 2000] lässt weitere Bedenken aufkommen, da er den Bedarf an besser modellierten Agentenverhalten verdeutlicht. In zahlreichen Versuchen konnte beobachtet werden, dass Abweichungen zwischen tatsächlichen menschlichen Verhalten und dem vorausberechneten Verhalten traditioneller Agenten bemerkbar waren. Entgegengesetzt der Ansicht Spieltheorie als eine Studie des rationellen Verhaltens zwischen untereinander agierenden Agenten anzusehen, versteht Gintis die Spieltheorie als Entwicklungsumgebung mit deren Hilfe das Verhalten mit weitlaufenden sozialen Abstimmungen untersucht werden kann. Er sieht das Spiel selbst als ein strategisches Interaktionsproblem an, welches in natürlichen und sozialen Prozessen eingeschlossen ist. Durch die Tatsache, dass Agenten wiederholt mit diesen Problemen konfrontiert werden, können sie die Fähigkeit entwickeln dieses Spiel auch effizient zu meistern. Viel mehr noch versteht Gintis die Entwicklung nicht auf Grund von kognitiven Prozessen, sondern als abgeänderte Arten der Imitation. Eine solche kann man sich als Nachahmung von früheren Kulturen und deren Ritualen vorstellen. Durch diese Bedenken sind viele Multi-Agenten-Forscher von der Idee abgekommen, dass die Lernalgorithmen von allgemein üblichen Standards abgeleitet werden. Der neue Ansatz besteht darin, systematische Studien über die Empfindlichkeit von Wirtschaftserfolgen in das Lernverhalten einfließen zu lassen. Beispielsweise führt Dawid [DAWID, 1999] eine systematische Studie von dynamischen Multi-Agenten-Modellen durch, wobei genetische Algorithmen zum Einsatz kommen um die Entwicklung von individuellen Strategien herauszuheben. Er veranschaulicht, dass bestimmte Aspekte der Implementierung einen starken Einfluss auf das Langzeitverhalten des Experiments bewirken. Diese Arbeit hat eine grundlegende Wirkung auf die Kreise der Multi-Agenten-Forscher, da seitdem genetische

Algorithmen bei der Repräsentation des Lernverhaltens der Agenten eine entscheidende Rolle spielt.

Entwicklung von Verhaltensnormen

Der Begriff "Norm" wurde schon durch viele Wissenschaftler definiert; Axelrod [AXELROD, 1997] erweiterte diese Definition durch ein abhängiges Verhalten: "A norm exists in a given social setting to the extent that individuals usually act in a certain way and are often punished when seen not to be acting in this way."

Er rechtfertigt seine Definition mit der Begründung, dass durch das Vorhandensein von Normen ein Maß entsteht, welches sich bei einem ökonomischen Prozess durch Zu- und Abnahme von Normen beobachten lässt. Durch Experimente mit Multi-Agenten-Simulationen veranschaulicht er, dass durch eigennützige, nicht in Verbindung stehenden Agenten mittels wechselseitigem Informationsaustausch, wobei kein oder nur wenig vorausschauendes Verhalten angewendet wird, gegenseitige Kooperation erreicht wird. Diese Erkenntnis hatte einen großen Einfluss sowohl bei Forschern der Ökonomie als auch bei jenen der Spieltheorie. Es hat die traditionelle Ansichtsweise der nicht-kooperativen Spieltheorie durch Berücksichtigung der begrenzten Rationalität und der evolutionären Dynamik enorm erweitert.

Thomas Schelling [SCHELLING, 1978], im Bereich der Verhaltensnormen tätig, arbeitet an bekannten Beispielen aus dem Alltag ohne die Hilfe von komplexen Computerwerkzeugen in Anspruch zu nehmen. Er zeigt damit wie musterbehaftetes soziales Verhalten aus einer unfreiwilligen Abfolge von lokalen Interaktionen zwischen Agenten entsteht, welche simplen Verhaltensregeln nachgehen. Durch eines seiner Experimente konnte Schelling veranschaulichen, wie eine lokale Kettenreaktion zu einer Aufteilung in verschiedene Gattungen führt. Dieses Szenario ergibt sich schließlich, wenn alle Agenten einigermaßen dem Ziel nachgehen, dass mindestens jeder dritte der benachbarten Agenten der seiner eigenen Gattung zugehören muss. Aufbauend auf Schellings Arbeit verwendeten Epstein und Axtell [EPSTEIN, 1996] ein Multi-Agenten-System, um zu beobachten wie die Agenten aus dem Befolgen und Anwenden einfacher Verhaltensregeln ein gemeinschaftliches Verhalten entstehen lassen kann. Speziell bei Verwendung beider Techniken, Analysen und computerberechnete Experimente, konnten sie aufzeigen wie im Verlauf der Zeit aus

eigentlich unwichtig gekennzeichneten Agenten jene mit einem hoch angesehenen sozialen Status werden.

Modellierung von Marktprozessen

Die Analyse der Entwicklung von Marktprozessen, insbesondere ihre Fähigkeit sich selbst zu organisieren, findet in der Forschung mittels Multi-Agenten-Simulationen einen hohen Stellenwert. Um einen kleinen Einblick in diese Thematik zu gewähren, wird im nächsten Abschnitt die Studie von Robert Marks [MARKS, 1992], welche einen großen Einfluss auf dieses Forschungsgebiet mit sich brachte, ein wenig erörtert. In Folge dessen werden weitere Studien präsentiert, welche sich speziell auf den Finanzmarkt und den Energiemarkt konzentrieren.

Robert Marks war einer der ersten Forscher, welche durch Hilfe einer Multi-Agenten-Simulation die Problematik der Selbst-Organisation eines Wirtschaftsraumes aufzeigen wollte. Durch seine Arbeit vermittelte er den Ökonomen auf verständliche Art und Weise, dass die Entwicklung, das Zusammenwirken und das Verständnis über den Ablauf von Marktergebnissen von großer Wichtigkeit sind. Hierfür nahm Marks ein Simulationsmodell eines Oligopols zur Hand und beobachtete Unternehmen, welche die Preise bestimmen und untereinander erfolgreich wetteifern. Sein Modell verwendete einen genetischen Algorithmus um rationelle induktive Unternehmen abbilden zu können. Die Menge an Preisstrategien, welche den Unternehmen zur Verfügung stehen, wird öfters mit Hilfe von Mutations- und Rekombinationsoperationen bearbeitet. Durch diese Vorgehensweise werden zwei Eigenschaften der Firmen simuliert: erstens experimentieren sie mit neuen Ideen (Mutation) und zweitens werden bereits erprobte Strategien, welche andere Unternehmen erfolgreich anwenden, einfach übernommen (Rekombination).

Marks stellte mittels seiner Beobachtungen fest, dass sich bei Firmen ohne jegliche Preisvorstellungen trotzdem ein globaler optimaler Preis einstellte. Eine weitere Eigenschaft, welche durch die Arbeiten Marks aufgezeigt wurde, war jene, dass die Entwicklung von Kooperationen unter den Unternehmen nicht gewährleistet war. In vielen der durchlaufenen Versuchen konnte ein Auftreten vieler Nischenstrategien beobachten werden, welche jedoch nur immer gegen eine spezielle Gruppe von Wettstreitern erfolgreich war.

Man kann sich leicht vorstellen, dass die Entwicklung der Strategie eines Unternehmens in Abhängigkeit mit den konkurrierenden Unternehmen einen komplexen Ablauf darstellt, welcher viele unterschiedliche Wege einschlagen kann und somit auch sehr stark vom Zufall bestimmt wird. Es kann durchaus geschehen, dass ein Unternehmen, welches seine optimale Preisstrategie gefunden hat, in einem Durchlauf der Simulation eine souveräne Stellung einnimmt, jedoch im nächsten Durchlauf mit der gleichen Strategie eine völlig andere Position erreicht.

Am Beispiel für den Finanzmarkt kann man schon langsam die Mächtigkeit der Multi-Agenten-Simulationen erahnen, da gewöhnliche Modelle für den Finanzmarkt auf Annahmen von rationalen Entscheidungen basieren und von einer idealen Markteffizienz ausgehen. Leider besitzt derzeit keines dieser Modelle die Fähigkeit die wesentlichen empirischen Eigenschaften der realen Finanzmärkte abzubilden. Zu diesen zählen breit gefächerte Vermögenseinnahmeverteilungen, hohe Umsätze, Nachwirkungen und Gruppierungen durch Schwankungen der Kapitalrenditen und der Zusammenhang zwischen Vermögenseinnahmen, Umsätzen und Kursschwankungen.

Auf Grund der oben genannten Schwierigkeiten nehmen sich viele Forscher der Problematik an und versuchen eine Lösung mittels Multi-Agenten-Simulationen herbeizuführen. In der Tat war es möglich durch Verwendung von Multi-Agenten-Simulationen Erklärungen für beobachtete Regelmäßigkeiten der Daten des Finanzmarktes zu erhalten.

Bis heute konzentrieren sich die meisten Forschungen auf einseitige Auktionen mit einer festen Anzahl von Agenten, welche auf einen einzelnen Auktionsgegenstand in einer einzigen Handelsperiode bieten. In Realität sieht es allerdings so aus, dass viele Auktionen aus einer kleinen, ungleich großen Menge an Käufern und Verkäufern besteht, welche sich regelmäßig trifft um Mengen und Preisangebote zu vereinbaren.

Im Fall des Energiemarktes beinhalten Auktionen wiederholte Stückzahl- und Preisangebote, welche zum Verkauf von großen Mengen an Strombündel von einer kleinen Anzahl von Generatoren ausgehandelt werden. Durch diesen Sachverhalt gestalten sich die resultierenden Marktprozesse als relativ komplex, und traditionelle analytische sowie statistische Anwendungen stoßen schnell an ihre Grenzen. Folglich versuchten Energieforscher die Möglichkeiten der Multi-Agenten-Simulation auch für diesen Bereich auszunützen.

Bower und Bunn [BOWER, 2001] untersuchten mit Hilfe eines Multi-Agenten-Systems den Energiegroßhandel von England und Wales. Der Schwerpunkt ihrer Arbeit lag darin folgende

Fragestellung zu beantworten: wie ändern sich die Energiepreise, wenn von einer Gleichpreisauktion, bei welcher der Preis für eine Einheit auch für alle weiteren Einheiten gelten, auf eine diskriminierende Auktion umgestellt wird. Der Markt wird als Spiel zwischen den Energiegeneratoren, welche Marktanteile und Gewinnziele besitzen, modelliert. In jeder Handelsperiode übermitteln alle Generatoren der Auktion eine Angebotsfunktion, welche Menge und Preis beschreibt. Jedes Kraftwerk für jeden Generator wird durch einen unabhängigen lernfähigen Agenten dargestellt, welche ihre eigenen Angebotsstrategien mittels eines Lernalgorithmus wählen können. Im Gegensatz dazu, findet man auf der Nachfrageseite des Marktes Agenten, welche ein passives Verhalten zeigen und Preise annehmen. Ihr Kaufverhalten wird mittels einer aggregierten Nachfragekurve festgelegt, die sich auf einen Tagesverbrauch eines typischen Wintertages bezieht.

Eine der wichtigsten Entdeckungen bei diesem Experiment war jene, dass wenn die Angebotsfunktionen nicht öffentlich bekannt sind, ein Wechsel von einer Gleichpreisauktion auf eine diskriminierende Auktion die Gewinne der größeren Generatoren im Vergleich zu den kleineren Generatoren ansteigen lässt. Dies lässt sich damit begründen, dass die großen Generatoren einen deutlichen Informationsvorteil bei der Auktion gegenüber den kleinen Generatoren haben, da sie mehr Angebote übermitteln und dadurch öfter die Gelegenheit bekommen durch den aktuellen Marktstatus zu lernen. Die Gleichpreisauktion entschärft diese Informationsungleichheit, da die kleinen Generatoren am gemeinsamen Wissen über den Markt teilhaben können, da sie die gleichen Marktpreise erhalten wie jeder andere auch.

Anordnung von Wirtschaftsnetzwerken

Interaktionsnetzwerke im Wirtschaftsumfeld werden meist mit Hilfe von "Transaction Cost Economics" analysiert. Dieser Ansatz geht davon aus, dass optimale Formen von Organisationen sowie Regierungen entstehen, welche am besten zu den entsprechenden Eigenschaften der Agententransaktionen geeignet sind. Das Problem dieses Ansatzes liegt darin, dass vorerst einmal die optimalen Organisations- bzw. Regierungsformen gefunden werden müssen.

Forscher, welche sich stark mit Multi-Agenten-Simulationen auseinander setzen und auch ein Interesse an Interaktionsnetzwerken zeigen, konzentrierten sich meist auf unvollständige

Wirtschaftsmärkte, welches eine strategische Interaktion zwischen einer kleinen Menge von Käufern und Verkäufern einschließt. Bei solchen Märkten spielt die Wahl der Partner durch Lernen und die Entwicklung von vertrauenswürdigen Verbindungen eine einflussreiche Rolle, wobei allerdings stets andere Formen von Interaktionsnetzwerken auftreten können. Im Gegensatz zu der "Transaction Cost Economics"-Methode werden mittels Multi-Agenten Simulationen mehr die dynamischen Ansätze gefordert. Es sollen beispielsweise Antworten auf folgende Fragen gefunden werden:

- Was veranlasst die Formgebung der Interaktionsnetzwerke zwischen Käufer und Verkäufer?
- Wie entwickeln sich diese Netzwerke über Zeit?
- Welche Auswirkungen bezüglich der Sozialfürsorge bringen diese Netzwerke mit sich?

Ein bestimmtes Interaktionsnetzwerk zieht besonders viel Aufmerksamkeit auf das Interesse der Forscher, da es sich durch eine offensichtliche empirische Allgegenwärtigkeit hervorhebt. Das Small-World-Netzwerk ist ein Netzwerk, welches zwei Eigenschaften mit sich bringt: erstens ist jeder Knoten an einen weiteren relativ gut verbundenen Knoten angeschlossen und zweitens lässt die Möglichkeit, Abkürzungen zwischen verschiedenen Knoten zu verwenden, die durchschnittliche minimale Pfadlänge zwischen den Knoten gering werden. Solche Netzwerke weisen sowohl gute lokale Verbindungen als auch globale Reichweite auf.
Wilhite [WILHITE, 2001] verwendet ein Multi-Agenten Modell einer bilateralen Austauschwirtschaft, um die Auswirkungen einer Einschränkung des Handels auf ein Small-World-Netzwerk aufzuzeigen. Er konzentriert sich hierbei auf den Ausgleich zwischen Markteffizienz und Transaktionskosten, wobei vier verschiedene Arten von Netzwerken zum Einsatz kommen:

- Vollständig vernetztes Handelsnetzwerk

  Alle können untereinander Geschäfte abschließen
- Lokal getrenntes Netzwerk

  Einzelne Handelsgruppen

- Lokal verbundenes Netzwerk

  Einzelne Handelsgruppen in ringförmig um Haupthandelspartner angeordnet

  Pro Gruppe besteht eine Verbindung zu diesem
- Small-World-Netzwerk

  Geschäfte ungleicher Handelsgruppen über zufällig gewählte Verbindungen

Nach der Festlegung der Netzwerkart werden alle teilnehmenden Agenten mit zwei Gütern ausgestattet und auf die Suche nach möglichen Geschäftspartnern geschickt. Nach diversen Verhandlungen wird der Handel dann jeweils dort abgeschlossen, wo auch der größte Nutzen ist. Eine der wichtigsten Erkenntnisse ist jene, dass das Small-World-Netzwerk die meisten Vorteile bezüglich Markteffizienz im Vergleich zu einem komplett vernetzten Markt mit sich bringt. Im Weiteren lässt diese Erkenntnis vermuten, dass eine natürliche Tendenz zu Small-World-Netzwerken besteht, da Agenten, welche durch diese Art von Netzwerk positive Erfahrungen machen, während andere diese Erfahrung nicht machen.
Eine logische Weiterführung von Wilhites Arbeit ist die Analyse der anfänglichen Formation von Handelsnetzwerken und deren Weiterentwicklung.

Modellierung von Organisationen

Eine Organisation wird dann gegründet, wenn eine Gruppe von Leuten sich Ziele bzw. Leistungskriterien setzt, welche durch die einzelnen Teilnehmer nur schwer oder gar nicht zu erreichen wären. Die computerberechnete Modellierung von Organisationen begann spätestens durch die RAND-Cooperation, die bereits fähig war Intelligenz symbolisch zu imitieren. Der Fortschritt war jedoch anfangs eher schleppend, bis dann die Entwicklung der objektorientierten Programmierung stattfand. Durch diese lassen sich Organisationen äußerst wahrheitsgetreu abbilden, da viele Programmiersprachen Analogien zu organisatorischen Phänomenen verwenden. Als Beispiel sei hier SmallTalk genannt.
Bei gemeinsamer Betrachtung aller Studien erkennt man, dass Organisationen als komplexe lernfähige Systeme aufgefasst werden können und meist objektorientierte Sprachen zum Einsatz kommen. Es wird ein großer Bereich von organisatorischen Problemen abgedeckt, wie auch die Firmenorganisation selbst. Die Arbeiten der Gruppe von Kathleen Carley von der Carnegie Mellon Universität blieb immer die treibende Kraft in dem immer größer

werdenden Forschungsgebiet, welches Multi-Agenten-Systeme im Allgemeinen und Organisationstheorie im Speziellen mit einander vereint. Obwohl wenige Ökonomen direkt in den Arbeiten integriert wurden, lässt die Tendenz erkennen, dass dieser Ansatz nicht mehr verfolgt wird. Van Zandt [VAN ZANDT, 1998] forderte in seinen Berichten, dass das Hauptaugenmerk wieder auf die Modelle der Multi-Agenten-Systeme gelenkt werden soll. Folglich ist der Bereich der Modellierung von Organisationen nur ein kleinen ergiebiges Forschungsgebiet der Multi-Agenten-Systeme.

Studien von Unternehmen in der Organisationstheorie, welche mit Hilfe von Multi-Agenten-Systemen eine Lösung herbeiführen wollen, tendieren dazu die Effekte der Organisationsstrukturen hervorzuheben, welche durch ihre eigenen Verhaltensmuster beeinflusst werden. Im Gegensatz dazu konzentrieren sich Multi-Agenten-Marktanalysen stets auf das Hervorheben der Effekte bestimmter Arten von Firmenverhaltensregeln, welche auf Grund von Änderungen der Preisdynamik, des Wachstums und der Marktstruktur angepasst werden.

Dawid [DAWID, 2001] schlägt hier einen neuen Weg ein und versucht die beiden Ansätze zu kombinieren. Mit Hilfe eines Multi-Agenten-Marktmodells soll ermittelt werden, in wie weit die Struktur des Marktes als auch die interne Organisation der beteiligten Firmen die Gestaltung der optimalen Verhaltensregeln beeinflussen.

Dawid konzentrierte sich immer nur auf einen Industriezweig, zum Beispiel die Getränkeherstellung, welcher eine Gruppe von mehreren teilnehmenden Firmen beherbergte. Am Beginn einer jeden Zeitperiode kann jede einzelne Firma für sich entscheiden, ob sie eine bereits bestehende Produktpalette weiterproduziert oder aber in die Entwicklung eines neuen Produktes investiert. Die Nachfrage eines jeden Produktes geht kontinuierlich zurück und stoppt schließlich nach einer stochastisch ermittelten Zeit. Daher ist jede Firma gezwungen einen gewissen Grad an Innovationen zu erreichen, damit die Gewinnausschöpfung aufrecht erhalten werden kann. Firmen unterscheiden sich in ihrer Fähigkeit bereits bestehende Produkte nachzuahmen sowie neue Produkte zu entwickeln, welche auf Grund zufällig generierter Effekte sowie der Tatsache, dass die Agenten während ihres Handelns stets dazulernen, erreicht wird. Dies führt zu einer Veränderung der Organisationsstruktur jeder einzelnen Firma.

Um zusätzliche Arbeitszeit zu sparen und die Sucheffizienz der Agenten zu erhöhen, findet ein automatischer Zusammenschluss dieser statt. Solche Multi-Agenten sind effizienter im Aufsuchen von nützlichen vertragsgemäßen Absprachen in Bezug auf den jeweiligen Markt, jedoch erfordert dies allerdings komplizierte Einstellungen und große Strategien. Auch auf diesem Forschungsgebiet der Multi-Agenten-Systeme findet sich eine große Menge an Wissenschaftlern ein, welche sich mit der Thematik automatisierter Märkte auseinandersetzen. Bis heute konzentrieren sich sämtliche Facharbeiten auf die Ausführung, den Ablauf und sicherheitstechnische Problemstellungen.

Im Allgemeinen kann man sagen, dass durch die vertragliche Bindung der Agenten, ihre Fähigkeit auf unerwartete Ereignisse zu reagieren, gehemmt wird. Es wurde eine Art der Vertragsbindung vorgeschlagen, welche den Agenten gestattet aus einer vertraglichen Bindung auszutreten. Jedoch zieht dies immer eine Geldstrafe mit sich, welche an den jeweiligen Vertragspartner zu zahlen ist. Folglich hängen der Verlauf der Vertragsbindungen sowie der -auflösungen stark von den gestaffelten Pönalen ab.

Andersson und Sandholm [ANDERSSON, 2001] erstellten ein Multi-Agenten-Modell eines automatisierten Verhandlungssystems, um die Empfindlichkeit der Erträge von den gestaffelten Bindungsverträgen zu analysieren. Dies bewerkstelligten sie indem sie Änderungen an der Strafgeldverteilung und den Verhandlungsagenten selbst vollzogen haben. Vier verschiedene Arten von Strafen wurden hierbei bedacht:

- Fix definierte Strafen
- Strafen sind prozentueller Anteil des Vertragswertes
- Erhöhung der Strafen abhängig von Vertragsabschlussdatum
- Erhöhung der Strafen abhängig von Vertragsauflösungsdatum

Die Agenten unterscheiden sich untereinander durch verschieden weite Voraussicht und dem Grad an eigennützigem Verhalten. Mehrere Fälle von Aufgabenzuteilungen wurden getestet, wobei jedem Fall fünf Verhandlungsrunden zugeschrieben wurden. Mit all den analysierten Einstellungen konnte gezeigt werden, dass die Auswahl relativ gering war, jedoch eine positive Strafe für Vertragsbrüche am besten funktionierten. Andersson und Sandholm konnten durch ihre Versuche noch einen weiteren interessanten Aspekt

aufzeigen: unter der Bedingung, dass die Strafen einer Vertragsauflösung gering gehalten werden, können stark eigennützige nicht vorausschauende Agenten im Gegensatz zu Agenten, welche kooperativ und mit wenig Voraussicht ausgeprägt sind, schnell ein hohes Sozialfürsorgeniveau erreichen. Je größer die Voraussicht eines Agenten, desto besser ist auch seine gebrachte Leistung; Agenten, welche wenig vorausschauend agieren, können nur über kurze Strecken mithalten.

Vergleich zwischen Realität und Multi-Agenten

Direkte Experimente mit menschlichen Subjekten sind zu einem wichtigen Forschungswerkzeug herangewachsen. Ein bleibendes Problem jedoch ist, dass man nie genau wissen kann warum eine spezielle Person die jeweilige Wahl getroffen hat. Vielmehr müssen die Präferenzen, Gedanken und Verhaltenseinstellungen über die getroffenen Entscheidungen und gewählten Aktionen abgeleitet werden. Im Vergleich dazu erstellt ein Forscher bei Multi-Agenten-Systemen ein bestimmtes Ausgangsszenario. Über eine gewisse Zeit bewirken die im System befindlichen Agenten eine Veränderung Anderer und somit ihrer Umwelt. Nach dem Durchlauf des Experiments kann der Forscher schließlich rückwirkend den gesamten Ablauf bis zum Ausgangswert analysieren. Die Schwierigkeit hierbei ist, dass sich der entwickelnde Lernvorgang nicht vollkommen realistisch implementieren lässt.

Diese Beobachtungen lassen eine synergetische Wirkung von Parallel-Experimenten erkennen, welche zwischen realen Personen und computerberechneten Agenten durchgeführt werden. Hierbei kann das menschliche Verhalten für eine genauere Abbildung bzw. Spezifikation des Lernverhaltens der Multi-Agenten hergenommen werden. Umgekehrt kann das Verhalten der Multi-Agenten dazu verwendet werden um Hypothesen zu formulieren, welche die Grundveranlassungen des menschlichen Verhaltens abbilden.

Duffy [DUFFY, 2001] verwendet diese Methode um ein Auftreten eines allgemeingültigen Zahlungsmittels, wie beispielsweise Geld, zu untersuchen. Bei diesen Parallelexperimenten kommen ähnliche Versionen von Suchmodellen für Geld zum Einsatz. Man geht davon aus, dass ein Agent ein bestimmtes Gut konsumiert, jedoch ein gleiches Gut, um eine Einheit erhöht, produziert. In jeder Simulationsperiode werden nach einer zufälligen Auswahl Agenten gepaart, welche dann entscheiden müssen ob sie untereinander Güter austauschen

wollen. Die Entscheidung einen Handel einzugehen kann entweder von der Tatsache ausgehen, dass das Gut für den eigenen Konsum vorgesehen ist oder um das Gut zu lagern, um es bei einem späteren Handel einzusetzen. Dabei ist zu beachten, dass jedes Gut unterschiedliche Lagerkosten mit sich bringt. Die Problemstellung bildet sich nun wie folgt: es soll beobachtet werden, ob das Konsumverhalten darauf hinausläuft, dass eine Verwendung von speziellen Güter stattfindet. Unter speziellen Gütern versteht man solche, die für einen Handel zwar akzeptiert werden, jedoch nicht direkt von dem jeweiligen Agenten konsumiert werden können. Die von den Multi-Agenten angewendeten Verhaltensregeln, um deren Handel zu steuern, werden aus Basisinformationen von bereits durchgeführten menschlichen Versuchen gewonnen. Die Agenten wählen eine aus einer Menge an anwendbaren Verhaltensregeln, wobei sie durch eine vereinfachte Form eines intensiven Lernens weiterlernen. Durch dieses Experiment konnte Duffy nachweisen, dass sowohl bei den Versuchen mit Multi-Agenten als auch bei jenen mit menschlicher Teilnahme die Grundeigenschaften übereinstimmten.

Duffy [DUFFY, 2001] ging dann noch einen Schritt weiter, indem er weitere Multi-Agenten-Simulationen startete. Welche Ergebnisse sind zu erwarten, wenn man nun zwei abgeänderte Suchmodelle für Geld zum Einsatz kommen lässt, welche so ausgelegt sind, dass bestimmte Agenten mehr ein spekulatives Verhalten zugeordnet wird. Spekulatives Verhalten tritt auf, wenn ein Agent ein Gut durch einen Handel erwirbt, welches höhere Lagerkosten hat als jenes, welches er bereits auf Lager hat. Dieser Handel wird deshalb durchgeführt, da der Agent erwartet, dass das kostenintensivere Gut eine höhere Akzeptanz bei den anderen Handelspartnern bei späteren Transaktionen mit sich bringt. Duffys Erkenntnisse aus den Experimenten mit den abgeänderten Suchmodellen waren jene, dass die Agenten schneller lernen ihre spekulativen Strategien zu entwickeln. Dies wiederum erhöht die Wahrscheinlichkeit eines Zusammenlaufens zu einem spekulativen Gleichgewichts. Bei dem Vergleich des Multi-Agenten-Modells mit einem mit menschlichen Akteuren konnte ein erfreuliches Ergebnis festgestellt werden: die Resultate beider Versuche wahren ungefähr miteinander zu vergleichen.

Programmierwerkzeuge für Multi-Agenten-Systeme

Viele Ökonomen befürworten die Verwendung von computerberechneten Multi-Agenten-Systemen um ökonomische Theorien zu testen und zu bestätigen. Nobelpreisträger Robert Lucas [LUCAS, 1987] meint sogar „(A theory) is not a collection of assertions about the behavior of the actual economy but rather an explicit set of instructions for building a parallel or analog system a mechanical, imitation economy. (Our) task as I see it (is) to write a FORTRAN program that will accept specific economic policy rules as input and will generate as output statistics describing the operating characteristics of time series we care about, which are predicted to result from these policies."

Um sich einen Vorteil aus der Einführung von leistungsstarken Multi-Agenten-Werkzeugen zu verschaffen, fordert Lane den Einsatz von computerberechneten Multi-Agenten-Systemen mehr zu forcieren. Durch diese Technik ist es sehr einfach ein maßgeschneidertes Modell, welches den eigenen speziellen Forschungskriterien genügt, zu erstellen. Mit dem Ansatz der objektorientierten Programmierung kann eine Umgebung erschaffen werden, welche eine Sammlung verschiedenster Arten von Agenten beherbergt. Aus dieser Sammlung kann man schließlich mit einem geeigneten Interface auf einfache Art und Weise verschiedene Objekte miteinander kombinieren, und so bestimmte wirtschaftliche Experimente durchführen.

Der derzeitige Nachteil von Multi-Agenten-Simulationen besteht für viele Ökonomen darin, dass zur Durchführung der Experimente ein großes programmiertechnisches Wissen erforderlich ist. Ein Beherrschen von mächtigen Programmiersprachen wie Java und C++ setzt einen hohen Zeitaufwand voraus, da ein Aneignen der jeweiligen Sprache von Grund auf notwendig ist um Multi-Agenten-Simulationen realisieren zu können. Natürlich gibt es auch leicht zu erlernenden Sprachen wie StarLogo und AgentSheets, jedoch weisen diese eine zu geringe Mächtigkeit auf, so dass man mit diesen komplexe ökonomische Anwendungen erstellen könnte. Glücklicherweise gibt es eine Menge von Multi-Agenten-Softwaresammlungen, welche derzeit ständig erweitert werden und den Einstieg in das Modellieren von derartigen Simulationen drastisch erleichtern. Eine der ersten Sammlungen war Swarm (basierend auf Objective C), welche auch gleich die Vorlage für die nachfolgenden Entwicklungen wie Ascape und Repast (basierend auf Java) bildete. Diese Entwicklungswerkzeuge bieten eine Menge an dienlichen Bibliotheken von Software an, welche speziell für die Entwicklung von Multi-Agenten-Systemen im Bereich der Sozialwissenschaften konzipiert wurden.

4.2 Gegenüberstellung der Vor- und Nachteile

Über die letzten fünfzig Jahre ist eine deutliche Kluft zwischen den Theoretikern aus dem Bereich der Ökonomie und denen aus dem Bereich der Sozialwissenschaften entstanden, da die Wirtschaftswissenschafter dahin tendieren ihre Studien über mathematische Systeme wiederzugeben. Diese bestehen erwartungsgemäß aus stochastischen nichtlinearen Differenzen- oder Differentialgleichungen, welche jedoch von den Sozialwissenschaftlern als eher ungeeignet angesehen werden, um die soziale Wirklichkeit abzubilden. Im Vergleich dazu vereinen Multi-Agenten-Systeme, durch die Anwesenheit der selbständigen lernfähigen Agenten, ökonomische, soziale sowie auch umweltbezogene Elemente. Die Dynamik des Ablaufes einer Multi-Agenten-Simulation ist bestimmt durch die Interaktion zwischen den Agenten, nicht durch exogen abhängige Gleichungen. Der jeweilige Zustand des gesamten Multi-Agenten-Systems wird zu jeder Zeit durch den jeweiligen Zustand aller Agenten definiert. Neben der Tatsache, dass die Forscher durch diese Technologie immer einen direkten Einblick auf den Ablauf besitzen, werden des Weiteren noch die Transparenz und die Klarheit des Modellierungsvorganges erhöht. Die bereits abgehandelten Studien können beweisen, dass durch einfaches individuelles Verhalten Regelmäßigkeiten in einer großen Ansammlung von Agenten gefunden werden können. Da Multi-Agenten-Systeme noch zusätzliche Unterstützung von empirischen Untersuchungen erhalten, kann man somit Verbesserungen in der Klarheit der Multi-Agenten-Systemmodellierung bewirken.

In den Arbeiten von Sargent [SARGENT, 1993] werden Multi-Agenten-Systeme dazu verwendet, bereits bestehende wirtschaftliche Theorien, welche mit konventionellen Methoden modelliert wurden, zu testen. Können sich die Agenten untereinander tatsächlich, abhängig von den in der Theorie bestimmten Arten von Gleichgewichten, koordinieren? Wenn mehrere Gleichgewichte vorliegen, welches wird dominieren und in den Vordergrund treten?

Multi-Agenten-Systeme können auch dazu verwendet werden, um die Robustheit der Theorien zu überprüfen und eine Bestätigung der bereits getroffenen Annahmen zu erbringen. Um die Stärken von Multi-Agenten-Systemen hervorzuheben, müssen sich Forscher auf bestehende kritische Problemstellungen aus den Bereichen ökonomischer und sozialer Wissenschaften konzentrieren. Es müssen mit Hilfe von computerberechneten Simulationen, welche kontrollierte und reproduzierbare Ergebnisse liefern, klar formulierte

Hypothesen getestet werden. Sie müssen ihre gefundenen Ergebnisse in statistischen Zusammenfassungen übermitteln, welche transparent und exakt aufgebaut sind. Das Vertrauen in die statistischen Ausgaben soll weiters gesteigert werden, indem sie in regelmäßigen Abständen mit Ergebnissen anderer Wissenschaftler, welche andere Wege benutzt haben um das Thema zu studieren, verglichen werden. Zu guter Letzt müssen sie ein Anwachsen von robusten Ergebnissen garantieren, so dass andere Forscher stets auf bereits vorliegenden passenden Studien aufbauen können. Das Erfüllen all der oben erwähnten Voraussetzungen stellt mit Sicherheit eine schwere Herausforderung dar, welche durch eine interdisziplinäre Zusammenarbeit zu erfolgen hat. Jedoch lässt die Erfahrung erahnen, dass dieses Vorhaben von Grund auf schwierig ist, da einfach die notwendige Routine in der Kommunikation unterschiedlicher Disziplinen fehlt.

In Wirklichkeit benötigt ein Forscher, um Multi-Agenten-Simulationen durchzuführen, Grundwissen auf dem Gebiet der Programmierung, statistische Kenntnisse und ein passendes Training für den jeweiligen Anwendungsbereich.

4.3 Notwendigkeit von Multi-Agenten Simulationen

Durch die Tatsache, dass wir uns in einer immer komplexer werdenden Welt aufhalten, wird eine Anwendung von agenten-basierte Computersimulationen geradezu notwendig. Die meisten zu analysierenden Systeme weisen eine so hohe Konzentration von gegenseitigen Abhängigkeiten auf, dass man diese mit gewöhnlichen Modellierungstechniken nicht abbilden könnte. Ein klassisches Beispiel der heutigen Zeit ist die Deregulierung der Energiemärkte. Die Abhängigkeiten der Infrastrukturen wie z.B. Elektrizität, Gas, Wasser oder Telekommunikation geraten immer mehr in den Blickpunkt der Öffentlichkeit, da diese Bereiche immer mehr an ihre Grenzen stoßen und regelmäßige Lieferengpässe oder sogar Totalausfälle mit sich bringen.

Des Weiteren gab es schon immer Modelle, welche mit einem traditionellen Ansatz nicht korrekt analysiert werden konnten. Für die Modellierung von ökonomischen Märkten mussten immer gleichzeitig Einschränkungen wie die Annahme von perfekten Märkten, homogenen Agenten oder Langzeitgleichgewichte in Kauf genommen werden. Gerade durch diese Beschränkungen ließen sich solche Problemstellungen analytisch und rechnerisch nicht

realitätsnah umsetzen. Genau hier lassen sich die Systeme mit Hilfe von agenten-basierten Simulationen viel realistischer abbilden.
Berücksichtigt man auch noch die Tatsache, dass die Daten viel organisierter in Datenbanken abgelegt werden können, wobei eine viel feinere Auflösung der Informationen erreicht wird, werden auch so genannte Mikrosimulationen realisierbar. Da die Computertechnologie mit großen Schritten voranschreitet und immer leistungsstärkere Prozessoren erhältlich sind, werden Computersimulationen immer effizienter und erfreuen sich daher auch immer größerer Beliebtheit.
Zusammenfassend kann man nun sagen, dass die traditionellen Modellierungstechniken nicht mehr den benötigten Ansprüchen genügen, und neue Ansätze gefunden werden müssen, um die heutige komplexe Welt zu beschreiben.

Hintergrundinformationen zu agenten-basierten Simulationen

Anwendungen von agenten-basierten Computersimulationen findet man in vielen wissenschaftlichen Bereichen wie z.B. System-, Computer-, Managementwissenschaften sowie traditioneller Modellierung und Simulation. Der theoretische Ansatz, die konzeptuelle Betrachtungsweise der realen Welt und die passenden Modellierungstechniken sind mitunter Gründe warum Agentensimulationen (CAS) in immer mehr wissenschaftlichen Bereichen zu finden sind. Geschichtlich hat sich diese Simulationstechnik aus den so genannten komplexen adaptiven Systemen entwickelt, welche bereits den Ansatz, Systeme von Beginn an zu analysieren, verfolgten. Dieser Vorgänger zeigte bereits auf welch komplexes Verhalten zwischen kurzsichtigen, selbständigen Agenten auftreten kann. John Holland [HOLLAND, 1995], ein Pionier auf diesem Gebiet, definierte Eigenschaften und Mechanismen, welche für alle CAS Gültigkeit haben. Die Eigenschaften wurden wie folgt eingeteilt:

- Aggregation erlaubt es Agenten Gruppen zu bilden
- Nicht-Linearität lässt einfache Hochrechnungen nicht zu
- Strömungen erlauben den Transfer von Ressourcen und Informationen
- Vielfältigkeit bietet den Agenten die Möglichkeit unterschiedlich zu reagieren

Folgende Mechanismen wurden durch John Holland definiert:

- Bezeichnungen bieten die Möglichkeit Agenten zu benennen und wieder zu erkennen
- Das interne Modell lässt Agenten innerhalb ihrer Welt überlegt agieren
- Bausteine erlauben ein großes komplexes System in mehrere kleinere verständlichere Komponente zu gliedern

Die oben genannten Eigenschaften und Mechanismen bilden zusammen ein nützliches Framework um agenten-basierte Computersimulationen zu erstellen. Während seinen Untersuchungen von CAS entwickelte John Holland weiters genetische Algorithmen, welche auf den Techniken der genetischen und natürlichen Selektion basieren und den Grundstein für Schwarmoptimierungsalgorithmen bilden.

4.4 Das junge Forschungsgebiet und seine Wertigkeit

Simulationen sind ein relativ neues, jedoch immer öfter eingesetztes Werkzeug im Bereich der Sozialwissenschaft geworden. Wie sooft bei neuen Forschungsgebieten konnten die gewonnenen Ergebnisse den anfänglichen Versprechen nicht Stand halten. Arbeiten aus dem Bereich der Sozialwissenschaft, welche Simulationen für die Lösung von Problemstellungen verwenden, werden derzeit grundsätzlich von simulationsinteressierten Forschern betrieben. An der Anzahl und der Verteilung der Publikationen auf die Journale, sie sind allesamt gleichmäßig auf die einzelnen Wissensportale verteilt, lässt sich erkennen, dass sich noch kein Journal gefunden hat, welches sich der Thematik Simulation als Spezialgebiet angenommen hätte.
Bratley, Fox und Schrage [BRATLEY, 1987] definierten den Begriff Simulation wie folgt: "Simulation means driving a model of a system with suitable inputs and observing the corresponding outputs". Obwohl diese Definition bereits einen Einblick gewährt wie diese Technik funktioniert, lässt sie allerdings nicht erahnen wie viele verschiedene Zwecke Simulationen erfüllen können.
Um den Stellenwert der Simulation in Bezug auf wissenschaftliche Vorgehensweisen abschätzen zu können, muss man die Durchführung wissenschaftlicher Experimente völlig

neu überdenken. Es gilt zumindest die Herausforderung mit den beiden Standardmethoden, nämlich Induktion und Deduktion, aufzunehmen. Während man unter Induktion eine Mustererkennung aus empirischen Daten versteht, wird bei der Deduktion eine Sammlung von Axiomen erstellt, welche danach erst bewiesen werden müssen. Die Simulation versucht nun einen dritten Weg einzuschlagen, um Wissenschaft zu betreiben. Wie bei der Deduktion werden Annahmen aufgestellt, jedoch werden diese nicht bewiesen, sondern statistische Daten generiert und induktiv analysiert. Dabei ist zu beachten, dass die von der Simulation gewonnenen Informationen von einer Menge streng definierter Regeln abhängen, und nicht von Daten, welche effektiv in der Wirklichkeit auftreten.

Man kann also eine Simulation als eine Art von Werkzeug auffassen, mit dessen Hilfe sich gedankliche Experimente durchführen lassen. Während einzelne Annahmen vielleicht noch leicht zu treffen sind, sind deren Konsequenzen nicht wirklich einfach zu erfassen. Die interagierenden Agenten lösen eine große Menge an Effekten aus, welche auch als auftretende Eigenschaften des Systems bezeichnet werden. Die Ergebnisse dieser Eigenschaften sind meist überraschend, da eine Vorhersage selbst bei trivialen Formen von Interaktionen schwer zu treffen sind.

Allerdings gibt es auch Modelle bei denen sich die auftretenden Eigenschaften ableiten lassen. Doch wenn Agenten eher eine lernende als eine optimierende Strategie verfolgen, wird ein Ableiten der Konsequenzen so gut wie unmöglich. Hier muss auf eine Simulation zurückgegriffen werden.

- Prognose

  Durch Übergabe von komplexen Inputs und Abarbeitung derer mittels eines Simulationsmodells können daraus Konsequenzen und daher Vorhersagen getroffen werden.

- Effizienz

  Simulationen können auch dazu verwendet werden, um bestimmte Aufgaben zu erfüllen. Ein beliebtes Forschungsgebiet ist hierbei die künstliche Intelligenz, welche sich unter anderen mit Themen wie medizinische Diagnose, Spracherkennung oder auch Funktionsoptimierung beschäftigt. Die Methodik der künstlichen Intelligenz kann als Simulation von menschlichen Empfindungen, Entscheidungsfindungen oder auch sozialen Interaktionen angesehen werden.

- Training

  Viele der erfolgreichsten Simulationssysteme wurden so eingesetzt, dass durch die Erschaffung einer naturgetreuen und dynamischen Umwelt, Leute durch Interaktion trainiert werden konnten. Flugsimulatoren sind vielleicht ein wichtiges Beispiel der Verwendung von Simulationen für spezielle Trainings.

- Unterhaltung

  Von ernst zu nehmenden Simulationen benötigt es nur mehr einen kleinen Schritt, um in die Unterhaltungsbranche vorzudringen. Flugsimulatoren oder auch Rennsimulationen sind auf Heimcomputern schon seit langer Zeit gerngesehene Spiele.

- Erziehung

  Durch die Kombination von Training und Unterhaltung kann man Simulationen auch für erziehungsmäßige Einsätze nützen. Wieder lassen sich Computerspiele, insbesondere Wirtschaftssimulationen, als gutes Beispiel nennen. Hier muss man durch richtige Organisation, Strategie und Politik eine künstliche Welt erschaffen, wobei eine Vielzahl an Faktoren einstellbar ist. Grundvoraussetzung ist allerdings, dass die Simulation sehr detailliert und naturgetreu ist, damit man die künstliche Welt auch in die reale umlegen kann. Dem Spieler sollen dadurch Zusammenhänge und Gesetzmäßigkeiten spielerisch näherer gebracht werden.

- Beweisführung

  Simulationen werden auch zum Ermitteln von Beweisen eingesetzt. Convays „Spiel des Lebens“ hat gezeigt, dass alleine durch Anwenden von einfachen Regeln komplexes Verhalten entstehen kann.

- Entdeckung

  Während Physiker genaue Simulationsmodelle von Elektronen- oder Planetenbewegungen besitzen, fällt es Sozialwissenschaftlern wesentlich schwerer die Bewegungsabläufe von z.B. Arbeitskräften exakt nachzubilden. Jedoch kann man den Nutzen der Simulationen bereits erkennen, da aus einfacheren Modellen bereits einige Gesetzmäßigkeiten und Zusammenhänge entdeckt wurden.

Entstehung einfacher Regeln aus komplexem Verhalten

Die Diskussionen rund um Agentensimulationen beginnen mit der Einführung eines einfachen Spieles des Mathematikers John Conway mit seinem Modell des "Spiel des Lebens" [GARDNER, 1970]. Es basiert auf einem netzartigen Automaten, mit welchen sich wahrscheinlich die Grundsätze von agenten-basierte Computersimulationen am besten beschreiben lassen. Sie sollten dazu verwendet werden um zu beweisen ob man eine logische Struktur entwickeln kann, welche komplex genug und sämtliche Informationen enthält, um sich selbst zu reproduzieren zu können. Schließlich konnte dieser Beweis durch eine mathematische Repräsentation einer Maschine erbracht werden, welche von netzförmigen Automaten abgeleitet wurde. Ein typischer Aufbau eines netzförmigen Automaten setzt sich aus einem 2-dimensionalen Rasters oder eines Netzwerkes aus Zellen zusammen. Jede einzelne Zelle nimmt einen bestimmten Wert einer begrenzten Zahlenmenge zu jedem Zeitpunkt an. Eine Sammlung von einfachen Regeln bestimmt die Folgewerte, welche auf den aktuellen Werten basieren. Bei jedem Zyklus werden sämtliche Zellen, abhängig von dem Regelsatz, aktualisiert. Des Weiteren fließt auch der Wert der nächsten Nachbarzelle in die Berechnung ein, wobei immer acht Nachbarzellen vorhanden sind. Das System ist deterministisch; ein bestimmter Status einer Zelle in Verbindung eines bestimmten Wertes der nächsten Nachbarzelle führt bei Anwendung einer Regel stets zum selben Updatewert.
Das Modell des "Spiel des Lebens" besitzt drei unterschiedliche Regeln, welchen den nächsten Wert, entweder 0 oder 1, einer Zelle bestimmt. Sie lauten wie folgt:

- Zelle erhält den Wert 1, wenn drei seiner Nachbarzellen den Wert 1 besitzen
- Zelle bleibt unverändert, wenn zwei seiner Nachbarzellen den Wert 1 besitzen
- Sonst erhält die Zelle den Wert 0

Werden viele Zellen zufällig verteilt und mehrere Zyklen durchlaufen, kann man eine charakteristische Musterbildung erkennen. In manchen Fällen stellt sich auch ein Gleichgewicht ein, welches entstandene Strukturen für immer bestehen lässt. Die Annahme einer Nachbarschaft von 8 Zellen bestimmt die Reichweite der Interaktionen, sowie die lokal verfügbaren Informationen jeder Zelle, um ihren eigenen Status zu ändern. Auf diese Art und

Weise konnten folgende Eigenschaften für die, sich immer weiterentwickelnden, Agentensimulationen gefunden werden:

- In Systemen, welche auf lokal verfügbaren Informationen basieren und komplett mit einfachen Regeln beschrieben sind, kann es zu resistenten Musterbildungen kommen.
- Die sich einstellenden Muster sind extrem sensibel, da bereits kleine Abweichungen in der Ausgangskonfiguration zu großen Veränderungen des Endergebnisses führen.

Mit Hilfe von einfachen Regeln und interagierenden Agenten lässt sich zeigen, dass sich in natürlichen Systemen eine Gemeinschaftsintelligenz einstellt, ohne dass eine Anwesenheit einer zentralen Autorität erforderlich ist. Solche Systeme können nicht nur fortbestehen, sondern können sich auch weiterentwickeln und sich ihrer Umwelt entsprechend anpassen, indem sie ihr eigenes Verhalten über die Zeit immer weiter optimieren. Eine Ameisenkolonie muss sich stets organisieren um komplexe Aufgaben wie die Beschaffung von Nahrung und den Bau eines Ameisenhügels zu bewältigen, ohne jedoch irgendwelche Einbußen der Elastizität in Kauf nehmen zu müssen; selbst eine vollkommene Zerrüttung der Struktur lässt das System unbeeinträchtigt weiterfunktionieren.
Eine derartige Gemeinschaftsintelligenz war sehr inspirierend für diverse Optimierungstechniken, konnte zum Beispiel die Optimierung der Ameisenkolonie in der Praxis verwendet werden um Planungs- und Wegbestimmungsprobleme zu lösen.

Unterstützung der Wissenschaften durch Agentensimulationen

In Agentensimulationen werden Personen oder Gruppierungen von Personen durch Agenten selbst und Prozesse der sozialen Interaktion durch Beziehungen zwischen Agenten dargestellt. Die grundlegende Annahme besteht nun darin, dass Personen und ihre sozialen Interaktionen zumindest für einen festgelegten Sachverhalt glaubwürdig modelliert werden können. Die eingeschränkte Reichweite des Verhaltens der einzelnen Agenten steht im Gegensatz zu den eher allgemeinen Zielen des Forschungsgebiets der künstlichen Intelligenz.

Bei Agentensimulationen muss man sich mit folgenden Fragen auseinandersetzen:

- Wie kann man das menschliche Verhalten modellieren?
- Wie lassen sich menschliche soziale Interaktionen abbilden?

Thomas Schelling [SCHELLING, 1971] ist einer der Pioniere auf dem Gebiet der agenten-basierten Simulationen, wobei er Ansätze von zellularen Automaten mit Rassentrennungsverhaltensmuster verband. Er stellte sich die Frage, ob es selbst dann zu Rassenabgrenzungen kommen würde, wenn sämtliche Beteiligten zwischen Hautfarben keine Unterscheidung treffen würden. Mit Hilfe seines Modells, welches später dann unter der Bezeichnung Schellingmodell allgemein bekannt wurde, konnte er aufzeigen, dass sich Ghettos spontan bilden können. Bei allgemeiner Betrachtung veranschaulichte Schelling, dass eine Musterbildung zu erkennen war, selbst wenn dies nicht das eigentliche Ziel der einzelnen Agenten war. Das Schellingmodell erzeugte derart viel Aufsehen, dass es einen immens großen Einfluss auf die Weiterentwicklung von Agentensimulationen mit sich brachte.
Auch im Bereich der Ökonomie sind Multiagentensysteme gern eingesetzte Analysewerkszeuge, mit welchen bereits folgende Annahmen getroffen werden konnten:

- Ökonomische Agenten sind rational. Dies bedeutet, dass sie eine vordefinierte Menge an Zielen besitzen und somit die Möglichkeit haben ihr Verhalten zu optimieren.
- Ökonomische Agenten sind homogen. Sämtliche Agenten im System verfügen über identische Eigenschaften und Verhaltensregeln.
- Ökonomische Prozesse sollen optimiert werden.
- Im Augenmerk der Untersuchungen steht meist das Langzeitgleichgewicht.

4.5 Allgemeiner Ablauf einer Simulation

Um die Kunst der Simulation zu verbessern ist es normalerweise unzureichend, wenn man sich nur über den Zweck der Simulation Gedanken macht. Man muss sich des Weiteren auch

über den Ablauf der Simulation selbst bewusst sein, welcher sich in drei Teilphasen unterteilen lässt: die Programmierung des Simulationsmodells, die Analyse der Ergebnisse und das Verteilen der gewonnen Informationen.

Programmieren des Simulationsmodells

Zunächst muss entschieden werden mit welcher Programmiersprache das Modell implementiert werden soll. Erfahrene Entwickler sollten sich eher an Java halten, da erstens sämtliche Programme auf fast allen Rechnern lauffähig sind und zweitens eine Vielzahl an Softwarepaketen verfügbar sind, welche dafür gestaltet worden sind um Simulationen durchzuführen. Die Umsetzung eines Simulationsmodells mit Hilfe einer Programmiersprache sollte jedoch zumindest drei Ziele erreichen: Gültigkeit, Benutzerfreundlichkeit und Erweiterbarkeit.

Die Gültigkeit setzt sich aus der inneren und der äußeren Gültigkeit zusammen. Die Äußere beschreibt wie gut die Realität durch das Simulationsmodell beschrieben wird. Die innere Gültigkeit hängt von der korrekten Umsetzung des Simulationsmodells ab, welches sich als schwieriges Unterfangen darstellt. Das Problem besteht nun darin einschätzen zu müssen, ob eine überraschende Abweichung auf Grund eines Programmierfehlers oder eines Effekts des Modells auftritt. Meist beansprucht die Überprüfung der Korrektheit der Umsetzung der Simulation mehr Zeitaufwand als die Programmierung selbst.

Unter Benutzerfreundlichkeit versteht man den Sachverhalt, dass nachfolgende Anwender stets die Ergebnisse richtig interpretieren und ein Verständnis vermittelt bekommen, wie sich die Werte zusammensetzen.

Der Sinn der Benutzerfreundlichkeit besteht darin, dass dem Forscher sowie all jenen, welche das Modell in späterer Hinsicht verwenden, stets ein Überblick geboten wird wie sich die Ergebnisse zusammensetzen und wie diese zu interpretieren sind. Beim Modellieren entsteht meist eine Reihe von unterschiedlichen Versionen, welche sich immer nur leicht voneinander unterscheiden. So können zum Beispiel die Art und Weise wie die Daten produziert werden, die unterschiedlichen Eingabeparameter oder die Regeln, welche das Verhalten der Agenten beeinflussen Abweichungen aufweisen. Bei einer unzureichenden Dokumentation ist es sehr schwer die einzelnen Versionen untereinander zu vergleichen.

Die Erweiterbarkeit soll weiteren Forschern ermöglichen ein bestehendes Modell für ihre eigenen Anwendungen einsetzbar zu machen. So soll es später möglich sein, kleine Abänderungen durchzuführen um eventuell ergänzende Fragen zu klären. Um die Erweiterbarkeit erzielen zu können muss dieser Faktor bei der Programmierung des Modells und der Dokumentation stets berücksichtigt werden.

Analyse der Ergebnisse

Simulationen generieren in der Regel eine große Menge an Daten. Falls einmal für die Analyse zu wenige Informationen vorliegen, kann die Simulation eine ihrer Stärken ausspielen, indem man mehrere Durchläufe startet und somit mehr Daten erhält.
Trotz der Reinheit und Vollständigkeit der durch die Simulation gewonnen Daten, ist die Analyse derer bei weiten nicht trivial. Mehrfache Simulationsläufe können sich untereinander unterscheiden, da sie von den Ausgangswerten und den stochastischen Ereignissen abhängen. Die Ergebnisse verstehen sich als pfadabhängig, das heißt, sie werden von der Vorgeschichte beeinflusst. Daher besteht die Herausforderung darin die erhaltenen Informationen mit ihrer Entstehungsgeschichte zu analysieren. Es bestehen drei verschiedene Methoden diesen Zusammenhang zu beschreiben:

- In Form von Nachrichten

  Die Beschreibungen werden chronologisch von Start bis Ende der Simulation protokolliert. Dies ist eine sehr direkte Art die Entwicklung zu vermitteln, jedoch bietet dieser Ansatz wenig Erklärungskraft.
- Aus der Sicht eines gewählten Agenten

  Die Entwicklungsgeschichte wird hier aus der Sicht eines bestimmten Akteurs wiedergegeben. Diese Methode lässt sehr leicht die Wirkungsweise des Simulationsmodells erkennen.
- Globale Sichtweise

  Dieser Ansatz eignet sich gut um die Entstehung von Mustern zu analysieren, jedoch ist hier ein Bedarf an einer detaillierten Aufzeichnung der Entstehungsgeschichte gegeben.

Die Beschreibung von Simulationsabläufen mittels der Analyse der Entwicklungsgeschichte kann sicher dazu genutzt werden um beispielsweise Muster zu erklären, sollte allerdings nicht an diesem Punkt stehen bleiben. Durch die Tatsache, dass es sich hierbei um virtuelle Versuche handelt, muss man stets von einem Auftreten zufälliger Abweichungen rechnen. Daher darf nicht ein einziger Durchlauf der Simulation für die Auswertung verwendet werden, da sonst eventuell falsche Annahmen getroffen werden könnten. Genauso wichtig wie die Beschreibung der Entwicklung eines einzelnen Durchlaufs ist die statistische Analyse einer ganzen Reihe von Abläufen, somit können die gewonnenen Schlussfolgerungen bestätigt, abgeschwächt oder gar abgelehnt werden. Hier finden wir auch einen weiteren Vorteil der Simulationstechnik, da Forscher die Durchläufe immer wieder neu starten können und so sehen, ob auftretende Muster nur zufällig oder regelmäßig auftreten. Neben dem Ansatz mehrere Simulationsläufe mit unterschiedlichen Initialwerten zu starten, besteht natürlich auch die Möglichkeit die Auswirkungen von abweichenden Simulationsparametern zu analysieren. Eine andere Methode wäre schließlich das Simulationsmodell selbst abzuändern und die Abweichungen der Ergebnisse zu vergleichen. In beiden Fällen müssen stets kontrollierte Experimente durchgeführt werden, welche eine Reihe von vollständigen Simulationsdurchläufen beinhalten. Um mittels der Statistik nun die Effekte zu analysieren, greift man auf die Regression zurück, wenn es sich um quantitative Abweichungen handelt. Anderes bei qualitativen Abweichungen, hier kommt die Varianzanalyse zum Einsatz. Wie es in der Statistik üblich ist, müssen allerdings zwei Punkte berücksichtigt werden: zum Einen sind die Abweichungen signifikant? Damit versteht man den Ausschluss einer möglichen Bildung der sich ergebenden Differenzen durch Zufall. Zum Zweiten sind die Abweichungen substantiell signifikant? Dadurch wird sichergestellt, ob die Unterschiede eine derartige Gewichtung haben, dass sie als bedeutend angenommen werden können.

Veröffentlichung der gewonnenen Informationen

Nach der Modellentwicklung, Implementierung der Simulation in einer dafür sinnvollen Programmiersprache und anschließender Analyse der sich ergebenden Informationen. kommt nun der letzte Schritt: die Veröffentlichung der Ergebnisse und somit Möglichkeit der Weiterverwendung des erarbeiteten Wissens für andere Forscher. Wie in den meisten

wissenschaftlichen Bereichen wird dies durch eine Publikation in namhaften Journalen oder in Sammlungen kurzer wissenschaftlicher Berichte bewerkstelligt. Im Bereich der Sozialwissenschaft muss man allerdings mit gewissen Einschränkungen der Veröffentlichung rechnen, da sich die erzielten Ergebnisse, welche anhand einer Simulation ermittelt wurden, meist nicht leicht erklären lassen. Grund dafür ist zum einen, dass die gewonnenen Daten in einem sehr starken Bezug zu dem jeweiligen Modell stehen. Falls der Detailgrad des Modells einfach zu hoch ist oder dieses einfach unzureichend genau beschrieben wurde, ist meist die Funktionsweise bzw. der gesamte Ablauf der Simulation sehr schwer nachzuvollziehen. Ein weiteres Problem besteht darin, dass die Beschreibung eines oder mehrerer Simulationsläufe meist in einer erzählerischen Form vorliegt und somit ein großer Interpretationsspielraum geschaffen wird. Im Gegensatz zu statistischen Analysen, welche mit konkreten Zahlenwerten, diversen Berechnungen, Tabellen oder Darstellungen erstellt werden, lassen sich Schlussfolgerungen anhand der Beschreibung der Entwicklungsgeschichte von den vorliegenden Durchläufen nur schwer wiedergeben. Dies liegt wiederum am notwendigen Verständnis der Wirkungsweise des jeweiligen Simulationsmodells, welches zwar für das Gesamtverständnis unabdingbar ist, allerdings wie vorher erklärt durch eine kurze Abhandlung meist nicht wiedergegeben werden kann.

4.6 Validierung eines agenten-basierten Modells

Traditionelle Modellvalidierung besitzt eine reichhaltige und gutdokumentierte Geschichte, welche die Grundlage für die Validierung von agenten-basierten Modellen darstellt. Gewöhnlich versteht sich eine Modellanalyse so, dass Ergebnisse des erstellten Modells systematisch mit denen des echten Systems verglichen werden. Einige wissenschaftliche Arbeiten erweiterten diesen Ansatz um die Einführung eines objektorientieren Verhaltens.
Burton [BURTON, 1998] veranschaulichte die Vorteile der Verwendung einer Kombination mehrerer Modelle für Vergleichszwecke, und demonstrierte somit ein spezielles Problem der Modellzusammenführung. Axelrod [AXELROD, 2005] bemerkte, dass eine Replikation ein entscheidender Qualitätsfaktor eines Simulationsmodells ist und bestimmt drei grundlegende Schritte, welche bei einem Modellvergleich zu beachten sind:

- Numerische Gleichheit
- Verteilte Übereinstimmung
- Relationale Übereinstimmung

Er lieferte eine systematische Analyse der auftretenden Probleme bei Versuchen eine Gleichheit herzustellen.

Analyse

Damit agenten-basierte Simulationen auch für die Wissenschaft Verwendung finden können, müssen zwei Kriterien erfüllt werden. Sie stellen erstens ein Hilfsmittel für Erklärungen dar und beinhalten zweitens Mechanismen um diese Erklärungen zu verfeinern. Die Größe der erklärenden Kraft eines Modells wird dadurch bestimmt, wie ein beobachtetes Metalevel-Phänomen durch das Verhalten von Mikrolevel-Akteuren abgebildet werden kann.
Dabei werden die generierten Ergebnisse eines Modells einerseits über Langzeitläufe und andererseits über wiederholte Durchläufe getestet. Diese Tests können wiederum mit den gleichen Parametereinstellungen vorgenommen werden, um die Spannweite der sich immer wieder anders einstellenden Ergebnisse zu analysieren. Jedoch lassen sich auch die Parameter systematisch ändern, damit man die jeweiligen Einflüsse beobachten kann. Schlussendlich werden alle Ergebnisse des Simulationsmodells mit dem Zielsystem verglichen.
Axelrod und Testfatsion [AXELROD, 2006] widmeten sich speziell der Problematik der Modellvalidierung und definierten vier Ziele:

- Empirisch

  Warum bilden sich eine Menge von beständiger Regelmäßigkeiten, auch wenn nur eine kleine Kontrolle von oben herab existiert?
- Verständlich

  Wie können agenten-basierte Systeme als Wissensfabriken verwendet werden um gute Designs zu entdecken?

- Heuristisch

  Wie kann man sich einen größeren Einblick über die grundlegenden Mechanismen der sozialen Systeme verschaffen?
- Methodische Verbesserung

  Wie kann man Anwender von agenten-basierten Simulationen mit Methoden und Werkzeugen unterstützen, so dass diese ernsthafte Untersuchungen an sozialen Systemen durchführen können?

Validierung und analytische Lösungen

Seit agenten-basierte Simulationen in Gebieten wie der Sozialwissenschaft, Wirtschaft und Politik immer häufiger angewendet werden, war somit auch gleichzeitig der Bedarf an Methoden zur Validierung und Verifikation gegeben. Durch eine Verifikation wird sichergestellt, dass sich das Modell so verhält wie es geplant war. In der Sozialwissenschaft kann dieser Prozess faktisch mit einer Vielzahl an praktischen Versuchen gleichgesetzt werden. Mittels der Validierung wird schließlich sicher gestellt, dass das Simulationsmodell auch wirklich das zu modellierende Echtsystem abbildet. In Bereichen wie Biologie führt man den Prozess meist anhand eines Hypothesentests durch.

Es ist allgemein bekannt, dass agenten-basierte Simulationen eine derartige Komplexität aufweisen können, dass ihre Ergebnisse jegliche Werte annehmen können. Dies führt jedoch dazu, dass das System nicht über die Eigenschaft der Generalisierung verfügt und somit keine voraussagende Kraft besitzt. In der Praxis stellt jedoch das Entwickeln und Testen von agenten-basierten Simulationen sowie das Anpassen der Datensätze eine schwierige Aufgabe dar.

Sobald ein Simulationsmodell erstellt wurde um die gewünschten Annahmen zu erreichen und die erhaltenen Datensätze durch minimale Korrekturen mit dem erwarteten Ergebnis übereinstimmen, kann man bereits von einer teilweisen Validierung sprechen. Natürlich ist die Qualität der Validierung umso höher je mehr Datensätze übereinstimmen, was gleichzeitig verdeutlicht, dass ein Modell prinzipiell weder als gültig noch als ungültig angesehen werden kann. Stattdessen kann ein Simulationsmodell wie eine wissenschaftliche Hypothese angesehen werden; sie werden umso glaubwürdiger je öfter eine Validierung stattgefunden hat, doch können sie nie die vollständige Korrektheit beweisen. Die einzige

Möglichkeit um ein totales Verständnis eines Modells zu erlangen besteht darin, dass logische und formelle Analysen durchgeführt werden.

Aus diesem Grund bevorzugen viele Forscher formell analytische Modelle anstatt agenten-basierter Modellen, jedoch gibt es einige Gründe warum man doch auf die Verwendung letzterer nicht verzichten sollte:

Auch formell korrekte Modelle müssen nicht unbedingt richtig sein, wenn bereits von falschen Annahmen und Voraussetzungen ausgegangen wird. Zusätzlich können agenten-basierte Simulationen mittels ihren experimentellem Ansatz dazu verwendet werden, um bereits gültige Modelle zu überprüfen.

Der Vorteil von agenten-basierten Simulationen besteht darin, dass ihre Ergebnisse meist sehr zugänglich und intuitiv sind. Solche Modelle spielen in der Wissenschaft eine entscheidende Rolle um mit deren Hilfe das Verständnis von Problemstellungen zu erhalten. Dies beinhaltet auch das Erlangen eines formell analytischen Verständnisses eines Systems mittels Ermittlung aller möglichen Lösungen. Es gibt viele Klassen von dynamischen Systemen, welche nicht empfänglich für geschlossene analytische Lösungen sind.

Wenden wir uns wieder dem Gegenstand der Verifikation zu; dieses Thema wird schändlicher Weise in rein formellen Systemen vernachlässigt, wo Validierung nicht mit Echtdaten begründet ist.

Formelle Systeme werden in Bereichen der Mathematik oder ähnlichen Disziplinen als Mechanismus der Wissensfindung angewendet, was verständlicherweise eine kritischere und kompliziertere Verifikation mit sich bringt. Wenn eine Validierung mittels Hypothesentests in Bezug zum Echtsystem vollzogen wurde, stellt sie selbst eine Art von Verifikation dar.

Wenn ein Simulationsmodell glaubwürdig das Echtsystem abbildet und vorausschauend die Werte berechnet, dann spricht man auch im formellen Sinn von einem Modell. Verifikation in dieser Form eines wissenschaftlichen Prozesses wird analog, aber nicht vollständig identisch, in einen Prozess des Modellverständnisses umgewandelt. Dieser Prozess kann das jeweilige Modell vereinfachen beziehungsweise generalisieren oder sogar Beziehungen der einzelnen Komponente aufdecken.

Agenten-basierte Simulationen als wissenschaftliche Hypothese

In wissenschaftlichen Disziplinen wie Biologie oder Psychologie bestehen prinzipiell nur zwei wichtige Kriterien um ein agenten-basiertes Modell zu validieren:

- Stimmt das Verhalten der agenten-basierten Simulation mit jenem des abzubildenden Echtsystems im Sinne der Standardwerte der psychologischen Beurteilung überein?
- Stimmen alle Eigenschaften der Agenten und der Umgebung der agenten-basierten Simulation mit denen des Echtsystems überein und sind diese auch nachvollziehbar?

In Anbetracht der psychologischen Beurteilung hängen diese Standardwerte vom Erfolg der vorangegangenen erklärenden Versuchen ab. Wenn keiner dieser sich entsprechend erklären lässt, dann ist es unter Umständen ratsam qualitative Übereinstimmungen zwischen dem Modell und dem Echtsystem zu ermitteln.
Existiert jedoch ein weiteres konkurrierendes Modell, dann müssen standardmäßige statistische Hypothesentests durchgeführt werden, um ermitteln zu können welches der beiden die besseren Übereinstimmungen liefert. Ein zweites Kriterium wird durch die Tatsache gebildet, ob der Entwickler des Modells den künstlichen Agenten Eigenschaften gegeben hat, welche Subjekte im Echtsystem nicht besitzen können. Folgendes Beispiel soll diesen Sachverhalt demonstrieren: bei dem Versuch die Ursprünge der Theorie des Verstands anhand von künstlichen Agenten, welche jederzeit über einen Zugang zu den Zuständen der anderen Agenten verfügen, zu erklären, wurde damit bereits der Endzustand des Systems modelliert. Wobei es dabei zu keiner Erklärung kommt, wie diese Eigenschaft überhaupt zustande gekommen ist. Man sollte jedoch berücksichtigen, dass ein solches Modell auch sinnvoll angewendet werden kann, wenn der wahre Endzustand des Systems angezweifelt wird. Man könnte damit zum Beispiel zeigen, dass die zuvor beschriebenen „allwissenden“ Agenten geringere soziale Fähigkeiten besitzen als Agenten, welche über ein unvollständiges Wissen verfügen. Dies könnte auch zu Änderungen von bereits getätigten Annahmen führen, da man leicht denken könnte, dass sozialere Agenten über einen höheren Wissensstand verfügen müssten.

Um eine Analyse eines agenten-basierten Modells durchführen zu können, bedarf es drei Prüfungsphasen:

- Replikation

  Die erste Phase wird durch die Replikation repräsentiert. Für bereits veröffentlichte Modelle scheint dieser Vorgang zwingend notwendig, da die Ergebnisse des jeweiligen Modells jederzeit durch eigene Durchläufe auf anderen Rechner kontrolliert werden können. Jedoch kann das Erstellen eines Modells anhand einer rein schriftlichen Beschreibung zu einer sehr schwierigen und herausfordernden Aufgabe ausarten. Die Bemühungen lohnen sich allerdings, da man unter Umständen entscheidende Aspekte des Modells erkennen kann, welche der ursprüngliche Autor entweder als gegeben angenommen hat oder sie bei seinen Ausarbeitungen einfach übersehen hat. Es ist gut möglich, dass Simulationsmodelle zwar als gültig eingestuft werden, ohne jedoch bereits vollkommen überprüft oder verstanden worden zu sein. Die Aufgabe jeder wissenschaftlichen Hypothese besteht darin, dass das Verständnis der Theorie verbessert wird und der wissenschaftlichen Gemeinschaft näher gebracht wird. Sind erst die kritischen Attribute des Modells soweit erklärt, dass sie vollends verstanden werden, kann man in die zweite Phase übergehen.
- Verständnis des Modells

  In dieser Phase werden die implizierten oder expliziten Beziehungen der Attribute sorgfältig ermittelt und gewissenhaft analysiert. Wie in jeder wissenschaftlichen Disziplin durchlaufen wir einen Prozess, der das Auffinden von überprüfbaren Voraussagen und die daraus resultierenden Folgerungen darstellt, welche aus den aufgestellten Hypothesen resultieren.
- Testen

  Die dritte und letzte Phase der Modellanalyse besteht nun darin, die aus Phase 2 gewonnenen Voraussagen und Folgerungen zu überprüfen. Wobei zunächst auf die bereits vorhandene Literatur zurückgegriffen wird, bevor man möglicherweise über neue Experimente nachdenkt und diese dann auch durchführt

Es sei angemerkt, dass die Güte der vorhandenen Daten nicht unbedingt ein erforderliches Maß der Bewertungsmodelle darstellt. Die Computerwissenschaft demonstriert, dass für

viele Arten von Computerberechnungen eine unzählige Anzahl an Mechanismen besteht, wodurch man grundlegend äquivalente Ergebnisse erzeugen kann. Steht man vor der Wahl von zwei Modellen, welche beide die gleichen Voraussagen treffen, wird man sich wohl meist für das einfachere der beiden entscheiden. Unterscheiden sich die Prognosen, hängt der Entscheidungsprozess von der Güte der Daten, der Einfachheit des Modells und der Möglichkeit der Vereinfachung ab. Dieser Vorgang kann aber durchaus kompliziert werden, und wird in zahlreichen Publikationen unter dem Begriff Modellselektion abgehandelt.
Agenten-basierte Simulationen wie auch die Wissenschaft selbst folgen dem Trend der Sparsamkeit. Komplexes individuelles Verhalten ist schwierig mit Programmiersprachen zu implementieren, bedarf einer langen Zeit um einen Durchlauf zu tätigen und bringt eine aufwendige Analyse mit sich. Aus diesem Grund achtet man darauf Problemstellungen auf simplen Wegen zu lösen.
Unser Verständnis über Evolution rechtfertigt bis zu einem gewissen Maß die Annahme, dass die leichteste Lösung, welche zu einer anpassungsfähigen Zielsetzung führt, auch die Glaubhafteste ist, da über längere Zeit am einfachsten genetisch Anpassungen vorgenommen werden können. Nichtsdestotrotz kann jede Voreingenommenheit, auch wenn sie im Allgemeinen brauchbar erscheint, gelegentlich zu Problemen führen.

4.7 Replikation einer Simulation

Bis heute unterteilt man den Forschungsprozess von Simulationen im Bereich der Sozialwissenschaften in drei Phasen: die Programmierung, die Analyse und die Veröffentlichung der Computersimulationen. Doch es gibt noch einen weiteren Aspekt, welcher zwar oft vernachlässigt wird jedoch für die Gültigkeit der jeweiligen Simulation von hoher Bedeutung ist. Hierbei handelt es sich um die Replikation eines Simulationsmodells. In der heutigen Zeit werden zwar Unmengen an Simulationen erstellt, doch nur wenige beachten dabei den Gesichtspunkt der Replikation. Dieser wird allerdings dazu benötigt um die Ergebnisse als glaubwürdig einstufen zu können, und diese somit immer wieder reproduzieren zu können. Ohne Replikation können veröffentlichte Ergebnisse starke Abweichungen aufzeigen, welche durch Fehler in der Programmierung, eine falsche Darstellung der Simulation selbst oder fehlerhafte Angaben während der Analyse und

Aufbereitung der Daten entstehen können. Im Weiteren kann durch Einsatz von Replikation die Robustheit der Schlussfolgerungen überprüft werden.
Nachdem dieser Aspekt jedoch eher spärlich angewendet wird, ist es ratsam diesen Prozess gleich an einem Beispiel zu erklären. Hierbei handelt es sich um eine Aufgabe, welcher sich Michael Cohen gewidmet hat. Er stellte sich die Frage ob ein Simulationsmodell, welches für einen bestimmten Zweck entwickelt wurde, mit einem zweiten allgemeinen Simulationsmodell kombinierbar wäre, um einen weiteren Sachverhalt untersuchen zu können. Als Zielmodell der Replikation wurde Axelrods Modell der Kulturveränderung herangezogen. Als zweites allgemeines Simulationsmodell soll das Sugarscape-System von Joshua Epstein und Rob Axtell dienen. Nach diversen Änderungen an dem Sugarscape-Modell wurden schließlich die Ergebnisse des Kulturänderungs-Modells repliziert.

Replikation einer Simulation in der Praxis

Der Vorgang der Replikation an sich ist nicht so schwierig und kompliziert, wie es vielleicht im Vorfeld anzunehmen wäre. Unter guten Voraussetzungen, hier ein einfaches Zielmodell und ein ähnlicher Aufbau beider Simulationsmodelle, entsteht ein annehmbarer Zeitaufwand um die Replikation durchzuführen.
Es gibt drei Stufen der Replikation, welche stets mit einzuschließen sind. Sie lassen sich wie folgt einteilen:

- Numerische Gleichheit

  Den am häufigsten angewendeten Standard bildet die numerische Gleichheit, welche die Ergebnisse exakt reproduziert. Doch seitdem Simulationen immer öfter stochastische Elemente beinhalten kann eine Gleichheit nur unter Verwendung mit denselben Zufallsgeneratoren und Startwerten erreicht werden.
- Relative Gleichheit

  Für viele Anwendungszwecke ist jedoch die relative Gleichheit erforderlich, welche erreicht wird, wenn man statistisch keine Abweichungen nachweisen kann. Sollte man nun mittels Statistik nachweisen können, dass die Abweichungen der Mittelwerte und Standardabweichungen in jenem Bereich liegen, dass sie auch zufällig passieren hätten können, dann spricht man von einer relativen Gleichheit.

- Verhältnismäßige Gleichheit

  Der schwächste Standard ist die verhältnismäßige Gleichheit, welche eine innere Beziehung zwischen zwei Simulationsmodellen auf Grund ihrer Ergebnisse liefert. Seitdem Simulationen immer öfter qualitative anstatt quantitativer Ergebnisse liefern, wird die verhältnismäßige Gleichheit allerdings zu einem immer beliebteren Standard.

Um testen zu können ob verteilte Gleichheit existiert, muss man vorher noch klären wie man die Nullhypothese anwenden soll. Man könnte leicht annehmen, dass diese Aufgabe zu lösen sein, indem man die Nullhypothese der verteilten Gleichheit ablehnt, jedoch könnte dieser Ansatz diverse Forscher motivieren die Gleichheit mit sehr kleinen Testwerten zu überprüfen. Je kleiner allerdings die Testwerte sind, desto höher ist der Grenzwert damit die Nullhypothese verworfen werden kann und damit ist die Wahrscheinlichkeit größer, dass eine verteilte Gleichheit gefunden werden kann. Um dieses Problem zu umgehen muss man bereits von Beginn an definieren wie groß der Unterschied sein muss, um realistische Ergebnisse zu erzielen, und mit entsprechend großen Testgrößen die Überprüfung auf verteilte Gleichheit durchführen.
Michael Cohen, Rick Riolo und Robert Axelrod stellten sich der Herausforderung acht ausgewählte Basismodelle zu replizieren. Sie wählten diese Modelle anhand von sechs Merkmalen:

- Einfachheit in Bezug auf Implementierung, Erklärbarkeit und Verständnis
- Relevanz zum Fachbereich der Sozialwissenschaft
- Vielfältigkeit zwischen anderen Disziplinen und Arten von Modellen
- Kurze Laufzeiten
- Nachgewiesene heuristische Werte
- Verfügbarkeit über vorhandene Publikationen

Viele der acht gewählten Modelle weisen mindestens fünf dieser oben genannten Merkmale auf. Jeder der Forscher nahm noch eines seiner eigenen Modelle in diese Sammlung auf, damit auch wirklich keine Unklarheiten auftreten konnten. Die Basismodelle lauten nun wie folgt:

- Conway's game of life
- Cohen, March und Olson's garbage can model
- Schelling's residential tipping model
- Axelrod's evolution of prisoner's dilemma strategies using genetic algorithm
- March's organizational code model
- Alvin and Foley's decentralized market
- Kauffmann, Macready und Dickenson's NK patch model
- Riolo's Prisoner's dilemma tag model

Die Forscher Cohen, Riolo und Axelrod implementierten nun jedes einzelne dieser Modelle im SWARM Simulationssystem, wobei sie noch zusätzliche Unterstützung von Chris Langton bekamen. In jedem Fall wurden die Kernergebnisse der Originalsimulation analysiert und daraus ermittelt welche Vergleichswerte von Nöten wären um auf eine Gleichheit testen zu können. Dieser Vorgang benötigte mehr Zeit als am Anfang angenommen wurde, danach konnten jedoch alle Modelle verhältnismäßige Gleichheit erzielen. In manchen Fällen waren die Ergebnisse so erfolgreich, dass möglicherweise sogar relative Gleichheit aufgetreten ist, jedoch wurde dies nicht weiter hinterfragt.
Selbst diese erfahrenen Wissenschaftler mussten feststellen, dass während dem Prozess des Replizierens viele Fehler und Probleme entstehen können. Deren Auflistung soll es einem erleichtern eventuell diese Probleme zu umgehen oder zumindest eine Möglichkeit bieten die Fehlerquelle schnell zu orten.

Durch Replikation gewonnene Erkenntnisse

Die erste Fehlerquelle stellte die Mehrdeutigkeit in verfassten Beschreibungen der Simulationsmodelle dar. Diese treten sowohl in der Beschreibung des Models selbst als auch in der Wiedergabe der numerischen Ergebnisse auf. Mehrdeutigkeit in der Beschreibung umfasst zum Beispiel die Reihenfolge nach welcher Agenten aktualisiert werden oder wie sich das Modell bei einem Gleichstand von Agenten verhält. Um diese Mehrdeutigkeiten aufzulösen bietet sich die Möglichkeit an zwei plausible Methoden zu wählen und im

Nachhinein die Ergebnisse mit den Originaldaten zu vergleichen. Jedoch ist diese Vorgehensweise sehr riskant, da durch ein mehrfaches Vorkommen von Mehrdeutigkeiten eine Reihe von Kombinationsmöglichkeiten entsteht. Um dieses Problem zu lösen sollte man, wenn überhaupt verfügbar, auf den originalen Quellcode zurückgreifen.

Die zweite Problematik, mit welcher sich die Wissenschaftler während dem Replizieren auseinandersetzen mussten, war jene, dass in vielen Publikationen lückenhafte Beschreibungen vorhanden waren. In zwei Fällen der acht replizierten Modelle waren die Qualität der publizierten Daten sogar so schlecht, dass eine Überprüfung auf verteilte Gleichheit nicht möglich war. Der Effekt der unzureichend genauen Beschreibung kann allerdings auch auf eine ganz andere Art und Weise auftreten. So können beispielsweise Variablen in Verwendung kommen, welche die Werte +1, 0, und -1 annehmen können. Jedoch durch ihre ungenaue Beschreibung könnten sie falsch interpretiert werden, und man könnte annehmen die Variable könnte nur zwei Werte annehmen.

Die dritte Schwierigkeit war jene, wenn zwar die Publikation vollständig jedoch auch falsch war. Cohen, Riolo und Axelrod berichten von einem Fall, bei dem in der Beschreibung die Abbruchbedingung nicht mit der der eigentlichen Simulationsläufe, welche die Daten des publizierten Modells lieferten, übereinstimmte. Mit anderen Worten stimmte der Haupttext der Publikation nicht mit dem Anhang überein.

Die vierte und letzte Problematik während Replikationen waren Schwierigkeiten mit dem Quellcode selbst. In einem Fall war der Ausdruck des Quellcodes so alt, dass einige Zeichen derart unkenntlich waren und der Text somit nicht mehr lesbar war. Dieser Fall stellte sich im Nachhinein als äußerst interessant heraus, als die Wissenschaftler die alten Programmdaten mit denen ihres neuimplementierten Programms gegenüberstellten. Auch wenn beide Ausgaben eine entsprechende Genauigkeit aufweisen und die Ergebnisse übereinstimmen, bedeutet das aber nicht gleichzeitig, dass sie nicht unterschiedlich sind. Betragen beide Ergebnisse beispielshalber 5, könnte der erste Wert durch 3+2 und der zweite durch 100/20 erreicht werden.

5 Das SARS-Modell im Detail

5.1 Das Modell der kleinen Welt

Die Weltbevölkerung beträgt derzeit um die 6,6 Milliarden Menschen. Trotz dieser hohen Zahl, kann man immer wieder Leute sagen hören wie klein denn die Welt sei, und in gewisser Weise haben sie mit dieser Aussage sogar Recht. Denn obwohl die Anzahl der gesamten Weltbevölkerung immens hoch ist, stellt sich die Struktur des sozialen Netzwerkes als relativ eng miteinander verbunden heraus. Stanley Milgram war der Erste, welcher sich mit dieser Problematik auseinandersetzte und ein einfaches Experiment durchführte. Er verteilte Briefe in Nebraska mit der Bitte diese an einen Aktienbroker in Boston, ohne jedoch dessen Adresse explizit zu erwähnen, weiterzuleiten. Eine Einschränkung legte Milgram noch zusätzlich fest: die Briefe dürfen nur zwischen Personen ausgetauscht werden, welche sich gegenseitig mit dem Vornamen anreden und somit einen sehr engen Kontakt zu einander haben. Da nicht davon auszugehen ist, dass die Initialsender den Aktienmakler in Boston persönlich kennen, waren zwei logische Strategien zu erkennen:

- Als Adressanten des Briefes wurden Bekannte im Finanzsektor gewählt
- Der Brief wurde an Bekannte gesendet, welche in Boston selbst oder dessen Umkreis wohnten

Die Briefe, welche tatsächlich ihr vorgesehenes Ziel erreichten, wurden schließlich von Milgram analysiert. Er stellte fest, dass es im Durchschnitt nur sechs Stationen erforderte, um die Kette von Nebraska nach Boston zu bilden. Aus diesem Experiment heraus stellte er die Schlussfolgerung auf, dass zwei, sich unbekannte Personen, über sechs Bekannte in Verbindung stehen. Er bezeichnete diese Situation auch als "six degrees of seperation".
Durch die Art und Weise wie das Experiment von Milgram durchgeführt worden ist, kann schnell die Annahme entstehen, dass die Zahl Sechs nicht unbedingt genau zutreffend ist. Durch die unzureichende Steuerung und die vielen enthaltenen Störeinflüsse des Experiments waren die Ergebnisse nicht gerade der Realität entsprechend. Nichts desto trotz war man sich aber im Großen und Ganzen darüber einig, dass zwei unabhängige Personen

über eine kurze Verbindung mehrerer, sich kennender Personen doch indirekt in Verbindung stehen. Die Idee der kleinen Welt kann für sämtliche soziale Verbindungen implementiert werden. So hat zum Beispiel Brett Tjaden das Netzwerk "The Six Degrees Of Kevin Bacon" entwickelt. Hier wird eine Filmdatenbank dahingehend verwendet, um eine Verbindung von einem beliebigen Schauspieler zu Kevin Bacon herzustellen.

Erklärung anhand Zufallsgraphen

Am einfachsten lässt sich das Modell der kleinen Welt über einen Zufallsgraphen erklären. Man nehme an es leben N Menschen auf der Welt und jeder einzelne besitzt z Bekannte. Daraus ergibt sich ein Wert von $1/2 \cdot N \cdot z$ Verbindungen zwischen den einzelnen Personen auf der ganzen Welt. Nun kann man ein einfaches soziales Netzwerk erstellen, indem man N Knotenpunkte mit $1/2 \cdot N \cdot z$ Kanten verbindet. Trägt man diese Verbindungen nach belieben ein, so erhält man einen Zufallsgraphen. Aus diesem Beispiel kann man leicht den Effekt des "small world"-Modells erkennen: nimmt man einen frei gewählten Knoten A heraus, so besitzt dieser laut Definition z Nachbarn, z^2 zweite Nachbarn, z^3 dritte Nachbarn, …. Nimmt man an, dass eine reale Person zwischen 100 und 1000 Bekanntschaften hat, so würde sich bereits nach der 4. Nachbarschaftgeneration ein Wert zwischen 10^8 und 10^{12} Verbindungen einstellen. Durch die Definition des Abtrennungsgrades d, welcher auch Durchmesser des Graphen genannt wird, lässt sich ermitteln wie viele Stationen erforderlich sind um alle Knoten des Netzwerkes erreichen zu können. Durch Umstellen der Formel $z^d = N$ ergibt sich schließlich $d = \log(N)/\log(z)$.
Dieser logarithmische Anstieg mit steigender Größe des Netzwerkes ist typisch für dieses Modell. Da $\log(N)$ nur sehr langsam mit steigendem N zunimmt, erlaubt es den Abtrennungsgrad d auch in sehr großen Netzwerken gering zu halten.
Zufallsgraphen stellen allerdings die Realität nur sehr unzureichend dar, und lassen dabei die Tatsache außer Acht, dass sich Freundeskreise meist überlappen. So könnte zum Beispiel der Freund des Freundes bereits wieder ein direkter Freund zu sich selber sein. Somit kann man allerdings feststellen, dass die Aussage man hätte z^2 Nachbarn nicht korrekt sein kann, da einige der Nachbarn der 2. Generation bereits zu der ersten zu zählen sind. Diese Eigenschaft bezeichnet man auch als Clustering des Netzwerkes.

Das Modell von Watts und Strogatz

Zufallsgraphen weisen, wie im vorigen Kapitel beschrieben, einen durchschnittlichen Knotenabstand auf, welcher sich bei steigender Knotenanzahl N nur logarithmisch erhöht. Dieser Ansatz lässt jedoch nicht das Clustering einfließen, welches die Eigenschaft darstellt, dass zwei beliebige Nachbarknoten wiederum zueinander im direkten Nachbarverhältnis stehen können. Das genaue Gegenteil zu einem Zufallsgraphen ist ein Gitternetz. In der einfachsten Form befinden sich alle Knoten auf einer Linie; man spricht hierbei von einem eindimensionalen Gitter. Verbindet man nun in solch einer Konstellation jeden Knoten mit seinen z nächsten Nachbarn, kann man leicht erkennen, dass die meisten unmittelbaren Nachbarknoten jeden Punktes sogleich direkte Nachbarn zueinander sind. In weiterer Folge wird die Kette zu einem Ring geformt, welches jedoch nur der Vereinfachung der Berechnung dient und nicht zwingend notwendig wäre.

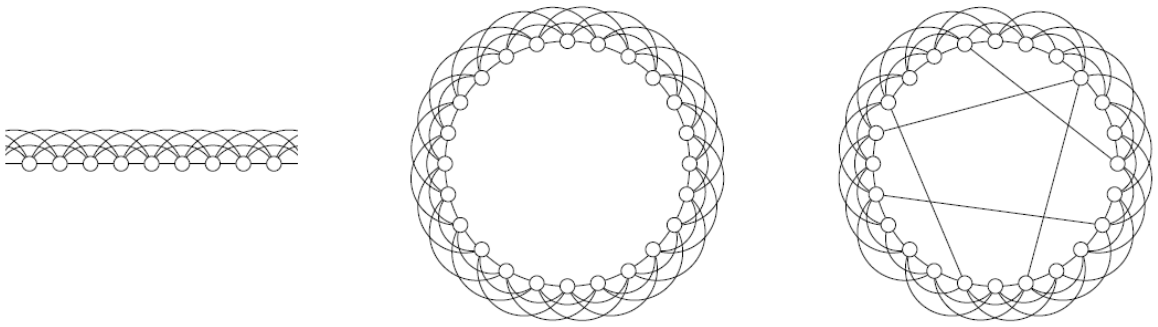

Abbildung 1

Für diesen Aufbau lässt sich der Clusteringkoeffizient C über die Formel $c = 3/4 \cdot (z-2)/(z-1)$ berechnen. Man kann leicht erkennen, dass C bei steigendem z gegen den Wert 3/4 tendiert. Jedoch können eindimensionale Gitternetze nicht den typischen Effekt der kleinen Welt, langsam anwachsende Knotenabstände mit steigender Systemgröße, abbilden. Betrachtet man dagegen Gitternetzwerke mit d Dimensionen, wie zum Beispiel Anordnungen in einem Rechteck, oder sogar in Form eines Hyperkubus mit L Seiten, so erhält man eine Knotenanzahl von $N = L^d$. Der mittlere Knotenabstand wächst mit steigendem L oder steigendem $N^{1/d}$. Setzt man nun für d kleine Werte ein, so kann man noch kein "small world"-Verhalten feststellen. Bei eindimensionalen Gitternetzen lässt sich daraus ableiten, dass der durchschnittliche Knotenabstand linear mit der Systemgröße

zusammenhängt. Nimmt man allerdings für d einen sehr hohen Wert an, so erkennt man, dass die Funktion $N^{1/d}$ nur sehr langsam mit einer steigenden Knotenanzahl N anwächst. In diesem Fall sind zwar die Eigenschaften des Modells der kleinen Welt gegeben, trotzdem drängt sich einem die Frage auf, ob man auf diese Weise ein reales Netzwerk erklären kann.

Watts und Strogatz [WATTS, 1999] entwickelten ein alternatives Modell um eine kleine Welt nachzubilden, welche unter Umständen besser unser tägliches Verhalten in einem natürlichen sozialen Netzwerk erklärt. Sie verfolgten den Ansatz eines wenig dimensionalen Gitternetzes, welches jedoch über einen gewissen Grad an Zufallsvariablen verfügt. Vergleichbar mit einem Zufallsgraphen, wollten sie dadurch den „small world"-Effekt abbilden.

Im Grunde genommen hat man ein eindimensionales Gitternetz, welches über eine ringförmige Anordnung an Knoten verfügt. Wie gewohnt wird jeder Knoten mit seinen z nächsten Nachbarn verbunden, jedoch werden nun einzelne Verbindungen mit der Wahrscheinlichkeit p einseitig gelöst und wieder mit einem beliebigen Knoten im System verbunden. Für ein klein gewähltes p, ergibt sich ein beinahe normales Gitternetz mit einigen Querverbindungen durch den Mittelpunkt des Ringes. Bei diesem Ansatz kann es nun vorkommen, dass die Anzahl der Verbindungen gewisser Knoten die Koordinationszahl z übersteigen oder umgekehrt kleiner als z ist.

Aus sozialer Sicht kann man diesen Aufbau so erklären, dass jeder seine unmittelbaren Nachbarn hat - Freunde, Arbeitskollegen oder Leute, welche einem im Laufe des Lebens vorgestellt werden. Manche haben aber auch ein bis zwei weitläufige Nachbarn, welche zum Beispiel Leute aus anderen Ländern, anderen sozialen Schichten oder anderen Lebensabschnitten sein können. Eben diese weitläufigen Bekanntschaften werden durch die neuen Querverbindungen im Modell von Watts und Strogatz repräsentiert.

Die Werte des Clusterkoeffizienten des Watts-Strogatz-Modells für kleine p-Werte verhalten sich ähnlich wie eine Kettenanordnung und tendieren gegen 3/4 für konstant kleine d- und große z-Werte. Watts und Strogatz konnten mit Hilfe einer numerischen Simulation nachweisen, dass die durchschnittlichen Knoten-Knoten-Abstände vergleichbar mit einem echten Zufallsgraphen sind.

Zum Beispiel beträgt der durchschnittliche Abstand eines Zufallsgraphen zwischen zwei zufällig gewählten Knoten $L = 3.2$ mit $N = 1000$ und $z = 10$. Das Modell von Watts und Strogatz weißt dabei einen nur wenig höheren durchschnittlichen Knotenabstand von

$L = 3.6$ bei einer eingestellten Querverbindungswahrscheinlichkeit von $p = 1/4$ auf. Im Vergleich dazu erhält man bei einem Gitternetz ohne jeglichen Querverbindungen einen durchschnittlichen Knotenabstand von $L = 50$ und mit $p = 1/64$ stellt sich schließlich der Wert $L = 7.4$ ein, welcher ungefähr das Doppelte des Zufallsgraphen darstellt. Demnach zeigt das Modell gleichzeitig sowohl die Eigenschaften des Clustering und als auch der kleinen Welt. Diese Ergebnisse wurden durch zahlreiche weitere Simulationen und viele analytischen Arbeiten über das „small world"-Modells bestätigt.

Weitere Modelle auf Basis der kleinen Welt

Eine Vielzahl von Autoren begannen schließlich diverse dynamische Systeme anhand der Vorgaben des Modells von Watts und Strogatz zu analysieren.

Watts und Strogatz selbst untersuchten zellulare Automaten, einfache spieltheoretische Modelle und Netzwerke von verbundenen Oszillatoren. Bei ihren Forschungen konnten sie unter anderen ermitteln, dass bei Verwendung eines „small world"-Graphen anstatt von gewöhnlichen Kettenanordnungen, zellulare Automaten die Aufgabe der so genannten „Klassifikation der Dichte" wesentlich einfacher lösen konnten, iterierende Mehrspielermodelle wie etwa das Gefangenendilemma weniger häufig Kooperationen bilden und Oszillatornetzwerke sich wesentlich einfacher synchronisieren lassen.

Monasson untersuchte das Eigenspektrum des Laplace'schen Operators bei dem Modell der kleinen Welt durch eine Matrixtransformation. Dieses Spektrum kann zum Beispiel die Maximalwerte eines Systems von Gewichten und Federn erklären, indem es auf das Prinzip der kleinen Welt zurückgreift. Des Weiteren kann es auch den Einfluss diffuser Dynamik auf einen „small world"-Graphen abbilden, da jeder Ausgangszustand eines diffusen Feldes in Eigenvektoren zerlegt werden kann, welche unabhängig voneinander exponentiell abfallen. Diese Absenkung ist wiederum konstant proportional zu dem entsprechenden Eigenwert. Eine diffuse Bewegung kann zum Beispiel ein einfaches Modell repräsentieren, welches die Verbreitung von Informationen über ein soziales Netzwerk darstellt.

Das Ising-Modell ist ein von Ernst Ising auf Anregung seines Doktorvaters Wilhelm Lenz um 1925 erstmalig genauer studiertes Modell der theoretischen Physik. Es beschreibt

insbesondere den Ferromagnetismus in Festkörpern (Kristallen) und zählt zu den meistuntersuchten Modellen der statistischen Physik.

Barrat und Weigt [BARRAT, 1999] konnten eine Lösung für das ferromagnetische Ising-Modell anhand eines „small world“-Netzwerks mit $d = 1$ finden, indem sie eine Nachbildungsmethode angewandt haben. Da das Ising-Modell eine kritische Dimension von zwei aufweist, kann man nicht davon ausgehen, dass ein Phasenübergang vollzogen wird, wenn p den Wert 0 besitzt und der Graph rein eindimensional ist. Andererseits steigt die effektive Anzahl der Dimensionen über 1 hinaus, sobald p einen größeren Wert als 0 einnimmt. Des Weiteren kann man davon ausgehen, dass für jedes endliche p bei jeder endlichen Temperatur ein Phasenübergang in dem immens großen System vollzogen wird, welches auch von Barrat und Weigt durch numerische und analytische Verfahren bewiesen werden konnte. Das Ising-Modell ist verständlicherweise stark vereinfacht und idealisiert, welches die Lösungen dadurch zu einem großen Teil nur angenähert wiedergeben kann.

Newman und Watts analysierten die Verbreitung von Krankheitserregern anhand eines „small world“-Graphen, wobei sie eine Krankheit wählten, welche nur für einen bestimmten Teil q der Gesamtbevölkerung infektiös wirkt. Der Erreger verbreitet sich von Nachbar zu Nachbar im Netzwerk der kleinen Welt, wobei dieser jedoch nur übertragen beziehungsweise wirksam wird, wenn die Nachbarn auch wirklich dafür empfänglich sind. Bei solch einem Modell kann sich der Erreger nur innerhalb des verbundenen Clusters von empfänglichen Nachbarn verbreiten. Diese Cluster können anfänglich klein sein, solange q kleine Werte annimmt, jedoch können diese auch rapid anwachsen oder sogar ins unendliche schnellen, wenn q steigt und somit größere Werte besitzt. Der Punkt an dem es ins Unendliche wechselt ist der Durchsickerungspunkt einer Gebietdurchsickerung mit einer Wahrscheinlichkeit q des „small world“-Graphen. Genau an diesem Punkt spricht man auch von einer Epidemie. Newman und Watts erarbeiteten eine annähernde Berechnung dieses Epidemiepunktes, welche sich auch verständnisvoller Weise mit ihren Ergebnissen der numerischen Simulationen deckt.

Kulkarni analysierte numerisch das Verhalten des Bak-Sheppen-Modells, und studierte so die Entwicklung verschiedener gemeinsam lebender Spezies mit Hilfe der kleinen Welt. Dieses Modell imitiert die evolutionären Effekte der Interaktionen zwischen einer Großen Anzahl an verschiedenen Spezies. Es ist bekannt, dass die Eigenschaft dieses Modells von dem Aufbau des Gitternetzes abhängt. Worauf hin Kulkarni angenommen hat, dass durch die Verwendung des „small world“-Netzwerks eine wesentlich bessere Abbildung des

ökologischen Systems möglich sei, als durch das gewöhnliche Netzwerk des Bak-Sheppen-Modells.
Durch die Simulation konnte veranschaulicht werden, dass die evolutionären Aktivitäten an jedem beliebigen Knoten des „small world“-Graphen stark von der Koordinationsnummer des jeweiligen Knoten abhängt. Knoten mit vielen verbundenen Nachbarn zeigten eine rege evolutionäre Aktivität, während Knoten mit wenigen Nachbarn eine geringe Aktivität aufwiesen.

5.2 Epidemien in der kleinen Welt

Es wurden bereits viele Arbeiten über Phänomene diverser epidemischer Situationen verfasst. Ein typischer mathematischer Ansatz besteht darin eine Durchmischung unterschiedlicher Bevölkerungen zu analysieren, wobei diese die Zustände empfänglich, infiziert und geheilt annehmen können und untereinander, in Abhängigkeit zu ihrer jeweiligen Größe, interagieren. Durch dieses nulldimensionale Modell war es möglich einige epidemische Eigenschaften wie die Existenz von Schwellwerten der Verbreitung von Krankheitserreger, die asymptotische Lösung der Dichte infizierter Personen und die Auswirkung stochastischer Schwankungen auf die Modellierung zu ermitteln.
Neben dem mathematischen Ansatz besteht auch noch die Möglichkeit sich erweiternde Bevölkerungen durch Gitternetze räumlich zu beschreiben. Dadurch kann die geographische Verbreitung einer Epidemie als Prozess zwischen Reaktion und Diffusion aufgefasst werden.
In der Realität lassen sich jedoch Bevölkerungen nicht wirklich kategorisieren, daher ist weder die Annahmen einer guten Durchmischung als auch die eines Gitternetzes nicht wirklich zutreffend. Das bereits erörterte „small world“-Modell von Watts und Strogatz versucht nun mit Hilfe eines Netzwerks den komplexen Aufbau von sozialen Interaktionen abzubilden. Diese Modelle spielen eine entscheidende Rolle in der Erforschung des Zusammenhangs von Netzwerkstrukturen und der Dynamik von vielen sozialen Prozessen wie zum Beispiel die Verbreitung von Krankheiten, die Entstehung von öffentlichen Meinungen, die Verteilung des Wohlstands oder die Übermittlung von kulturellen Traditionen. Im Vergleich mit anderen epidemischen Modellen kann man erkennen, dass die Ausbreitung der Epidemie bei einem „small world“-Netzwerk wesentlich schneller abläuft als

bei einem Reaktions-Diffusions-Modell oder einem diskreten Modell, welches durch ein gewöhnliches Gitternetz des sozialen Umfelds dargestellt wird.

Im originalen Modell der kleinen Welt wird der Aufbau durch den Parameter p beeinflusst, welcher alle Werte von 0 bis 1 einnehmen kann. Er repräsentiert die Anzahl der zusätzlich eingefügten Querverbindungen und kann somit von einem gewöhnlichen Gitternetz bis hin zu einem Zufallsgraphen verlaufen. Es wurde veranschaulicht, dass sowohl geometrische Eigenschaften als auch einige statistische Eigenschaften einen Übergang bei $p_c = 0$ und einer großen Anzahl von Knoten ($N \to \infty$) aufweisen. Daher führt jeder endlicher Wert für die Unordnung des Netzwerkes zu einem „small world"-Verhalten.

Ein einfaches epidemisches Modell

Das folgende Modell soll eine Verbreitung von Krankheitserregern abbilden und auf einfache Art und Weise erklären welchen Einfluss die Netzwerkstruktur auf die temporären dynamischen Effekte der Epidemie mit sich bringt. Man unterscheidet drei verschiedene Stadien, welche wie folgt unterschieden werden:

- empfänglich (S)
- infiziert (I)
- unempfänglich (R)

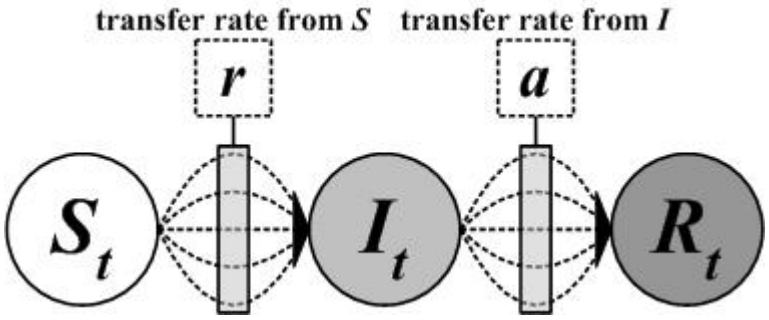

Abbildung 2

Zunächst wird jedem einzelnen Element der Bevölkerung eine der drei Eigenschaften zugewiesen. Empfängliche Elemente können den infizierten Status erreichen, indem sie von

bereits infizierten Elementen angesteckt werden. Infizierte Elemente erreichen den unempfänglichen Status nach einer bestimmten Zeit t_I, während unempfängliche Elemente nach der Zeit t_R wieder den Status „empfänglich“ erreichen. Diese Art von Modellierung wird auch SIRS genannt, da jedes Element durch die jeweiligen Stadien voranschreitet um schließlich wieder am Anfangsstadion anzukommen.

Die Infektion ist nur während der S-Phase möglich und kann auch nur von einem I-Element übertragen werden. Während der R-Phase sind die betroffenen Elemente immun und können nicht infiziert werden. SIRS-Modelle sind sehr aufregend, da man mit ihrer Hilfe Entspannungsphasen abbilden kann. In räumlich erweiterten Versionen kann es zu Raum-Zeit-Schwankungen kommen, welche durch Interaktion von Nachbarelementen hervorgerufen werden. Beide der Eigenschaften verhalten sich analog zu denen von Reaktion-Diffusion-Systemen wie zum Beispiel der Belousov-Zhabotinskii Reaktion.

Da SIR-Systeme eines eindimensionalen Gitternetzes mit lokalen und globalen Interaktionen bereits ausgiebig untersucht wurden, befassen wir uns hier weiter mit einem sensiblen sozialen Netzwerk, um die Ausbreitung der Infektion genauer zu untersuchen. Die Interaktion zwischen den einzelnen Elementen wird anhand eines „small world“-Netzwerks beschrieben, wobei Knoten die Personen darstellen und die Verbindungen die Kontakte zwischen den einzelnen Personen repräsentieren. Eine Übertragung des Krankheitserregers kann nur über solche Verbindungen vollführt werden.

Wie schon bereits vorher erwähnt, handelt es sich bei dem Modell von Watts und Strogatz um ein zufälliges Netzwerk, welches sich aus einem topologischen Ring mit einer Knoten Anzahl N entwickelt. Die Knoten werden jeweils ringförmig miteinander verbunden um anschließend zufällig gewählte Querverbindungen zu erhalten. Die Anzahl dieser ist abhängig von der Wahrscheinlichkeit p, wobei jedoch zu berücksichtigen ist, dass Eigenverbindungen sowie mehrfache Verbindungen nicht zulässig sind. Mit dieser Vorgehensweise erhält man bei $p=0$ ein gewöhnliches Gitternetz, bei $p>0$ erhält man Eigenschaften des „small world“-Effekts und bei $p=1$ sind alle Verbindung, welche überhaupt nur möglich sind, gesetzt und man kann fast von einem Zufallsgraphen sprechen. Es ist jedoch zu beachten, dass dieser Algorithmus mit Vorsicht anzuwendend ist, da eine falsche Anwendung auch zu nicht-verbundenen Graphen führen kann.

Jedes Element erhält seinen eigenen Zeitnehmer $\tau_i(t) = 0,1,2,...,\tau_I + \tau_R \equiv \tau_S$, welches die jeweilige Phase der Erkrankung beschreibt. Die Zeit selbst wird mittels diskreten Schritten durchlaufen und stetig erhöht.

Ein empfängliches Element behält solange die Zeit $\tau = 0$, bis es infiziert wird. Fand die Infektion einmal statt, schreitet die Zeit für die Dauer von τ_0 Zeitschritte voran. Während der ersten τ_I Zeitschritte gilt ein Element daher als infiziert und besitzt die Möglichkeit die Krankheit empfänglichen Nachbarn weiterzugeben. Während der letzten Zeit τ_R des Zyklus verweilt das Element in der R-Phase, und ist somit immun jedoch nicht ansteckend. Nach dieser Zeit gelangt es wieder in die S-Phase.

Die Ansteckung eines empfänglichen Elements durch ein bereits infiziertes und die nachträgliche Wiederaufnahme des Erkrankungszyklus erfolgt stochastisch auf einem lokalen Level. Betrachten wir beispielsweise ein Element i, welches empfänglich ist und k_i Nachbarn besitzt, von denen bereits k_{inf} infiziert wurden. Dann kann man davon ausgehen, dass die Wahrscheinlichkeit der Infektion des Elements i bei k_{inf} / k_i liegt. Dadurch ergibt sich auch die Tatsache, dass das Element i mit der Wahrscheinlichkeit von 1 infiziert wird, wenn bereits all seine Nachbarn erkrankt sind. Betrachtet man beispielsweise ein empfängliches Element, welches eine Ansteckungswahrscheinlichkeit von $q = 0.2$ zu all seinen Nachbar hat, stellt sich eine Infizierungswahrscheinlichkeit von $1-(1-q)^{k_{\text{inf}}}$ ein.

Gewonnene Erkenntnisse

Eine entscheidende Eigenschaft von „small world“-Graphen besteht darin sowohl kurze durchschnittliche Knotenabstände mit der Bildung von Cluster zu verbinden. Jedoch darf man die Unterschiede zwischen dem Modell der kleinen Welt und gewöhnlichen Gitternetzen sowie Zufallsgraphen nicht aus den Augen verlieren. Nachfolgend werden noch einmal kurz die entscheidenden Erkenntnisse aufgelistet:

Mit steigender Anzahl an Knotenpunkten verfügen diese Graphen auch über linear anwachsende durchschnittliche Knotenabstände, während bei einem „small world“-Graphen diese nur logarithmisch ansteigen.

Dies bewirkt gleichzeitig, dass die Weiterleitung von Informationen oder die Verbreitung eines Krankheitserregers anfangs Knoten erreichten, wobei die Anzahl dieser mit der Zeit anwächst. Später wechselt dieser Anstieg auf einen exponentiellen Verlauf, und wird bei einer Sättigung des Graphen später wieder gesenkt.

Krankheitsübertragungsmodelle, welche ein Maß von Empfänglichkeit zu Infektion mit einbeziehen, haben einen Durchsickerungsübergang ab dem man von einer Epidemie spricht. Diese Situation wird stark von der Eigenschaft des „small world“-Netzwerkes beeinflusst.

Dynamische Systeme wie Spiele oder zellulare Automaten zeigen ein unterschiedliches quantitatives Verhalten bei „small world“-Graphen oder gewöhnlichen Gitternetzen. Einige Problemstellungen wie beispielsweise die Ermittlung der Klassifikation der Dichte erscheinen mit Hilfe eines „small world“-Graphen leichter, andere wie die Lösung von Planungsproblemen wiederum schwerer.

Einige reale Graphen weisen andere Eigenschaften als jene von „small world“-Graphen auf, welche sie jedoch für ihre Funktion zwingend benötigen. Beispielsweise verfügt das World Wide Web über eine skalierungsfreie Verteilung an Koordinationsnummern von Knotenpunkten, jedoch ohne über die hohe Clusterdichte eines sozialen Netzwerkes zu verfügen.

5.3 Der SARS-Virus und seine Eigenschaften

Allgemeines

Das schwere akute Atemwegsyndrom, kurz SARS, ist die erste heftige, leicht übertragbare Erkrankung des 21. Jahrhunderts. Vieles an diesem Virus bleibt vorerst missverstanden und rätselhaft, doch zeigte die rasche Verbreitung über den internationalen Flugverkehr die potentielle Gefahr. Die größte Angst hat man daher davor, dass der Virus an einem stark frequentierten internationalen Flughafen oder einem dicht besiedelten Gebiet auftritt, da in einem solchen Fall die Isolation des Virus als enorm schwer gestalten würde und eine drohende Epidemie fast unvermeidbar wäre.

Die ersten bekannten Fälle von SARS traten im November 2002 in einer chinesischen Provinz auf. Die ersten offiziellen Berichte über eine atypische Lungenentzündung, welche auf 305 Personen übertragen wurde und 5 Tode forderte, erreichte erst im Februar 2003 die Weltgesundheitsorganisation (WHO). Die hauptgefährdeten Personen kamen aus der medizinischen Betreuung, da diese auch regelmäßigen Kontakt zu den Infizierten hatten. Eine Erhebung veranschaulichte, dass ein Drittel aller Infektionen dem Pflegepersonal zugerechnet werden konnte.

Somit kann man sich leicht erklären, dass ein behandelnder Arzt, welcher sich in der Zwischenzeit bereits selbst angesteckt hatte, den Virus nach Hong Kong brachte. Er besuchte den neunten Stock eines 4-Sterne Hotels, woraufhin alle Gäste und Besucher mit dem Virus infiziert wurden und diese den Erreger nach Hong Kong, Vietnam und Singapur weiter verschleppten. Zeitgleich konnte sich der Erreger über die internationalen Flugrouten weiter verbreiten, als Gäste des Hotels abreisten und diverse Ziele ansteuerten sowie behandelnde Ärzte aus Vietnam und Singapur internationale Reisen aus beruflichen oder anderen Gründen antraten.

SARS verdeutlichte drastisch das globale Chaos, welches durch eine neuauftretende infektiöse Krankheit entstehen kann. Als viele noch hofften man könnte den Virus isolieren, versuchten weltweit Gesundheitsbehörden, Ärzte, Pflegepersonal Wissenschaftler und Labormitarbeiter weiterhin den Virus bewältigen zu können. Während Ökonomen und Marktanalysten die auftretenden Kosten und Folgekosten abzuschätzen versuchten, und alleine für den Fernen Osten Einbußen von etwa 30 Milliarden Dollar kalkulierten. Natürlich verbreitete sich schnell Panik in der Öffentlichkeit, viele Regierungsbeamte mussten zurücktreten und ihren Posten freigeben und die soziale Stabilität war in den am stärksten betroffenen Regionen sehr gefährdet. Krankenhäuser sowie Schulen wurden geschlossen und viele Regierungen verhängten eine Sperre von schwerbetroffenen Regionen.

Der Erreger

Bislang wusste man, dass bakterielle Erreger wie Chlamydien, Mykoplasmen oder Legionellen zu einer atypische Lungenentzündung führen können. Da diese jedoch nicht auftraten, wusste man vorerst nicht wie man dieses Problem in den Griff bekommen könnte, da selbst die Verabreichung von Antibiotika nicht die gewünschten Erfolge mit sich

brachten. Daher wurde schnell angenommen, dass es sich hierbei um einen Virus handelt. Durch dauerhaftes Forschen konnte man schließlich ermitteln, dass der Erreger eine Abwandlung des Coronavirus ist. Wahrscheinlich ist ein bereits bekanntes Coronavirus mutiert oder eine bisher unbekannte Art des Virus, welcher bislang nur bei Tieren auftrat, konnte sich anpassen und ist nun auch für den Menschen gefährlich geworden.

Die Übertragung

Neueste Erkenntnisse zeigten, dass der Virus außerhalb des menschlichen Körpers 24 Stunden überleben kann. Die Theorie, dass der Virus an der Luft nicht beständig ist, musste daher fallengelassen werden. Jedoch sehen Wissenschaftler eine Übertragung über ein Klimaanlagensystem für sehr unwahrscheinlich an.
Durch eine Tröpfcheninfektion aus geringer Entfernung, welche zum Beispiel bei der Versorgung von hustenden und niesenden Infizierten zustande kommt, kann der Erreger weitergegeben werden. Bislang konnte eine Infektion auf indirektem Weg nicht eindeutig nachgewiesen werden, jedoch werden diese, wie auch die Übertragung über Körperausscheidungen, nicht ausgeschlossen.

5.4 Das zu modellierende SARS-Modell

Das von Chung-Yuan Huang, Chuen-Tsai Sun, Ji-Lung Hsieh und Holin Lin vorgeschlagene Modell basiert auf zwei Ebenen. Die obere Ebene stellt ein Multiagentensystem dar, welches die heterogenen Eigenschaften der wirklichen Welt widerspiegeln soll. Die untere Ebene besteht aus einem zweidimensionalen zellförmigen Automaten, der die Aktivitäten der realen Welt abbildet. Diese beiden Schichten werden durch das sogenannte „mirror identity concept“ miteinander verbunden, wodurch schließlich ein „small world“-Netzwerk entsteht, mit dessen Hilfe man die Übertragung des SARS-Virus sowie dessen Auswirkungen analysieren kann.
Das abstrakte „mirror identity“-Konzept bildet menschliche Interaktionen und tägliche Handlungen der modernen Gesellschaft ab. Es definiert die gesellschaftlichen Eigenschaften

eines Individuums und legt dabei unter anderen den Bewegungsfreiraum oder tägliche Besuche von bestimmten Orten fest. In diesem Modell werden die einzelnen simulierten Akteure als je eine Entität eines Agenten angesehen, welche dem computerbasierten Multiagentensystem der oberen Schicht angehören. Jene Orte, welche der Agent in regelmäßigen Abständen aufsucht, werden durch die „mirror identity“ bestimmt.

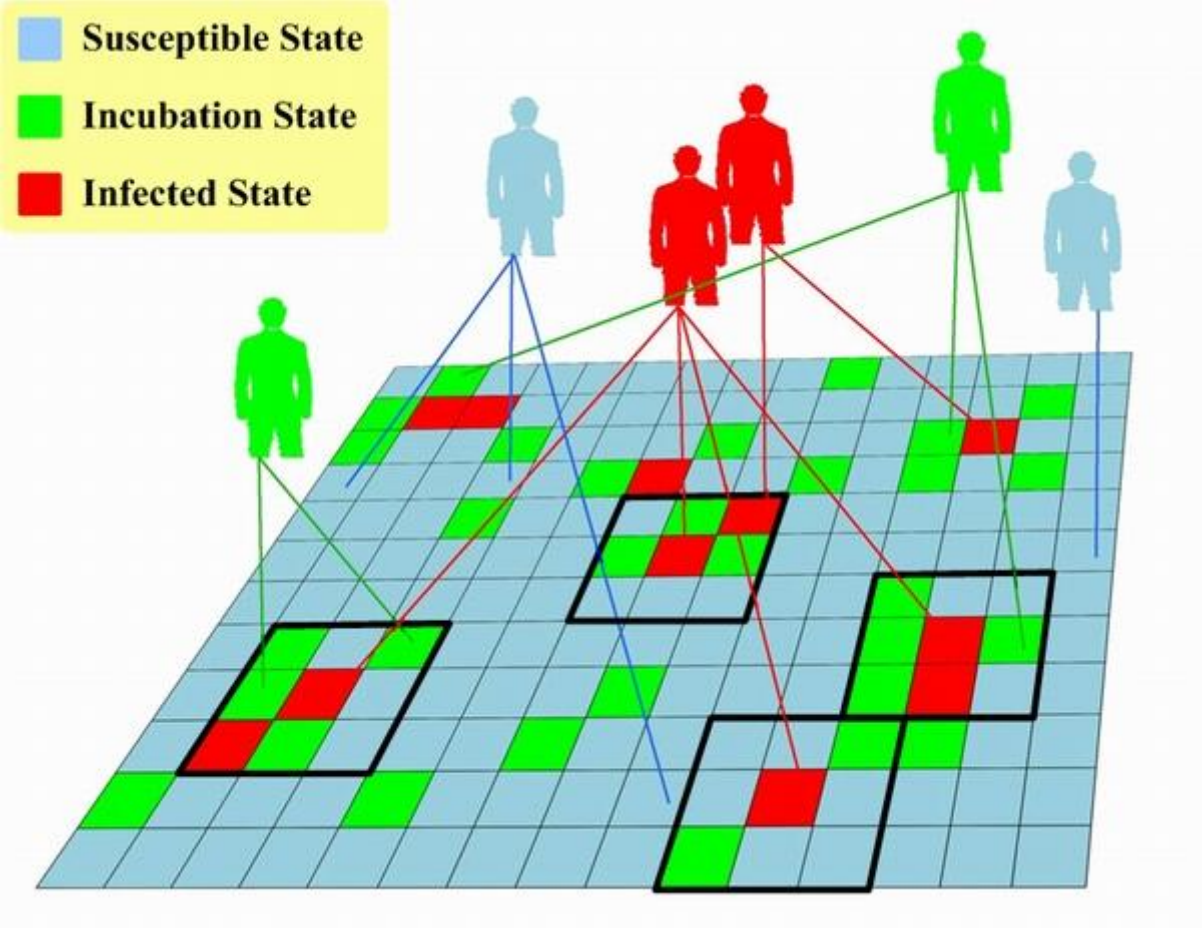

Abbildung 3

Die Autoren des Modells beschreiben eine fiktive Person namens Andy, ein bereits pensionierter Bürger, und geben somit ein Bespiel eines Agenten. Jeden Tag um 8:00 in der Früh fährt Andy mit einem Moped zu einem nahegelegenen Pflegeheim und bietet dort seine Dienste auf freiwilliger Basis an. Er unterstützt eine Krankenschwester namens Cindy um drei dort drei Bewohner, Bob, Frank und Eric, zu versorgen. Wie gewohnt geht er täglich um 18:00 abends zu seinem japanischen Stammlokal, wobei er stets mit dem Eigentümer, dem Küchenchef und weiteren Stammgästen ins Gespräch kommt. Nach dem Abendessen begibt es sich wieder nach Hause, wechselt dort seine Kleidung und besucht eine

nahegelegene Kneipe um seine Freunde Michael und Gerry zu treffen. Dies ist der übliche Tagesablauf von Andy, welcher nur selten abgeändert wird.

Überträgt man nun diese Vorlage auf das Modell, so stellen Andy und alle Leute mit denen er Kontakt hat wie Cindy, Bob, Eric, Frank, Michael und Gerry einzelne Agenten des übergeordneten Multiagentensystems dar. Sämtliche Lokalitäten wie Andys Zuhause, das Pflegeheim, das japanische Restaurant und die Kneipe werden durch die „mirror identities“ der unteren Schicht repräsentiert. Es ist zu beachten, dass sein Moped als Erweiterung seines Zuhauses aufgefasst wird, da Andy ohnehin immer alleine fährt.

Jeder Agent besitzt seinen eigenen Satz an Attributen, welcher Informationen über den epidemischen Fortschritt und der sozialen Bewegungsfreiheit liefert. Jede „mirror identity“, welche jederzeit die Attribute der Agenten beeinflussen können, haben zusätzlich noch private Eigenschaften. Diese Repräsentieren den aktuellen Status sowie zusätzliche Angaben über die Lokalität. Jeder Agent hat jederzeit Zugriff auf die Eigenschaften der „mirror identity“, mit der sie in Verbindung stehen. Darüber hinaus besitzen die Agenten die Fähigkeit mit Hilfe ihrer „mirror identities“ Gruppierungen mit anderen Agenten zu bilden.

Im obigen Beispiel würde Andy drei Gruppen angehören: erstens der im Pflegeheim, zweitens der im japanischen Restaurant und drittens der in der Kneipe. Alle diese „mirror identities“ werden durch Andy miteinander verbunden und bilden eine sternförmige Anordnung, wobei der Agent den Mittelpunkt repräsentiert und die „mirror identities“ die Endpunkte darstellen.

Attribut	Typ	Beschreibung	Wert
ID	Integer	Eindeutige Erkennungsnummer, identifiziert jeden einzelnen virtuellen Agenten	1…P
E	Symbol	Status des epidemischen Fortschritts; zu Beginn gibt es zwei unterschiedliche Typen: jene, welche immun auf den Erreger sind, und den Rest, der anfangs als „Susceptible“ eingestuft wird. Dies bedeutet, dass der Agent noch nicht infiziert, jedoch für den Erreger empfänglich ist	Susceptible, Incubation, Infected, Recovered, Immune, Dead

Mobility	Symbol	Der Initialwert ist stets auf „Free“ gesetzt, wodurch jedem Agenten freie Mobilität geboten wird. Hat sich der Agent einmal infiziert wird sein Status auf „Quarantined“ oder „Isolated“ gesetzt, je nach dem welchen „mirror identities“ er unterliegt (z.B.: Krankenhaus oder Zuhause). Während dieser Zeit ist der Agent fix stationiert und es kommt zu keiner Interaktion mit seinen restlichen „mirror identities“.	Free, Quarantined, Isolated
Count	Integer	Dies legt die Anzahl der „mirror identities“ eines Agenten fest, wobei jeder Agent mindestens eine und maximal M „mirror idetities“ besitzt. Diese Werte sind normalverteilt.	1…M
MirrorIdentity	Set	Stellt eine Datenstruktur dar, welche aus mindestens einer „mirror identity“ besteht.	
Age	Symbol	Das Modell teilt alle Agenten in drei Altersgruppen: jung (1-20 Jahre), mittel(21-60 Jahre) und alt (61 und älter).	Young, Prime, Old
Super	Boolean	Bestimmt ob es sich bei dem Agenten um einen sogenannten Superspreader handelt. Mittels Superspreader werden Personen nachgebildet, welche besonders viele Leute infizieren.	true, false
ImmunityPermanent	Boolean	Bestimmt ob ein Agent permanent immun ist.	true, false
Day	Integer	Die Anzahl der Tage wird in drei epidemischen Phasen verwendet: bestimmt erstens die Anzahl der Tage die ein Infizierter benötigt um sich wieder zu erholen, zweitens die ein erholter Agent in Anspruch nimmt um sich vollständig zu regenerieren und drittens die einem Agententemporär als immun zustehen.	1…X
RateContact	Real	Die Rate mit welcher sich ein bestimmter Agent mit anderen Agenten trifft. RateContact ist normalverteilt.	0…1

WearingMask	Boolean	Zu Beginn gibt es für niemanden eine Tragepflicht eines Atemschutzes. Erst während der Epidemie kann entschieden werden ob generell die gesamte Bevölkerung oder nur das Pflegepersonal Masken tragen muss.	true, false
MaskType	Real	Gibt den durchschnittlichen Wirkungsgrad der Schutzmasken an. Je höher die Zahl, desto größer die Effizienz.	0…1
QuarantinedDay	Integer	Anzahl der Tage, während Hausquarantäne vorgeschrieben wird.	0…P

Tabelle 1

Attribut	Typ	Beschreibung	Wert
Root	Boolean	Bestimmt den Ort welcher von dem Agenten während einer Epidemie am häufigsten genutzt wird. (Zuhause, Krankenhaus, Quarantäne,…) Jeder Agent besitzt genau eine „mirror identity“ mit dem Wert „true“. Alle anderen besitzen den Wert „false“.	true, false
Suspend	Boolean	Zu Beginn der Simulation auf „false“ gesetzt, sind alle „mirror identities“ für den Agenten zugänglich. Im Laufe der Simulation kann das Attribut den Wert „true“ einnehmen (ausgenommen Root=true). Es sollen dadurch Bewegungseinschränkungen für den Agenten entstehen. Bei einem infizierten Agenten wird das Attribut „Suspend“ aller „mirror identities“ (ausgenommen Root=true) auf „true“ gesetzt, welches dazu führt, dass er nur mehr an einem Ort verweilt. Ist der Agent verstorben, so erhalten all seine „mirror identities“ den Wert „true“.	true, false

Location	(Integer, Integer)	Die „mirror identities“ werden auf einem 2-dimensionalen Raster räumlich voneinander getrennt und auch unterschieden. Dabei gibt der erste Wert den Abstand auf der x-Achse und der zweite Wert den Abstand auf der y-Achse an. Jede „mirror identity“ ist genau mit einem eindeutigen Koordinatenpaar verbunden.	
Neighbor	Set	Repräsentiert die Koordinatenpaare der acht benachbarten „mirror identities“ eines Agenten. Hier wird die Methode von Moore verwendet, welche definiert, dass jede „mirror identity“ acht benachbarte Agenten besitzt.	

Tabelle 2

Die meisten Agenten besitzen zwischen 2 und 5 „mirror identities“, wobei diese Anzahl eine Normalverteilung darstellt. Je mehr „mirror identities“ einem Agenten zugeteilt werden, desto mehr Aktivitätsknoten besitzt dieser und desto höher ist die Wahrscheinlichkeit, dass sich der Agent infiziert bzw. den bereits empfangenen Virus an einen anderen Agenten weitergibt. Raster, welche sich in dem zellförmigen Automaten umgeben, stellen Nachbarn dar. In dem oben erwähnten Beispiel würden Andy, Cindy, Bob, Dick und Eric angrenzende „mirror identities“ besitzen; die Kneipe stellt eine weitere „mirror identity“ dar, welche Andy, Frank und Gerry zugeordnet ist und aneinander angrenzende Raster sind.

In dem von Chung-Yuan Huang, Chuen-Tsai Sun, Ji-Lung Hsieh und Holin Lin erstellten Modell entspricht ein einzelner diskreter Zeitschritt der Simulation einem gesamten Tag der abzubildenden echten Welt. Sowohl die Zustände der Agenten sowie deren zugewiesenen „mirror identities“ ändern sich gleichzeitig von Schritt zu Schritt, wobei jede „mirror identity“ eines einzelnen Agenten mit angrenzenden „mirror identities“ in Kontakt steht. Die Eigenschaften der Agenten, deren „mirror identities“ mit denen anderer Agenten angrenzen, beeinflussen sich gegenseitig durch den definierten Regelsatz. Dadurch ändern sich die Simulation und dessen festgelegte epidemischen Parameter, die Parameter der öffentlichen Gesundheitsmaßnahmen und diverse Zufallsvariablen. Durch die Kombination des zellförmigen Automaten und den „mirror identities“ ist es möglich vielfache Netzwerkeigenschaften abzubilden: täglich besuchte Orte, unterschiedliche Reichweiten im Sinne der Bewegungsfreiheit oder lokale Gruppenbildungen.

Attribut	Typ	Beschreibung
PopulationAgent	Set	Speichert die Agentenpopulation des Simulationssystems. Die maximale Kapazität beträgt dabei P Agenten.
P	Integer	Anzahl der Agenten.
M	Integer	Maximale Anzahl der „mirror identities“ eines Agenten.
H	Integer	Höhe des 2-dimensionalen Rasters.
W	Integer	Breite des 2-dimensionalen Rasters.
N	Integer	Anzahl der verfügbaren Raster des zellförmigen Automaten.
DayIncubation	Integer	Durchschnittliche Anzahl an Tagen der Inkubationszeit.
DayInfectious	Integer	Durchschnittliche Anzahl an Tagen der Infektionszeit.
DayRecovered	Integer	Durchschnittliche Anzahl an Tagen der Regenerationszeit.
DayImmune	Integer	Durchschnittliche Anzahl an Tagen der Immunitätszeit.
RateSuper	Real	Prozentuelles Vorkommen von Superspreadern in Relation zur Gesamtpopulation.
RateYoung	Real	Prozentuelles Vorkommen von jungen Agenten (0 bis 20 Jahre) in Relation zur Gesamtpopulation.
RatePrime	Real	Prozentuelles Vorkommen von Agenten mittleren Alters (21 bis 60 Jahre) in Relation zur Gesamtpopulation.
RateOld	Real	Prozentuelles Vorkommen von alten Agenten (60 Jahre und älter) in Relation zur Gesamtpopulation.
RateForeverImmunity	Real	Prozentuelles Vorkommen von konstant immunen Agenten in Relation zur Gesamtpopulation.
RateInfection	Real	Durchschnittliche Infektionsrate.
RateDeath	Real	Durchschnittliche Sterberate.

Tabelle 3

Strategie	Attribut	Typ	Beschreibung	Wert
WearingMaskInGP	RateParticipation	Real	Gibt die Rate an, wie viele Agenten sich an die Vorschreibung halten	0~1
	RatePrevention	Real	Rate der Effizienz der Strategie	0~1
WearingMaskInHW	RateParticipation	Real	Gibt die Rate an, wie viele	0~1

			Agenten sich an die Vorschreibung halten	
	RatePrevention	Real	Rate der Effizienz der Strategie	0~1
TemperatureMeasuring	RateDetection	Real	Rate der vorzeitigen Fiebererkennung	0~1
	RateParticipation	Real	Gibt die Rate an, wie viele Agenten sich an die Vorschreibung halten	0~1
HomeQuarantine	Class	Symbol	Unterscheidung zwischen A- und B-Quarantäne	A, B
	DayQuarantined	Integer	Anzahl der Tage, welche in Heimquarantäne verbracht werden müssen	0~1
	RateParticipation	Real	Gibt die Rate an, wie viele Agenten sich an die Vorschreibung halten	0~1
RestrictingAccessToHospitals	RateParticipation	Real	Gibt die Rate an, wie viele Agenten sich an die Vorschreibung halten.	0~1
ReducingPublicContact	RateParticipation	Real	Gibt die Rate an, wie viele Agenten sich an die Vorschreibung halten	0~1

Tabelle 4

Basierend auf der eingestellten Kontaktrate Agent.Parameter.RateContact und einer zufällig gewählten Zahl c ermittelt die „mirror identity“ jedes Agenten ob sie mit den acht angrenzenden „mirror identities“ interagiert. Falls c niedriger ist als die Kontaktrate findet eine Interaktion der „mirror identity“ des Agenten mit jener des Nachbaragenten statt. Die Größe des Parameters Agent.Parameter.RateContact kann variieren und ist unter anderem davon Abhängig ob eine allgemeine Verordnung verhängt wurde, welche den Kontakt der Agenten und somit die Infektionsgefahr reduzieren soll.

Angenommen es grenzt eine „mirror identity“ des Agenten A an eine des Agenten B und Agent A wurde bereits infiziert. Somit ist dieser ansteckend, während Agent B für Infektionen empfänglich ist. Treffen nun diese beiden Agenten aufeinander so hängt die

Weitergabe des Infekts sowohl von dem eingestellten Parameter System.Parameter.RateInfection als auch von einer zufällig bestimmten Zahl n, welche bestimmt ob Agent B von Agent A infiziert wurde, ab. Ist n kleiner als die Infektionsrate bekommt Agent B den Status N (Inkubation) und die Zeitvariable Agent.AttributeDay wird auf 1 gesetzt. In diesem Stadium können noch keine Symptome erkannt werden und der neuinfizierte Agent kann den Virus nicht an weitere Agenten übertragen. Die Infektionsrate System.Parameter.RateInfection wird durch Faktoren wie z.B. die Immunitätsrate, dem Superspreader-Status oder dem momentanen Aufenthaltsort bestimmt.

Der epidemische Zustand eines Agenten wechselt von N zu I (Infected), sobald die Inkubationsdauer System.Parameter.DayIncubation überschritten wurde. Nach Ablauf der Infektionsdauer I, welche durch den Parameter System.Parameter.DayInfectious bestimmt wird, entscheidet die Kombination aus eingestellter Sterberate (System.Parameter.RateDeath) und einer zufällig gewählten Zahl d, ob der Agent den Zustand D (Death) oder R (Recovered) erreicht. Die Sterberate ist wiederum abhängig von Faktoren wie dem Alter, ob sich der Agent während der Inkubations- bzw. der Infektionszeit in Quarantäne befand oder ob er richtige Behandlung erhielt.

Erreicht der Agent den Zustand R (Recovered), so wir er automatisch nach einer bestimmten Zeitspanne System.Parameter.DayRecovered in den Zustand M (Immune) übergeleitet. In dieser Phase wird der Parameter Agent.AttributeForeverImmune angewendet, um zu ermitteln ob der Agent permanent oder nur temporär im Zustand M verweilt.

Familien und Krankenhäuser

Das von Chung-Yuan Huang, Chuen-Tsai Sun, Ji-Lung Hsieh und Holin Lin gewählte Modell kann auch so verwendet werden, um Verbindungen wie beispielsweise Eigenheime, Schlafsäle oder Krankenhäuser abzubilden. Die Tabelle 2 veranschaulicht, dass alle „mirror identities“ zwei Attribute aufweisen: Root und Suspend. Nur eine einzige „mirror identity“ eines Agenten darf bei dem Attribut Root den Wert true erhalten, den anderen „mirror identities“ wird false zugewiesen. Anders verhält es sich bei dem Attribut Suspend, wo zu Beginn der Simulation alle „mirror identities“ den Wert false erhalten.

Verordnet die Gesundheitsbehörde eine Heimquarantäne für den Agenten A, so wird das Attribut Suspend aller „mirror identities“ des Agenten, mit Ausnahme der Stamm-„mirror

identity“, auf true gesetzt. Somit bleibt dem Agenten nur mehr eine einzige Lokation übrig in der er sich aufhalten kann.
Die angrenzenden Rasterelemente der „mirror identity“ des Agenten repräsentieren demnach jene „mirror identities“ anderer Agenten. (Zuhause – Familie, Arbeitsplatz – Kollegen, Freizeit – Freunde). Wird die Quarantäne wieder aufgehoben, so wird das Suspend-Attribut aller „mirror identites“ des Agenten wieder auf false zurückgesetzt, wodurch dieser wieder vollkommene Bewegungsfreiheit erhält.

Modellierung von Strategien

Die Vorschreibung eines Atemschutzes für die normale Bevölkerung umfasst zwei Parameter: Beteiligungsrate und Vorbeugungseffizienz. Ersteres stellt den Anteil der Personen dar, welche tatsächlich einen Atemschutz verwenden, während die Vorbeugungseffizienz den Wirkungsgrad der Masken beschreibt.
Befindet sich Agent A im epidemischen Zustand S und trägt jedoch einen Atemschutz, so verringert sich die Infektionswahrscheinlichkeit in Abhängigkeit des Parameters für die Vorbeugungseffizienz. Betrachtet man den Fall, dass sich Agent A im epidemischen Zustand I befindet und bereits vor und nach auftreten der Symptome eine Gesichtsmaske trägt, so reduziert sich folglich auch die Wahrscheinlichkeit andere Agenten zu infizieren.
Wenn die Messung der Körpertemperatur vorgeschrieben ist, wird über die „mirror identities“ aller Agenten bestimmt ob eine Temperaturmessung der angrenzenden Agenten durchzuführen ist bevor es zu einem Kontakt kommt. Diese Entscheidung wird über die Kombination aus der Beteiligungsrate und einer zufällig gewählten Zahl n bestimmt. Falls n unterhalb der Beteiligungsrate ist, sind alle benachbarten Agenten verpflichtet sich an die Fiebermessungsverordnung zu halten und müssen bevor ein Kontakt zustande kommt die Temperatur messen. Das Ergebnis der jeweiligen Messung wird noch zusätzlich durch den Parameter der Erkennungsrate bestimmt. Dies führt dazu, dass eine hohe Erkennungsrate und eine hohe Anzahl an Temperaturmessungen zu einer Eindämmung der Verbreitung des Erregers führen.
Prinzipiell kann man sagen, dass die Reduzierung des öffentlichen Kontakts zu einer der effizientesten Maßnahmen zählen, um die Verbreitung des Virus kontrollieren zu können. Sollte diese Verordnung aktiv werden, bestimmt die Kombination aus der Beteiligungsrate

und einer zufälligen Zahl n ob die „mirror identities“ von zwei Agenten interagieren bevor diese physisch in Kontakt treten. Liegt n über der Beteiligungsrate so repräsentiert das den Sachverhalt, dass mindestens einer der Agent sich dagegen entschieden hat mit den Agenten, welche einer bestimmten „mirror identity“ zugehören, in Kontakt treten zu wollen oder einfach keinen Grund hatten mit dieser „mirror identity“ zu interagieren.

6 Replikation und Gegenüberstellung der Simulationsergebnisse

6.1 Verständnis und kritische Betrachtung des zu replizierenden Modells

Bevor die erneute Umsetzung des SARS-Modells nun wirklich stattfinden kann, muss natürlich das gesamte Modell zunächst durchdacht und verstanden werden. Hierbei war es vorerst ein wenig problematisch, da anfangs nur eine Zusammenfassung des Modells vorlag. Man konnte sich zwar schon ungefähr die Zusammenhänge und Ansätze der Wissenschaftler erklären, jedoch fehlten viele Informationen um die Replikation auch nur halbwegs brauchbar durchführen zu können. Nach längerem Suchen wurde schließlich eine ausführlichere Version der Abarbeitung gefunden und diese auch für das Replizieren des SARS-Modells herangezogen.

Die enthaltenen Tabellen veranschaulichten die Parameter des Modells, der Agenten und deren „mirror identities“, welche in späterer Folge auch den groben Aufbau der Java-Klassen festlegten. Die Business-Logik der Simulation wurde anhand von Prosatext und kurzen Pseudo-Codes veranschaulicht. Nach kurzer Eingewöhnung konnte sich relativ leicht ein Bild gemacht werde wie die Klassen mit einander arbeiten sollten.

Abgesehen davon, dass es leider nicht möglich war den gesamten Quellcode der Originalsimulation zu bekommen und die vorhandenen Unterlagen nicht das gesamte Potential des Modells abdeckten, fällt auch eine nicht gerade realistische Abbildung der realen Gesellschaft auf. Durch die fest definierten Bewegungsmöglichkeiten der Agenten ändern sich die Nachbarn der jeweiligen Positionen nie, sondern es kommt lediglich durch die An- bzw. Abwesenheit der benachbarten Agenten zu unterschiedlichen Mustern. Eine Abbildung von Massentransportmitteln wie etwa Busse, Züge oder Flugzeuge sind somit nicht wirklich vorhanden, obwohl diese bestimmt das größte Potential mit sich bringen, um einen eingeschleppten Erreger an viele dort befindliche Menschen zu übertragen.

Wissenschaftler des Max-Planck-Instituts haben sogar die Ausbreitung der Epidemie über die Flugverkehrsströme simuliert und sind zu einer sehr nahen Abbildung des tatsächlichen Verlaufs der SARS-Bedrohung gekommen. Somit ist dieser Aspekt sicher nicht zu vernachlässigen.

6.2 Wahl des Simulationstoolkits und gewonnene Erfahrungen

Da ausreichend Java-Kenntnisse vorhanden waren, fiel vorerst die Wahl auf Repast Symphony um das SARS-Modell zu implementieren. Die zumindest grobe Erstellung der Programmteile mittels einer Diagrammerstellung und einer „Drag&Drop“-Steuerung versprach weitere Vorteile mit sich zu bringen. Jedoch wurde die anfängliche Freude, dieses System für die Implementierung heranzuziehen, sehr schnell gebremst, da bereits beim „nachprogrammieren“ des ersten Beispielprogramms nichtnachvollziehbare Fehler auftraten. Nach einigen Neuinstallationen und Durchläufen des Tutorials blieb lediglich die Schlussfolgerung, dass es sich hierbei um eine noch nicht wirklich ausgereifte Version handeln müsse. Daher fiel die Wahl auf die Vorversion, welche zwar nicht den gleichen Komfort mit sich bringt, jedoch bereits beim ersten Versuch ein einfaches Modell zu implementieren die Mächtigkeit des Toolkits zeigte.
Es bedarf sicher einer längeren Eingewöhnungszeit um mit diesem Framework arbeiten zu können, doch hat man einmal das Prinzip verstanden sollte der Aufwand nicht mehr so enorm sein.

6.3 Erstellung der benötigten Klassen

Die Klasse SarsModel

Prinzipiell Bedarf es für die Implementierung des SARS-Modells nur drei Klassen. Die Hauptklasse SarsModel erbt die Funktionalität von der Elternklasse SimpleModel, welche von dem Repast-Toolkit angeboten wird. Dabei werden die Funktionen setup, buildModel, preStep und postStep überschrieben, damit der Ablauf des zu implementierenden Modells übernommen werden kann.
Die Methode setup wird nur einmalig bei jedem Öffnen der Simulation durchlaufen und ist dadurch für die Festlegung von Systemparametern verantwortlich. In diesem Fall wird eigentlich nur der Konstruktor der Elternklasse SimpleModel aufgerufen und ein Fenster zur Darstellung der Rasteransicht initialisiert.

Die Methode buildModel wird stets dann ausgeführt, wenn über die GUI von Repast ein neuer Simulationsdurchlauf gestartet wird. Dabei werden die Benutzeroberfläche frei wählbare Modellparameter übernommen und das Modell dementsprechend definiert und eingerichtet.

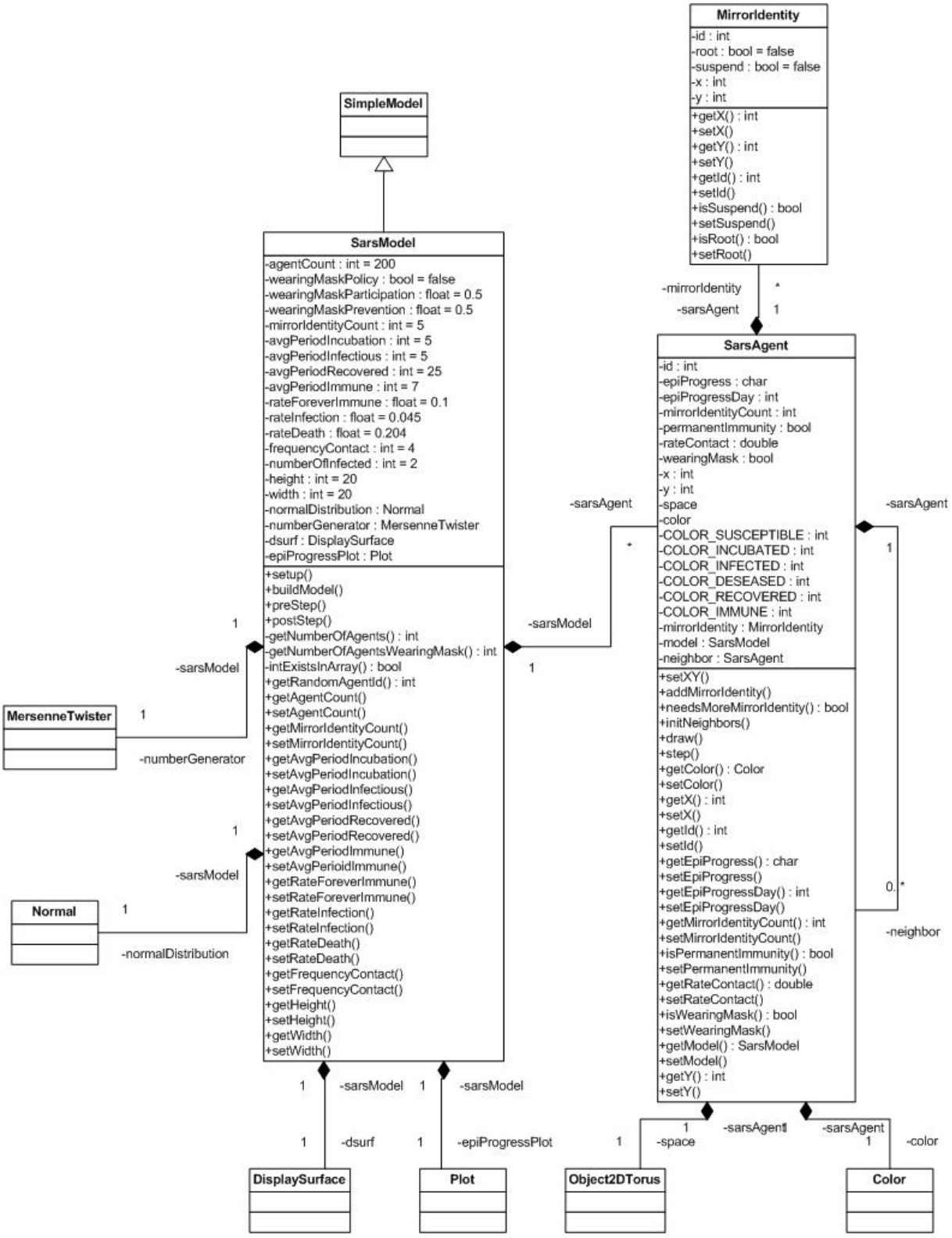

Abbildung 4

Neben den, für die Simulation notwendigen Modellparametern, enthält die Klasse SimpleModel auch alle benötigten Agenten-Objekte und initialisiert diese.
Die überschriebenen Methoden preStep und postStep werden entweder vor oder nach dem jeweiligen Simulationsschritt aufgerufen. Dadurch hat man immer Einfluss auf die Systemparameter und kann diese je nach Bedarf anpassen. In diesem Fall wird die Routine postStep auch dazu verwendet, um die Anzeige der Agenten bzw. den Plot über den aktuellen epidemischen Zustands des Systems zu informieren und die Ansichten zu aktualisieren.

Die Klasse SarsAgent

Die Klasse SarsAgent implementiert die vom Framework zur Verfügung gestellten Klassen Drawable und Stepable. Durch die Verwendung der Klasse Stepable und der zu definierenden Methode step ist jedes Agentenobjekt unabhängig von den anderen Agenten. Die Routine step eines jeden Agenten wird pro Simulationsschritt genau einmal durchlaufen.
Da jeder Agent nur bestimmte Positionen auf dem Raster einnehmen kann, müssen diese vorerst festgelegt werden und in einer einfachen Struktur abgelegt werden. Genau dafür wird die Klasse MirrorIdentity herangezogen.

Die Klasse MirrorIdentity

Dabei handelt es sich um eine einfache Datenstruktur, welche Positionen eines Agenten festhält, welche er während der Simulation auf dem Raster besuchen darf. Zusätzlich zu den Koordinatenangaben werden auch Informationen über den jeweiligen Standort selbst gespeichert. Somit kann man beispielsweise definieren ob es sich bei diesem Ort um das Zuhause des Agenten handelt oder ob dieser Ort derzeit für den jeweiligen Agenten gesperrt ist.

6.4 Grober Ablauf der Simulation

Nach dem Start der Simulation werden vorerst grundlegende Einstellungen vorgenommen, welche von dem Simulationsmodell selbst eigentlich nicht beeinflusst werden. Dies wird durch die Methode setup der Klasse SarsModel vorgenommen, welche beispielsweise Ausgabemöglichkeiten während der Simulation erzeugt.

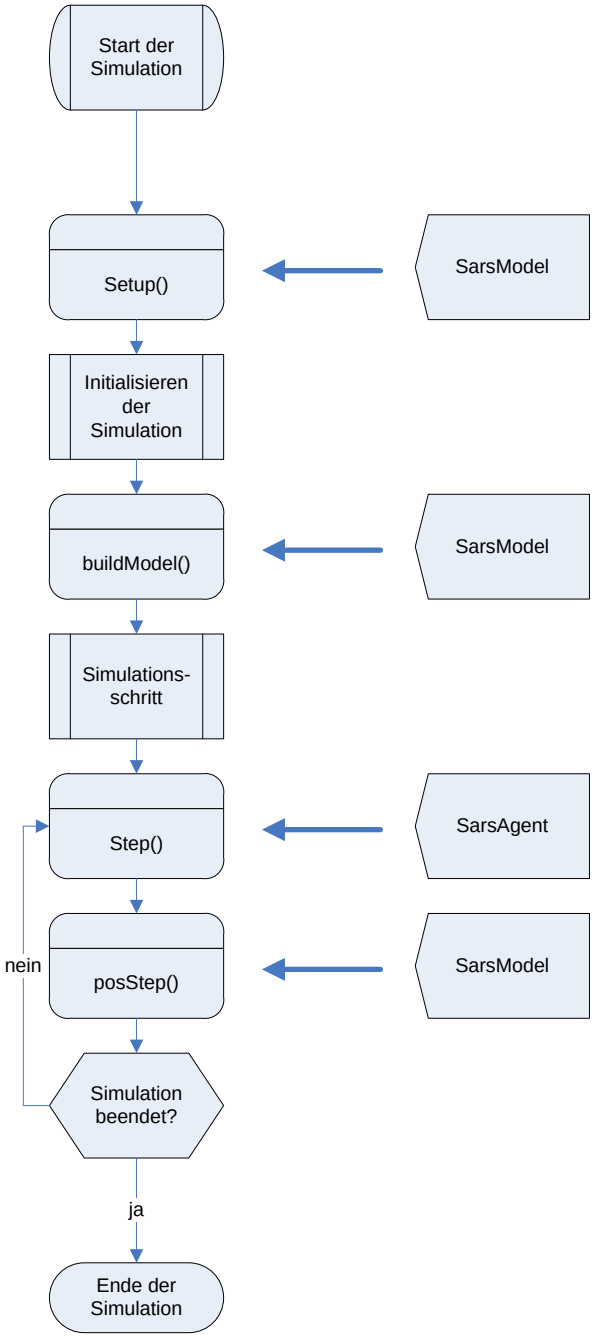

Abbildung 5

Danach muss man die Simulation initialisieren, welches durch die Funktion buildModel realisiert wird. Alle, über die Benutzeroberfläche frei definierbaren Werte, werden dann für den jeweiligen Simulationsdurchlauf übernommen.
Danach steht es einem frei die Simulation Schritt für Schritt oder in einem Zug durchlaufen zu lassen, wobei bei letzterem zu jeder Zeit die Möglichkeit besteht eine Pause einzulegen um sich die derzeitigen Ergebnisse anzusehen.

6.5 Analyse der eigenen Simulationsergebnisse

Während der Simulationsdurchläufe hat sich nun doch der Verdacht bestätigt, dass das vorliegende SARS-Modell einen weiteren Fehler enthält. Zwar handelt es sich hierbei eher um einen Logikfehler als um einen Modellfehlern, doch lassen sich dadurch sicher die Simulationsergebnisse beider Versionen nicht direkt miteinander vergleichen.
Die Simulation wird anfangs eindeutig mit einer bestimmten Anzahl an Agenten sowie einer bestimmten Größe des Raters definiert. Da jeder Rasterpunkt einem Ort eines einzelnen Agenten entspricht und stets alle möglichen Positionen vergeben werden, kann es unter Umständen auch vorkommen, dass Agenten überhaupt nicht auf dem Raster zu finden sind. BSP: Bei 1000 Agenten, welche max. 5 Aufenthaltsorte aufweisen können, ist es nicht möglich alle auf einen Raster von 10x10 unterzubringen, selbst wenn alle Agenten lediglich nur einen Aufenthaltsort besitzen.
Interessant wird es allerdings den Vergleich einer Epidemie mit und ohne Schutzvorkehrung zu vergleichen und dessen Auswirkungen auf den Verlauf zu analysieren. Dabei wird hierbei die Schutzmaskentragepflicht herangezogen, welche mit 2 Parametern verfeinert werden kann. Zum Einen kann man mit dem Parameter wearingMaskParticipation festlegen wie groß der Anteil ist, welcher sich überhaupt an diese Verordnung hält, zum Anderen bestimmt der Parameter wearingMaskPrevention wie hoch der Wirkungsgrad der Masken ist.

- Keine Schutzvorkehrungen

    - Simulationsparameter

      AgentCount = 1500

      AvgPeriodImmune = 0

      AvgPeriodIncubation = 5

      AvgPeriodInfectious = 25

      AvgPeriodRecovered = 7

      FrequencyContact = 4

      Width = 34

      Heigth = 33

      MirrorIdentityCount = 5

      NumberOfInfected = 5

      RateDeath = 0.204

      Rate ForeverImmune = 0.1

      RateInfection = 0.045

      WearingMaskParticipation = 0

      WearingMaskPrevention = 0

    - Erkenntnisse des Simulationslaufes

      Durch das Setzen der Parameter WearingMaskParticipation und WearingMaskPrevantion auf Null, wird eine vollkommene Ablehnung bzw. Abwesenheit von jeglichen Atemschutzmaßnahmen simuliert. Dies hat zur Folge, dass die Übertragung des Erregers nur mehr von der Wahrscheinlichkeit des Kontakts der Agenten sowie der Wahrscheinlichkeit der eigentlichen Infizierung abhängt.

      In Abbildung 6 kann man deutlich erkennen, dass die Anzahl der Infizierten nach kurzer Zeit drastisch ansteigt. Ein wenig zeitlich versetzt folgt diesem Verlauf auch die Kurve der verstorbenen Agenten, wobei sie durch die Anzahl der Agenten, welche sich noch regenerieren konnten, abgeschwächt wird.

      Durch die hohe Anzahl der verstorbenen Agenten kommt es in Folge auch zu einer Minderung der Kontakte und somit zu einer Senkung der Infektionen.


Legend
Susceptible
Infected
Incubated
Recovered
Deceased
Epidemiological Progress
Number of Agents [in %]
0
25
50
75
100
Number of Days
0
100
200
300
400
500
600
700

Abbildung 6

- Schutzmaskentragepflicht (Anteil: 50% / Effizienz: 50%)

    - Simulationsparameter

      AgentCount = 1500

      AvgPeriodImmune = 0

      AvgPeriodIncubation = 5

      AvgPeriodInfectious = 25

      AvgPeriodRecovered = 7

      FrequencyContact = 4

      Width = 34

      Heigth = 33

      MirrorIdentityCount = 5

      NumberOfInfected = 5

      RateDeath = 0.204

      Rate ForeverImmune = 0.1

      RateInfection = 0.045

      WearingMaskParticipation = 0.5

      WearingMaskPrevention = 0.5

    - Erkenntnisse des Simulationslaufes

      Abbildung 7 veranschaulicht nun einen Simulationslauf mit mäßiger Teilnahme an der Tragepflicht eines Atemschutzes, welcher nur über eine durchschnittliche Güte verfügt.

      Bereits hier lässt sich eine drastische Verbesserung erkennen, da nicht nur die Kurve der Infizierten deutlich gedämpft ist und daher nicht so stark ansteigt, sondern auch das Ende der Epidemie einen wesentlich positiveren Ausgang findet. Während bei der Simulation ohne Atemschutz (Abbildung 6) die Anzahl der Todesfälle fast bei 50 % lag und somit die Anzahl der Agenten, welche für den Erreger empfänglich sind, deutlich überstiegen hat, zeigt dieser Simulationsdurchlauf eine erkennbare Senkung der Anzahl der verstorbenen Agenten auf unter 40%.

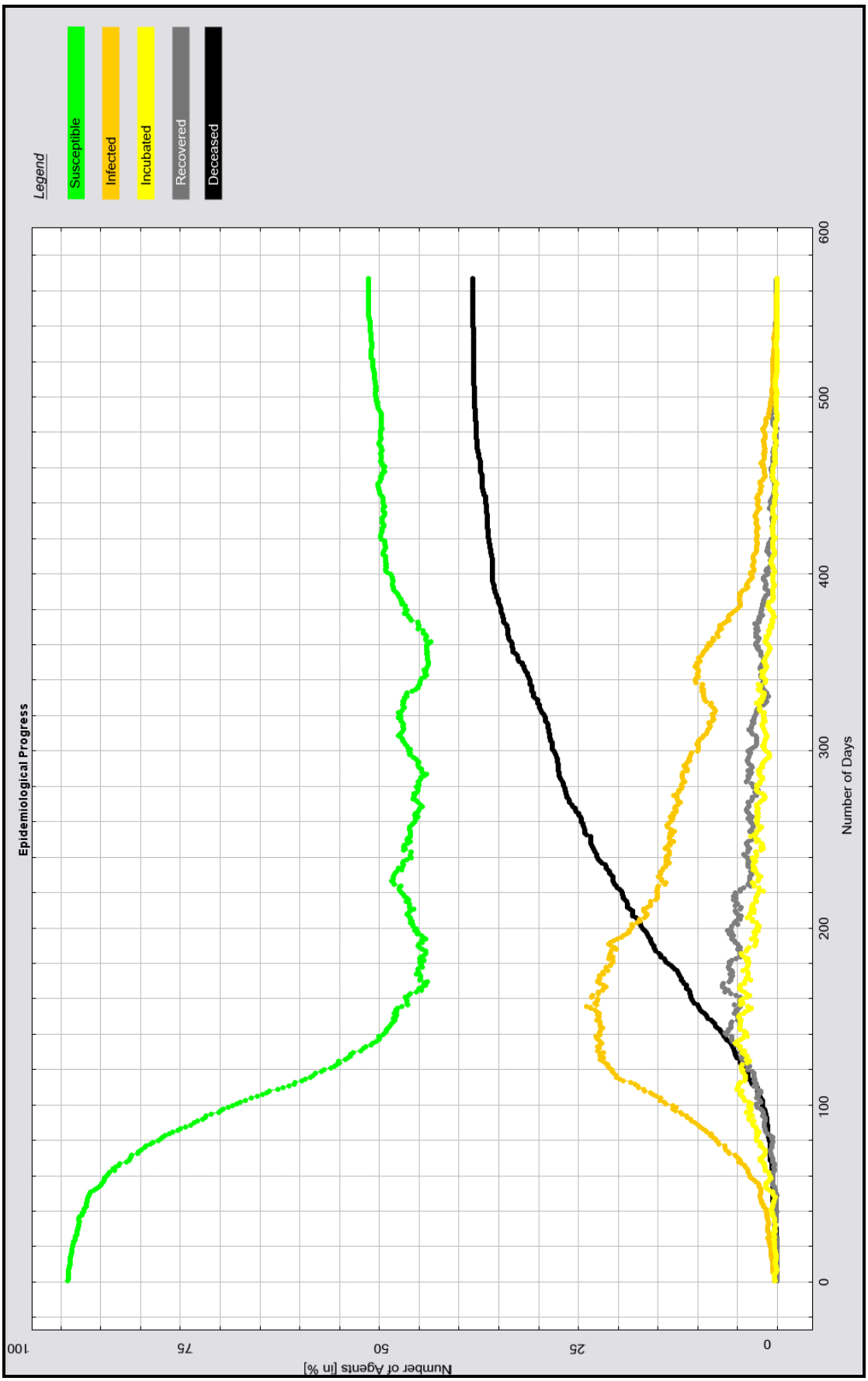
Legend
Susceptible
Infected
Incubated
Recovered
Deceased
Epidemiological Progress
Number of Agents [in %]
0
25
50
75
100
Number of Days
0
100
200
300
400
500
600

Abbildung 7

- Schutzmaskentragepflicht (Anteil: 50% / Effizienz: 75%)

    - Simulationsparameter

      AgentCount = 1500

      AvgPeriodImmune = 0

      AvgPeriodIncubation = 5

      AvgPeriodInfectious = 25

      AvgPeriodRecovered = 7

      FrequencyContact = 4

      Width = 34

      Heigth = 33

      MirrorIdentityCount = 5

      NumberOfInfected = 5

      RateDeath = 0.204

      Rate ForeverImmune = 0.1

      RateInfection = 0.045

      WearingMaskParticipation = 0.5

      WearingMaskPrevention = 0.75

    - Erkenntnisse des Simulationslaufes

      Abbildung 8 veranschaulicht nun einen Simulationslauf mit mäßiger Teilnahme an der Tragepflicht eines Atemschutzes, welcher über eine höhere Güte verfügt.

      Wiederum lässt sich eine Verbesserung feststellen: die Infektion kann sich nicht mehr so rasch Verbreiten und erreicht bei weiten nicht so viele Empfänger. Die Anzahl der verstorbenen Agenten wird auf 25% gesenkt und bestätigt bereits hier eine Anwendung von Schutzmasken im Falle einer Epidemie. Interessant wird schließlich der Vergleich mit dem Simulationsdurchlauf bei dem 75% der Agenten eine nur mäßig qualitative Schutzmaske verwenden. Dies wird dann beim nächten Simulationsdurchlauf analysiert.


Legend
Susceptible
Infected
Incubated
Recovered
Deceased
Epidemiological Progress
Number of Agents [in %]
0
25
50
75
100
Number of Days
0
100
200
300
400
500

Abbildung 8

- Schutzmaskentragepflicht (Anteil: 75% / Effizienz: 50%)

    - Simulationsparameter

      AgentCount = 1500

      AvgPeriodImmune = 0

      AvgPeriodIncubation = 5

      AvgPeriodInfectious = 25

      AvgPeriodRecovered = 7

      FrequencyContact = 4

      Width = 34

      Heigth = 33

      MirrorIdentityCount = 5

      NumberOfInfected = 5

      RateDeath = 0.204

      Rate ForeverImmune = 0.1

      RateInfection = 0.045

      WearingMaskParticipation = 0.75

      WearingMaskPrevention = 0.5

    - Erkenntnisse des Simulationslaufes

      Abbildung 9 zeigt den Simulationslauf mit einer höheren Teilnahme an der Tragepflicht eines Atemschutzes, welcher allerdings nur über eine mäßige Güte verfügt.

      Vergleicht man nun die Simulationsergebnisse mit jenen der Vorangegangen (Abbildung 8), welche von einer 50%-igen Teilnahme der Tragepflicht und einer qualitativ besseren Atemschutzmaske ausgegangen ist, lässt sich eine Ähnlichkeit beider Kurvenverläufe erkennen. Jedoch kann man beobachten, dass die Anzahl der infizierten Agenten in diesem Simulationsdurchlauf gerade in der kritischen Phase, jene in welcher die Anzahl der verstorbenen Agenten anwächst, eher konstant bleibt und erst später absinkt. Dies führt scheinbar dazu, dass am Ende der eigentlichen Epidemie ein schlechteres Ergebnis erzielt wird, da knapp über 30% der Agenten den Angriff des Erregers nicht überlebt haben (Abbildung 9).

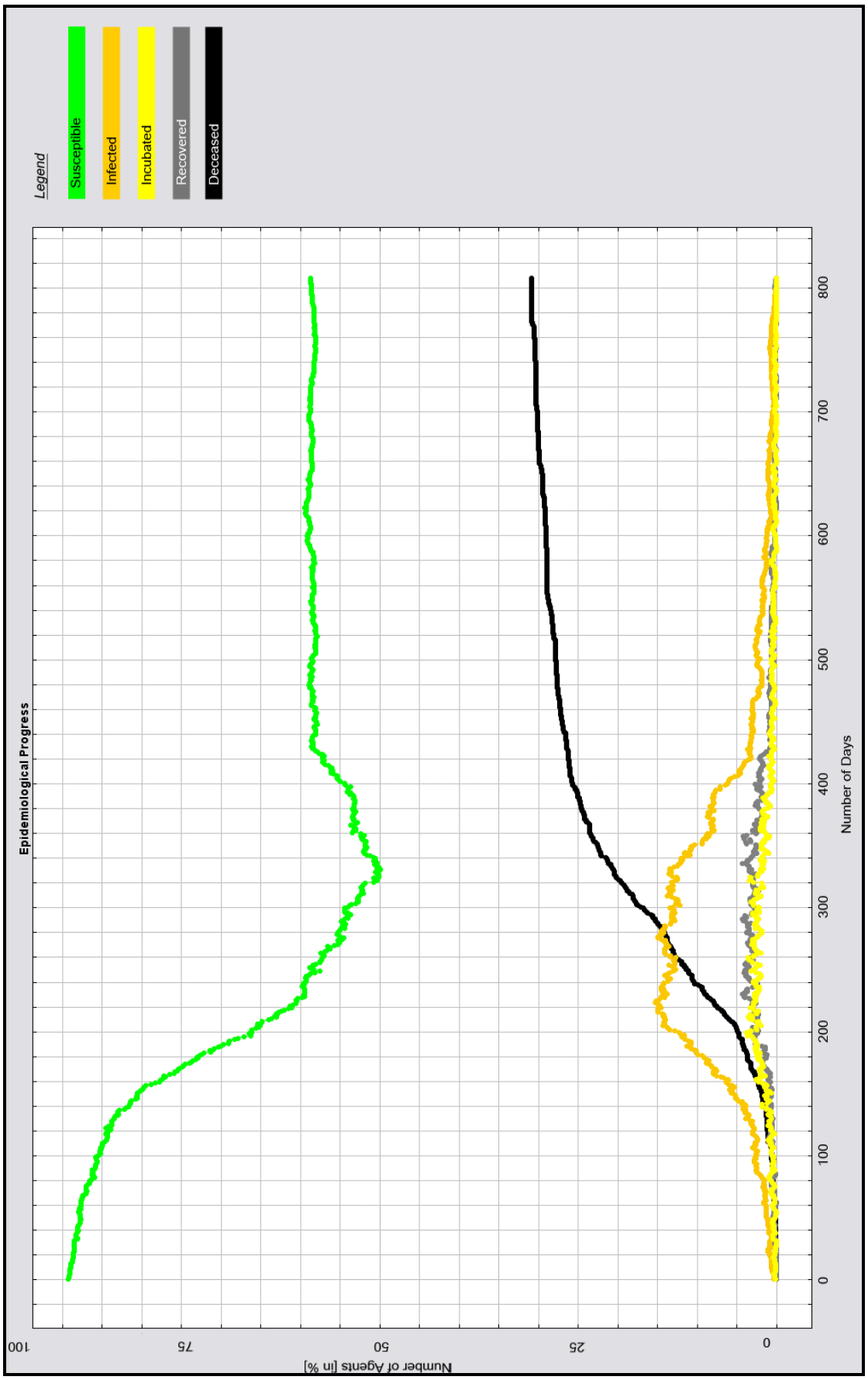
Legend
Susceptible
Infected
Incubated
Recovered
Deceased
Epidemiological Progress
Number of Agents [in %]
0
25
50
75
100
Number of Days
0
100
200
300
400
500
600
700
800

Abbildung 9

- Schutzmaskentragepflicht (Anteil: 75% / Effizienz: 75%)

    - Simulationsparameter

      AgentCount = 1500

      AvgPeriodImmune = 0

      AvgPeriodIncubation = 5

      AvgPeriodInfectious = 25

      AvgPeriodRecovered = 7

      FrequencyContact = 4

      Width = 34

      Heigth = 33

      MirrorIdentityCount = 5

      NumberOfInfected = 5

      RateDeath = 0.204

      Rate ForeverImmune = 0.1

      RateInfection = 0.045

      WearingMaskParticipation = 0.75

      WearingMaskPrevention = 0.75

    - Erkenntnisse des Simulationslaufes

      Der letzte Simulationsdurchlauf soll nun eine rege Beteiligung an der Tragepflicht veranschaulichen, wobei Schutzmasken mit einer hohen qualitativen Güte zum Einsatz kommen.

      Abbildung 10 lässt auch deutlich erkennen, dass beide Parameter einen wesentlichen Einfluss auf die Epidemie haben können, oder so wie in diesem Fall eine solche gar verhindern können. Die Anzahl der Infizierten Agenten kann drastisch minimiert werden, welches eine gleichzeitige Auswirkung auf die Anzahl der verstorbenen Agenten mit sich bringt. Diese pendelt sich nämlich am Ende der Simulation bei ungefähr 7% ein.

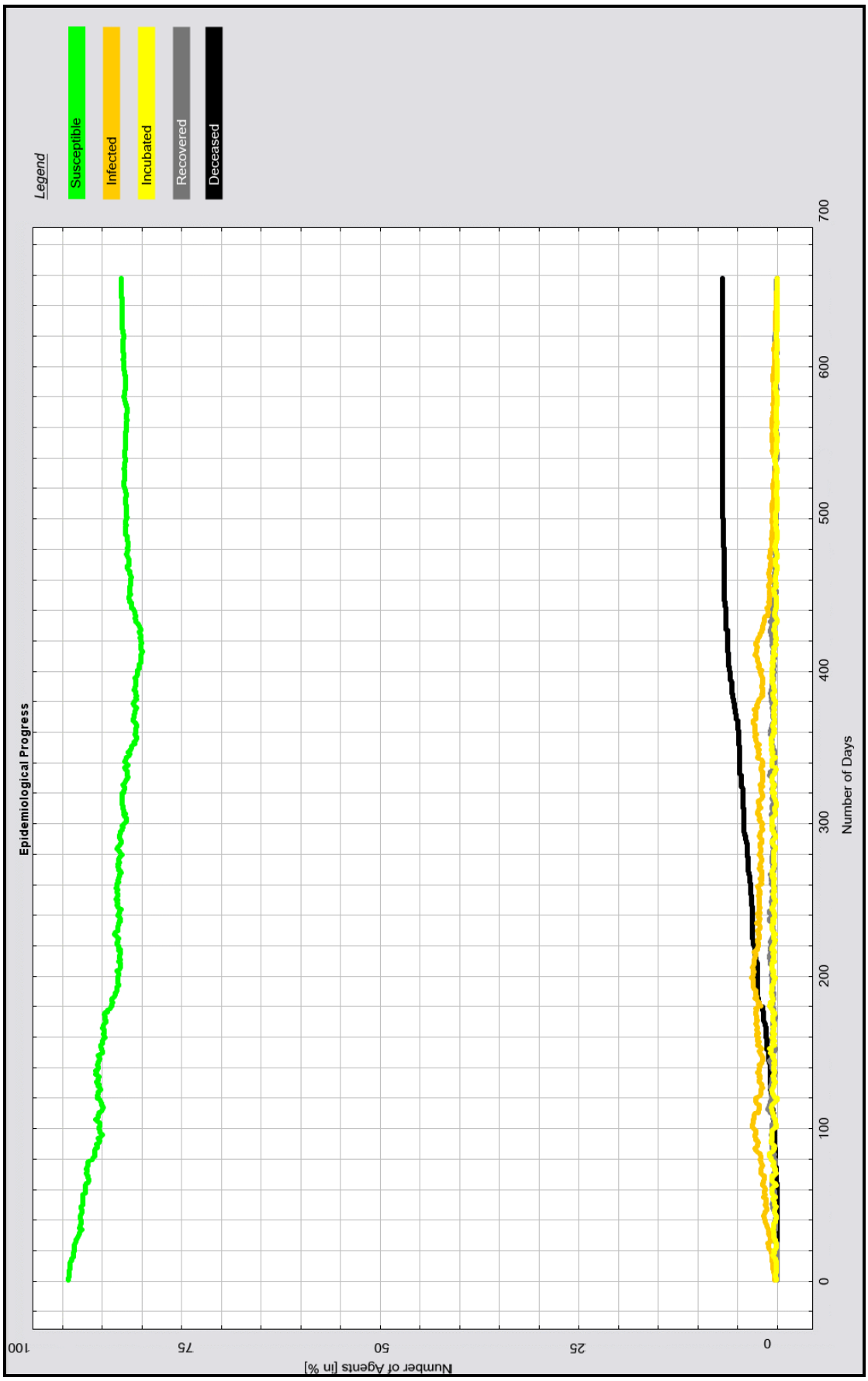

Abbildung 10

6.6 Die Simulationsmodelle und deren Ergebnisse im Vergleich

Ein direkter Vergleich, welcher Schwachstellen des Ausgangsmodells hätte liefern können, konnte jedoch nicht angestellt werden, was bereits auf die schlechte Umsetzung des Ausgangsmodells zurückzuführen ist. Die zweifache Definition der Agentenanzahl und der Anzahl der verfügbaren Aufenthaltsorte der Agenten auf dem Raster führen bereits bei einer unsachgemäßen Angabe der Werte zu Konflikten. Eine übergroße Anzahl an Agenten im Vergleich zu möglichen Positionen hat beispielsweise zur Folge, dass viele Agenten schließlich keine Wirkung auf das Gesamtsystem ausüben können und somit für die Simulation unnötig werden.

Zusätzlich waren leider nicht genügend Informationen über Ergebnisse mehrerer Durchläufe verfügbar, somit war auch hier eine Einschränkung der Analyse gegeben.

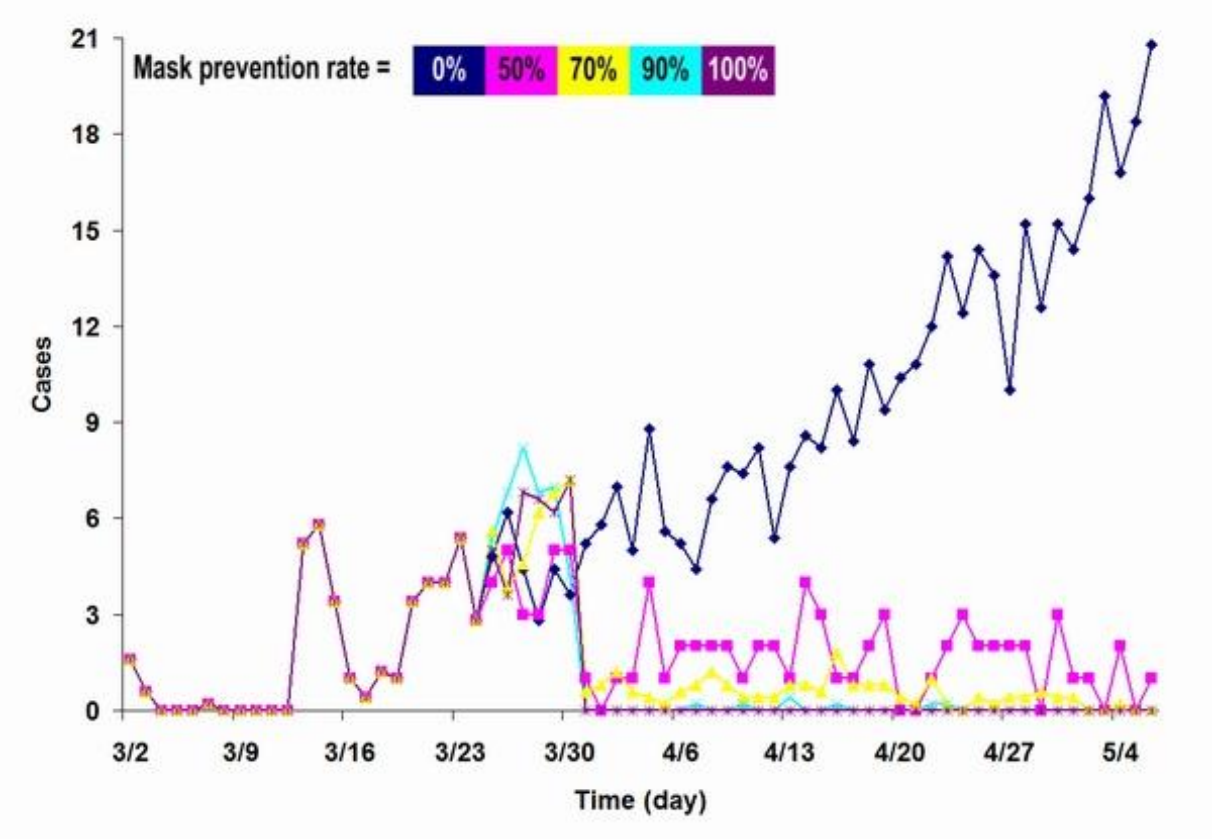

Abbildung 11

Die obige Graphik veranschaulicht den unterschiedlichen Verlauf einer Epidemie bei verordneter Schutzmaskenpflicht und unterschiedlichen Wirkungsgraden der verwendeten Masken. Hier lässt sich deutlich erkennen, dass vollkommen unwirksame Masken einen

drastischen Anstieg der Infektionsfälle mit sich bringen, während bereits auch minderqualitative Masken das Wachstum der Infizierten deutlich bremsen können.
Aus den angeführten Graphiken in Kapitel 6.6 sieht man, dass die replizierte Simulation ein ähnliches Verhalten mit sich bringt. Jedoch kann hier auch abgelesen werden, dass zusätzlich zum Wirkungsgrad der Masken auch die Teilnahme der Bevölkerung an der Schutzmaskentragepflicht eine erhebliche Rolle spielt. Die besten Masken können die Epidemie nicht verhindern, wenn diese nur von wenigen Personen auch getragen werden. Somit geht der Wirkungsgrad der Schutzmasken mit der Teilnahme der Bevölkerung Hand in Hand.

6.7 Erfahrungen und gewonnene Erkenntnisse

Die Replikation an sich ist ein zwingend notwendiges Werkzeug um die Gültigkeit bzw. die Qualität eines Modells zu überprüfen. Jedoch ist der Arbeitsaufwand dabei nicht zu unterschätzen, da bereits das Erlangen des vollständigen Wissens um die Funktion des zu replizierenden Modells eine Menge Zeit in Anspruch nimmt. Hier sollte man auch bereits beginnen die Eigenschaften und deren Zusammenspiel kritisch zu hinterfragen. Die Annahme, dass Agenten nur fixe Positionen auf dem Raster besitzen und somit stets mit den gleichen Nachbaragenten in Berührung kommen, scheint nicht besonders Realitätsnahe. Sind doch gerade öffentliche Massentransportmittel oder Großveranstaltungen der ideale Platz, um eine bestehende Infektion an viele unterschiedliche Personen weiterzugeben, welche meist im Alltag keinerlei Bezug zueinander haben.
Ein bereits vorhandenes Beherrschen einer Programmiersprache, in welcher die Replikation durchgeführt wird, ist dringend anzuraten, da die Verwendung des Simulationstoolkits ohne solche Kenntnisse sicher nur schwer möglich ist. Die Ansätze gehen zwar in Richtung von graphisch unterstützten Simulationsabläufen, doch scheinen diese Systeme noch nicht ganz ausgereift zu sein.
Eine große Erleichterung für die Replikation eines bestehenden Modells wäre ein Zugang zu dem verfügbaren Quellcode der zu replizierenden Simulation. Dies würde eine Menge Zeit ersparen und könnte bereits im Vorfeld einige Probleme lösen beziehungsweise Fragestellungen beantworten. In diesem Fall waren nur Pseudo-Fragmente erhältlich,

welche sicher auch unterstützend wirken, jedoch oft auch missverständlich sein können. Somit ist hier wieder eine Gefahrenquelle gegeben, welche eine korrekte Umsetzung verhindern könnte. Um schließlich eine Analyse des erstellten Simulationsmodells zu bewerkstelligen, ist es nötig genügend Ergebnisse von unterschiedlichen Simulationsläufen zur Verfügung zu haben. Nur so ist es auch möglich beide Modelle gewissenhaft zu untersuchen und Unterschiede erkennen und somit eventuelle Fehler aufdecken zu können. Es wäre von Vorteil wenn die vorliegenden Ergebnisse in einer verständlichen allgemeinen Form dargestellt sind, da dies beim Replizieren wieder Zeit ersparen würde. So wurde beispielsweise in dem vorgegebenen SARS-Modell die Ausgabe der Ergebnisse stets von einem konkreten Datum abhängig gemacht. Dadurch ist eigentlich wieder eine Anpassung auf dieses spezielle Format notwendig.

7 Literaturverzeichnis

Papers

ANDERSSON M. R., SANDHOLM T. W. (2001): "leveled commitment contracts with miopic and strategic agents", *Journal of Economic, Dynamics and Control 25,* p615-640

AXELROD R. (1997): "the complexity of cooperation: agent-based models of conflict and cooperation", *Princeton, NJ: The Princeton University Press,* p47

AXELROD R. (2005): "advancing the art of simulation in the social sciences", *University of Michigan*

AXELROD, R. AND TESTFATSION, L. (2006): "a guide for newcomers to agent-based modeling in the social sciences.", *Handbook of Computational Economics, chapter Appendix A, Elsevier, Amsterdam*

BARRAT A. AND WEIGT M. (1999): "on the properties of small world network models", *European Physical Journal B*

BOWER J., BUNN D. (2001): "experimental analysis oft he efficiency of uniform-price versus discriminatory auctions in the England and Wales electricity market", *Journal of Economic*, Dynamics and Control, p561-592

BRATLEY P., FOX B. AND SCHRAGE L. (1987): "a guide to simulation", *2nd edition, New York*

BRYSON J. J., YASUSHI A. AND LEHMANN H. (2007): "agent-based models as scientific methodology - a case study analysing primate social behaviour", *Artificial models of natural Intelligence, Department of Computer Science, University of Bath*

BURTON, R. (1998): "validating and docking: an overview, summary and challenge" *in Simulating Organizations: Computational Models of Institutions and Groups, M. Prietula, K. Carley and L. Gasser (eds.), MIT Press: Cambridge, MA.*

CAMERER C. F. AND LOWENSTEIN G. (2003), "behavioral economics: past, present, future“, *Advances in Behavioral Economics: p3-51*

DAWID H. (1999): "adaptive learning by genetic algorithms: analytical results and applications to economic models", 2nd edition, Berlin

DAWID H., REIMANN M. AND BULLNHEIMER B. (2001): "to inovate or not to inovate?", *IEEE, Transactions on Evolutionary Computations 5,* p471-481

DUFFY J. (2001): "learning to speculate: experiments with artificial and real agents", *Journal of Economic, Dynamics and Control 25,* p295-319

EDMONDS B. AND HALES D. (2002): "replication, replication and replication – some hard lessons from model alignment", *Centre for Policy Modelling (http://cfpm.org)*

EPSTEIN J. M., AXTELL R. (1996): "growing artificial societies: social science from the bottom up", *Cambridge, MA: MIT Press.*

GARDNER, M. (1970): "the fantastic combinations of John Conway's new solitaire game 'life', *Scientific American*, p120-123

GINITS H. (2000): "game theory evolvoing", *Princeton, NJ: The Princeton University Press*

GLEDITSCH N. P. AND METELITS C. (2003): "the replication debate", *International Studies Perspectives, 4(1): p72–79*

HOLLAND J. (1995): "Can There Be A Unified Theory of Complex Adaptive Systems?", *in: Harold J. Morowitz, Jerome L. Singer, editors. The Mind, The Brain, and Complex Adaptive Systems. Addison-Wesley.*

KING G. (1995): "replication, replication.", *PS: Political Science and Politics 28: p444-452*

KOLLMAN K., MILLER J. H. AND PAGE S. E. (2003): "computational models in political economy", ISBN: 0262112752, p2-11

KUPERMAN M. AND ABRAMSON G. (2001): "small world effect in an epidemiological model", *Physical Review Letters, Volume 86, Number 13: 2909-2912*

LUCAS R. E. Jr. (1987): "methods and problems in business cycle theory", *Studies in Business-Cylce Theory, Cambridge, UK: Cambridge University Press,* p271-296

MACAL C. M. AND NORTH M. J. (2005): "tutorial on agent-based modeling and simulation", *Proceedings of the 2005 Winter Simulation Conference*

MARKS R. E. (1992): "breeding hybrid strategies: optimal behavior in oligopolies", *Journal of Evolutionary Economics 2*, p17-38

MOORE C. AND NEWMAN M. E. J.(2000): "epidemics and percolation in small-world networks", *Departments of Computer Science and Physics, University of New Mexico*

NEWMAN M. (1999): "small worlds: the structure of social networks", *Working Papers 99-12-080, Santa Fe Institute*

SARGENT T. (1993): "Bounded rationality in macroeconomics. ", *The Arne Ryde Memorial Lectures, Oxford, UK: Clarendon Press.*

SCHELLING, T. C. (1971): "Dynamic models of segregation", *Journal of Mathematical Sociology 1: 143-186.*

SCHELLING T. C. (1978): "micromotives and macrobehavior", *New York, NY: W.W. Norton and Co.*

TESFATSION L. (2002): "agent-based computational economics: growing economies from the bottom up", *Department of Economics, Iowa State University*

TESFATSION L. (2003): "agent-based computational economics", *Department of Economics, Iowa State University*

TESFATSION L. (2005): "agent-based computational economics: a constructive approach to economic theory", *Economics Department, Iowa State University*

VAN ZANDT, T. (1998): "organisations with an endogenous number of information processing agents", *Organizations with incomplete information, Cambridge, UK: Cambridge University Press*

WATTS D. J. (1999): "small worlds", *Princeton University Press, Princeton*

WILHITE, A. (2001): "bilateral trade and ‘small-world‘ networks", *Computational Economics 18*, p49-64

Internetadressen

Wikipedia Online Lexikon, http://www.wikipedia.com

Home Page of Leigh Tesfatsion, http://www.econ.iastate.edu/tesfatsi/

Repast Agent Simulation Toolkit, http://repast.sourceforge.net/

8 Anhang

Quellcode der Klasse SarsModel

```
/**
 * Diese Klasse stellt die eigentliche Verbindung zum Repast-Framework dar. Die
 * Funktionen setup(), buildModel() und postStep() bzw. preStep werden durch sie
 * überschrieben.
 * Die anfängliche Initialisierung des Modells sowie das Festlegen der Simulations-
 * parameter wird durch diese Klasse realisiert.
 *
 * @author Christoph Kaniak
 * @version 1.0
 */

import uchicago.src.sim.engine.SimpleModel;
import uchicago.src.sim.engine.SimInit;
import uchicago.src.sim.gui.DisplaySurface;
import uchicago.src.sim.gui.Object2DDisplay;
import uchicago.src.sim.space.Object2DTorus;
import uchicago.src.sim.analysis.*;
import cern.jet.random.engine.*;
import cern.jet.random.*;

public class SarsModel extends SimpleModel {

    // Attribute der Simulation
    private int agentCount = 10000;
    private boolean wearingMaskPolicy = false;
    private double wearingMaskParticipation = 0.5;
    private double wearingMaskPrevention = 0.5;
    private int mirrorIdentityCount = 5;
    private int avgPeriodIncubation = 5;
    private int avgPeriodInfectious = 25;
    private int avgPeriodRecovered = 7;
    private int avgPeriodImmune;
    private double rateForeverImmune = 0.1;
    private double rateInfection = 0.045;
    private double rateDeath = 0.204;
    private int frequencyContact = 4;
    private int numberOfInfected = 2;
    private int height = 50;
    private int width = 50;

    // Objekte für die Berechnung der Normalverteilung
    private MersenneTwister numberGenerator;
    private Normal normalDistributionMICount;
    private Normal normalDistributionContact;

    // Definition der Ausgabeobjekte
    private DisplaySurface dsurf;
    private Plot epiProgressPlot;

    // Konstruktor
    public SarsModel() {

        // Paramter, welche über die Benutzeroberfläche festzulegen sind
        // !!! Alle Parameter benötigen zwingend Setter- und Getter-Methoden
        params = new String[] {
                "height", "width", "agentCount", "wearingMaskPolicy",
                "wearingMaskParticipation", "wearingMaskPrevention",
                "mirrorIdentityCount", "avgPeriodIncubation",
                        "avgPeriodInfectious", "avgPeriodRecovered",
                        "avgPeriodImmune", "rateForeverImmune", "rateInfection",
                        "rateDeath",  "frequencyContact", "numberOfInfected"};
```

```
    // Bei jedem einzelnen Schritt, wird die Methode step() aller
    // Agentenobjekte aufgerufen
    autoStep = true;
}

/**
 * Initialisiert Kernfunktionen der Simulation. (z.B.: graphische Ausgabe)
 */
public void setup() {

    // Konstruktor der Elternklasse
    super.setup();

    // Erzeugung eines Containers für den Raster
    if (dsurf != null)
        dsurf.dispose();
    dsurf = new DisplaySurface(this, "Agent Environment");
    super.registerDisplaySurface("Agent Environment", dsurf);
}

/**
 * In Abhängigkeit der festgelegten Simulationsparameter wird das Modell neu
 * initialisiert.
 */
public void buildModel() {
    Object2DTorus space = new Object2DTorus(this.width, this.height);

    // IDs von Agenten, welche permanent immun sind
          int tempCount = (int) Math.ceil(this.agentCount *
                           this.rateForeverImmune);
          int[] immuneAgentId = this.getRandomAgentId(this.agentCount,
                        tempCount, null);
    int[] wearingMaskAgentId;

    // Maskentragepflicht vorgegeben?
    if (this.wearingMaskPolicy){

        // IDs von Agenten, welche eine Maske tragen
        tempCount = (int) Math.ceil(this.agentCount *
        this.wearingMaskParticipation);
                wearingMaskAgentId = this.getRandomAgentId(this.agentCount,
                tempCount, null);
    } else
        wearingMaskAgentId = null;

          // Erzeugung von Zahlengeneratoren, welche stets normalverteilte Werte
          liefern
    this.numberGenerator = new MersenneTwister();
          this.normalDistributionMICount = new Normal(3.0, 1.0,
          this.numberGenerator);
          this.normalDistributionContact = new Normal(0.5 , 0.15,
          this.numberGenerator);

    // Durchlauf für alle Agenten
    for (int i = 0; i < this.agentCount; i++){

        // Erzeuge ein SarsAgent-Objekt
        SarsAgent agent = new SarsAgent(this, space);

        // Definiere eindeutige ID
        agent.setId(i);

        // Festlegen der max. Anzahl der Mirror Identitys
                                                agent.setMirrorIdetityCount(
                    this.normalDistributionMICount.nextInt());

        // Festlegen der Kontaktrate
        agent.setRateContact(
                    this.normalDistributionContact.nextDouble());
```

```
    // über ID wird ermittelt ob Agent permanent immun
    if (intExistsInArray(immuneAgentId, agent.getId())){
        agent.setEpiProgress('M');
        agent.setPermanentImmunity(true);
    }

    // über ID wird ermittelt ob Agent eine Schutzmaske trägt
    if (this.wearingMaskPolicy){
        if (intExistsInArray(wearingMaskAgentId, agent.getId()))
            agent.setWearingMask(true);
    }

    // Festlegung welche Agenten von Beginn an infiziert sind
    if (i < this.numberOfInfected)
        agent.setEpiProgress('I');

    // Anhängen des SarsAgent-Objekts an die Agentenliste
    agentList.add(agent);
}

      // Jeder Raster stellt einen möglichen Aufenthaltsort eines Agenten
// dar. Verknüpfung jedes Rasters mit zufällig gewählten Agenten
int mirrorIdentityId = 0;
for (int y = 0; y < this.height; y++){
    for (int x = 0; x < this.width; x++){
        // Erzeugung der MirrorIdentity
        MirrorIdentity tempMirrorIdentity =
                            new MirrorIdentity(x, y);

        // Jede MirrorIdentity erhält eine eindeutige ID
        tempMirrorIdentity.setId(mirrorIdentityId++);

        // Verbinden der MirrorIdentity mit Agenten
        boolean mirrorIdentityConnected = false;
        while (!mirrorIdentityConnected){

            // Wahl eines beliebigen Agenten
            int agentId = getNextIntFromTo(0,
                                    this.agentCount - 1);
                            SarsAgent agent = (SarsAgent)
                            agentList.get(agentId);

            // Agent kann weitere MirrorIdentitys aufnehmen?
            if (agent.needsMoreMirrorIdentity()){

                // Verknüpfung MirrorIdentity mit Agent
                agent.addMirrorIdentity(tempMirrorIdentity);
                mirrorIdentityConnected = true;
            }
        }
    }
}

// Festlegen der Root-MirrorIdentity
for (int i = 0; i < agentList.size(); i++){
    SarsAgent agent = (SarsAgent) agentList.get(i);
    MirrorIdentity[] agentMirrorIdentity =
        agent.getMirrorIdentity();
    try{
        int tempMirrorIdentityCount =
                            agentMirrorIdentity.length-1;
        if (tempMirrorIdentityCount < 0)
            tempMirrorIdentityCount = 0;
        int index = getNextIntFromTo(0, tempMirrorIdentityCount);
        agentMirrorIdentity[index].setRoot(true);
        agent.setMirrorIdentity(agentMirrorIdentity);
            agent.setXY(agentMirrorIdentity[index].getX(),
                            agentMirrorIdentity[index].getY());
    } catch (Exception e){
```

```
        }

    }

    // Entfernen aller Agenten die nicht am Raster Platz gefunden haben
    for (int i = 0; i < this.agentList.size(); i++) {
        SarsAgent agent = (SarsAgent) this.agentList.get(i);
        if (agent.getMirrorIdentity() == null ||
            agent.getMirrorIdentity().length == 0)
            this.agentList.remove(i);

    }
    this.agentCount = this.agentList.size();

    // Einrichten des Rasters
    Object2DDisplay display = new Object2DDisplay(space);
    dsurf.addDisplayable(display, "Agent Environment");
    dsurf.display();

    // Einrichten des Graphen
    epiProgressPlot = new Plot("Epidemiological Progress");
        epiProgressPlot.addLegend(0,
                "Susceptible", SarsAgent.COLOR_SUSCEPTIBLE, Plot.FILLED_CIRCLE);
        epiProgressPlot.addLegend(1,
                "Incubated", SarsAgent.COLOR_INCUBATED , Plot.FILLED_CIRCLE);
        epiProgressPlot.addLegend(2,
                "Infected", SarsAgent.COLOR_INFECTED, Plot.FILLED_CIRCLE);
        epiProgressPlot.addLegend(3,
                "Recovered", SarsAgent.COLOR_RECOVERED, Plot.FILLED_CIRCLE);
        epiProgressPlot.addLegend(4,
                "Deceased", SarsAgent.COLOR_DECEASED, Plot.FILLED_CIRCLE);
        epiProgressPlot.setAxisTitles("days", "agents");
        epiProgressPlot.setConnected(true);
        epiProgressPlot.setXRange(0.0, 10.0);
        epiProgressPlot.setYRange(0.0, this.agentCount);
        epiProgressPlot.setXIncrement(10);
        epiProgressPlot.display();
}


/**
 * Wird nach dem eigentlichen Simulationsschritt ausgeführt. Aktualisiert
 * alle Ausgaben und bricht Simulation ab sobald keine Infektion mehr
 * vorhanden.
 */
public void postStep(){

    // Aktualisieren der Anzeige
    this.dsurf.updateDisplay();
    this.epiProgressPlot.plotPoint(this.getTickCount(),
                this.getNumberOfAgents('S'), 0);
    this.epiProgressPlot.plotPoint(this.getTickCount(),
        this.getNumberOfAgents('E'), 1);
    this.epiProgressPlot.plotPoint(this.getTickCount(),
        this.getNumberOfAgents('I'), 2);
    this.epiProgressPlot.plotPoint(this.getTickCount(),
        this.getNumberOfAgents('R'), 3);
    this.epiProgressPlot.plotPoint(this.getTickCount(),
        this.getNumberOfAgents('D'), 4);
    this.epiProgressPlot.updateGraph();
    this.epiProgressPlot.fillPlot();

    // Abbruch der Simulation wenn Infektion vorbei
    if (this.getNumberOfAgents('I') == 0 &&
        this.getNumberOfAgents('E') == 0 &&
        this.getNumberOfAgents('R') == 0)
        this.stop();
}
```

```
    /**
     * Hauptroutine
     */
    public static void main(String[] args) {
        SarsModel sarsModel = new SarsModel();
        SimInit init = new SimInit();
        init.loadModel(sarsModel, null, false);
    }


////////////////////
// SIMULATION FUNCTIONS
////////////////////

    /**
     * Liefert die Anzahl der Agenten, welche einen bestimmten epidemischen
     * Zustand aufweisen. Bei Übergabe von '*' wird die Anzahl aller Agenten
     * ausgegeben.
     *
     * @param char Kürzel für den epidemischen Zustand (S, E, I, R, D)
     * @return int Anzahl von Agenten
     */
    private int getNumberOfAgents(char epiProgress) {
        int numberOfAgents = 0;
        if (epiProgress == '*')
                numberOfAgents = this.agentList.size();
        else {
            for (int i = 0; i < this.agentList.size(); i++) {
                SarsAgent agent = (SarsAgent) this.agentList.get(i);
                if (agent.getEpiProgress() == epiProgress)
                    numberOfAgents++;
            }
        }
        return numberOfAgents;
    }


    /**
     * Liefert die Anzahl der Agenten, welche eine Maske tragen
     *
     * @return int Anzahl von Agenten
     */
    private int getNumberOfAgentsWearingMask() {
        int numberOfAgents = 0;
        for (int i = 0; i < this.agentList.size(); i++) {
            SarsAgent agent = (SarsAgent) this.agentList.get(i);
            if (agent.isWearingMask())
                numberOfAgents++;
        }
        return numberOfAgents;
    }

    /**
     * Liefert die Anzahl der Agenten, welche eine Maske tragen
     *
     * @param int[] Zu überprüfendes Array
     * @param int Integer, nach der zu suchen ist
     * @return boolean Liefert true, wenn Wert gefunden. Sonst false.
     */
    private boolean intExistsInArray(int[] array, int value) {
        try{
            for (int i = 0; i < array.length; i++) {
                if (array[i] == value)
                    return true;
            }
        } catch (Exception e){
        }
        return false;
    }
```

```java
/**
 * Liefert Integerarray zufällig gewählter Zahlen, welche alle in dem Bereich
 * von 0 bis maxId liegen. Durch die Festlegung von lockedId kann man
 * bestimmte Zahlen sperren lassen und werden nicht in das Array abgelegt.
 *
 * @param int höchstzugelasserer Wert
 * @param int Anzahl der zu generierenden Zahlen, entspricht Länge des Arrays
 * @param int[] gesperrte Werte
 * @return int[] Array mit zufällig gewählten Integerwerten
 */
private int[] getRandomAgentId(int maxId, int idCount, int[] lockedId) {
    int[] id;
    int tempId;
    int i = 0;

    // Definition des Ausgabearrays
    id = new int[idCount];

    while (idCount != i) {

        // Erzeugung einer Zufallszahl
        tempId = this.getNextIntFromTo(0, maxId);

        // Zahl bereits in Array vorhanden?
        if (!intExistsInArray(id, tempId)){

            // Zahl ist gesperrt?
            if (lockedId != null) {
                if (!intExistsInArray(lockedId, tempId))
                    id[i++] = tempId;
            } else
                id[i++] = tempId;
        }
    }
    return id;
}

////////////////////
// GETTER- AND SETTER FUNCTIONS
////////////////////

/**
 * @return the agentCount
 */
public int getAgentCount() {
    return agentCount;
}

/**
 * @return the mirrorIdentityCount
 */
public int getMirrorIdentityCount() {
    return mirrorIdentityCount;
}

/**
 * @return the avgPeriodIncubation
 */
public int getAvgPeriodIncubation() {
    return avgPeriodIncubation;
}

/**
 * @return the avgPeriodInfectious
 */
public int getAvgPeriodInfectious() {
    return avgPeriodInfectious;
}

/**
```

```
     * @return the avgPeriodRecovered
     */
    public int getAvgPeriodRecovered() {
        return avgPeriodRecovered;
    }

    /**
     * @return the avgPeriodImmune
     */
    public int getAvgPeriodImmune() {
        return avgPeriodImmune;
    }

    /**
     * @return the rateForeverImmune
     */
    public double getRateForeverImmune() {
        return rateForeverImmune;
    }

    /**
     * @return the rateInfection
     */
    public double getRateInfection() {
        return rateInfection;
    }

    /**
     * @return the rateDeath
     */
    public double getRateDeath() {
        return rateDeath;
    }

    /**
     * @return the frequencyContact
     */
    public int getFrequencyContact() {
        return frequencyContact;
    }

    /**
     * @return the height
     */
    public int getHeight() {
        return height;
    }

    /**
     * @return the width
     */
    public int getWidth() {
        return width;
    }

    /**
     * @param agentCount the agentCount to set
     */
    public void setAgentCount(int agentCount) {
        this.agentCount = agentCount;
    }

    /**
     * @param mirrorIdentityCount the mirrorIdentityCount to set
     */
    public void setMirrorIdentityCount(int mirrorIdentityCount) {
        this.mirrorIdentityCount = mirrorIdentityCount;
    }

    /**
```

```
     * @param avgPeriodIncubation the avgPeriodIncubation to set
     */
    public void setAvgPeriodIncubation(int avgPeriodIncubation) {
        this.avgPeriodIncubation = avgPeriodIncubation;
    }

    /**
     * @param avgPeriodInfectious the avgPeriodInfectious to set
     */
    public void setAvgPeriodInfectious(int avgPeriodInfectious) {
        this.avgPeriodInfectious = avgPeriodInfectious;
    }

    /**
     * @param avgPeriodRecovered the avgPeriodRecovered to set
     */
    public void setAvgPeriodRecovered(int avgPeriodRecovered) {
        this.avgPeriodRecovered = avgPeriodRecovered;
    }

    /**
     * @param avgPeriodImmune the avgPeriodImmune to set
     */
    public void setAvgPeriodImmune(int avgPeriodImmune) {
        this.avgPeriodImmune = avgPeriodImmune;
    }

    /**
     * @param rateForeverImmune the rateForeverImmune to set
     */
    public void setRateForeverImmune(double rateForeverImmune) {
        this.rateForeverImmune = rateForeverImmune;
    }

    /**
     * @param rateInfection the rateInfection to set
     */
    public void setRateInfection(double rateInfection) {
        this.rateInfection = rateInfection;
    }

    /**
     * @param rateDeath the rateDeath to set
     */
    public void setRateDeath(double rateDeath) {
        this.rateDeath = rateDeath;
    }

    /**
     * @param frequencyContact the frequencyContact to set
     */
    public void setFrequencyContact(int frequencyContact) {
        this.frequencyContact = frequencyContact;
    }

    /**
     * @param height the height to set
     */
    public void setHeight(int height) {
        this.height = height;
    }

    /**
     * @param width the width to set
     */
    public void setWidth(int width) {
        this.width = width;
    }

    /**
```

```
     * @return the numberGenerator
     */
    public MersenneTwister getNumberGenerator() {
        return numberGenerator;
    }

    /**
     * @return the dsurf
     */
    public DisplaySurface getDsurf() {
        return dsurf;
    }

    /**
     * @param numberGenerator the numberGenerator to set
     */
    public void setNumberGenerator(MersenneTwister numberGenerator) {
        this.numberGenerator = numberGenerator;
    }

    /**
     * @param dsurf the dsurf to set
     */
    public void setDsurf(DisplaySurface dsurf) {
        this.dsurf = dsurf;
    }

    /**
     * @return the numberOfInfected
     */
    public int getNumberOfInfected() {
        return numberOfInfected;
    }

    /**
     * @return the epiProgressPlot
     */
    public Plot getEpiProgressPlot() {
        return epiProgressPlot;
    }

    /**
     * @param numberOfInfected the numberOfInfected to set
     */
    public void setNumberOfInfected(int numberOfInfected) {
        this.numberOfInfected = numberOfInfected;
    }

    /**
     * @param epiProgressPlot the epiProgressPlot to set
     */
    public void setEpiProgressPlot(Plot epiProgressPlot) {
        this.epiProgressPlot = epiProgressPlot;
    }

    /**
     * @return the wearingMaskPolicy
     */
    public boolean isWearingMaskPolicy() {
        return wearingMaskPolicy;
    }

    /**
     * @return the wearingMaskParticipation
     */
    public double getWearingMaskParticipation() {
        return wearingMaskParticipation;
    }

    /**
```

```
     * @return the wearingMaskPrevention
     */
    public double getWearingMaskPrevention() {
        return wearingMaskPrevention;
    }

    /**
     * @param wearingMaskPolicy the wearingMaskPolicy to set
     */
    public void setWearingMaskPolicy(boolean wearingMaskPolicy) {
        this.wearingMaskPolicy = wearingMaskPolicy;
    }

    /**
     * @param wearingMaskParticipation the wearingMaskParticipation to set
     */
    public void setWearingMaskParticipation(double wearingMaskParticipation) {
        this.wearingMaskParticipation = wearingMaskParticipation;
    }

    /**
     * @param wearingMaskPrevention the wearingMaskPrevention to set
     */
    public void setWearingMaskPrevention(double wearingMaskPrevention) {
        this.wearingMaskPrevention = wearingMaskPrevention;
    }

}
```

Quellcode der Klasse SarsAgent

```
/**
 * Diese Klasse beinhaltet sämtliche Informationen des Agenten und bietet des
 * Weiteren auch die Möglichkeit die aktuelle Position auf dem Raster zu zeichen.
 * Durch eine farbige Differenzierung kann man den jeweiligen epideischen Zustand
 * des Agenten sofort erkennen.
 *
 * @author Christoph Kaniak
 * @version 1.0
 */

import uchicago.src.sim.space.Object2DTorus;
import uchicago.src.sim.gui.Drawable;
import uchicago.src.sim.gui.SimGraphics;
import uchicago.src.sim.engine.Stepable;
import java.util.*;
import java.awt.*;


public class SarsAgent implements Drawable, Stepable {

    // Attribute des Agenten
    private int id;
    private char epiProgress;
    private int epiProgressDay;
    private int mirrorIdetityCount;
    private MirrorIdentity[] mirrorIdentity;
    private boolean permanentImmunity;
    private double rateContact;
    private boolean wearingMask;
    private SarsAgent[] neighbor;
    private int x;
    private int y;
    private Object2DTorus space;
    private Color color;
```

```
// Definitionen der Farben aller epidemischen Zustände
public final static Color COLOR_SUSCEPTIBLE = Color.GREEN;
public final static Color COLOR_INCUBATED = Color.YELLOW;
public final static Color COLOR_INFECTED = Color.ORANGE;
public final static Color COLOR_DECEASED = Color.BLACK;
public final static Color COLOR_RECOVERED = Color.GRAY;
public final static Color COLOR_IMMUNE = Color.BLUE;

private SarsModel model;

// Konstruktor
public SarsAgent(SarsModel model, Object2DTorus space) {
    this.model = model;
    this.space = space;
    this.epiProgress = 'S';
    this.permanentImmunity = false;
    this.wearingMask = false;
}

/**
 * Aktualisiert die Position des Agenten auf dem Raster
 *
 * @param int x-Position des Agenten
 * @param int y-Position des Agenten
 */
public void setXY(int x, int y) {
    if (space.getObjectAt(this.x, this.y) == this)
        space.putObjectAt(this.x, this.y, null);
    this.x = x;
    this.y = y;
    space.putObjectAt(x, y, this);
}

/**
 * Fügt eine neue MirrorIdentity zu dem Agenten hinzu
 *
 * @param MirrorIdentity neuer Aufenthaltsort des Agenten
 */
public void addMirrorIdentity(MirrorIdentity newMirrorIdentity){
    try{
        int mirrorIdentityCount = this.mirrorIdentity.length;
        MirrorIdentity[] tempMirrorIdentity = new
            MirrorIdentity[mirrorIdentityCount + 1];
        // copy old values
        for (int i = 0; i < this.mirrorIdentity.length; i++){
            tempMirrorIdentity[i] = this.mirrorIdentity[i];
        }
        // insert new mirror identity
        tempMirrorIdentity[mirrorIdentityCount] = newMirrorIdentity;
        this.mirrorIdentity = tempMirrorIdentity;
    } catch (Exception e){
        MirrorIdentity[] tempMirrorIdentity = new MirrorIdentity[1];
        // insert new mirror identity
        tempMirrorIdentity[0] = newMirrorIdentity;
        this.mirrorIdentity = tempMirrorIdentity;
    }
}

/**
 * Fügt eine neue MirrorIdentity zu dem Agenten hinzu
 *
 * @return boolean Liefert true, falls die Kapazität es noch zulässt.
     *         Sonst false.
 */
public boolean needsMoreMirrorIdentity(){
    try{
        if (this.mirrorIdentity.length < this.mirrorIdetityCount)
            return true;
        else
```

```
            return false;
        } catch (Exception e){
            return true;
        }
    }

    /**
     * Initialisiert alle Nachbarn des Agenten
     */
    public void initNeighbors(){
        Vector neighbors = space.getMooreNeighbors(this.x, this.y, false);
        SarsAgent[] tempNeighbor = new SarsAgent[neighbors.size()];
        for (int i = 0; i < neighbors.size(); i++){
            tempNeighbor[i] = (SarsAgent) neighbors.get(i);
        }
        this.neighbor = tempNeighbor;
    }

    /**
     * Zeichnet den Agenten an der neuen Position mit dem aktuellen epidemischen
     * Zustand.
     *
     * @return SimGraphics Zeichenobjekt des Repast-Frameworks
     */
    public void draw(SimGraphics g) {
        switch (this.epiProgress){
        case 'E':
            color = COLOR_INCUBATED; // YELLOW
        break;
        case 'I':
            color = COLOR_INFECTED; // ORANGE
        break;
        case 'R':
            color = COLOR_RECOVERED; // GRAY
        break;
        case 'M':
            color = COLOR_IMMUNE; // >BLUE
        break;
        case 'D':
            color = COLOR_DECEASED; // RED
        break;
        case 'S':
            color = COLOR_SUSCEPTIBLE; // GREEN
        break;
        }
        g.drawFastCircle(color);
    }

    /**
     * Funktion, welche bei jedem Simulationsschritt durchlaufen wird. Agent
        * springt zu seinem nächsten (zufälligen) Aufenthaltsort, versucht dort mit
        * den Nachbarn in Kontakt zu kommen. Dabei wird in Abhängigkeit der Zustände
        * der beteiligten Agenten der Gesamtzustand des Systems beeinflusst.
     *
     */
    public void step(){
        int contactFrequency = this.model.getFrequencyContact();
        int nextId;
        int maxLoopCount = 20;
        while (contactFrequency > 0 &&
                    this.mirrorIdentity != null &&
                    maxLoopCount > 0){

            // Suche nächsten Aufenthaltsort auf
            if (this.mirrorIdentity.length > 1)
                nextId = this.model.getNextIntFromTo(0,
                    this.mirrorIdentity.length-1);
            else nextId = 0;
            MirrorIdentity mi = this.mirrorIdentity[nextId];
            this.setXY(mi.getX(), mi.getY());
```

```
            // Ermittle alle Nachbaragenten
            initNeighbors();
            for (int i = 0; i < this.neighbor.length; i++){
                SarsAgent neighborAgent = this.neighbor[i];

                // Kommt Kontakt mit Nachbarn zustande?
                if(this.model.getNextDoubleFromTo(0.0, 1.0) <=
                    this.rateContact){

                    // Ist Nachbar infiziert?
                    if (this.epiProgress == 'S' &&
                        neighborAgent.epiProgress == 'I'){

                        // Maskentragepflicht verordnet?
                        if (this.model.isWearingMaskPolicy()){

                            // Sowohl Agent als auch Nachbar tragen Maske
                            if (this.isWearingMask() && neighborAgent.isWearingMask())
{
                                if (this.model.getNextDoubleFromTo(0.0, 1.0) >
                                    this.model.getWearingMaskPrevention() &&
                                    this.model.getNextDoubleFromTo(0.0, 1.0) >
                                    this.model.getWearingMaskPrevention() &&
                                    this.model.getNextDoubleFromTo(0.0, 1.0) <=
                                    this.model.getRateInfection()){
                                    this.epiProgress = 'E';
                                }
                            }

                            // Nur Agent trägt Maske
                            else if (this.isWearingMask() &&
                                !neighborAgent.isWearingMask()) {
                                if (this.model.getNextDoubleFromTo(0.0, 1.0) >
                                    this.model.getWearingMaskPrevention() &&
                                    this.model.getNextDoubleFromTo(0.0, 1.0) <=
                                    this.model.getRateInfection()){
                                    this.epiProgress = 'E';
                                }
                            }

                            // Nur Nachbar trägt Maske
                            else if (!this.isWearingMask() &&
                                neighborAgent.isWearingMask()) {
                                if (this.model.getNextDoubleFromTo(0.0, 1.0) >
                                    this.model.getWearingMaskPrevention() &&
                                    this.model.getNextDoubleFromTo(0.0, 1.0) <=
                                    this.model.getRateInfection()){
                                    this.epiProgress = 'E';
                                }
                            }

                            // Keiner trägt eine Maske
                            else if (this.model.getNextDoubleFromTo(0.0, 1.0) <=
                                this.model.getRateInfection()){
                                this.epiProgress = 'E';
                            }
                        } else if (this.model.getNextDoubleFromTo(0.0, 1.0) <=
                            this.model.getRateInfection()){
                            this.epiProgress = 'E';
                        }
                    }
                    contactFrequency--;
                }
            }
            maxLoopCount--;
        }

        // Bearbeitung des epidemischen Zustands
        if (this.epiProgress != 'S' && this.epiProgress != 'D')
```

```
            this.epiProgressDay++;

        switch (this.epiProgress){
            case 'E':
                if (this.epiProgressDay > this.model.getAvgPeriodIncubation()){
                    this.epiProgress = 'I';
                    this.epiProgressDay = 0;
                }
            break;
            case 'I':
                if (this.epiProgressDay > this.model.getAvgPeriodInfectious()){
                    double d = this.model.getNextDoubleFromTo(0.0, 1.0);
                    if (d <= this.model.getRateDeath())
                        this.epiProgress = 'D';
                    else
                        this.epiProgress = 'R';
                    this.epiProgressDay = 0;
                }
            break;
            case 'R':
                if (this.epiProgressDay > this.model.getAvgPeriodRecovered()){
                    this.epiProgress = 'M';
                    this.epiProgressDay = 0;
                }
            break;
            case 'M':
                if (!this.isPermanentImmunity() &&
                    this.epiProgressDay > this.model.getAvgPeriodImmune()){
                    this.epiProgress = 'S';
                    this.epiProgressDay = 0;
                }
            break;
        }
    }

////////////////
// GETTER- AND SETTER FUNCTIONS
////////////////

    public Color getColor(){
        return color;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    /**
     * @return the id
     */
    public int getId() {
        return id;
    }

    /**
     * @return the epiProgress
     */
    public char getEpiProgress() {
        return epiProgress;
    }

    /**
     * @return the epiProgressDay
     */
    public int getEpiProgressDay() {
        return epiProgressDay;
```

```
}

/**
 * @return the mirrorIdetityCount
 */
public int getMirrorIdetityCount() {
    return mirrorIdetityCount;
}

/**
 * @return the permanentImmunity
 */
public boolean isPermanentImmunity() {
    return permanentImmunity;
}

/**
 * @return the rateContact
 */
public double getRateContact() {
    return rateContact;
}

/**
 * @return the wearingMask
 */
public boolean isWearingMask() {
    return wearingMask;
}

/**
 * @return the model
 */
public SarsModel getModel() {
    return model;
}

/**
 * @param id the id to set
 */
public void setId(int id) {
    this.id = id;
}

/**
 * @param epiProgress the epiProgress to set
 */
public void setEpiProgress(char epiProgress) {
    this.epiProgress = epiProgress;
}

/**
 * @param epiProgressDay the epiProgressDay to set
 */
public void setEpiProgressDay(int epiProgressDay) {
    this.epiProgressDay = epiProgressDay;
}

/**
 * @param mirrorIdetityCount the mirrorIdetityCount to set
 */
public void setMirrorIdetityCount(int mirrorIdetityCount) {
    this.mirrorIdetityCount = mirrorIdetityCount;
}

/**
 * @param permanentImmunity the permanentImmunity to set
 */
public void setPermanentImmunity(boolean permanentImmunity) {
    this.permanentImmunity = permanentImmunity;
```

```
    }

    /**
     * @param rateContact the rateContact to set
     */
    public void setRateContact(double rateContact) {
        this.rateContact = rateContact;
    }

    /**
     * @param wearingMask the wearingMask to set
     */
    public void setWearingMask(boolean wearingMask) {
        this.wearingMask = wearingMask;
    }

    /**
     * @param x the x to set
     */
    public void setX(int x) {
        this.x = x;
    }

    /**
     * @param y the y to set
     */
    public void setY(int y) {
        this.y = y;
    }

    public void setColor(Color color) {
        this.color = color;
    }

    /**
     * @param model the model to set
     */
    public void setModel(SarsModel model) {
        this.model = model;
    }

    /**
     * @return the mirrorIdentity
     */
    public MirrorIdentity[] getMirrorIdentity() {
        return mirrorIdentity;
    }

    /**
     * @param mirrorIdentity the mirrorIdentity to set
     */
    public void setMirrorIdentity(MirrorIdentity[] mirrorIdentity) {
        this.mirrorIdentity = mirrorIdentity;
    }

    /**
     * @return the neighbor
     */
    public SarsAgent[] getNeighbor() {
        return neighbor;
    }

    /**
     * @param neighbor the neighbor to set
     */
    public void setNeighbor(SarsAgent[] neighbor) {
        this.neighbor = neighbor;
    }

}
```

Quellcode der Klasse MirrorIdentity

```
/**
 * Diese Klasse speichert Informationen über den möglichen Aufenthaltsort der
 * Agenten. Neben der Position auf dem Raster wird weiters noch festgehalten ob
 * dieser Ort das Zuhause des Agenten ist oder ob es überhaupt verfügbar ist.
 *
 * @author Christoph Kaniak
 * @version 1.0
 */

public class MirrorIdentity {

    // Attribute des Aufenthaltsortes
    private int id;
    private boolean root;
    private boolean suspend;
    private int x;
    private int y;

    // Parameterloser Konstruktor
    public MirrorIdentity() {
    }

    // Konstruktor
    public MirrorIdentity(int x, int y) {
        this.x = x;
        this.y = y;
        this.root = false;
        this.suspend = false;
    }

/////////////////
// GETTER- AND SETTER FUNCTIONS
/////////////////
    /**
     * @return the x
     */
    public int getX() {
        return x;
    }

    /**
     * @return the y
     */
    public int getY() {
        return y;
    }

    /**
     * @param x the x to set
     */
    public void setX(int x) {
        this.x = x;
    }

    /**
     * @param y the y to set
     */
    public void setY(int y) {
        this.y = y;
    }

    /**
```

```
     * @return the id
     */
    public int getId() {
        return id;
    }

    /**
     * @return the root
     */
    public boolean isRoot() {
        return root;
    }

    /**
     * @return the suspend
     */
    public boolean isSuspend() {
        return suspend;
    }

    /**
     * @param id the id to set
     */
    public void setId(int id) {
        this.id = id;
    }

    /**
     * @param root the root to set
     */
    public void setRoot(boolean root) {
        this.root = root;
    }

    /**
     * @param suspend the suspend to set
     */
    public void setSuspend(boolean suspend) {
        this.suspend = suspend;
    }

}
```