FAKULTÄT FÜR !NFORMATIK

Simulationsumgebungen als didaktische Instrumente

MASTERARBEIT

zur Erlangung des akademischen Grades

Magister der Sozial- und Wirtschaftswissenschaften

im Rahmen des Studiums

Informatikmanagement

eingereicht von

Jürgen Pfeffer, bakk. techn.
Matrikelnummer 9626384

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuer: O.Univ.Prof. i.R. Dipl.-Ing. Dr.techn. Peter Fleissner

Wien, 15.11.2008

(Unterschrift Verfasser) (Unterschrift Betreuer)

Kurzfassung

Die zentrale Fragestellung der vorliegenden Arbeit ist der mögliche Einsatz von Simulationsumgebungen im didaktischen Kontext. Als Simulationsumgebungen werden dabei inhaltsneutrale, computerbasierte Tools verstanden, in denen komplexe, reale Systeme konstruiert und simuliert werden können. Durch den Vorgang der Nachbildung eines mentalen Modells in eine mathematische Struktur und ihrer Simulation in Simulationsumgebungen entsteht Erkenntnis über das zugrundeliegende reale System. Der theoretische Teil der Arbeit gibt einerseits eine Einführung in das computerunterstützte Lernen, wobei den verschiedenen Lerntheorien ein wesentlicher Teil gewidmet ist. Andererseits sollen eine Einführung in die Simulation als wissenschaftliche Methode sowie eine Vorstellung unterschiedlicher Simulationsparadigmen (ereignisorientierte Simulationen, System Dynamics, Multiagentensimulation) gegeben werden. Der praktische Teil der Arbeit untersucht konkrete Simulationsumgebungen auf ihre Tauglichkeit als didaktisches Instrument. In der abschließenden Diskussion finden sich Anmerkungen zum computerbasiertem Lernen sowie Hinweise für Lehrende, die den Einsatz von Simulationsumgebungen im Unterricht in Erwägung ziehen.

Abstract

The basic research question of this thesis is the potential adoption of simulation environments in a learning context. Simulation environments are topic neutral computer based tools for constructing and simulation complex real systems. The process of mapping a mental model into a mathematical structure and the simulation of that structure within a simulation environment lead to knowledge about the underlying real system. The theoretical part of the thesis gives an introduction into computer based learning. The main part focuses on various learning theories. On the other hand simulation as a scientific method and specific simulation paradigms (event based simulation, System Dynamics, multi agent simulation) will be introduced. The practical part of the thesis analyses specific simulation environments for their suitability as a didactical instrument. The

closing discussion gives comments on the topic of computer based learning and contains references for teachers who want to use simulation environments in education.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

The only source of
knowledge is experience.

Albert Einstein

Lernen ist nach Zimbardo und Gerrig „ein Prozess, der in einer relativ konsistenten Änderung des Verhaltens oder des Verhaltenspotentials resultiert, und basiert auf Erfahrung.“[1] Erfahrung ist demnach und auch nach vielen anderen Quellen der zentrale Schritt zum Lernen. Umso erstaunlicher ist die Tatsache, dass Bestrebungen im Bereich computerunterstütztes Lernen nach wie vor in eine stark behavioristische Richtung gehen. Klein portionierte Lerneinheiten, die den Lernenden „antrainiert“ werden, erinnern mehr an Konditionierungsübungen denn an modernes Lernen. Simulationen sowie der Einsatz von Simulationsumgebungen im didaktischen Kontext verfolgen dagegen ganz andere Ansätze.

Simulationen sind Nachbildungen von Abläufen in realen und meist komplexen Systemen. Simulationsbasiertes Lernen stellt als konstruktivistische Lernmethode einen Teilbereich des computerunterstützten Lernens dar und findet mittels Computerprogrammen statt, in denen BenutzerInnen „Phänomene oder Aktivitäten“ modellieren und „die dafür vorgesehen sind, dass die Nutzer durch Interaktionen mit ihnen etwas über diese Phänomene und Aktivitäten lernen.“ (Rieber, 2005). Simulationsumgebungen bieten Lernenden die Möglichkeit, durch Nachbildung eines realen Systems in ein Modell und durch die darauffolgende Simulation dieses Modells, Erkenntnisse über das zugrundeliegende reale System zu gewinnen.

Die vorliegende Diplomarbeit bereitet im theoretischen Teil die beiden Themenbereiche Simulation und computerunterstütztes Lernen auf und beschäftigt sich im praktischen Teil mit der Frage der Tauglichkeit konkreter Simulationsumgebungen als didaktische Instrumente.

Im Anschluss an diese Einleitung dient Kapitel 2 der Einführung in das computerunterstützte Lernen. Nach einleitenden Definitionen der Begrifflichkeiten der Didaktik und des Lernens ist ein wesentlicher Teil des Kapitels den verschiedenen Lerntheorien gewidmet. Im Weiteren gibt es kurze Ausführungen, die den Bogen von den ersten Lernmaschinen bis zum aktuellen Konzept des Blended Learning spannen sollen. Anschließend finden sich Einführungen und Definitionen sowie Aspekte des Lernens in Spielen, Simulationen und Mikrowelten.

Kapitel 3 beschäftigt sich mit der Simulation als Methode, beginnend mit Definitionen aus dem Bereich Modellbildung und Simulation. Als konkrete unterschiedliche Sparten der Simulation werden System Dynamics, ereignisorientierte Simulation sowie agentenbasierte Simulation vorgestellt. Abschließend werden allgemeine Hinweise zu Modellkonstruktion und Simulation gegeben und die Simulation aus der Perspektive eines virtuellen Experimentes betrachtet.

In den praktischen Teil wird in Kapitel 4 durch einen Kriterienkatalog eingeleitet, nach welchem die zu testenden Simulationsumgebungen evaluiert werden.

Die anschließende Besprechung der verschiedenen Simulationsumgebungen im Kapitel 5 soll einerseits eine Beschreibung derselben bieten und andererseits einen Eindruck vermitteln, ob sie für den Einsatz im didaktischen Kontext geeignet sind, und wenn ja, für welche Zielgruppe.

Die abschließende Zusammenfassung in Kapitel 6 gibt didaktische Anmerkungen zu Simulationsumgebungen und eine Zusammenfassung der Ergebnisse des Autors über den Einsatz von Simulationsumgebungen als didaktische Instrumente.

Im Fokus der vorliegenden Arbeit steht keine spezielle Lernsoftware, sondern die Bewertung der Eignung von per se inhaltsneutralen Simulationsumgebungen für

didaktische Zwecke. Die Arbeit soll für all jene, die Simulation als didaktisches Instrument einsetzen wollen, eine argumentatorische Unterstützung für diesen Einstieg sein und gleichzeitig konkrete Möglichkeiten der praktischen Anwendung skizzieren. Die Arbeit versteht sich auch als Plädoyer gegen den Einsatz des Computers als bloße Konditionierungsmaschine und für die Anwendung und Entwicklung von computerunterstützten Instrumenten in der Lehre, die unter Ausschöpfung der vielfältigen neuen technischen Möglichkeiten entstehen und dabei die tatsächlichen menschlichen Fähigkeiten des Lernens zentral berücksichtigen.

2 Computerunterstütztes Lernen

I hear and I forget.
I see and I remember.
I do and I understand.

Konfuzius

Die rasante Verbreitung von Computer und Internet in den 1990-er Jahren führten zu einer Durchdringung der unterschiedlichsten Lebensbereiche mit diesen Medien. Das Lernen, sowohl das institutionelle in Schulen und Universitäten als auch das individuelle, waren schnell geprägt von den Möglichkeiten des Einsatzes computerunterstützter Instrumente im didaktischen Kontext. Die Vorstellung, dass Lernen automationsunterstützt ablaufen kann, ist jedoch schon viel älter. Die Entwicklung der ersten Lernmaschinen reicht in das 19. Jahrhundert zurück. Das „programmierte Lernen“ zur Mitte des 20. Jahrhunderts unterteilte den zu lernenden Inhalt in kleine Stücke und verlief zuerst nach einem fixen Ablaufplan. Später waren auch, je nach Lernerfolg, Verzweigungen im Lernablauf möglich.

Eine moderne Bezeichnung von computerunterstütztem Lernen ist E-Learning. Definitionen von E-Learning finden sich in der Literatur unterschiedliche. Erik Minass gibt in seinem Buch über die Dimensionen des E-Learnings [2] an, dass folgende drei Aspekte in allen Definitionen des Begriffs E-Learning vorkommen:

- Systeme, die Lernen ermöglichen und Lerninhalte darbieten
- Örtliche Unabhängigkeit
- Individuelles und gruppenbezogenes Lernen

Der Fokus der vorliegenden Arbeit liegt aber nicht auf Distanzunterricht, sondern generell auf dem Einsatz von Computern im Kontext des Lernens und im speziellen auf der Verwendung von Simulationsumgebungen als didaktische Instrumente. Aus diesem Grund wird in weiterer Folge der Begriff computerunterstütztes Lernen verwendet.

Dieses Kapitel beginnt mit einer Einführung in die Begrifflichkeiten der Didaktik und des Lernens sowie einer Auflistung didaktischer Prinzipien. Der Großteil dieses Kapitels beschäftigt sich mit Theorien des Lernens und des Wissens. Kleinere Einführungen werden in die Geschichte der Lernmaschinen sowie in Definitionen und Aspekte des Lernens in Spielen, Simulationen und Mikrowelten gegeben.

2.1 Einführung in das Lernen

Lernen ist nach Zimbardo und Gerrig [1] „ein Prozess, der in einer relativ konsistenten Änderung des Verhaltens oder des Verhaltenspotentials resultiert, und basiert auf Erfahrung.“ Da das Lernen kein beobachtbarer Prozess ist, muss die Veränderung des Verhaltens in Form von Leistung (das was sich im offenen Verhalten ausdrücken lässt) oder die Veränderung des Verhaltenspotentials in Form von Verständnis, als Ergebnis des Lernens beschrieben werden. Die Konsistenz der Veränderung bedeutet, dass die Veränderung des Verhaltens oder des Verhaltenspotentials eine dauerhafte oder zumindest längerfristige ist. Die Eigenschaft, dass „Lernen ausschließlich durch Erfahrung stattfindet“ [1], unterscheidet Lernen auch vom biologischen und genetischen Verhalten (Reflexe, Prägungen, Instinkte, Reifung).

Didaktik stammt vom griechischen „didaskein“ ab und steht für unterrichten bzw. lehren. Nach Raithel et al. [3] unterscheidet man zwischen Didaktik im engeren und im weiteren Sinn. Im weiteren Sinn steht Didaktik für die Theorie des Lernens und Lehrens in jeglichem Kontext. Im engeren Sinn ausschließlich für die Erforschung des schulischen Lernens.

Raithel et al. geben in ihrem Lehrbuch zur Pädagogik [3] die fünf Prinzipien der Didaktik von Kaiser und Kaiser, Studienbuch Pädagogik (2001) wieder:

Prinzip der Situationsbezogenheit

Lernen soll im Endeffekt zur besseren Bewältigung von Situationen egal welcher Art beitragen. Dem Lernvorgang sollte aus dieser Perspektive Rechnung getragen werden, indem das Gelernte auf die möglichen Situationen der Anwendung bezogen wird.

Prinzip der Handlungsorientierung

Die Bewältigung der Situationen aus dem ersten Punkt findet durch Handlung statt. Das Lernen soll also Unterstützung und Anleitung für Handlungen und Entscheidungen bieten.

Prinzip der Wissenschaftsorientierung

Im Lernprozess wird Wissen vermittelt. Dieses Wissen ist das Ergebnis von wissenschaftlichen Prozessen. Die Lerninhalte sollen demnach auf dem aktuellen Stand der Wissenschaft ausgerichtet sein und das Vorgehen an wissenschaftlichen Methoden orientiert sein.

Prinzip des Exemplarischen

Das zu Lernende kann immer nur ein Ausschnitt der gesamten Wissensmenge sein. Die Reduktion auf exemplarische Sachverhalte ist also zwingend und ein Grundprinzip der Didaktik.

Prinzip der Struktur

Das einzeln zu Lernende muss vertretbar als exemplarisch gelten und sich in ein Ganzes einordnen. Dem Lernenden muss das Wissen gegliedert und geordnet vermittelt werden.

2.2 Theorien des Lernens und Wissens

Lerntheorien sind Thesen darüber, wie Lernen funktioniert. Im Kern lassen sich drei verschiedene Lerntheorien aufzählen: Behaviorismus, Kognitivismus und Konstruktivismus. Jede Theorie des Lernens basiert explizit oder implizit auf mindestens einer dieser drei Paradigmen. Diese Paradigmen widerspiegeln auch die wichtigsten Ansätze des 20. Jahrhunderts aus erkenntnistheoretischer Sicht [4]. Eine Gegenüberstellung unterschiedlicher Aspekte dieser drei Lernparadigmen gibt Tabelle 2.1.

	Behaviorismus	**Kognitivismus**	**Konstruktivismus**
Paradigma	Stimulus-Response	Problemlösung	Konstruktion
Gehirn ist ein…	passiver Behälter	informationsverarbeitendes Gerät	informationell geschlossenes System
Wissen wird…	abgelagert	verarbeitet	konstruiert
Wissen ist…	eine Input-Outputrelation	ein interner Verarbeitungsprozess	mit einer Situation operieren können
Lernziel	richtige Antworten	richtige Methoden zur Antwortfindung	komplexe Situationen bewältigen
Lehrer ist…	Autorität	Tutor	Coach, Trainer
AutorInnen	Pawlow, Skinner, Thorndike	Ausubel, Brunner, Gagné, Bower/Hilgard	Maturana/Varela

Tabelle 2.1: Übersicht der Lernparadigmen nach [3] und [4]

Als kognitivistische Theorie und dennoch als eigenständiger Abschnitt im Folgenden findet die Gestalttheorie Berücksichtigung. Neben den drei „klassischen“ Lerntheorien sollen zwei weitere aktuelle lerntheoretische Modelle betrachtet werden, die für sich den Anspruch erheben, als eigenständige Lerntheorien zu gelten. Dies ist einerseits die von David Kolb definierte erfahrungsbasierte Lerntheorie und der von George Siemens entwickelte Konnektivismus, der sich aus Prinzipien der Chaos-, Netzwerk-, Komplexitäts- und Selbstorganisations-Theorien zusammensetzt. Vorstellungen darüber, in welcher Form Wissen kogni-

tiv gespeichert wird, soll der Abschnitt zu Kognitiven Wissensrepräsentationen geben.

2.2.1 Behaviorismus

VertreterInnen des Behaviorismus sind der Meinung, dass ausschließlich beobachtbares Verhalten analysiert werden kann. Der einzelne Organismus, egal ob Tier oder Mensch, wird dabei als eine Black-Box empfunden, deren inneres Funktionieren für die BeobachterInnen ungeklärt bleibt. Das beobachtbare Verhalten wird von Reizen und Verstärkungen gesteuert [3]. Ein bestimmter Reiz (Stimulus) löst dabei eine bestimmte Reaktion im Verhalten (Response) aus. Diese Reaktion erhält aus der Umwelt des Individuums eine Rückmeldung und erzeugt positive oder negative Verstärkungen. Dadurch entstehen Verhaltensreaktionen. Lernen passiert somit durch Konditionierung. Abbildung 2.1 zeigt das Black-Box Modell aus [4].

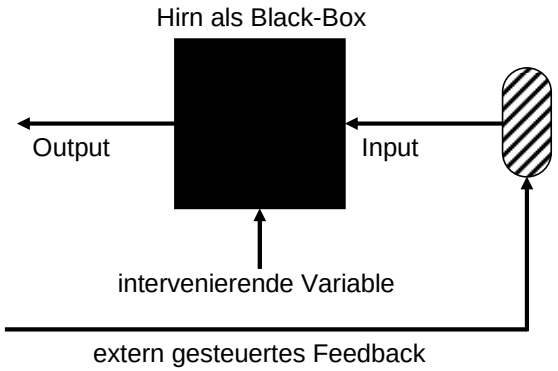

Abbildung 2.1: Schematisches Lernmodell des Behaviorismus nach [4]

Das Ziel des Lernens ist aus behavioristischer Sicht die Aneignung von Faktenwissen [5]. Dies wird erreicht, indem Lernende Reize in Form von Fragen oder Aufgaben präsentiert bekommen. Das richtige Antwortverhalten wird antrainiert und verstärkt. Das zu Lernende ist aus Sicht der BehavioristInnen die Wahrheit, wodurch VertreterInnen des Behaviorismus auch als ObjektivistInnen bezeichnet werden können.

Ein zentraler Aspekt der behavioristischen Lernforschung ist, dass zur Theoriebildung Versuche mit Tieren (z.B. Ratten, Hunde) eine zentrale Rolle spielen und dass aus diesen Lern- und Verhaltensexperimenten mit Tieren Schlussfolgerungen auf den Menschen gezogen werden. Berühmtheit erlangten zum Beispiel die Versuche mit Hunden, die der russische Mediziner und Physiologe Iwan Petrowitsch Pawlow (1849-1936) durchgeführt hat. Pawlow nutzte die Eigenschaft, dass Hunde beim Anblick von Futter Speichel absondern und verband die Futtergabe mit einem gleichzeitig ertönenden Glockenton. Pawlows Hunde wurden auf den Glockenton konditioniert und sonderten in weiterer Folge auch Speichel ab, wenn nur der Glockenton zu hören, aber kein Futter für sie zu sehen war.

Der Behaviorismus als Lerntheorie ist in der Diskussion der letzten Jahre auf starke Kritik gestoßen (z.B. [6]). Ein Argument gegen den Behaviorismus stellt zum Beispiel die Erkenntnis dar, dass Menschen, aber auch zum Teil Tiere Probleme „durch Einsicht“ lösen (vgl. Gestaltpsychologie im Abschnitt 2.2.4). Dennoch hat der Behaviorismus nach Baumgartner und Payr [4] dort eine berechtigte Grundlage, wo es um das Trainieren von körperlichen Fertigkeiten geht, wie Maschinschreiben, Jonglieren oder Klavierspielen.

Im Zusammenhang mit dem Lernen hat sich in weiterer Folge eine Vielzahl anderer Theorien entwickelt (vgl. die folgenden Abschnitte), dennoch spielt der Behaviorismus in den meisten Bereichen des computerunterstützten Lernens eine zentrale Rolle. Baumgartner und Payr [4] nennen diese Art von Lernsoftware „Drill & Test“-Software.

2.2.2 Kognitivismus

Unter Kognition versteht man sämtliche „Prozesse des Wissens“ [1]. Als kognitive Fähigkeiten können das Denken, die Erinnerung, die Verwendung der Sprache, die Kreativität, die Wahrnehmung und einige mehr bezeichnet werden. Bei VertreterInnen des Kognitivismus wird der Lernende als Individuum definiert, „das äußere Reize aktiv und selbständig verarbeitet und deshalb nicht durch äußere Stimuli steuerbar ist“ [3]. Je nach persönlichem Entwicklungsstand verar-

beiten Lernende Eindrücke. Dieses Verarbeiten geschieht in einer inneren kognitivistischen Struktur. Die Vorstellung dieser Struktur entspricht dem Computermodell von Bower und Hilgard (vgl. Abbildung 2.2). Der sensorische Speicher ist Teil des Zentralnervensystems und nimmt Reize aus der Außenwelt als Nervenimpulse auf. Dieses Bild der Sinneswahrnehmung ist im sensorischen Speicher jedoch nur für ca. 0,3 Sekunden aufgebaut, danach geht der Großteil dieser Stimuli wieder verloren. Nur ein geringer Teil der Sinneswahrnehmungen wird in das Kurzzeitgedächtnis übernommen und kann in diesem für ca. 10 Sekunden erhalten bleiben. Durch innere Wiederholung kann diese Frist verlängert werden, z.B. durch das permanente Wiederholen einer zu merkenden Telefonnummer. Der Lernvorgang von Informationen aus dem Kurzzeitgedächtnis geschieht durch einen Transfer in das Langzeitgedächtnis. Dieser Transfer bedeutet „die Integration in das bestehende Wissen“ [7] durch Anknüpfen des neuen Wissens an bestehendes.

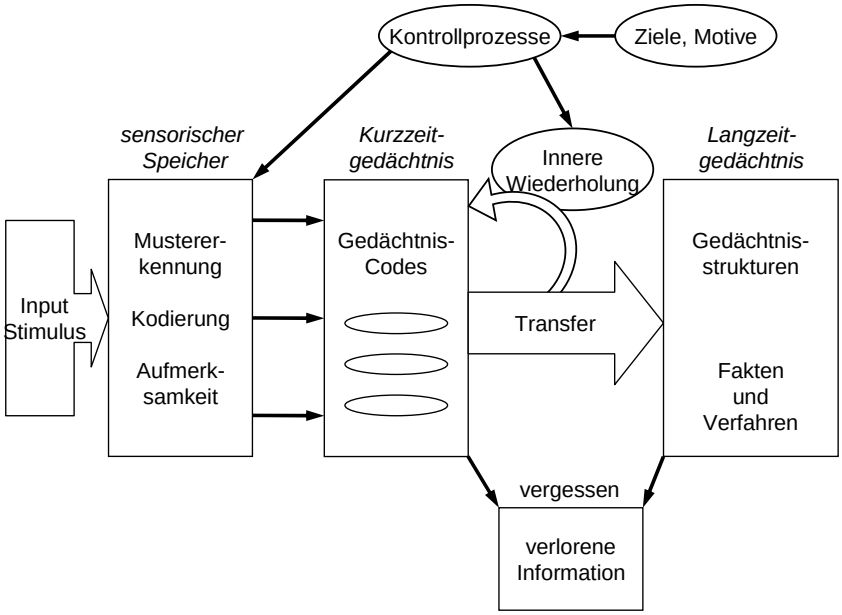

Abbildung 2.2: Computermodell von Bower/Hilgard, aus [3]

Der Kognitivismus spielt für die Entwicklung von Anwendungen im Bereich der künstlichen Intelligenz eine große Rolle, da im Sinne der Modellkonstruktion (vgl. Kapitel 3) ein Verständnis des realen Systems, in dem Fall der kognitiven Vor-

gänge, Voraussetzung für die technische Reproduktion darstellt. Ein Kritikpunkt am Kognitivismus ist, dass sich bestimmtes Wissen und bestimmte Fähigkeiten nicht sprachlich ausformulieren lassen [4]. Als Beispiel für „sprachloses Wissen“ wird die menschliche Fähigkeit des Erkennens von Gesichtern genannt. So sind Menschen nicht in der Lage, Gesichter verbal ausreichend zu beschreiben, noch zu beschreiben, wie sie diese identifizieren, dennoch gelingt eine Identifikation sehr gut, auch wenn das Gesicht gravierende Veränderungen erfahren hat (z.B. altern, Bartwuchs). Die Unfähigkeit der Beschreibung der kognitiven Vorgänge in diesem Beispielt geht einher mit den Schwierigkeiten der technischen Implementierung von automatischen Gesichtserkennungssystemen. Eventuell führen Erkenntnisse aus der Dual Coding Theorie (vgl. Abschnitt 2.2.3) hier zu neuen Antworten.

2.2.3 Kognitive Wissensrepräsentationen

Durch den Transfer in das Langzeitgedächtnis wird Wissen in Gedächtnisstrukturen abgelegt. Vorstellungen darüber, in welcher Form dieses erlernte Wissen kognitiv langfristig gespeichert wird, gibt es bei VertreterInnen des Kognitivismus einige:

Prototypen und Schemata/Skripts

Eine Grundfunktion des Gedächtnisses ist es, „ähnliche Erfahrungen zusammenzufassen, um Muster in der Interaktion mit der Umwelt aufzudecken. [1]“ Die unzähligen Ereignisse und Sinneseindrücke, denen der Mensch täglich ausgesetzt ist, müssen stark vereinfacht werden, damit sie kognitiv verarbeitbar werden. Durch das Kategorisieren und Zusammenfassen dieser Eindrücke entstehen Prototypen. Als Prototyp kann „das repräsentativste Exemplar einer Kategorie“ [1] definiert werden. Die prototypische Vorstellung eines Hauses stellt demnach den Durchschnitt aller gesammelten Eindrücke eines Individuums aller verschieden betrachteten Häuser dar.

Eine andere Art der kognitiven Vereinfachung wird durch Schemata erzielt. Schemata sind „allgemeine konzeptuelle Rahmen oder Cluster von Wissen“ [1].

Die von Frederic Bartlett (1886-1969) begründete Schematheorie beschreibt, wie Wissen über bestimmte Bereiche zu Schemata gruppiert wird. Mit einem Restaurantbesuch sind zum Beispiel verschiedene Handlungen verbunden (Restaurant betreten, Tisch suchen, bestellen, warten, essen, bezahlen, Restaurant verlassen). Dieses Schema wird unabhängig vom tatsächlich besuchten Restaurant aktiviert und kann in jedem Punkt in weitere Schemata aufgeteilt werden. Aufgrund dieser Ablaufanordnungen werden Schemata ähnlich wie in Computerprogrammen oder bei der Filmproduktion auch als Skripts bezeichnet.

Semantische Netze

Ross Quillian entwickelte das Konzept der semantischen Netze [8], das der Vorstellung folgt, dass einzelne Begriffe kognitiv in einem Netzwerk gespeichert werden. Einzelne Begriffe wie zum Beispiel „Baum“ lösen Assoziationen zu anderen Begriffen wie zum Beispiel „Blatt“, „Tanne“ oder „grün“ aus. Durch diese Assoziationen werden unterschiedliche Begriffe kognitiv miteinander verbunden (vgl. Abbildung 2.3).

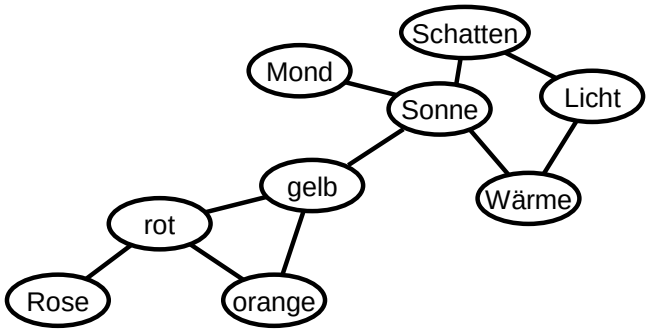

Abbildung 2.3: Semantisches Netz um den Begriff „gelb“

Mentale Modelle

Eine besondere Bedeutung im Zusammenhang mit der Interaktion zwischen Menschen und Computer spielt das Konzept der mentalen Modelle. Ein mentales Modell ist „eine gedankliche Konstruktion (Vorstellung) des interessierenden Sachverhaltes, die uns beim Handeln anleitet [4]“. Dabei handelt es sich um ver-

einfachte, weniger komplexe und oft auch verfälschte Abbildungen des realen Systems. Wenn Lernende mit Simulationsumgebungen experimentieren, handeln sie nach dieser Theorie nicht im Rahmen des realen abgebildeten Systems, sondern im Rahmen ihrer Vorstellungen über die Funktionsweise dieses Systems. Ein zentraler Aspekt des simulationsbasierten Lernens stellt also auch die Veränderung des mentalen Modells im Lernprozess dar, das sich im positiven Fall an das reale Modell anpasst.

Dual Coding

Im Bereich des computerunterstützten Lernens bietet die Dual Coding Theorie eine Argumentationsunterstützung für die Verwendung multimedialer Elemente an. Die duale Kodierungstheorie von Allan Paivio unterteilt die Kognition in zwei Systeme, in ein verbales (semantisches) und ein nonverbales, das als ein visuelles System verstanden werden kann [9]. Während das Lesen und Hören von Begriffen das verbale System aktiviert und so erlerntes Wissen auch dort gespeichert wird, sprechen optische Sinneseindrücke das nonverbale System an. Der Vorteil von Bildern, Animationen und Simulationen ist aus didaktischer Perspektive, dass es damit möglich ist, Wissen in beiden Systemen verbal und visuell abzuspeichern. Durch eine zusätzliche kognitive Verbindung des Gelernten zwischen den beiden Systemen kann eine effizientere und nachhaltigere Aufnahme in das Gedächtnis erfolgen.

2.2.4 Gestalttheorie

Einen weiteren Bereich im Kontext des Wahrnehmens und des Lernens, der auch für multimediales Lernen bedeutend ist, stellt die Gestalttheorie dar. Die Gestalttheorie kann lerntheoretisch als kognitivistischer Ansatz betrachtet werden und wurde 1912 von Max Wertheimer begründet. Als weitere zentrale Vertreter gelten Kurt Koffka, Wolfgang Köhler sowie Kurt Lewin, welcher als der Begründer der Feldtheorie gilt. Sämtliche hier aufgezählten Vertreter der Gestaltpsychologie emigrierten in Folge der nationalsozialistischen Machtergreifung aus Europa in die USA. Als Gestalt wird ein Ganzes definiert, „dessen Einzelheiten in einer be-

stimmten Relation zueinander stehen“ [10]. Nach der Theorie der GestaltpsychologInnen kann die einzelne Wahrnehmung nicht atomar betrachtet werden, sondern muss immer einer ganzheitlichen Betrachtung unterzogen werden. Das Ganze bestimmt demnach, wie die einzelnen Teil wahrgenommen werden (=perzeptuelle Organisation, vgl. [6]). Während in der Vorstellung von VertreterInnen des Behaviorismus ein Individuum in einer neuen Problemsituation sämtliche bisherigen vergleichbaren Erfahrungen anwendet, um zu einer Lösung zu gelangen, entsteht der Lösungsansatz in der Vorstellung der VertreterInnen der Gestaltpsychologie durch eine kognitive Bewertung, Selektion und Strukturierung der wahrgenommenen Reize innerhalb des beobachteten Gesamtkontextes. Erkenntnis geschieht demnach durch Einsicht. Einsicht wird bei höher entwickelten Individuen als „plötzliche Wahrnehmung von Beziehungen zwischen den zentralen Elementen einer Problemsituation“ [10] definiert. Durch diese Einsicht kommt es auch zu einer Generalisierung. Einmal erkannte Lösungsansätze können auf andere Problemstellungen übertragen werden,

Die oben beschriebene Gestalt kann sich sowohl auf die einzelnen Sinne (optische, akustische, taktile, gustatorische und olfaktorische Gestalten) als auch auf kognitive Leistung (z.B. Denk- und Gefühlsgestalten) stützen. Das Erkennen von Gestalten erfolgt nach der Gestalttheorie im Denken zwingend. Der Prozess der Gestaltung vereinfacht die Wahrnehmung aus vielen unterschiedlichen Eindrücken zu einem Gesamtbild. VertreterInnen der Gestalttheorie haben mehrere Dutzend Gestaltgesetze entwickelt, von denen einige hier beschrieben werden:

- **Gesetz der Prägnanz / Gesetz der guten Gestalt**: Eine komplexe Figur wird auf ihre einfache Form zurückgeführt; z.B. wird etwas, das wie ein Kreis ausschaut, als Kreis wahrgenommen.
- **Gesetz der Nähe:** Einzelne Elemente, die nahe beieinander liegen, werden als Einheit wahrgenommen (siehe Abbildung 2.4 links).
- **Gesetz der Ähnlichkeit:** Elemente, die einander in ihrer Form ähnlich sind, werden eher als zusammengehörig wahrgenommen, als einander nicht ähnliche Elemente (siehe Abbildung 2.4 rechts).

- **Gesetz der Kontinuität:** Reize, die aufeinander folgen, werden als zusammengehörig wahrgenommen. Dazu vergleichend kann auch das Phi-Phänomen betrachtet werden, welches eine Bewegung beschreibt, die aus zusammenhängenden stationären Eindrücken erzeugt wird (z.B. Leuchtschrift, Fernsehbild).

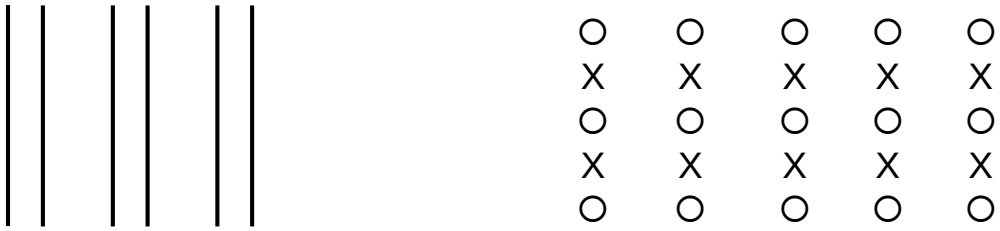

Abbildung 2.4: Gestaltgesetz der Nähe (links) und der Ähnlichkeit (rechts)

Nach Bodenmann et. al [10] findet man gestalttheoretische Elemente im didaktischen Kontext überall dort, „wo die Lehrpersonen darum bemüht sind, die Einsichten der Schülerinnen und Schüler zu fördern, indem der Sinn für das Ganze ebenso wie die Erfassung der Beziehungen zwischen den einzelnen Elementen stimuliert wird.“ Erkenntnisse der Gestaltpsychologie gehen daher zentral in das computerunterstützte Lernen ein und stellen eine wichtige Argumentation für die Sinnhaftigkeit des Einsatzes von Simulationen im didaktischen Kontext dar.

2.2.5 Konstruktivismus

Der Konstruktivismus hat ideengeschichtlich eine lange Tradition und unterschiedliche Ausprägungen (vgl. z.B. [11][12]). Als Vorgänger kann unter anderem Immanuel Kant (1724-1804) gesehen werden, für den die Wirklichkeit, „eine Widerspiegelung unseres menschlichen Erkenntnisapparates, nicht der Wirklichkeit an sich“ darstellt [11]. Der zentrale Gedanke des Konstruktivismus ist, dass Realität nicht objektiv wahrgenommen werden kann (wie dies z.B. von VertreterInnen des Behaviorismus angenommen wird), sondern von den BetrachterInnen konstruiert wird. Realität wird demnach „als eine interaktive Konzeption verstanden, in der Beobachter und Beobachtetes gegenseitig und strukturell miteinander

gekoppelt sind“ [4]. Aus konstruktivistischer Perspektive funktioniert Lernen, indem Menschen ihr Wissen konstruieren. Dies geschieht in Interaktion mit der aktuellen Lebenssituation und dem früher erworbenen Wissen. Aus der Sicht des Konstruktivismus ist das menschliche Gehirn, wie Abbildung 2.5 zeigt, ein geschlossenes System, das zwar energetisch offen, aber informationell geschlossen ist. Was also in der Kognition verarbeitet wird, wird innerhalb des Systems Gehirn selbst erzeugt.

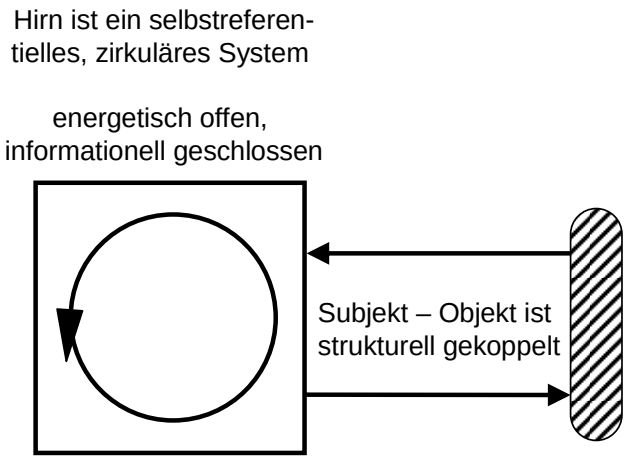

Abbildung 2.5: Schematisches Lernmodell des Konstruktivismus nach [4]

Der zentrale Unterschied zum Kognitivismus ist, dass beim Konstruktivismus nicht „das Lösen bereits präsenter Probleme im Vordergrund steht, sondern das eigenständige Generieren von Problemen“ [4].

Aus dem Konstruktivismus haben sich folgende drei konkrete Lehransätze entwickelt [5]:

- **Cognitive Apprenticeship:** Lernen im Lehrlingsverhältnis, die Lehrerin oder der Lehrer beraten als Coach, ziehen sich aber bei Lernfortschritten immer mehr zurück.
- **Knowledge Communities:** Lernen durch Kommunikation in Wissensgemeinschaften. Die Interaktion mit der sozialen Umwelt beeinflusst die Wissenskonstruktion.

- **Cognitive Tools:** Kognitive Werkzeuge ermöglichen es den Lernenden, durch Modellierung eigene Konzepte zu generieren.

2.2.6 Erfahrungsbasierte Lerntheorie

David Kolb entwickelte in den 1980-er Jahren die erfahrungsbasierte Lerntheorie. Erfahrungen sammeln und über diese Erfahrungen nachdenken steht im Zentrum dieser Theorie [2]. Seine Theorie visualisierte Kolb in einem Lernzyklus (vgl. Abbildung 2.6). Jede Position des Zyklus muss in einem Lernprozess durchlaufen werden.

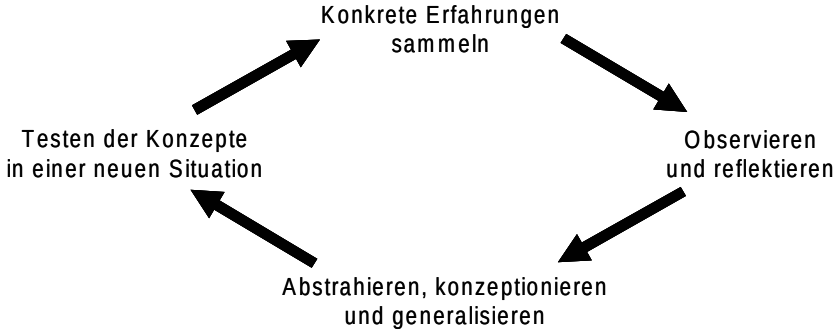

Abbildung 2.6: Lernzyklus von David Kolb

Im Kontext des Lernens mit Simulationsumgebungen spielt dieser Lernzyklus eine zentrale Rolle und beschreibt auch gleichzeitig exakt den richtigen Umgang mit diesem.

2.2.7 Konnektivismus

Der Begriff des Konnektivismus bzw. des Konnektionismus wird im Kontext der Erforschung künstlicher Intelligenz verwendet und beschäftigt sich mit der Informationsverarbeitung in neuronalen Netzen. Diesen liegt die Annahme zugrunde, dass „Informationsverarbeitung durch die Interaktion einer großen Anzahl einfacher Elemente erfolgt, die einander stimulierende und hemmende Signale senden“ [13].

Den Anspruch, mit dem Konnektivismus „eine neue Lerntheorie für das digital vernetzte Zeitalter zu definieren“ [14], stellte im Jahr 2004 der Kanadier George Siemens mit seiner Arbeit „Connectivism: A Learning Theory for the Digital Age”. Siemens argumentiert, dass ausgehend von der Tatsache, dass Wissen sich in immer kürzer werdenden Intervallen verdoppelt und, dass das in Schulen und Hochschulen erworbene Wissen nur mehr ein Schnappschuss des Wissens zu einem bestimmten Zeitpunkt ist, „Bildung in Form von reinem Inhaltskonsum durch die Lernenden schnell irrelevant wird“ [15]. An den dominierenden Lerntheorien Behaviorismus, Kognitivismus, Konstruktivismus kritisiert Siemens, dass diese Wissen als Ziel begreifen, das erreichbar ist. Die zentrale Aussage von Siemens im Gegensatz dazu ist: „The pipe is more important than the content within the pipe“ [15]. Die Verbindungen zwischen unterschiedlichem Wissen und das Auffinden des Wissens ist im Konnektivismus wichtiger als das Wissen selbst, da das Wissen nicht mehr ein abgrenzbarer Bereich ist, sondern einen sich permanent verändernden Prozess darstellt.

Als weitere wichtige Fähigkeiten im Umgang mit immer größer werdenden Wissensmassen nennt Siemens die Fertigkeit, Wissen schnell zu evaluieren. Die Lernenden müssen also effizient entscheiden, welches Wissen für die jeweilige Fragestellung relevant ist und welches nicht.

Die meisten KritikerInnen sprechen dem Konnektivismus die Eigenschaft ab, eine eigene Lerntheorie zu sein. Kerr [16] ist zum Beispiel der Meinung, dass die klassischen Lerntheorien auch für das heutige Lernen, das zunehmend von der Technik dominiert wird, ausreichen. Generell kann festgehalten werden, dass der Konnektivismus aus einer Verbindung von Kognitivismus und Konstruktivismus hervorgegangen ist und, durch die Einbindung der Veränderungen des Wissens und des Lernens und dem Einsatz von Computer und Internet, eine modernisierte und attraktive Variante dieser beiden klassischen Lerntheorien darstellt.

2.3 Methoden des Computerunterstützen Lernens

Jede Form des klassischen und des computerunterstützten Lernens basiert explizit oder implizit auf einer der oben genannten Lerntheorien. Neben einer Einteilung in diese Lerntheorien können computerunterstützte Lernsysteme auch nach der verwendeten Methode eingeteilt werden[5]:

- **Tutorielle Systeme:** älteste Form computerunterstützter Lernsysteme, weitgehend linear organisierte Programme, EntwicklerInnen legen „optimale“ Instruktionsreihenfolge fest, Lernende arbeiten vorgefertigte Sequenzen ab, werden auch als „Drill & Practice-Systeme“ bezeichnet, streng behavioristisch.
- **Adaptive Systeme/Intelligente Tutorielle Systeme (ITS):** Lernsystem diagnostiziert Unterstützungsbedarf der Lernenden und passt sich den Fähigkeiten im Einzelnen an, IST ermöglichen Individualisierung des Lernens, hoher Aufwand der Implementierung meist nicht gerechtfertigt [5].
- **Computerspiele:** Edutainment verbindet Bildung mit Unterhaltung, meist hohe Kosten in der Erstellung, auch Spiele ohne expliziten Bildungsanspruch können Fähigkeiten stärken (z.B. divergentes Denken).
- **Simulationen und Mikrowelten:** interaktiv veränderbares Computermodell eines Systems, unterstützt explorierendes Lernen (vgl. Kapitel 3 Simulation).
- **Kombinierte Ansätze:** Kombination der oben genannten Methoden, seltenes Vorkommen von „reinen“ Typen in modernen Programmen.
- **Werkzeuge und Programmierumgebungen:** keine spezielle Lernsoftware, Nutzung von inhaltsneutralen Umgebungen für didaktische Zwecke. Möglichkeiten gehen meist weit über den Lerneinsatz hinaus, z.B. Simulationsumgebungen.

Diese methodischen Grundtypen können wiederum den Lernparadigmen zugeordnet werden. Diese Zuordnung ist in Tabelle 2.2 dargestellt.

	Behaviorismus	**Kognitivismus**	**Konstruktivismus**
Interaktion	starr vorgegeben	dynamisch in Abhängigkeit des externen Lernmodells	selbstreferentiell, zirkulär, strukturdeterminiert (autonom)
Programmmerkmale	starrer Ablauf, quantitative Zeit- und Antwortstatistik	dynamisch gesteuerter Ablauf, vorgegebene Problemstellungen, Antwortanalyse	dynamisch, komplex vernetzte Systeme, keine vorgegebenen Problemstellungen
Paradigma	Lernmaschine	Künstliche Intelligenz	sozio-technische Umgebung
idealer Softwaretypus	Course-, Teachware, Computer Aided Instruction (CAI)	Tutorensysteme, Computer Based Training (CBT)	Simulationen, Mikrowelten

Tabelle 2.2: Lernparadigmen und Softwaretypologie nach [4]

2.4 Von den Lernmaschinen zum Blended Learning

Halycon Skinner meldete 1866 ein Patent an, das eine erste Lernmaschine beschrieb. Über die Tastatur einer Schreibmaschine musste die Bezeichnung eines Bildes eingegeben werden, das in einem Kasten angezeigt worden war. Herbert Aiken erfand 1911 eine Buchstabiermaschine. Wieder war ein Bild zu sehen, das richtig benannt werden musste. Bei der Lernmaschine von Aiken mussten Kärtchen mit den richtigen Buchstaben in eine Vorrichtung gesteckt werden. In den folgenden Jahrzehnten wurden hunderte Patente für Lernmaschinen angemeldet, bis 1936 ca. 700 [17].

Die ersten Lernmaschinen von Burrhus F. Skinner und James G. Holland in den 30-er Jahren des letzten Jahrhunderts folgten der behavioristischen Lerntheorie und funktionierten nach dem Schema der Konditionierung. Das Prinzip ihrer Lernmaschinen bestand darin, den zu lernenden Stoff in kleine Einheiten zu zerteilen und nach jeder Einheit zum vorher gelesenen Text Fragen zu stellen. Der Lernende konnte die eigene Eingabe im Anschluss mit der richtigen Lösung vergleichen. Ein Aufteilen des zu lernenden Stoffes in kleine Lerneinheiten, die eindeutig abprüfbar sind, ist bis heute die dominierende Vorgehensweise im Bereich des computerunterstützten Lernens.

Die Lernmaschinen von Skinner und Holland können als lineare Lernprogramme bezeichnet werden, da der Ablauf für jeden Lernenden der gleiche war. Die Lernprogramme, die Norman Crowder 1959 entwarf, sahen Verzweigungen vor. Je nachdem, welcher Fehler bei der Eingabe der Antwort gemacht wurde, erhielt der Lernende entsprechenden Rückmeldungen und auch der weitere Fortgang der Lerneinheit wurde entsprechend verzweigt. Der Grundgedanke der Verzweigung ist bis heute in computergestützten Lernumgebungen ein zentrales Element.

Nach weiteren intensiven weltweiten Forschungen und Entwicklungen (vgl. z.B. [17]) in den 1960-er Jahren ließ Ende der 1970-er und Anfang der 1980-er Jahre das Interesse am automatisieren Lernen stark nach. Erst mit der steigenden Anzahl von Computern in privaten Haushalten und vor allem seit der stark ansteigenden Verbreitung des Internets ab Mitte der 1990-er Jahre, erlebte E-Learning, wie es mittlerweile bezeichnet wurde, wieder einen starken Aufschwung.

Das E-Learning, das in vielen Fällen mit Distanzunterricht gleichgesetzt wird, betont andere Aspekte als das traditionelle Lernen in Klassenzimmern und Hörsälen. Im Folgenden finden sich Vorteile und Nachteile des E-Learnings, wie sie unter anderen bei [18], [19], [20] und [21] angeführt werden:

Vorteile des E-Learning:

- Lernen unabhängig von Zeit und Ort
- Selbstbestimmung des Lerntempos
- Einsatz unterschiedlicher Medien kommt unterschiedlichen Lerntypen entgegen

Nachteile des E-Learning:

- fehlender sozialer Kontakt zu KollegInnen
- keine ganzheitliche Kommunikation möglich
- hohe Selbstlernkompetenz erforderlich

Das reine E-Learning konnte aufgrund der hier angeführten Nachteile nicht die ursprünglich angenommen Erwartungen erfüllen. Vor allem die fehlenden sozialen Interaktionen (gemeinsames Lernen, ganzheitliche Kommunikation mit den Vortragenden) sind gegenüber dem traditionellen Unterricht gravierende Nachteile [20]. Aus diesen Gründen wurde das Blended Learning entwickelt. Blended Learning steht für vermischten Unterricht und stellt eine Kombination aus Präsenzunterricht und E-Learning dar. Eine mögliche synonyme deutschsprachige Bezeichnung lautet „hybrides Lernen“. Abbildung 2.7 zeigt ein typisches Muster einer Lehrveranstaltung, die nach dem System des Blended Learning abläuft.

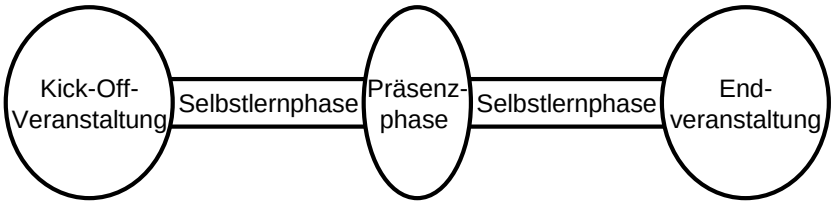

Abbildung 2.7: Phasen des Blended Learning

Elektronisch unterstütztes Lernen tendiert in der Regel dazu, Multimediaelemente so in die Lerneinheit einzubauen, dass diese auf eine bestimmte Art in einer bestimmten Reihenfolge vom Lernenden zu konsumieren sind. Simulationsbasiertes Lernen folgt anderen Regeln. Anstelle im Detail zum Beispiel die Funktionsweise eines Autos zu beschreiben und in einer Videosequenz das Fahrverhalten zu illustrieren, versetzt eine Simulation über das Autofahren den Lernenden in die Rolle der Fahrerin oder des Fahrers eines Autos. Simulationsbasiertes Lernen stellt somit eine konstruktivistische Lernmethode dar und findet mittels Computerprogrammen statt. Reale Systeme werden durch Nachbildung in ein Computermodell übertragen und Lernen geschieht durch die Interaktion mit diesem Computermodell.

2.5 Multimediales Lernen in Spielen, Simulationen und Mikrowelten

In seiner Arbeit über die Pädagogik und das Spiel [22] zitiert Rimmert van der Kooij eine Studie, nach der ein Kind bis zum Alter von sieben Jahren ungefähr 15.000 Stunden spielt, das sind ca. 25 % des bis dahin gelebten Lebens, Schlaf inklusive. Spielen ist demnach die wichtigste Aktivität kleiner Kinder. Und dennoch ist das Spielen in der Literatur nicht eindeutig defniert. Theorien darüber, warum Kinder spielen, gibt es viele. Eine Übersicht findet sich ebenfalls im eingangs erwähnten Artikel von Kooij, in dem auch folgende vier Kategorien des Spiels definiert werden:

- **Wiederholungsspiel, sensomotorisches Übungsspiel**: Das Wiederholen von Bewegungen, meist zur Erkundung der Umgebung.
- **Imitationsspiel, Phantasiespiel**: Vorgeführte Bewegungen erhalten einen Sinn.
- **Konstruktionsspiel**: Verschiedene bedeutungslose Spielelemente werden zu einem Ganzen zusammengefasst um einen Sinn zu ergeben (z.B. Bauen mit Bausteinen).
- **Gruppierungsspiel**: Bereits sinnvolle Spielelemente, z.B. Bäume, Häuser, werden zueinander in Beziehung gesetzt.

Als bekanntestes frühes Simulationsspiel kann SimEarth genannt werden. SimEarth simuliert die Evolution der Erde sowie der Zivilisation. Ein Problem, das Simulationen wie SimEarth aus didaktischer Sicht aufweisen, ist eine zu hohe Komplexität, die die strategischen und systematischen Vorgehensweisen für BenutzerInnen erschweren und damit den tatsächlichen Lernerfolg einschränken. SimEarth ist nach Angabe des Herstellers kein Spiel, sondern ein Spielzeug. Nach [4] wird ein Spiel (Game) als „ein System von Regeln mit einer vorgegebenen Zielsetzung“ verstanden. Ein Spiel zeichnet in der Regel die Eigenschaft des Nullsummenspiels aus, jede Verbesserung des Ergebnisses einer Spielerin ist

automatisch mit der Verschlechterung des Ergebnisses einer anderen Spielerin verbunden. Ein Spielzeug oder auch ein Rollenspiel (Play) ist ein Spiel, bei dem zwar Regeln vorhanden sind, jedoch die eindeutige Zielsetzung (Gewinn des Spieles) sowie die Eigenschaft des Nullsummenspiels fehlen. Spiele, in denen auch die Regeln beliebig veränderbar sind, werden im Kontext von Software als Mikrowelten bezeichnet.

Ein wichtiger Aspekt des Spiels ist, dass es keinen Zweck verfolgt, sondern der Sinn des Spiels in sich selbst ist. Dies stellt ein Grundproblem des spielbasierten Lernens dar, da dieses immer einem Zweck, einem Lernziel, folgt [6]. Rieber [9] berichtet von Studien, bei denen Simulationen zum Erlernen physikalischer Gesetze Aspekte des Spiels hinzugefügt worden sind, indem für das Erreichen bestimmter Vorgaben Punkte vergeben worden sind. Das Ergebnis dieser Studien war, dass sich das Hinzufügen dieser Spielelemente zu den Simulationen negativ auf den Lernerfolg der beobachteten Lernenden ausgewirkt hat. Durch das Fokussieren auf das Erhalten einer maximalen Punkteanzahl optimierten die BenutzerInnen zwar die Beherrschung der Simulation, reflektierten aber nicht auf das dahinterliegende physikalische System.

3 Simulation

Nothing is built on stone;
all is built in sand.
But we must build as
if the sand were stone.

Jorge Luis Borges

In diesem Kapitel wird Simulation als Methode zur Beschreibung realer Systeme aufbereitet. Simulation ist nach der Richtlinie 3633 des Vereins Deutscher Ingenieure e.V. „das Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierfähigen Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind“ [23]. Im Folgenden werden die drei verbreitetsten Verfahren der Simulation vorgestellt: ereignisorientierte Simulation, System Dynamics und Multiagentensimulation. Als Unterscheidungskriterium der Paradigmen wird herangezogen. wie die generierten Modelle die Wirklichkeit einteilen.

Ereignisorientierte Simulationen beschreiben reale Systeme als Prozesse, die durch das Eintreffen von Ereignissen ausgelöst werden. Eintretende Ereignisse, die entweder innerhalb des Systems erzeugt werden oder von außen auf das System treffen, sind die einzige Möglichkeit, den Zustand eines Systems zu ändern. Ereignisorientierte Simulationen werden zur Beschreibung von Prozessabläufen zum Beispiel in Betrieben verwendet.

System Dynamics teilt das zu beschreibende System in Bestands- und Flussgrößen ein. Diese Größen stehen miteinander in Wechselwirkung und beschreiben in positiven und negativen Rückkopplungsschleifen auf Makroebene das Verhalten

des Systems. System Dynamics ignoriert das Verhalten von einzelnen Akteuren oder aggregiert diese in eine kleine Anzahl von Gruppen. System Dynamics kann zur Beschreibung von technischen, physikalischen, biologischen oder ökonomischen Systemen verwendet werden. Die Einteilung in Bestands- und Flussgrößen stellt dabei ein relatives Konzept dar, so wirkt zum Beispiel das Element „Geschwindigkeit“ im Zusammenspiel mit dem „Weg“ als verändernd und wird als Flussgröße dargestellt, während „Geschwindigkeit“ im Zusammenspiel mit „Beschleunigung“ als Bestandsgröße modelliert wird.

Im Zentrum von Multiagentensimulationen stehen die einzelnen Akteure eines Systems und ihr Verhalten. Die Beschreibung der Akteure erfolgt auf lokaler (Mikro-) Ebene. Durch das lokale Wirken der Akteure kann auf Makroebene eine qualitative Veränderung des Systems entstehen (Emergenz), welche aus dem Verhalten der einzelnen Akteure nicht vorhersehbar ist. Multiagentensimulationen werden für Nachbildung von Systemen verwendet, die sich aus vielen Akteuren zusammensetzen (z.B. Verhalten von Ameisenkolonien, die Verbreitung von Informationen oder Krankheiten in der Bevölkerung).

Eine Übersicht über zentrale Eigenschaften dieser drei Simulationsparadigmen bietet Tabelle 3.1.

	Ereignisorientierte Simulation	**System Dynamics**	**Agentenbasierte Simulation**
Objekte der Modellierung	Ereignisse	Bestands- und Flussgrößen	Akteure
beschrieben wird…	Abhängigkeit der Elemente	Auswirkungen der Elemente aufeinander	Verhalten der Akteure
Art	ereignis-diskret	kontinuierlich	zeit-diskret
Ebene	lokal	global	lokal

Tabelle 3.1: Eigenschaften der Simulationsparadigmen

Dieses Kapitel bietet neben Definitionen zu Modell und Simulation eine Einführung in die oben genannten Simulationsparadigmen. Einen Überblick über die zeitliche Einordnung der Entwicklungen und den Zusammenhang dieser und anderer Simulationsparadigmen findet sich in Abbildung 3.1. Zum Abschluss

dieses Kapitels werden grundsätzliche Hinweise für den Modellbildungsvorgang gegeben sowie Simulation mit der in der Naturwissenschaft erprobten Methode des Experiments verglichen.

Zu den drei in diesem Kapitel vorgestellten Arten der Simulation, ereignisorientierte Simulationen, System Dynamics und Multiagentensimulation finden sich im darauffolgenden praktischen Teil Softwaretests von konkreten Simulationsumgebungen.

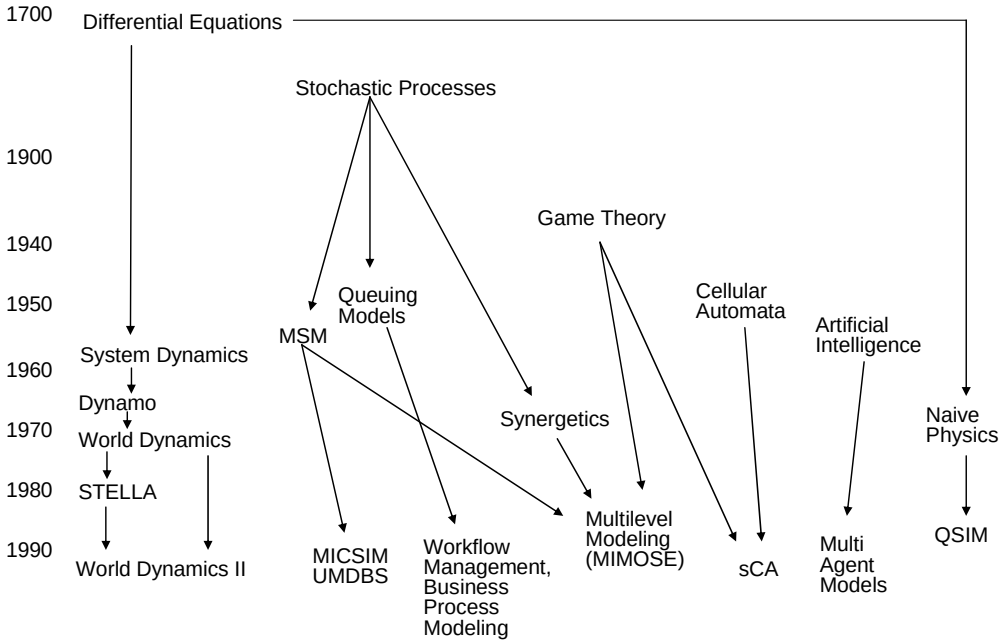

Abbildung 3.1: Die Entwicklung der Simulation aus [24]

3.1 Modellkonstruktion und Simulation

Modellierungsprozesse sind gleichzeitig Konstruktions- und Abbildungsprozesse. Die ModellbauerInnen trennen durch Abstraktion das Wesentliche vom Unwesentlichen und konstruieren damit aus dem Blickwinkel des Konstruktivismus (vgl. 2.2.5) einen Entwurf des realen Systems. Andererseits sind Modelle im erkenntnistheoretischen Sinn Abbilder der Wirklichkeit, da diese Eigenschaften des abzubildenden Objekts beinhalten.

Nach Gilbert und Troitzsch [24] ist Simulation „eine besondere Form der Modellierung“. Eine Definition eines Modells wird 1973 von Herbert Stachowiak in seiner allgemeinen Modelltheorie [25] vorgenommen. Demnach zeichnet ein Modell drei Kriterien aus:

- **Abbildung.** Ein Modell ist immer eine Repräsentation von einem realen System.
- **Verkürzung.** Es ist weder möglich noch erwünscht, alle Eigenschaften des Originals abzubilden. Es wird versucht, die für die Beobachtung relevanten Eigenschaften zu extrahieren.
- **Pragmatismus.** Ein Modell steht nicht für sich selbst, sondern muss interpretiert werden. Diese Interpretation orientiert sich an der Nützlichkeit für die ModelliererInnen.

Für die Generierung von Simulationen bedeutet das, dass ein Modell zwar einerseits tatsächlich die Struktur der Wirklichkeit (vergleiche aber dazu die Anmerkungen zur Wirklichkeit im Abschnitt 2.2.5 Konstruktivismus) nachbilden soll, stellt aber andererseits auch klar, dass diese Nachbildung nicht umfassend sein kann und auch nicht sein soll. Modelle sind also Entwürfe, die „eine wirklichkeitsnahe, jedoch einfachere, billigere oder ungefährlichere Untersuchung als das Objekt erlauben“ (Brockhaus, 1983).

Ziel der Modellbildung und der darauf aufbauenden Simulation ist es, sich auf jene Eigenschaften des Originalsystems zu beschränken, die für die Fragestellungen relevant sind. Außerdem muss das Modell sich mit jenen Eigenschaften begnügen, die messbar und nachbildbar sind. Ein Teil der Interpretation der Ergebnisse von Simulationen muss sich demnach der Frage widmen, inwiefern das Modell mit dem Original in Beziehung steht, sodass Interpretationen und Rückschlüsse auf die Realität gültig sind (vgl. Abschnitt 3.8).

Die „Logik der Simulation“ kann nach Gilbert und Troitzsch [24] wie in Abbildung 3.2 dargestellt werden. Durch Abstraktion wird aus der Beschreibung des realen Systems ein Modell erzeugt. Als Ergebnis von Simulationsläufen wer-

den simulierte Daten gewonnen, welche mit den durch Beobachtung des realen Systems gesammelten Daten verglichen werden.

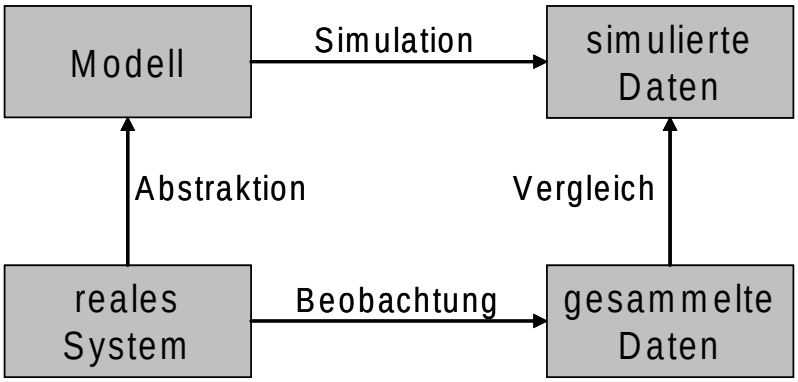

Abbildung 3.2: Die Logik der Simulation nach [24]

Simulation wird unter anderem zu folgenden Zwecken verwendet (vergleiche z.B. Gilbert und Troitzsch [24], Gilbert [26] und Bossel [27]:

- Simulation als eine Methode zur Theoriebildung
- Zum Erlangen eines besseren Verständnisses von bestimmte Eigenschaften von Systemen der Wirklichkeit
- Als Vorhersageinstrument, um in die Zukunft blicken zu können, z.B. Demographie
- Als Substitut für menschliche Kapazitäten, z.B. Expertensysteme
- Zur Systementwicklung im technischen Bereich
- Zu Trainingszwecken, z.B. Flugsimulatoren
- Zu Unterhaltungszwecken in Spielen, z.B. SimEarth
- Simulation zur Entdeckung und zur Formalisierung

3.2 Ereignisorientierte Simulation

Ereignisorientierte Modelle beschreiben Systeme, in denen eine Änderung des Systemzustandes ausschließlich durch das Eintreten von Ereignissen ausgelöst

werden kann [24]. Diese Ereignisse können sowohl innerhalb des modellierten Systems auftreten (z.B. bewirkt der Abschluss eines Prozesses den Start eines anderen Prozesses) oder von außerhalb wirken (z.B. ein virtueller Kunde betritt das simulierte Kaufhaus und löst Prozesse aus). Der Ablauf in solchen Systemen geschieht in diskreten Schritten. Diese Schritte sind jedoch nicht zeitdiskret, sondern erfolgen auf Grund von Ereignissen. Zwischen zwei aufeinanderfolgenden Ereignissen verändert sich das System nicht. Als Beispiele für ereignisorientierte Systeme können organisatorische und betriebliche Abläufe in Produktionsstätten oder Lagern genannt werden. Da die Umsetzung von ereignisorientierten Simulationen in Softwareprogrammen mit Hilfe von Warteschlangen implementiert wird, werden ereignisorientierte Systeme auch „queuing models“ genannt. Eine konkrete Form der Beschreibung von ereignisorientierten Systemen sind die von Carl Adam Petri 1962 in seiner Dissertation vorgestellten Petri-Netze (vgl. z.B. [28]).

Abbildung 3.3 zeigt die ereignisorientierte Simulation eines Warenlagers eines Großhandelsunternehmens in der Simulationsumgebung AnyLogic (vgl. Abschnitt 5.3). Waren werden angeliefert und von den entsprechenden MitarbeiterInnen ausgeladen, kontrolliert und in das Lager gebracht, um zu einem späteren Zeitpunkt in umgekehrter Reihenfolge das Lager wieder zu verlassen.

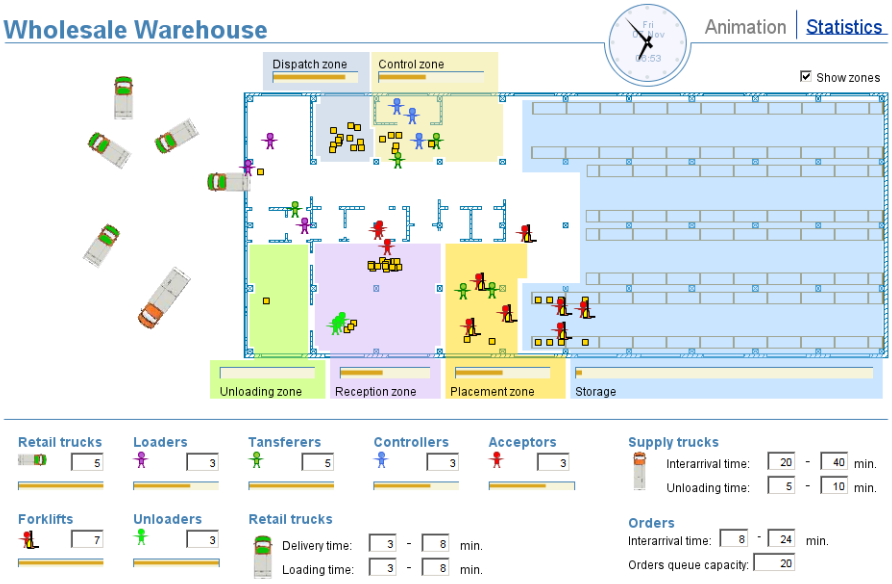

Abbildung 3.3: Ereignisorientierte Simulation eines Lagers in AnyLogic

3.3 Kausaldiagramme

Kausaldiagramme (engl. causal loop diagrams, influence diagrams) zeigen die kausalen Zusammenhänge zwischen Elementen eines dynamischen Systems [29]. Diese Zusammenhänge werden in Kausaldiagrammen nur als positive und negative Abhängigkeiten angegeben. In System Dynamics (vgl. den folgenden Abschnitt) kommen noch ausformulierte Gleichungen dazu, die diese Zusammenhänge beschreiben. Kausaldiagramme sind keine unmittelbare Simulationsmethode, sondern dienen der Beschreibung der grundlegenden Strukturen in dynamischen Systemen.

Ein zentraler Aspekt der Analyse von Kausaldiagrammen ist die Beschreibung von Rückkopplungsschleifen. Rückkopplungsschleifen sind „in sich geschlossene Kausalketten“ [30]. Ob eine Rückkopplung positiv oder negativ ist, kann einfach errechnet werden, indem die einzelnen Vorzeichen der geschlossenen Kausalkette miteinander multipliziert werden. Positive Rückkopplungen können durch ein durch diese Multiplikation erhaltenes positives Ergebnis identifiziert werden und finden sich in den unterschiedlichsten biologischen, demographischen oder ökonomischen Systemen, zum Beispiel: höhere Löhne führen zu höherem Konsum, höherer Konsum führt zu höheren Umsätzen der Unternehmen, höhere Umsätze der Unternehmen führen zu höheren Löhnen. Ergibt das Ergebnis der Multiplikation der Vorzeichen dagegen ein negatives Ergebnis, liegt eine negative Rückkopplungsschleife vor. Beispiele für negative Rückkopplungen findet man dort, wo Zielwerte zu erreichen sind, zum Beispiel: ein großer Unterschied zwischen der gewünschten Temperatur in einem Raum und der tatsächlichen führt zu einer höheren Heizleistung, eine höhere Heizleistung führt zu einer Erhöhung der Raumtemperatur, eine höhere Raumtemperatur führt zu einer geringeren Differenz zur gewünschten Temperatur. Kommt es in negativen Rückkopplungen zu Verzögerungen, wie dies beim vorhin genannten Beispiel des Thermostats der Fall ist, kommt es zu Schwingungen im System, da die verzögerte Handlung bereits in einem veränderten System zu wirken beginnt, in dem diese Handlung eventuell nicht mehr notwendig oder sogar kontraproduktiv ist.

Abbildung 3.4 zeigt ein Kausaldiagramm, das eine vereinfachte Darstellung der Abläufe innerhalb eines Betriebes im Zusammenhang mit der Menge der zu erledigenden Arbeit und der eingesetzten Arbeitskräfte abbildet. Dieses Beispiel ist im Zuge der Softwaretests in Kapitel 5 in der Simulationsumgebung Vensim entstanden. Der linke rote Kausalkreis bildet eine positive Rückkopplung, die beiden Kreise auf der rechten Seite der Abbildung stellen eine negative Rückkopplung dar. Das Element „productivity“ in dieser Abbildung kann als Konstante oder exogene Variable des Systems bezeichnet werden, da diese keine kausalen Abhängigkeiten innerhalb des Systems hat. Die Verzögerung, die zwischen den benötigten Arbeitskräften und der angepassten tatsächlichen MitarbeiterInnenmenge durch Neuanstellung und Kündigung entsteht, kann eine Schwingung im System erzeugen, da zum Beispiel zum Zeitpunkt des Wirksamwerdens der Anstellung von neuen MitarbeiterInnen diese eventuell gar nicht mehr gebraucht werden.

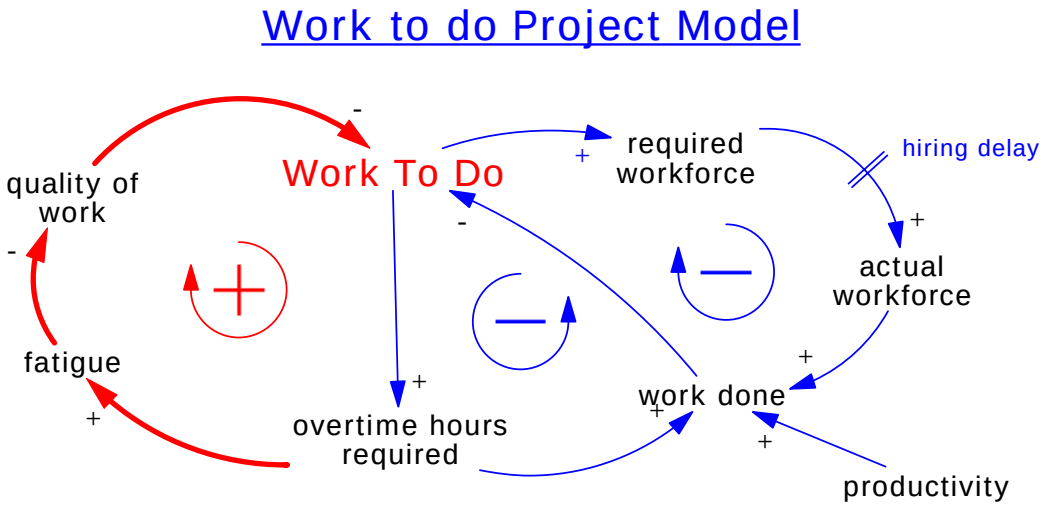

Abbildung 3.4: Kausaldiagramm erstellt in Vensim nach [29]

3.4 System Dynamics

System Dynamics ist eine Simulationsmethode, die von Jay Wright Forrester in den 1950er Jahren am Massachusetts Institute of Technology (MIT) entwickelt wurde [31] und zur Analyse und Simulation von dynamischen Systemen verwen-

det wird. Im Gegensatz zur agentenbasierten Simulation (vgl. Abschnitt 3.6), in der das Verhalten der einzelnen Akteure eines Systems beschrieben und simuliert wird, beschreibt System Dynamics das Verhalten des Systems mithilfe aggregierter Variablen auf der Makroebene.

Forrester verwendete seine systemdynamischen Methoden ursprünglich zur automatischen Steuerung von Fliegerabwehrkanonen. Nach dem Zweiten Weltkrieg entwickelte er die Methode für friedliche Zwecke weiter. Im Buch „Industrial Dynamics“ [32] wandte er sie auf die Modellierung von betrieblichen Prozessen an. Unter dem Titel „Urban Dynamics“ [33] simulierte Forrester in weiterer Folge soziale Problemstellungen in Städten. Durch die Arbeiten an den Weltmodellen (vgl. Abbildung 3.7 auf Seite 43), die Forrester im Auftrag des Club of Rome zur Vorhersage der weiteren globalen ökonomischen und ökologischen Entwicklung erstellte, und deren erste Veröffentlichung im Buch „World Dynamics“ [34] erlangte System Dynamics hohe Bekanntheit auch über die Grenzen der Wissenschaft hinaus, vor allem durch die literarische Verarbeitung im Bestseller „Die Grenzen des Wachstums“ [35] von Dennis Meadows et al.

Unter einem System versteht man im Gebiet von System Dynamics „eine Ansammlung von Elementen, die kontinuierlich über die Zeit miteinander interagieren und dadurch ein gemeinsames Ganzes ergeben“ [31]. Als Struktur werden die Beziehungen der Elemente in diesem System bezeichnet. „Dynamisch“ sind die Modelle von Forrester deshalb, weil eine kontinuierliche Veränderung der Variablen des Systems über die Zeit stattfindet die als Verhalten des Systems bezeichnet wird. Ein zentraler Aspekt von System Dynamics ist der Zusammenhang zwischen Struktur und Verhalten des beobachteten Systems. Während die Struktur das Verhalten des Systems bestimmt, lässt das Verhalten Rückschlüsse auf und Erkenntnisse über die zugrundeliegende Struktur zu. Martin [31] nennt System Dynamics den „mentalen Link“ zwischen Struktur und Verhalten eines Systems und verweist in diesem Zusammenhang auch auf die Bedeutung von Simulationen für die Verbesserung mentaler Modelle (vgl. Abschnitt 2.2.3) hin.

System Dynamics basiert auf den Gedanken der Differenzen- und der Differentialgleichungen, welche eine „mathematische Beschreibung des zeitlichen Ablaufs von Vorgängen in Naturwissenschaft und Technik ermöglichen“ [36].

Zeitdiskrete Prozesse werden mit Hilfe von Differenzengleichungen modelliert. Von zeitdiskret oder auch zeitgetaktet spricht man, wenn die unterschiedlichen Zustände nach bestimmten Zeitabständen (z.B. Sekunden, Jahre) auftreten. Eine Differenzengleichung liegt dann vor, wenn jeder Zustand sich durch eine Funktion vergangener diskreter Zustände ergibt. Beispiele für zeitdiskrete Abläufe sind Fertigungssysteme, betriebliche Abläufe, Kommunikationssysteme.

Kontinuierliche Prozesse dagegen werden in physikalischen und biologischen Systemen und auch im Wirtschaftssystem beobachtet. Kontinuierlich bedeutet dabei, dass die Veränderungen im System nicht zu bestimmten Zeitpunkten, sondern permanent passieren. Zur Modellierung solcher kontinuierlicher Prozesse werden Differentialgleichungen angeschrieben. Der Vergleich zwischen diskreten und kontinuierlichen Prozessen wird in Abbildung 3.5 illustriert.

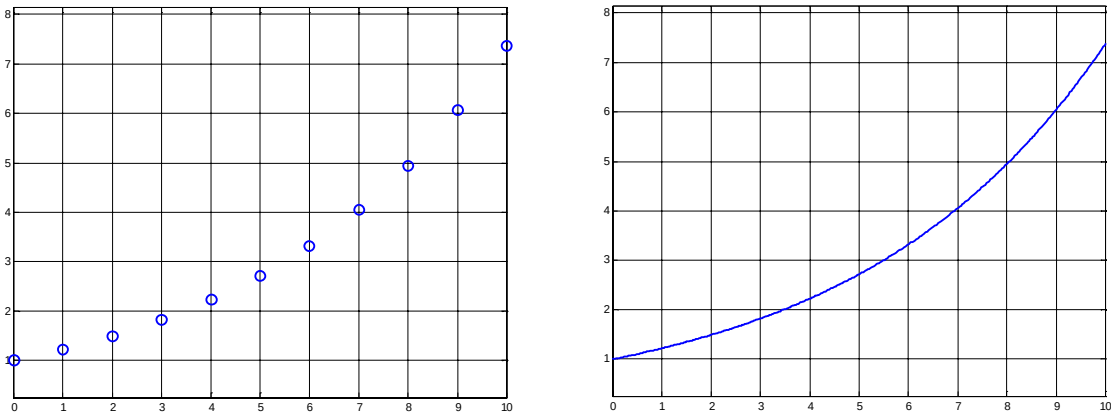

Abbildung 3.5: Zeitdiskreter (links) und kontinuierlicher (rechts) Prozess

Viele Systeme, die mit System Dynamics beschrieben werden, laufen als kontinuierliche Prozesse ab. Eine explizite mathematische Lösung ist jedoch nur für eine geringe Anzahl von Differentialgleichungssystemen möglich. Simulationsprogramme wie Stella, Vensim, usw. nehmen daher eine sogenannte Diskretisierung des Gleichungssystems vor. Die kontinuierlichen Differentialgleichungen werden in

diskrete Differenzengleichungen übertragen, die iterativ gelöst werden. Simultane Gleichungssysteme können mit System Dynamics nicht behandelt werden.

Obwohl eine Simulation in einer System Dynamics Simulationsumgebung aus vielen Differenzen- oder Differentialgleichungen besteht, ist für die tatsächliche Verwendung dieser Tools kein umfangreiches Wissen aus diesen Bereichen der Mathematik notwendig (vgl. Kapitel 5).

Forrester modellierte und simulierte seine Modelle mit dem Computerprogramm DYNAMO, bei dem sowohl Ein- als auch Ausgabe (über eine IBM-Kugelkopf-schreibmaschine) ausschließlich aus Textzeichen bestanden haben. Die Elemente der Modellierung in System Dynamics Modellen werden im Folgenden anhand der Simulationsumgebung Stella (vgl. Abschnitt 5.4) erklärt. Stella war das erste gebräuchliche und weit verbreitete Tool für die grafische Erstellung von System Dynamics Modellen und später folgende Tools haben zentrale Aspekte der Notation von Stella übernommen. Modelle, die in System Dynamics Umgebungen erstellt werden, bestehen aus vier verschiedenen Hauptelementen (vgl. Abbildung 3.6):

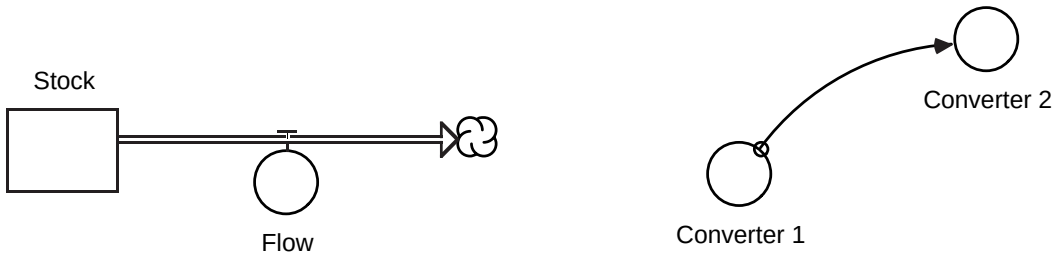

Abbildung 3.6: Elemente der System Dynamics Modellkonstruktion

- **Stock** (auch Zustandsgröße, Bestandsgröße, Niveaugröße, Niveau): Symbolisiert die Anzahl oder Menge einer darzustellenden Größe, z.B. Wasserstand in einem Behälter, Anzahl der Lebewesen einer bestimmten Spezies.
- **Flow** (auch Veränderungsgröße, Flussgröße, rates): Stellt die Rate der Veränderung eines Stocks dar und kann als Zu- oder Abfluss verstanden

werden, z.B. Wasserzufluss, Abgang durch Tod von Lebewesen. Die Wolke in Abbildung 3.6 wird dann verwendet, wenn Ursprung oder Ziel eines Flusses nicht Teil der Modellierung ist.

- **Converter**: Wird dazu verwendet, Eingaben von BenutzerInnen zu speichern. Ein Converter kann auch als Variable verstanden werden. Auch der Kreis als Teil des Flow-Elements wird als Converter bezeichnet, da dieser den Fluss reguliert.
- **Connector**: Ein Pfeil, der jeweils zwei der oben genannten Elemente miteinander verbindet und als Abhängigkeit verstanden werden kann. In Abbildung 3.6 symbolisiert der Pfeil, dass „Converter 2“ sich aus einer Funktion berechnet, die „Converter 1“ als Variable beinhaltet.

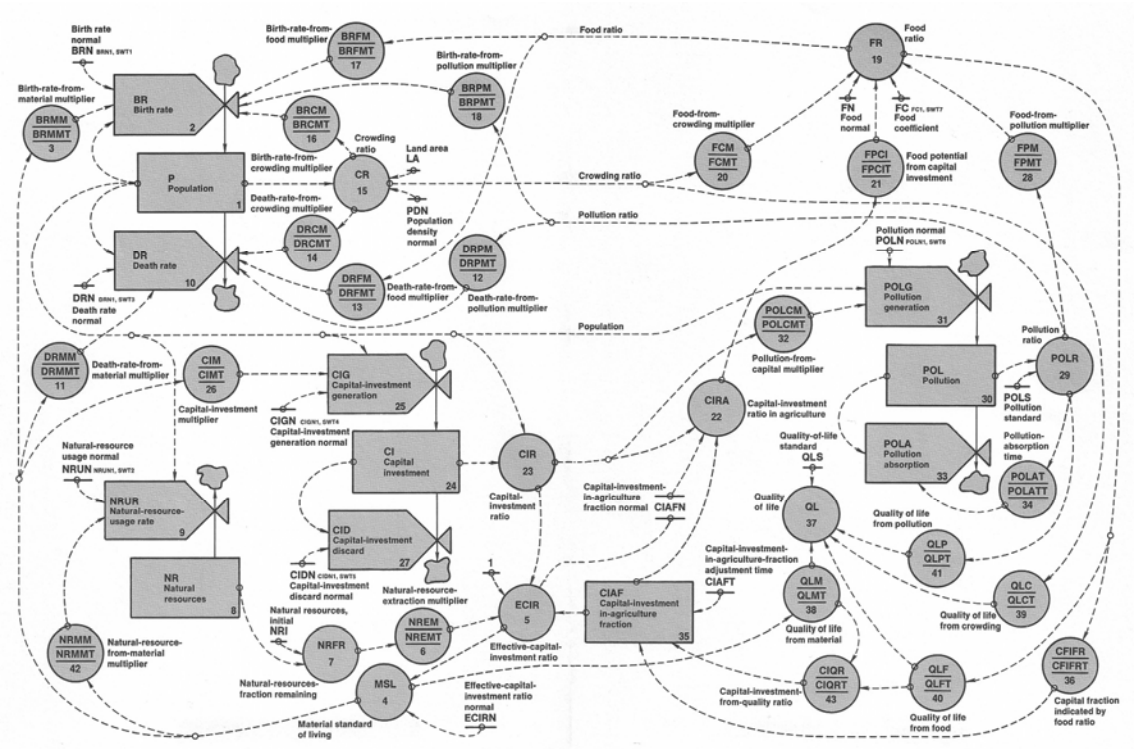

Abbildung 3.7: Weltmodell von Forrester, Quelle: [34]

3.5 Zellulare Automaten

„Ein zellularer Automat besteht aus eine Vielzahl von identischen Zellen, die in einem regelmäßigen Raster angeordnet sind“ [24]. Diese Zellen können verschiedenartig angeordnet sein, 1-, 2- oder 3-dimensional. Der Ablauf erfolgt zeitdiskret. Jede Zelle hat zu jedem Zeitpunkt t einen eindeutig definierten Zustand, welcher aus dem Zustand t-1 der Zelle selbst und bei Bedarf der Nachbarzellen errechnet wird. Diese Interaktion mit anderen Zellen ist auf die direkten Nachbarn begrenzt. Zellulare Automaten sind daher sehr gut geeignet für die Simulation von Phänomenen, die jeweils lokal begrenzt ablaufen.

Eine der einfachsten und bekanntesten Beispiele für Simulation mit zellulären Automaten ist das „Game of Life“ von John H. Conway (publiziert 1982 in „Winning Ways for Your Mathematical Plays“, Berlekamp et al.). Für jede Zelle gibt es dabei zwei Zustände (tot, lebend) und nur zwei Regeln:

1. Eine lebende Zelle bleibt lebend, wenn sie genau zwei oder drei lebende Nachbarn hat.

2. Eine tote Zelle bleibt tot, solange sie nicht genau drei lebende Nachbarn hat.

Diese zwei einfachen Regeln erzeugen je nach Ausgangszustand der Zellen eine immer neue Evolution der Population. Abbildung 3.8 zeigt die 14 Schritte eines sich wiederholenden Musters, das bei der in Bild 1 gewählten Ausgangssituation entsteht.

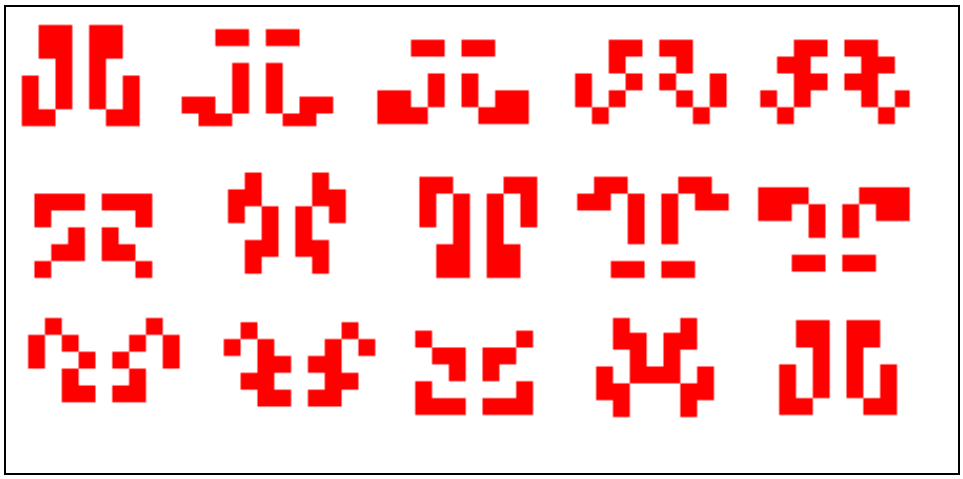

Abbildung 3.8: Zyklisches Muster einer Game of Life Simulation

3.6 Agentenbasierte Simulationen

Der Grundgedanke, auf dem Simulationen mit Multiagentensystemen beruhen, ist jener der Dezentralisierung und Individualisierung [37]. Steven Johnson beschreibt in seinem Buch [38] verschiedene Phänomene der Emergenz. Darunter versteht man das Entstehen von Bottom-Up Strukturen aufgrund lokalen Verhaltens. Ausgangspunkt seiner Schilderungen sind dabei die Forschungen von Deborah Gordon über das Verhalten der Ameisen. Gordon untersuchte über mehrere Jahre rote Ernteameisen (Pogonomyrmex barbatus) in der Wüste von Arizona [39] und beobachtete dabei neben dem Verhalten der einzelnen Ameise vor allem Eigenschaften und Verhalten unterschiedlicher Ameisenkolonien. So beobachtete sie, dass alte Kolonien andere Verhaltensweisen als jüngere Kolonien aufweisen, obwohl die Lebensspanne einer einzelnen Ameise nur einen Bruchteil des Alters einer Kolonie ausmacht. Die Makrostruktur der Kolonie entwickelt also unabhängig von einer zentralen Steuerung ein Verhalten, dass nicht durch das Verhalten der einzelnen Akteure erklärbar ist. Die Ameisenkolonie stellt hier eindeutig mehr dar, als die Summe der einzelnen Ameisen. Abbildung 3.9 zeigt die Simulation einer Ameisenkolonie auf Futtersuche mit dem Programm StarLogo (vgl. Abschnitt 5.1), einer Software zur Erstellung agentenbasierter Simulationen.

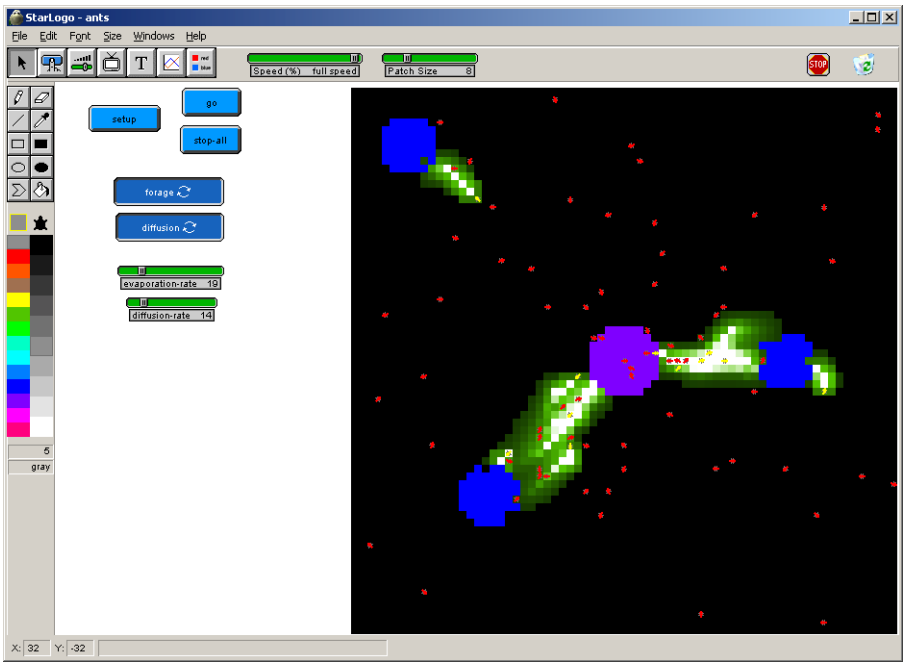

Abbildung 3.9: StarLogo Simulation einer Ameisenkolonie auf Futtersuche

Die Simulation einzelner Akteure auf der Mikroebene mit relativ einfachem lokalem Verhalten und die Beobachtung der dadurch entstehenden Veränderung des Gesamtsystems auf der Makroebene sind die Kernelemente der agentenbasierten Simulationen. Als Agent in solchen Systemen bezeichnet man die Repräsentation einer realen Einheit (Ameise, Mensch, Firma,...) innerhalb eines Computerprogramms [26]. Eine wichtige Eigenschaft dieser Agenten ist die Interaktion mit anderen Agenten oder der Umwelt. Die Agent-zu-Agent Interaktion kann aus dem Austausch von Informationen oder Ressourcen bestehen. Als Umwelt wird die virtuelle Welt bezeichnet, in der sich die Agenten bewegen. Das Verhalten dieser Akteure folgt in jedem Simulationsschritt eindeutigen und meist einfachen Regeln.

Nach Nigel Gilbert kann man für agentenbasierte Modellbildung und Simulation folgende Eigenschaften angeben [26]:

Ontologische Entsprechung

Eine Simulation einer Ameisenkolonie wie in Abbildung 3.9 setzt sich aus der Simulation der einzelnen Ameisen und ihrem Verhalten zusammen. Eine Einheit in der Simulation entspricht also direkt einer Einheit des realen Systems, was

sowohl die Beschreibung des Verhaltens der einzelnen Agenten, als auch die Interpretation der Ergebnisse vereinfacht.

Verschiedenartige Agenten

Ein zentraler Aspekt agentenbasierter Simulationen ist die Einheit des Agenten. Jeder Agent besitzt eigene „Variablen“ und kann somit eigenes Verhalten generieren. Eine Auswirkung dieser Eigenschaft ist, dass die zu simulierenden Einheiten der des realen Systems in ihrer Unterschiedlichkeit ohne großen Aufwand einzeln und ohne Aggregation in die Simulationsumgebung übernommen werden können. Systemdynamische Modellierungen, wie sie in Abschnitt 3.4 vorgestellt worden sind, würden bei vergleichbarem Detaillierungsgrad eine sehr hohe Komplexität erreichen, unübersichtlich werden oder überhaupt nicht mehr konstruierbar sein.

Interaktion zwischen den Agenten

In agentenbasierten Simulationen können Agenten mit anderen Agenten interagieren. Dies erfolgt in der Regel durch Austausch von Information im weitesten Sinne. So lassen sich sehr einfach z.B. biologische und soziale Diffusionsprozesse simulieren (z.B. Verbreitung von ansteckenden Krankheiten oder Informationen).

Repräsentierung der Umwelt

Agenten sind in einer Umwelt verortet und können sich durch diese bewegen. Das ermöglicht Interaktionen mit dieser Umwelt. Die Umwelt kann unterschiedlichste Elemente der realen Welt repräsentieren, z.B. Hindernisse, die die Agenten nicht überwinden können, oder Informationsquellen, die das Verhalten der Agenten verändern. Durch die Umwelt und die möglichen Bewegungen der Agenten entsteht eine Art physischer Raum (vgl. Abbildung 3.9).

Die explizite Darstellung der räumlichen (zwei- oder dreidimensionalen) Umwelt ist keine zwingende Eigenschaft von Multiagentensystemen. Zum Beispiel ist bei der Simulation der demographischen Entwicklung der Bevölkerung eine Verortung der Akteure in einer Umwelt keine Voraussetzung. Ein anderes Beispiel für Interaktion zwischen den Akteuren unabhängig von deren räumlicher Anordnung

stellen Simulationen von Vorgängen in sozialen Netzwerken dar. Demgegenüber gibt es aber auch agentenbasierte Simulationen, die z.B. mit geographischen Informationssystemen interagieren.

Beschränkte Rationalität

Vor allem bei der Simulation sozialer Vorgänge erweist es sich in der Regel als eine der Realität besser entsprechende Variante, Agenten zu erzeugen, die nicht immer das tun, was objektiv am besten für sie ist. Die Implementierung eines bestimmten Grades an Irrationalität ist eine zentrale Stärke agentenbasierter Simulationen. Ein weiterer Aspekt, der sich aus dieser Überlegung ergibt, ist die eingeschränkte Fähigkeit des einzelnen Akteurs, Informationen über andere Akteure oder das Gesamtsystems zu erhalten. Steven Johnson fasst diese Eigenschaften mit „think local, act local“ [38] zusammen.

Lernen

Jeder einzelne Agent ist lernfähig. Diese Eigenschaft ermöglicht eine ständige Entwicklung und Veränderung des Verhaltens der Akteure ausgelöst durch die Interaktion mit anderen Akteuren, der Umwelt oder akteursinternen Veränderungsprozessen, z.B. altern, wachsen,…

3.7 Hinweise zu Modellbildung und Simulation

Miller und Page führen im Anhang ihres Buches [40] 18 Handlungsanweisungen für qualitätsvolles computerunterstütztes Modellieren an. Einige dieser Hinweise werden im Folgenden kurz erläutert und auf die Erstellung und Verwendung von Simulationen übertragen, da sie auch in diesem Zusammenhang von großer Relevanz sind. Da die englischen Originalbezeichnungen der einzelnen Punkte sehr treffend und klar sind, wird bei den Bezeichnungen der Überschriften auf diese zurückgegriffen.

Keep the model simple

Die reale Welt zu beschreiben, kann mitunter ein sehr komplexes Unterfangen sein. Wenn das reale zu beschreibende System noch dazu ein soziales ist, ist in vielen Fällen eine zufriedenstellende umfassende Beschreibung nicht möglich, da unzählige Dimensionen ineinander greifen und einander beeinflussen. Diese Beschreibung des realen Systems ist jedoch Voraussetzung für die Übertragung in eine computerunterstützte Simulation. Die zentrale Herausforderung bei der Übertragung einer beobachteten Realität in eine computergestützte Nachbildung ist es nach Miller und Page, dass genau so viele Eigenschaften des realen Systems übertragen werden, wie unbedingt zur Beschreibung notwendig sind – und nicht mehr.

Focus on the science, not the computer

Vor allem in den ersten Stadien der Erschaffung von Simulationen sollte die Erforschung des realen Systems im Vordergrund stehen und nicht die Attraktivität der grafischen Aufbereitung. Vor allem in Zusammenhang mit spielbasiertem Lernen führt dieser Punkt jedoch schnell zu einem Widerspruch, da das technische Niveau von herkömmlichen Computerspielen mittlerweile ein sehr hohes ist und die Anwenderinnen und Anwender computerbasierter Lerninstrumente eine hohe Erwartungshaltung mitbringen.

Avoid black boxes

Jeder Teilaspekt der Simulation muss klar ersichtlich und nachvollziehbar sein. Und auch das Verhalten des gesamten Systems muss vorhersagbar und kontrollierbar sein. Als Black-Box wird ein Objekt bezeichnet, dessen innere Funktionsweise nach außen verborgen bleibt. Solche Elemente in einer Simulation stören den Erkenntnisprozess empfindlich und verhindern somit den erwünschten Lerneffekt.

Have tunable dials

Der Einsatz von Reglern zur Steuerung von Simulationsparametern erhöht die Möglichkeiten der Interaktion der Lernenden mit der Lernumgebung. Außerdem wird dadurch der Umgang mit abstrakt-logischen Operationen und die systematische Planung und Durchführung von Problemlösungsstrategien gefördert.

3.8 Die Simulation als virtuelles Experiment

Als Einsatzgebiet im Unterricht für Simulationen sieht Rieber [9] den Ansatz, den Lernenden eine Fragestellung zu geben und die Simulation als Labor zum Finden der Lösung zu verwenden. Mit den Modellen in den Simulationsumgebungen experimentieren, indem verschiedene Parameter verändert und die dabei entstandene Auswirkung auf einzelne Elemente oder das gesamte System beobachtet werden, hat einen spielerischen Charakter (vgl. z.B. [41]). Gleichzeitig ist dies mit dem Sammeln von Erfahrungen über das System verbunden und demnach eine Quelle des Lernens. In Verbindung mit einer von den Lehrenden gestellten Fragestellung, die mit Hilfe der Simulation beantwortet werden muss, wird aus dem spielerischen Experimentieren ein Experiment, das dem aus der naturwissenschaftlichen Forschung bekannten und etabliertem Experiment in vielem gleicht. Im Folgenden sollen die Übereinstimmungen sowie die Unterscheidungen zwischen herkömmlichen naturwissenschaftlichen Experimenten und virtuellen Simulationsexperimenten herausgearbeitet werden.

Unter einem Experiment versteht man ein Untersuchungsdesign, in dem „der Forscher einzelne Bedingungsfaktoren (unabhängige Variablen) variiert, um zu sehen, welche Effekte (abhängige Variablen) sich daraus ergeben“ [42]. Als unabhängige Variablen werden dabei jene bezeichnet, die von den ForscherInnen „absichtsvoll und geplant“ verändert werden. Durch diese gewollte Veränderung geschieht auch die Veränderung von anderen, den abhängigen Variablen. Als Hypothese bezeichnet man die Vorhersage dieses Effekts. Störvariablen sind jene, die diesen experimentell beobachteten Effekt verfälschen.

Von den in [42] aufgezählten Formen des Experiments entsprechen Computersimulationsexperimente der Definition von Laborexperimenten, indem ein kontrolliertes Verändern der unabhängigen Variablen sowie eine Kontrolle der Störvariablen möglich sind.

Der Ablauf eines Experimentes folgt in unterschiedlichen wissenschaftlichen Bereichen im Kern diesem Ablauf [43]:

- Festlegen eines Ziels, einer Hypothese
- Erstellen eines Umsetzungsplanes des Experiments
- Vorbereitungen und Organisation des Experiments
- Testphase des Experiments
- Durchführungsphase inklusive Datensammlung
- Wiederholung des Experiments
- Analyse der Daten
- Schlussfolgerungen inklusive Überprüfung der Hypothesen
- Erstellung eines Berichts

Diese Punkte finden sich auch in Simulationsexperimenten wieder, allerdings mit veränderter Bedeutung. Die Vorbereitungen sind weit weniger aufwändig und bestehen bei einem fertigen Modell nur aus der Einrichtung der Startwerte der Variablen. Die Testphase ist bei Simulationen im Computer nicht notwendig, da aufgrund der Geschwindigkeit, der Verfügbarkeit und der Kosten von Computern der eigentliche Simulationslauf ein unwesentlicher Aufwand ist. Die Datensammlung verläuft bei Computersimulationen in der Regel automatisch. Wiederholungen sind in Computersimulationen kein zusätzlicher Aufwand, daher werden in vielen wissenschaftlichen Projekten tausende Simulationsdurchläufe durchgeführt.

Einen relevanten Aspekt bei der Verwendung einer Computersimulation als virtuelles Experiment stellen die Schlussfolgerungen dar. Abbildung 3.10 ist eine

Erweiterung der Logik der Simulation aus Abbildung 3.2. Wie eingangs in diesem Kapitel beschrieben, entsteht durch Abstraktion ein Modell. Durch Simulation werden Daten gewonnen, die mit den durch Beobachtung gesammelten realen Daten des Systems verglichen werden. Erweitert man die Simulation um eine Hypothese zu einem Experiment, wirkt diese ebenfalls auf die Simulation und die daraus erzielten Effekte ein. Diese Effekte, die in herkömmlichen Experimenten Rückschlüsse auf die Hypothese zulassen und zu einer Adaption dieser führen können, stehen aber auch in direkter Abhängigkeit zum zugrundeliegenden Modell, und sind durch die Grundannahmen der Abstraktion determiniert.

Einfach gesagt kann aus einem unerwarteten Ergebnis nicht klar gefolgert werden, ob die Hypothese oder das Modell falsch sind. Das Modell kann also als eine Art Störvariable in Experimenten mit Simulationsumgebungen gesehen werden bzw. mit verzerrenden Einflussfaktoren, wie sie aus der sozialwissenschaftlichen Forschung bekannt sind (z.B. Interviewereffekt), verglichen werden.[1]

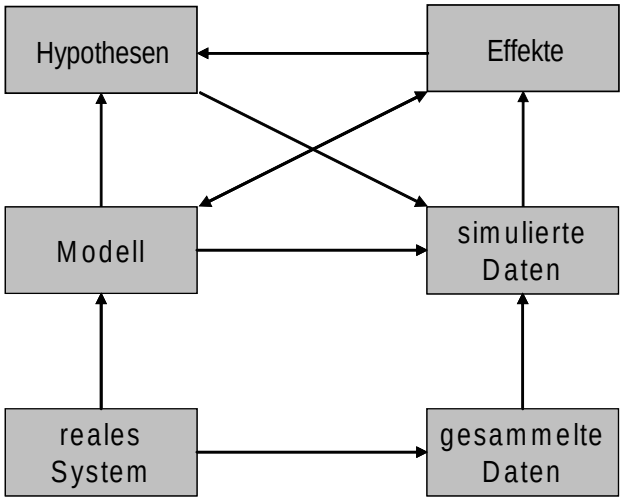

Abbildung 3.10: Logik des computersimulierten Experiments

[1] Tatsächlich findet die Abstraktion des realen Systems in ein Modell über den Umweg des mentalen Modells statt, das die zentrale Voraussetzung einer Simulation darstellt.

4 Untersuchungskriterien

Eine Frage hätte ich da noch...

Inspektor Columbo

Im praktischen Teil dieser Diplomarbeit werden verschiedene Simulationsumgebungen vorgestellt. Nach einer kurzen Übersicht über die Produkte und deren Anwendungsmöglichkeiten sollen die Programme nach den unten angeführten Kriterien untersucht werden. Der Schwerpunkt ist dabei auf den Einsatz im Unterricht gerichtet. Die unterschiedlichen Punkte werden also aus der Perspektive der Verwendung im Rahmen einer Lehrveranstaltung betrachtet.

Alle Eindrücke und Bewertungen spiegeln die persönlichen Erfahrungen des Autors wider und erheben keinen Anspruch auf allgemeine Gültigkeit (vgl. Abschnitt 4.8).

4.1 Zugang, Kosten

Der Kostenfaktor ist im Zusammenhang mit computergestütztem Lernen eines der zentralen Probleme (vgl. z.B. [6]). In diesem Abschnitt sollen Fragen nach Zugang und Kosten zu den einzelnen Softwarepaketen geklärt werden. Unter Zugang wird dabei der gesamte Vorgang verstanden, der notwendig ist, damit die Software auf dem lokalen Computer benutzt werden kann, umfasst also den Erwerb sowie die Installation. Bei der Erwerbungsfrage ist zu unterscheiden, ob die Software im Internet zum Download zur Verfügung steht oder auf anderem Weg erworben werden muss. Zudem ist das Auffinden der Software im Internet ein

Kriterium, die Möglichkeit also, die Software über die gängigen Suchmaschinen aufzufinden.

Die Frage der Kosten ist vor allem im schulischen oder universitären Kontext zentral. So können bereits niedrige Kosten den Einsatz im Rahmen einer Lehrveranstaltung erschweren oder verunmöglichen. Sollte die Software nicht gratis sein, soll außerdem geklärt werden, ob verbilligte Ausbildungs- oder Testversionen existieren.

Einen weiteren Aspekt stellen die Transaktionskosten dar. Darunter kann der zusätzliche Aufwand verstanden werden, den mögliche BenutzerInnen der Software abseits finanzieller Investitionen zu tragen haben, zum Beispiel Aneignung von Vorwissen, um die Software bedienen zu können, Aufwand im Zusammenhang mit Beschaffung und Installation der Software. Diese Punkte werden, sofern sie nicht die Installation betreffen, in den Abschnitten „Gesamteindruck“ und „Eignung für den Unterricht“ besprochen.

4.2 Erste Schritte

Der Einstig in eine neue Software und die damit verbundenen ersten Eindrücke und Erlebnisse mit der Benutzung haben einen großen Einfluss auf die Motivation der Lernenden (vgl. dazu Punkt 4.5). Im Abschnitt „Erste Schritte“ wird untersucht, inwiefern dieser Einstieg für neue UserInnen unterstützt wird oder nicht. Als positiv wird dabei eine Anleitung für die ersten Schritte gewertet, welche auf der Projekthomepage auffindbar ist, oder eine interaktive Einschulung im Rahmen der Software. In diesem Abschnitt wird auch über das Vorhaben und Gelingen der Erstellung einer ersten Simulation mit der jeweiligen Software berichtet, vergleichbar mit dem ersten Programmierbeispiel in allen gängigen Lehrbüchern für Programmiersprachen, in denen die Meldung „Hello World“ ausgegeben wird.

4.3 Dokumentation

Gute Dokumentation dient als Einstieg für AnfängerInnen und als Nachschlagewerk für Erfahrene. Es soll geklärt werden, ob eine Dokumentation für die Tools verfügbar und zugänglich ist.

4.4 Beispielbibliothek

Beispiele, die entweder mit der Software mitgeliefert werden oder leicht im Internet auffindbar sind, erhöhen die Attraktivität einer Simulationsumgebung. Durch das Nachvollziehen einer von anderen erstellten Simulation werden Eindrücke über die Funktionsweise der Software gewonnen, Ideen für mögliche eigene Simulationen angeregt und schneller Erfolgserlebnisse produziert.

4.5 Motivation, Erfolgserlebnisse

Claudio Dondi et al. führen in einer Aufzählung über die Vorteile des spielbasierten Lernens [44] als einen der zentralen Punkte die Motivation an. In weiterer Folge beschäftigen sich die Autoren dieses Artikels auch mit der Entstehung, der Förderung und den Problemen der Motivation. Als die Motivation fördernd wird in dieser Aufzählung eine unmittelbare Rückmeldung genannt. Dies bedeutet im Umgang mit Simulationen, dass funktionierende, sprich ablauffähige Simulationsexperimente Motivation erzeugen. Diese Erfolgserlebnisse sind vor allem dann notwendig, wenn die Lernenden angeregt werden sollen, eigene Modellkonstruktionen und Simulationen vorzunehmen. Die technische Umsetzung darf hier eine nicht allzu große Hürde darstellen, da sonst das Stadium des entdeckenden Lernens nicht erreicht werden kann. Neben dem generellen Gefühl der Motiviertheit im Umgang mit der Simulationsumgebung soll in den folgenden Produkttests konkret festgehalten werden, ob es EinsteigerInnen möglich sein kann, eine erste eigenständige Simulation innerhalb von zwei Stunden ablaufen zu lassen.

4.6 Gesamteindruck

Hier werden die gewonnenen Eindrücke zusammengefasst und die Vorteile den Nachteilen der einzelnen Tools gegenübergestellt. Auch soll eine Einschätzung darüber gegeben werden, welche Art von Vorwissen für potentielle BenutzerInnen nötig ist bzw. wie hoch die sogenannten Transaktionskosten sind, sich dieses anzueignen.

4.7 Eignung für den Unterricht

Abschließend wird eine Einschätzung abgegeben, ob die getestete Simulationsumgebung für den Einsatz im Unterricht geeignet ist und wenn ja, für welche Zielgruppen (z.B. SchülerInnen, Studierende aller Fächer, Studierende mit Programmiererfahrung). Zusätzlich werden in diesem Abschnitt Hinweise über den Grad der benötigten Erfahrungen der Lehrenden mit der Software gegeben.

4.8 Testumgebung

Sämtliche folgenden Softwaretests wurden auf einen Home-PC mit einem AMD Athlon XP 3000+ Prozessor mit 2,1 GHz und einem Arbeitsspeicher von 1 GB durchgeführt. Als Betriebssystem des Testrechners wird Windows XP Professional in der Studierendenversion des Zentralen Informatikdienstes (ZID) der Technischen Universität Wien mit Service Pack 3 verwendet. Der Tester ist der Autor der vorliegenden Arbeit und als Informatiker geübt, sich die Arbeitsweisen von neuen Programmen und Entwicklungsumgebungen anzueignen. Bei den durchgeführten Tests wird dennoch versucht, Schlüsse auch für weniger versierte ComputernützerInnen zu ziehen. Der Testzeitraum der folgenden Softwaretests umfasst für alle Programme den Zeitraum Oktober bis November 2008.

5 Simulationsumgebungen

The use of simulations in learning derives from a belief that exploratory learning is of value and that the learner must be allowed freedom.

Jos J. A. van Berkum

Im vorherigen Abschnitt wurde ein Katalog von Untersuchungskriterien erarbeitet. Mit diesen Kriterien werden im Folgenden verschiedene Simulationsumgebungen untersucht. Einführend werden die einzelnen Tools kurz vorgestellt und die jeweiligen Anwendungsbereiche skizziert. Im Weiteren wird versucht, ein Projekt mit Hilfe der Dokumentation der Software zu erstellen. Neben der Evaluierung kann so auch ein Einblick in die Funktionsweisen der einzelnen Tools gegeben werden.

Eine Zusammenfassung der Ergebnisse der Beurteilungen finden sich nach den Detailbesprechungen im Abschnitt 5.7.

5.1 StarLogo/NetLogo

StarLogo (siehe Abbildung 5.1 links) ist eine "programmierbare Modellierungsumgebung zur Erforschung der Arbeitsweise von dezentralisierten Systemen" [45]. Das am Massachusetts Institute of Technology (MIT) entwickelte Programm ist eine Weiterentwicklung der Programmiersprache Logo. Mit der ursprünglichen Logo Programmiersprache (eine lauffähige Windows Version aus dem Jahre 2002 findet sich unter [46]) können ComputernutzerInnen, die keine Programmierer

sind, durch einfache Befehle ein Symbol auf dem Bildschirm bewegen. Dieses Symbol, das zwar keiner Schildkröte gleicht, aber als „turtle“ bezeichnet wird, hinterlässt eine gezeichnete Spur seiner Bewegungen. So können einfache, aber auch komplexere geometrische Figuren entstehen. Dieses Grundprinzip, dass ein Element mit einfachen Befehlen über den Bildschirm bewegt wird, wird in StarLogo um den Aspekt erweitert, dass es nicht ein Element gibt, sondern viele. StarLogo ist demnach eine Umgebung für Multiagentensimulationen.

NetLogo (siehe Abbildung 5.1 rechts) kann als Weiterentwicklung von StarLogo betrachtet werden, obwohl StarLogo und NetLogo zwei unterschiedliche Programme von unterschiedlichen Entwicklungsgruppen sind. Die Aufmachung von NetLogo ist attraktiver gestaltet und die Modellbibliothek ist sehr viel umfangreicher. Diese beiden Programme werden in dieser Aufzählung gemeinsam genannt, da sie in der Anwendung fast identisch sind.

Abbildung 5.1: Startseite der Homepage von StarLogo [45] und NetLogo [46]

Der Hauptfokus der Beschreibung liegt in diesem Abschnitt auf dem Programm StarLogo, da die Entwicklung dieser Software sehr stark vom direkten Einsatz im Unterricht motiviert ist. So schreiben die Autoren in ihrem Lehrbuch zu StarLogo [41], dass StarLogo gestalten kann, was und wie Menschen lernen. StarLogo wurde entwickelt „um Menschen zu befähigen, ihre eigenen Modelle komplexer, dynamischer Systeme zu bauen“. Die Autoren sprechen davon, dass es von zentra-

ler Bedeutung ist, Softwareumgebungen zu entwickeln, die die „intellektuelle Neugierde“ der Lernenden fördern.

Zugang, Kosten

Das Auffinden der Software im Internet verläuft ohne Schwierigkeiten. Im Gegensatz zur Suchmaschine Google ist bei Live Search die Projekthomepage beim Suchwort „StarLogo“ nicht der erste Treffer, dennoch wird diese auch beim Suchdienst von Microsoft innerhalb der ersten Bildschirmseite dargestellt. Der Downloadbereich der Software findet sich auf der Startseite (siehe Abbildung 5.1) und bietet Downloadversionen für Windows, Macintosh und Unix-Systeme. Der Download erfordert eine Bekanntgabe von Name und E-Mail Adresse und beinhaltet in der Version für Windows einen automatischen Installer, sodass die Installation von StarLogo ohne irgendwelche Vorkenntnisse möglich und mit keinem zusätzlichen Aufwand verbunden ist.

Erste Schritte

Für den Start mit StarLogo bieten sich zwei Varianten an. Zum einen über die von StarLogo bereitgestellte Dokumentation, die im Rahmen der Installation lokal eingerichtet wird, aber auch online auf der Projektseite verfügbar ist. Eine andere Variante, die ersten Schritte in StarLogo vorzunehmen, die sich vor allem für den Einsatz im Unterricht eignet, ist das von den Autoren des Programms geschriebene Lehrbuch zu StarLogo [41]. Im Rahmen dieses Tests wurde nach der lokal installierten Dokumentation vorgegangen und die Lektion „Getting Started“ bearbeitet. Eingangs erfolgt eine kurze Einführung in die Grundelemente von StarLogo. „Turtles“ werden in StarLogo nicht nur Schildkröten genannt. Diese Bezeichnung wurde aus dem ursprünglichen Programm Logo übernommen und steht ganz allgemein für Agenten, egal ob Ameisen, Moleküle oder Menschen simuliert werden. Die Anordnung der Simulationen in StarLogo erfolgt auf einem Raster, der die Welt darstellt. Die Felder dieses Hintergrunds werden als „Patches“ bezeichnet. „Observer“ schließlich umfasst alle Vorgänge, die nicht im Einflussbereich einzelner Agenten stehen, wie zum Beispiel das Erzeugen von

Agenten oder statistische Auswertungen über alle Agenten hinweg. Nach einer kurzen Beschreibung des Interfaces folgt in Punkt 4 die Erstellung eines eigenen Projektes. Schon nach Eingabe der ersten Befehle beginnen sich die Turtles durch die Simulationsumgebung zu bewegen. Weiters finden sich die Beschreibung zu weiteren Befehlen sowie zur Verwendung von Funktionen im Einführungstext.

Dokumentation

Die installierte Dokumentation ist identisch mit der online zur Verfügung stehenden Dokumentation. Der Einstieg fällt mit den Abschnitten „Getting Started“ und „Tutorial“ sehr leicht. Eine Gesamtauflistung aller möglichen Befehle mit Beschreibung und eine FAQ-Sammlung sind ebenfalls vorhanden.

Beispielbibliothek

Am Ende der oben beschriebenen einführenden Dokumentation findet sich der Hinweis, dass die Beschäftigung mit Beispielprojekten der beste Weg ist, um StarLogo zu erlernen. Die mitgelieferten Beispiele sind aus den Bereichen Biologie, Mathematik, Physik und Soziale Systeme. Die Beispiele sind Teil der Standardinstallation und sind leicht zu öffnen. Der Programmcode der Beispiele ist offen einsehbar und kann zu Versuchszwecken beliebig verändert werden. zeigt das Beispielprojekt „Rabbits“ aus dem Bereich Biologie.

Motivation, Erfolgserlebnisse

Nach Starten des Programms wird nur ein Befehl benötigt, den die BenutzerInnen in das richtige Fester („Turtle Command Center“) eingeben müssen, damit sich am Bildschirm etwas zu bewegen beginnt. Die Interaktion mit StarLogo produziert also auch für EinsteigerInnen schon nach wenigen Minuten positive Rückmeldungen. Die einfach gehaltenen Beispiele mit den klaren Interaktionsmechanismen motivieren zusätzlich, durch experimentelle Veränderungen die Simulation zu beeinflussen und zu erforschen.

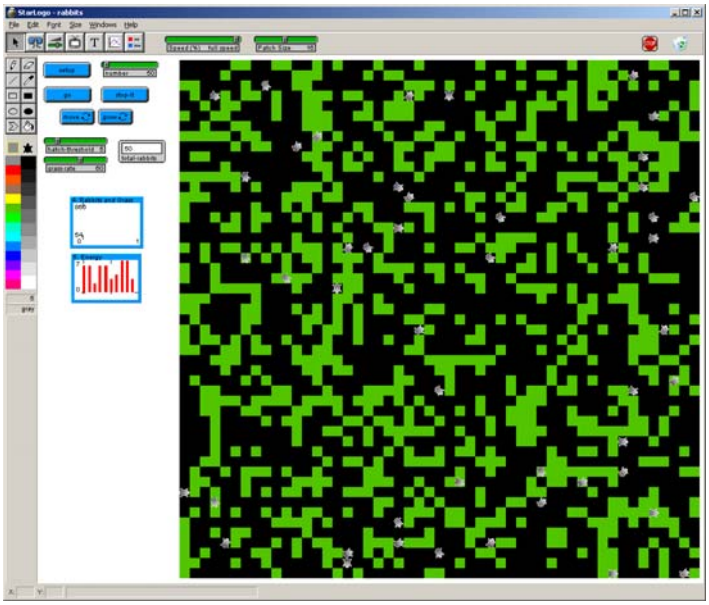

Abbildung 5.2: Beispielprojekt „Rabbits“ in StarLogo

Gesamteindruck

Der Gesamteindruck im Rahmen der Fragestellung des Einsatzes im didaktischen Kontext kann als sehr zufriedenstellend beurteilt werden. Das Programm ist übersichtlich gestaltet und wirkt überschaubar und damit erlernbar. Die Programmiersprache von StarLogo ist klar und für EinsteigerInnen in die Programmierung gut geeignet. Ein Nachteil aus der proprietären Sprache ist die fehlende Kompatibilität mit anderen Systemen. Die grafische Oberfläche von StarLogo ist schlicht gehalten. Zudem wirkt StarLogo aufgrund der Ablaufgeschwindigkeit der Beispielsimulationen als weniger gut geeignet für Simulationen mit sehr vielen Akteuren.

Eignung für den Unterricht

StarLogo ist sehr gut für den Einsatz im Unterricht geeignet. Als Zielgruppe können Studierende aller Fächer genannt werden. Die Einfachheit der Bedienung und die raschen motivierenden Interaktionen auch ohne Programmierkenntnisse legen den Einsatz auch im schulischen Bereich nahe. Zudem geben die AutorInnen von StarLogo als eine zentrale Motivation zur Erstellung ihrer Software an, Kindern und Jugendlichen den Einstieg in Programmierung und Simulation

erleichtern zu wollen. Für den Einsatz an Schulen wird das Lehrbuch [41] empfohlen, eine Abbildung der Turtles aus diesem Lehrbuch findet sich in Abbildung 5.3.

Abbildung 5.3: Turtles aus dem StarLogo Lehrbuch [41]

5.2 Repast

Die Bezeichnung Repast [47] ist ein Abkürzung für „Recursive Porous Agent Simulation Toolkit“. Repast bzw. Repast Simphony, wie der eigentliche Name der aktuell verfügbaren Version lautet, ist eine open-source agentenbasierte Programmierumgebung die verspricht, Modellbildung und Simulation zu vereinfachen (vgl. Abbildung 5.4). Repast ist vollständig objektorientiert und in Java entwickelt. Repast ist sehr an die fensterzentrierte Entwicklung („point-and-click“) orientiert, wie diese in modernen Betriebssystemen vorherrschend ist. Details der Eigenschaften und des Verhaltens der Agenten werden in Java programmiert, dennoch ist es mit der Repast Entwicklungsumgebung möglich, Agenten auch ohne Programmierkenntnisse zu erzeugen und diese in Simulationen agieren zu lassen. Die EntwicklerInnen von Repast versprechen, dass ihre Software auch bei sehr großen Simulationen mit mehr als 100.000 Agenten noch gute Dienste leistet.

Eine weitere Stärke von Repast sind die Schnittstellen zu anderen Programmen, z.B. das Programmpaket für Statistik „R“, das wissenschaftliche Visualisierungspaket „VisAD“, die Mathematikumgebung „MATLAB“. Für Menschen, die sich mit sozialer Netzwerkanalyse beschäftigen, bietet Repast eine vollständige Integration der JUNG Netzwerkbibliothek [48] sowie eine Importfunktion für UCINET DL-Files [49].

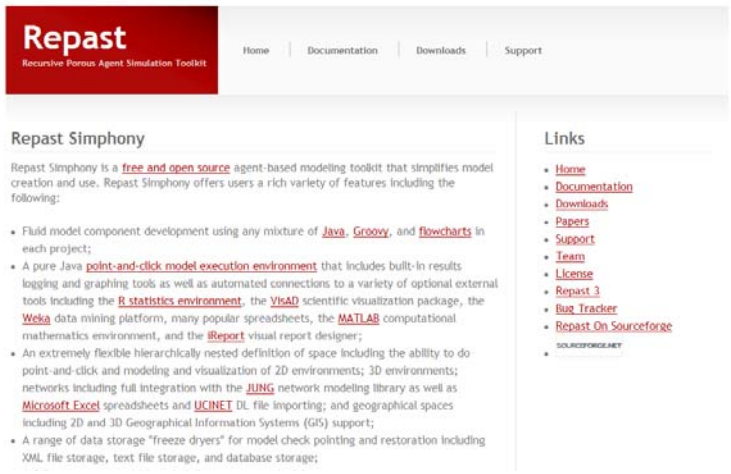

Abbildung 5.4: Startseite der Homepage von Repast [47]

Zugang, Kosten

Der Zugang bzw. das Auffinden der Software gestaltet sich als sehr einfach. Der erste Treffer mit dem Suchbegriff „repast“ im Internetsuchdienst Google zeigt auf das „Repast Agent Simulation Toolkit“ im Open Source Software Portal sourceforge.net. Im Downloadbereich findet sich der Link für den „Windows Installer“, der die 308 MB große Installationsdatei auf den lokalen Computer kopiert. Da Repast in Java programmiert ist, ist eine aktuelle Java Installation Voraussetzung für die Installation von Repast. Die Installation der Software verläuft in der Windows-Version voll automatisch und erfordert keinerlei Spezialwissen. Die Installation auf anderen Betriebssystemen ist in der Dokumentation [50] ausführlich beschrieben.

Erste Schritte

Repast läuft in der Eclipse Entwicklungsumgebung, was für geübte JavaProgrammiererInnen ein Vorteil ist, jedoch für den Neuling in Eclipse eine Hürde darstellt. In jedem Fall ist das Kapitel 2 „Creating a Network Infrastructure Model“ der Dokumentation [50] als Einstieg zu empfehlen. Ein vergleichbares Beispiel findet sich auch unter [51], dies beschreibt jedoch die erforderlichen Arbeitsschritte stark verkürzt und für NeueinsteigerInnen ungeeignet. Der erste Versuch, in die Simulationsumgebung im Rahmen der vorliegenden Arbeit einzusteigen, erfolgt nach dem oben erwähnten Beispiel in Kapitel 2 der

Dokumentation. Ziel des Übungsbeispiels, das sehr detailliert Schritt für Schritt vorgeführt wird, ist es, ein Infrastrukturmodell bestehend aus mehreren Gasspeicherstellen, die durch Pipelines miteinander verbunden sind, aufzubauen und zu simulieren. Die Reihenfolge der Arbeitsschritte der Erstellung einer Simulation in Repast kann wie folgt zusammengefasst werden:

- Erstellen eines neuen Projekts in Repast
- Definition der Modelleinstellungen
- Erzeugen der Agenten und Definition des Verhaltens der Agenten
- Erstellung der Simulationsumgebung und der Simulationsanordnung
- Einstellung der Darstellungsoptionen
- Simulation

Abbildung 5.5 zeigt Ansichten aus Repast aus dem oben erwähnten Übungsbeispiel der Dokumentation. Im linken Bild wird die Definition von Eigenschaften und Verhalten eines Akteurs vorgenommen. Das rechte Bild zeigt einen Simulationslauf mit sieben hintereinander verbundenen Akteuren.

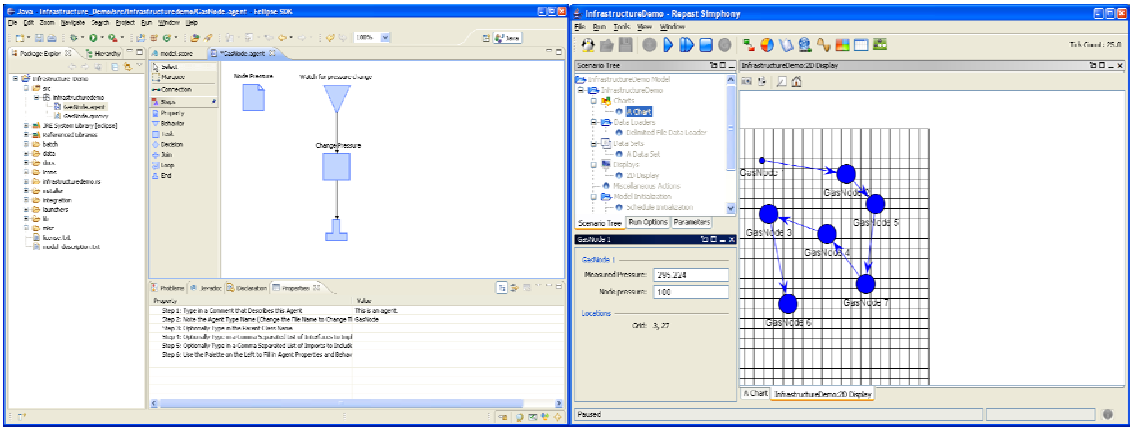

Abbildung 5.5: Simulationssoftware Repast

Dokumentation

Die Online-Dokumentation der Software wirkt umfangreich. Vor allem die FAQ sowie die Dokumentation, die als PDF vorliegt [50], erleichtern neuen UserInnen den Einstieg in die Entwicklungsumgebung und machen mit den Grundgedanken,

denen Repast folgt, vertraut. Eine Problematik der Dokumentation ist, dass diese sich zum Testzeitpunkt nicht auf die aktuelle Version der Software bezogen hat. Die dadurch feststellbaren Abweichungen zwischen Dokumentation und Programmoberfläche können für EinsteigerInnen Verwirrung erzeugen.

Beispielbibliothek

Laut Homepage gibt es für Repast Modellbibliotheken für die Bereiche genetische Algorithmen, neuronale Netze, Regressionen, Erzeugung von Zufallszahlen und spezielle Bereiche der Mathematik. Da diese Modelle Java-Klassen sind, müssen diese in Eclipse eingebunden werden, was für mit Eclipse unerfahrene Menschen zu einer Herausforderung werden kann.

Motivation, Erfolgserlebnisse

Die Gruppe der BenutzerInnen ohne Erfahrung mit der Entwicklungsumgebung Eclipse, zu der auch der Autor der vorliegenden Arbeit gehört, hat neben der Aneignung der Fertigkeiten zur Kontrolle der Simulationsumgebung die zusätzliche Hürde der Eclipse-Plattform zu bewältigen. Bei Fehlermeldungen kann so zum Beispiel nicht klar sein, ob diese aufgrund eines Java Problems (falsche Pfade für Klassen usw.) oder eines Fehlers in der Simulation entstehen, was die Fehlersuche erschwert bis unmöglich macht. Im Rahmen des hier durchgeführten Softwaretests ist es nicht gelungen, innerhalb von zwei Stunden eine ablauffähige Simulation zu erstellen. Als Nachteil für die Motivation kann außerdem festgehalten werden, dass sehr viele Schritte notwendig sind, bevor die erste sehr einfache Simulation gestartet werden kann.

Gesamteindruck

Das Tool wirkt sehr umfangreich, aber gut organisiert. BenutzerInnen mit Erfahrungen in der Programmierung mit Java, vor allem mit Eclipse, haben bei Repast einen entscheidenden Startvorteil. Als wesentlicher Vorteil dieser Entwicklungsumgebung kann die Skalierbarkeit für unterschiedliche UserInnengruppen festgestellt werden. EinsteigerInnen haben die Möglichkeit, fast ausschließlich

durch Mausklicks sehr einfache Simulationen zu erzeugen. Dies gilt jedoch nicht mehr für die logischen Komponenten, die zwar als Grafik angeordnet werden, dann jedoch eine schriftliche Ausfüllarbeit erfordern, die nicht als intuitiv bezeichnet werden kann. Die auf den ersten Blick ansprechende grafische Darstellung der logischen Komponenten (vgl. Abbildung 5.5) kann auch als aufwändig und umständlich empfunden werden.

Für fortgeschrittene UserInnen öffnet die Einbettung in die bestehende Programmiersprache Java fast unbegrenzte Möglichkeiten der Komplexitätssteigerungen des zu simulierenden Systems. Der Vorteil der offenen Java/Eclipse-Plattform kann sich für einen schnellen Einstieg von nicht Java/Eclipse versierten Menschen aber als Hauptschwierigkeit erweisen. Eine Aneignung der Programmiersprache Java oder aber auch der Eclipse-Umgebung als Voraussetzung für die eigentliche Verwendung von Repast kann im Kontext der Transaktionskosten als sicher zu aufwändig eingestuft werden.

Eignung für den Unterricht

Für den Unterricht ist Repast im akademischen Bereich geeignet und hier vor allem in technischen oder naturwissenschaftlichen Studienrichtungen. Die Komplexität der Software und die Einbettung in Eclipse erschweren den Einstieg für UserInnen, welche wenig Erfahrung mit vergleichbaren Entwicklungsumgebungen oder mit Programmierung haben. Dennoch sollte es mit Repast möglich sein, im Rahmen einer Lehrveranstaltung konkrete Projekte zu entwickeln. Voraussetzung dafür sind jedoch technisch versierte ÜbungsleiterInnen, die bei der Einrichtung der Umgebung und der Erstellung der ersten Beispiele anwesend sind. Technisch weniger versierte UserInnen können im Selbststudium auf unüberwindbare Hindernisse stoßen, lange bevor die eigentlichen Simulationsdurchläufe beginnen.

5.3 AnyLogic

AnyLogic ist nach Eigenbeschreibung [52] das erste und einzige dynamische Simulationstool, das System Dynamics, ereignisorientierte Simulation und agentenbasierte Simulation innerhalb einer Softwareumgebung vereint und kann daher auch als multiparadigma Modellierungstool verstanden werden. Auch ist ein Wechsel zwischen diesen Paradigmen innerhalb einer Simulation möglich. AnyLogic wird seit 1991 vom Unternehmen XJ Technologies in St. Petersburg, Russland entwickelt. AnyLogic ist für sehr große agentenbasierte Simulationen geeignet. Laut Selbstdarstellung ist die Version 6 bis zu 20 Mal schneller als ihre Vorgängerversion sowie auch effizienter in der Speicherverwaltung. Damit soll es möglich sein, auf den aktuell verfügbaren Standard-PCs Simulationen mit mehreren millionen Agenten laufen zu lassen.

Die Software ist in Java implementiert und vollständig objektorientiert. Java ist auch die Sprache, mit der im Rahmen dieser Simulationsumgebung Simulationen beliebig erweitert werden können. AnyLogic Simulationen werden als Java Code kompiliert und können daher in andere Anwendungen eingebunden werden.

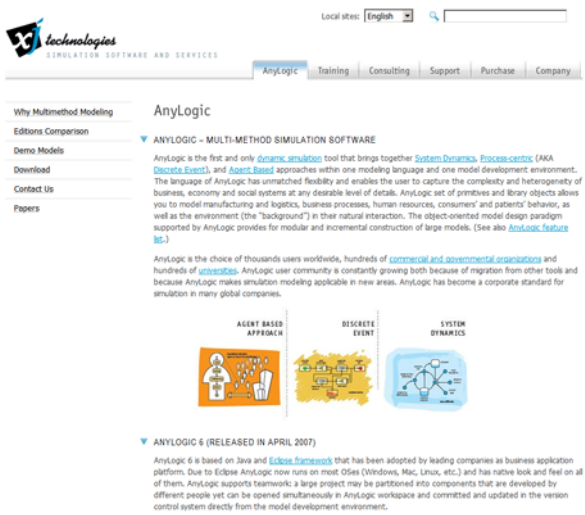

Abbildung 5.6: Projektseite der Software AnyLogic [52]

Zugang, Kosten

Der erste Treffer in verschiedenen Suchdiensten bei Eingabe des Titels der Software führt direkt zur Projekthomepage von AnyLogic [52] der XJ Technologies Company. AnyLogic bietet drei verschiedene Vollversionen der Software an, die sich in unterschiedlichen Preisen niederschlagen. Die kostengünstigste Variante ist mit $ 330,- eine einfache Lizenz der „Educational“ Version, die „Advanced“ Version kostet $ 4.800,-, die „Professional“ Version $ 11.990,-. Für MitarbeiterInnen der Technischen Universität Wien bietet der Zentrale Informatikdienst (ZID) eine Lizenz für EUR 13,- pro Quartal.

AnyLogic ist auch in einer 15 Tage gratis Testversion verfügbar. Der folgende Softwaretest wird in dieser Version durchgeführt. Unter dem Link für den Download findet sich eine Registrierungsseite. Nach Registrierung wird der Link für den Download per E-Mail zugeschickt. Die Installation der Software verläuft automatisch und erfordert keine zusätzlichen Aktivitäten.

Erste Schritte

Anleitungen für erste Schritte sind sowohl für die Konstruktion von agentenbasierten Simulationen als auch für System Dynamics Modelle auf der Projekthomepage verfügbar [52]. Im Rahmen des vorliegenden Softwaretests wird anhand des Dokuments „Agent Based Modeling Tutorial“ [53] der Einstieg in AnyLogic versucht. Als Beispiel wird das Bass Diffusionsmodell konstruiert, also der Lebenszyklus eines Produkts zur Abschätzung der Verkaufszahlen. Als erstes werden Personen als neue Objekte erzeugt, diese werden als die AkteurInnen der agentenbasierten Simulation definiert und in die Umwelt des Gesamtprojekts integriert. Als nächstes wird die Adoptionsrate, das ist die Wahrscheinlichkeit, dass Personen das Produkt kaufen, als Parameter dem Modell hinzugefügt. Den Zustand der Akteure definiert ein Zustandsgraph, in dem der Ausgangszustand den „potentiellen Käufer“ darstellt und bei einer bestimmten Kaufwahrscheinlichkeit in den Zustand des „Käufers“ übertritt. Die Einrichtung von Variablen und Datensatzelementen ist die Voraussetzung für das Einfügen von Diagrammen zur Beobachtung der Veränderung der Eigenschaft der Akteure.

Im zweiten Teil des Einführungsbeispiels wird Kommunikation zwischen den Akteuren dem Modell hinzugefügt um so Effekte der Mund-zu-Mund Propaganda zu implementieren. Abbildung 5.7 zeigt im Vordergrund die Abbildung der Diffusionskurve aus dem Ablauf der Simulation und im Hintergrund den Zustandsgraphen der Agenten innerhalb der Simulation.

Für das Einführungsbeispiel existieren mehrere Dateien, die den Fortschritt der Modellkonstruktion in unterschiedlichen Stadien darstellen und auf die im Falle von nicht behebbaren Problemen zurückgegriffen werden kann. Dies kann leicht geschehen, weil schon bei diesem einfachen Beispiel sehr viele Definitionen vorgenommen werden müssen. Im Rahmen des vorliegenden Tests ist es jedoch nach einigen Ausbesserungen gelungen, das Beispiel fehlerfrei zu kompilieren und ablaufen zu lassen.

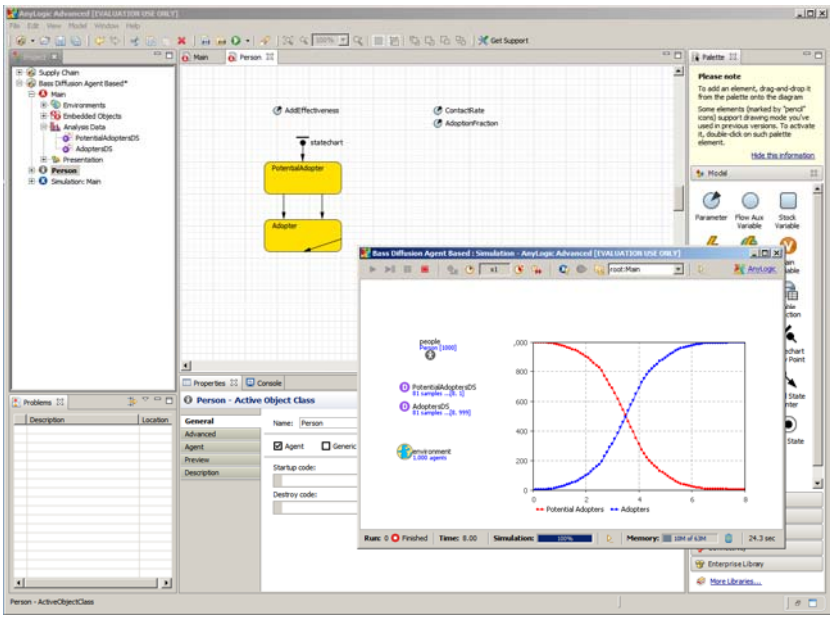

Abbildung 5.7: Agentenbasierte Simulation in AnyLogic

Dokumentation

Die Dokumentation ist Teil des Programms und kann im Menü aufgerufen werden. Im Downloadbereich der Projektseite [52] finden sich zusätzlich einige Dokumente zur Einführung in AnyLogic allgemein, sowie Einführungen zur Simulation von agentenbasierten Simulationen und von System Dynamics Modellen.

Die Dokumentation für EinsteigerInnen ist sehr detailliert und gut nachvollziehbar.

Beispielbibliothek

Ca. 50 Modellbeispiele sind Teil der Installation und können über das Startfenster aufgerufen werden. Diese Beispiele sind aus unterschiedlichen Bereichen und beinhalten auch einige bekannte Standardbeispiele für Simulationen. Die Beispiele sind kurz beschrieben und lassen sich ohne zusätzlichen Aufwand öffnen und ausführen. Die Beispiele laden zum Experimentieren ein, da sie interaktiv gestaltet sind.

Motivation, Erfolgserlebnisse

AnyLogic ist ein anspruchsvolles Tool. Doch im Gegensatz zum von der Entwicklungsumgebung vergleichbaren Programm Repast (siehe Abschnitt 5.2) funktioniert AnyLogic auf Anhieb und bedarf keiner zusätzlichen Java-Einstellungen oder sonstiger Adaptionen. Bis zum Ablauf der ersten agentenbasierten Simulation in AnyLogic sind viele Elemente und Einstellungen notwendig, die für AnfängerInnen nicht intuitiv sind. Es sollte jedoch der Mehrzahl der BenutzerInnen möglich sein, die Einführungsbeispiele erfolgreich zu absolvieren. Motivierend zum Experimentieren sind vor allem die Beispielmodelle, die zur Interaktion einladen.

Gesamteindruck

Obwohl AnyLogic in einer typischen Umgebung für Java Programmentwicklung auftritt, entstehen dadurch keine zusätzlichen Transaktionskosten, da nach Installation das Tool sofort einsatzfähig ist. Das Programm ist beeindruckend und für all jene empfehlenswert, die professionell Simulationen erstellen wollen. Der Einstieg ist nicht trivial, zum Beispiel erfordert die einfache Anzeige von statistischen Werten aus der Simulation für nicht in objektorientierter Programmierung geübte BenutzerInnen komplizierte Formulierungen.

Eignung für den Unterricht

Die Logik der objektorientierten Modellierung und Programmierung ist ein Grundgedanke für die Konstruktion von Modellen in AnyLogic. Studierende, die Vorbildung in diesem Bereich haben, schaffen den Einstieg in dieses Tool sicher um einiges schneller. Für den Einsatz im didaktischen Kontext ist AnyLogic daher vor allem in technischen Studienrichtungen geeignet. Für Studierende ohne Erfahrungen in Programmierung wird die Welt von AnyLogic zu aufwändig sein. Den eigentlichen Hindernisgrund für den Einsatz in der Lehre stellen jedoch die hohen Kosten dar. Das Beispiel der TU Wien (vgl. Abschnitt Kosten) zeigt jedoch, dass die EntwicklerInnen von AnyLogic Bereitschaft zeigen, mit Bildungseinrichtungen Kooperationen für den Erwerb günstigerer Einzellizenzen einzugehen.

5.4 Stella

Die Kurzbezeichnung Stella steht für „System Thinking Educational Learning Laboratory with Animation“ und trägt damit im Titel die Intention des Programms, im Bereich des Lernens eingesetzt zu werden. Stella war das erste verfügbare benutzerfreundliche Tool für die grafische Implementierung von System Dynamics Modellen. Laut Eigenbeschreibung [54] bietet Stella die Möglichkeit zu visualisieren und zu vermitteln, wie dynamische Systeme funktionieren. Stella wendet sich als Zielgruppe an Lehrende, Studierende und ForscherInnen.

Zugang, Kosten

Aufgrund der Bezeichnung der Programmierumgebung reicht es nicht aus, nur diese in Standardsuchdiensten als Suchbegriff einzugeben. Mit dem Zusatz „simulation“ findet man jedoch schnell zur Projekthomepage von „isee systems“, einer amerikanischen Firma, die in ihrer Selbstdarstellung als Unternehmensziel „to improve the way the world works“ angibt. Der wirtschaftliche Hintergrund von Stella schlägt sich auch in den Bezugskosten nieder. Je nach gewünschtem

Einsatzgebiet kostet eine Lizenz der aktuellen Version 9.1 von Stella bis zu $ 1.899,-. die Studierendenlizenz kostet immerhin auch noch $ 129,-. Für EDV-Labors gibt es die „Lab Pack License“, die für die ersten 20 Arbeitsplätze je $ 150,- kostet. Es gibt eine eingeschränkte „Save-Disabled Version“ um $ 50,-.

Es wird jedoch auch eine Trial-Version der „Save-Disabled Version“ zum gratis Download angeboten. Diese ist 30 Tage gültig. Aus Kostengründen wurden die folgenden Tests mit dieser Testversion durchgeführt. Das Installationsprogramm entspricht Standards, wie sie von Windows-Programmen bekannt sind, womit das Programm ohne Probleme installierbar ist.

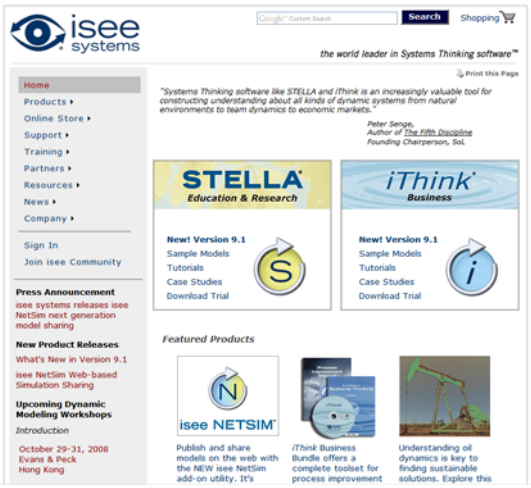

Abbildung 5.8: Homepage der Programmierumgebung Stella [54]

Erste Schritte

Durch die Implementierung von Stella als eigenständig ausführbares Exe-Programm sind keinerlei sonstige Installationen oder Adaptionen nötig, um das Programm ablaufen zu lassen. Als erstes Beispiel wird im Rahmen des vorliegenden Tests versucht, das „Population Model“ aus Abschnitt 6 aus der Einführung von Leslie A. Martin [31] zu erstellen. Das Übungsprojekt ist in Version 2 von Stella erstellt, jedoch sind zur Version 9 optisch nur marginale Änderungen erkennbar. Stella besteht aus drei verschiedenen Ebenen, die höchste Ebene für die Präsentation der fertigen Modelle, die zweite Ebene für die Konstruktion der Modelle und die dritte Ebene für die mathematischen Gleichungen, die im Modell

verwendet werden. Diese Ebenen können hierarchisch verstanden werden und sind im User-Interface des Programms übereinander angeordnet.

Die einzelnen Elemente für die Konstruktion von System Dynamics Modellen (vgl. Abschnitt 3.3) finden sich in der Symbolleiste gut sichtbar angeordnet. Der Ablauf der Erstellung eines Modells in Stella verläuft in folgenden Schritten:

1. Anordnung der Elemente der Simulation
2. Definition der Startwerte für stocks und der Flussfunktionen für flows
3. Einrichtung eines Graphen zur Beobachtung der Simulation
4. Berechnung der Simulation

Das Beispiel aus der Einführung von Martin [31] ist sehr detailliert und erfordert keinerlei Vorwissen. Im Rahmen der Erklärungen werden zudem die Grundlagen von System Dynamics erklärt. Das Beispiel konnte im Rahmen des vorliegenden Tests in kurzer Zeit nachvollzogen werden und konnte damit vollständig implementiert werden (vgl. Abbildung 5.9).

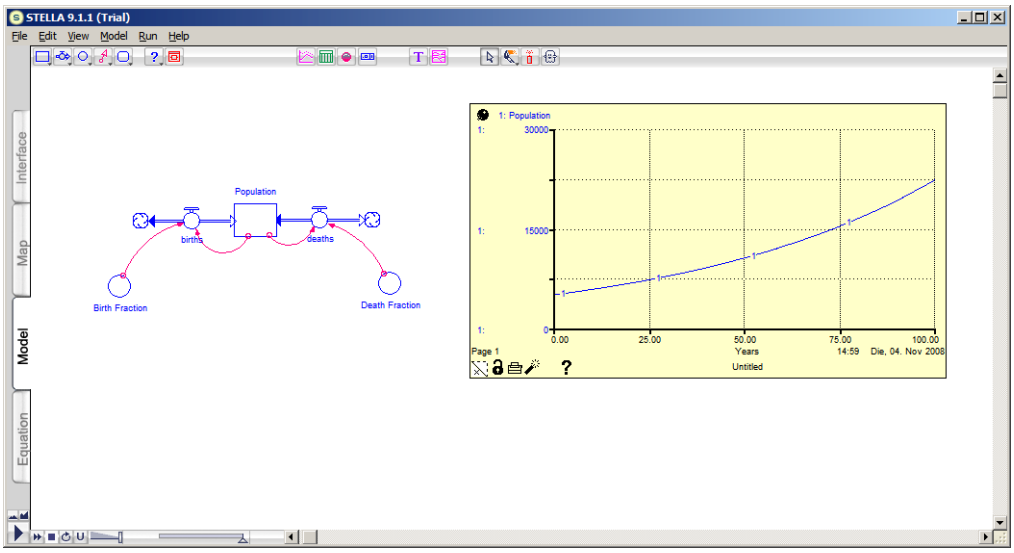

Abbildung 5.9: Einführungsbeispiel aus [31] in Stella

Dokumentation

Auf der Projektseite [54] finden sich ein „Tutorial“ und „Case Studies“. Da diese Software seit den 1980-er Jahren eine weite Verbreitung gefunden hat, finden sich im Web unzählige Einführungen und Erklärungen. Explizit soll an dieser Stelle das Projekt „Road Maps“ (vgl. z.B. [31][55]; der Link zu der gesamten Sammlung der Road Maps findet sich in den Linksammlungen im Abschnitt 8.1) genannt werden, das unter der Anleitung von Forrester entstandenen ist. Die oben angeführten ersten Schritte wurden nach diesen Unterlagen durchgeführt.

Beispielbibliothek

Die Projektseite von Stella [54] bietet ca. 20 Modellbeispiele aus verschiedenen Bereichen. Ausführliche Beschreibungen finden sich in der Modelldatei selbst. Auf einer den oben vorgestellten drei Ebenen des Modells hinzugefügten vierten Ebene „Interface“ wurden sehr anschauliche und interaktive Beschreibungen erstellt. Eine weitere Quelle für Beispieldateien unterschiedlichster Art stellt das Internet dar. Durch den Vorteil der großen Verbreitung von Stella finden sich dort viele System Dynamics Beispiele, die in Stella implementiert sind.

Motivation, Erfolgserlebnisse

Der optische Eindruck von Stella, eines einfachen Programms, unterstützt die Motivation, damit zu arbeiten. Alle für die Konstruktion und die Veränderung eines Modells notwendigen Elemente finden sich in der Taskleiste. Mit Stella ist es möglich, nach einer kurzen Einführung in nur wenige Begriffe von System Dynamics innerhalb von Minuten ein Modell zu konstruieren und ablaufen zu lassen. Durch das Funktionieren einfacher Modelle und die wenigen Elemente werden BenutzerInnen von Stella motiviert, selbst Veränderungen und Erweiterungen an Modellen vorzunehmen. Die Simulationsumgebung stellte zu keinem Zeitpunkt der Testung ein Hindernis für die Exploration von Modellen dar, so ist zum Beispiel während des gesamten Testlaufes der oben beschriebenen ersten Schritte keine einzige Fehlermeldung zu verzeichnen gewesen.

Gesamteindruck

Stella wirkt sehr gut überschaubar, da die Anzahl der Buttons und Menüfunktionen gering ist. Obwohl die neuen Versionen viele Zusatzmöglichkeiten der Präsentation und Interaktion anbieten, bleiben im Kern für die Erstellung eines Modells die vier Grundelemente des System Dynamics, wie sie in Abschnitt 3.3 vorgestellt wurden. Außer einem Grundwissen in System Dynamics, das jedoch mit Hilfe der Dokumentation auch im Selbststudium erlernt werden kann, sind für die Verwendung von Stella weder Vorkenntnisse notwendig, noch entstehen aus der Beschäftigung mit Stella irgendwelche zusätzlichen Transaktionskosten.

Eignung für den Unterricht

Der Untertitel der „Road Maps“ von Forrester lautet „System Dynamics in Education Project“. Diese Unterlagen wollen kein Lehrplan sein, sondern das Selbststudium von System Dynamics und Stella unterstützen. Dennoch sind diese für den Einsatz im Unterricht hilfreich, da es damit auch möglich ist, System Dynamics zu lernen und zu verstehen, ohne zu wissen, wie man Differentialgleichungen löst. Stella ist gut für den Unterricht geeignet, da es unkompliziert zu verwenden ist und Erfolgserlebnisse innerhalb kürzester Zeit zu erzielen sind. Als Zielgruppen sind Studierende aller Fächer geeignet und wahrscheinlich auch mathematisch interessierte Schülerinnen und Schüler.

Das zentrale Problem für den Einsatz im Unterricht stellen die Kosten dar. Ein Erwerb wäre auf Instituts- oder Bildungseinrichtungsebene für ein EDV-Labor notwendig, hätte aber damit auch den Nachteil, dass Studierende die Software nicht auf ihren eigenen Computern ausführen können.

5.5 Vensim

Vensim ist ein Tool zur visuellen Konstruktion von System Dynamics Modellen und wird von der in Harvard ansässigen Firma Ventana Systems, Inc. entwickelt. Die Bezeichnung Vensim ist die Abkürzung für „Ventana Simulation Environ-

ment“. Nach Angaben auf der Vensim Projekthomepage [56] sind die Vorteile von Vensim im Gegensatz zu anderen System Dynamics Simulationsumgebungen zum einen die bessere Integration von Kausaldiagrammen und zum anderen verbesserten Analyseverfahren, unter anderem zum schnelleren Finden von Problemen in den Modellen. Vensim ist zudem optimiert für die Analyse sehr großer komplexer Systeme.

Abbildung 5.10: Projektseite der Simulationsumgebung Vensim [56]

Zugang, Kosten

Der erste Treffer bei Eingabe von „Vensim“ als Suchbegriff im Suchdienst Google führt zur Projektseite [56]. Die umfassendste Version von Vensim kostet $ 1.995,-. Die im Funktionsumfang eingeschränkte PLE Version (Personal Learning Edition) ist ab $ 50,- erhältlich. Für den Einsatz im didaktischen Kontext ist diese PLE Version gratis. Nach Registrierung kommt man zur eigentlichen Downloadseite, die Downloads für Windows und für Macintosh anbietet. Die Windowsversion der Installationsdateien wirkt veraltet, so ist etwa von sechs verschiedenen Disketten die Rede, die man zur Installation runterladen muss. Die eigentliche Installation benötigt dann jedoch keine Disketten startet von der lokalen Festplatte.

Erste Schritte

Vensim startet aus dem Windows-Menü ohne zusätzliche Einrichtungen oder Ergänzungen. Als Einführungsbeispiel werden die Kapitel 4 „Causal Loop Di-

agramming“ und 6 „Building a Simulation Model“ aus der Dokumentation [29] gewählt.

Die Erstellung eines Simulationsprojektes in Vensim folgt in der Regel folgenden Schritten, wobei in Klammer die jeweils verwendeten Zusatzprogramme von Vensim angegeben sind:

- Konstruktion des Modells
- Analyse der Struktur des Modells (Tree Diagrams, Loop Tool, Document Tool)
- Simulation mit Veränderung der Parameter zur Erkundung des Modells
- Detaillierte Analyse im Detail (Dataset Analysis Tool)
- Kontrollierte Simulationen zur Verbesserung des Modells
- Präsentation des Modells (SyntheSim)

Das Beispiel für das Kausaldiagramm modelliert die Abläufe eines Projekts im Zusammenhang mit der Menge der zu erledigenden Arbeit und der eingesetzten Arbeitskräfte (vgl. Abbildung 3.4 im Kapitel 3.3 Kausaldiagramme). Im ersten Schritt erfolgt die Konstruktion des Kausaldiagramms nach der sehr detaillierten Anleitung. Das verläuft ohne Probleme, zudem ist die Verwendung der Elemente der Softwareumgebung sehr intuitiv und schließt an das Wissen an, das BenutzerInnen in Office-Anwendungen erlernt haben. Eine Abbildung des hier erstellten Kausaldiagramms findet sich in Abschnitt 3.3 auf Seite 39. Im nächsten Schritt der Anleitung werden verschiedene Strukturanalyse Werkzeuge von Vensim vorgestellt, von denen jedoch ca. die Hälfte in der gratis Version nicht frei geschalten ist.

Als Einführungsbeispiel zur Erstellung einer System Dynamics Simulation verwendet auch die Dokumentation von Vensim [29] das bereits bekannte Modell der Populationsentwicklung, dieses Mal für eine Hasenpopulation. Nach Eingabe der notwendigen Elemente biete Vensim die Möglichkeit einer automatischen Überprüfung des Modells. Ergebnisse der Simulation können tabellarisch oder als

Diagramm dargestellt werden (vgl. Abbildung 5.11). Vensim bietet für erste Analyseschritte von System Dynamics Modellen viele kleine hilfreiche Tools, die auch in der Gratisversion verfügbar sind (z.B. einfacher Vergleich von Durchläufen mit unterschiedlichen Variablen, Schieberegler für die Veränderung von Variablen).

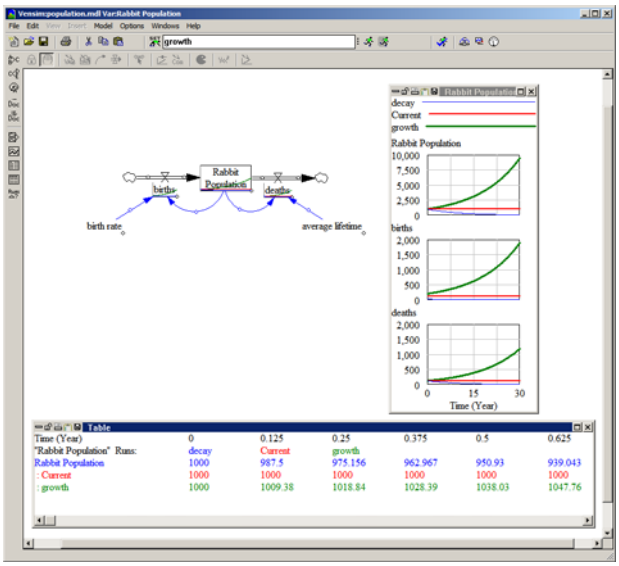

Abbildung 5.11: Das Populationsmodell in Vensim

Dokumentation

Bei der Installation können optional als Komponenten Dokumentation und Modellbeispiele installiert werden. Die über 300 Seiten starke Dokumentation [29] findet sich auch auf der Homepage von Vensim. Diese Dokumentation unterstützt gut den Einstieg in Vensim und gibt viele Hinweise und Anmerkungen zu Modellkonstruktion und Simulation im Allgemeinen.

Vensim wird als mögliche Alternative zu Stella in den Road Maps von Forrester [31] genannt. Daher gibt es auch seit dem Jahr 2000 eine Erweiterung der Road Maps für die Implementierung der Beispiele in Vensim [57].

Beispielbibliothek

Im Rahmen der Installation werden einige Vensim Modelle automatisch installiert. Diese Modelle sind aus unterschiedlichen fachlichen Bereichen. Leider konnte zu den einzelnen Modellen keine Dokumentation gefunden werden. Wei-

ters finden sich alle Beispiele aus der offiziellen Dokumentation [29] im Installationsverzeichnis der Software.

Motivation, Erfolgserlebnisse

Vensim wirkt auf den ersten Blick komplizierter als Stella. Doch das Konstruieren von Kausaldiagrammen oder System Dynamics Modellen mit Vensim erinnert an die Verwendung von Word oder PowerPoint aus dem Microsoft Office-Paket oder vergleichbaren Programmen. Die beiden oben genannten Beispiele aus der Dokumentation konnten ohne Probleme implementiert werden. Die automatische Modellüberprüfung wirkt zudem hilfreich und motivierend, da sie Rückmeldung über die korrekte Eingabe der Modellelemente und Parameter gibt. Dies soll vor allem für EinsteigerInnen in Vensim ein langwieriges Suchen von zum Beispiel syntaktischen Fehlern verkürzen.

Gesamteindruck

Nach den ersten Eindrücken eines komplizierten veralteten Programms (z.B. Hinweis auf Disketten in der Installation) werden neue BenutzerInnen positiv von der Einfachheit und der Klarheit überrascht. Die Dokumentation ist klar verständlich und nachvollziehbar. Die verschiedenen Tools zur Analyse und zur Manipulation der konstruierten Modelle laden zum Ausprobieren ein.

Durch eine den weitverbreiteten Windows-Office-Anwendungen verwandte Benützung können die Elemente intuitiv richtig eingesetzt und manipuliert werden, wodurch keinerlei Transaktionskosten durch die Verwendung der Software entstehen.

Eignung für den Unterricht

Vensim ist für den Einsatz im Unterricht geeignet. Da keinerlei technische Voraussetzungen für den Einsatz notwendig sind, ist eine Verwendung für Studierende aller Fachrichtung möglich. Das Zeichnen von Kausaldiagrammen kann auch im schulischen Kontext zu Systembeschreibungen herangezogen wer-

den. Durch die PLE Version, die für den Einsatz im didaktischen Kontext gratis ist, spricht auch die monetäre Frage für den Einsatz von Vensim.

5.6 ExtendSim

ExtendSim ist eine Software zur Abbildung von Prozessen und demnach ein Tool zur Konstruktion und Simulation von ereignisbasierten Systemen. ExtendSim gibt in der Funktionsübersicht auf der Homepage des Projekts [58] an, dass damit auch agentenbasierte Simulationen durchgeführt werden können, im Rahmen dieses Tests wird nur der ereignisbasierten Ansatz berücksichtigt. ExtendSim kann durch Codes aus anderen Programmiersprachen (Delphi, C++, Visual Basic) erweitert werden. Die Konstruktion von Modellen in ExtendSim ist grafikorientiert. Das Zusammenstellen von Abläufen erfolgt daher zum Großteil mit der Maus und es ist keinerlei Programmierung notwendig. ExtendSim bietet auch die Möglichkeit, 3D-Modelle zu erstellen.

Abbildung 5.12: Projektseite der Software ExtendSim [58]

Zugang, Kosten

ExtendSim ist über Suchmaschinen leicht zu finden. Schon die Startseite der Homepage eröffnet potentiellen BenutzerInnen der Software, dass ExtendSim ein kommerzielles Produkt ist. Die Kosten für die einfachste Version der Software für

eine „Stand Alone License“ belaufen sich auf $ 995,-, für die teuerste Version müssen $ 4.995,- investiert werden. Für Bildungseinrichtungen werden Versionen ab $ 495,- angeboten. Bei der Abnahme von mindestens 5 Lizenzen für EDV-Labore kostet eine Lizenz $ 125. Es findet sich allerdings auch eine zeitlich unlimitierte, aber im Funktionsumfang eingeschränkte Demoversion zum Download. Hauptdefizit ist die fehlende Möglichkeit, selbst erstellte Modelle abspeichern zu können.

Die Installation der Software erfolgt automatisch und ExtendSim kann sofort nach Installation ausgeführt werden. Der vorliegende Softwaretest der Simulationsumgebung ist mit der Demoversion durchgeführt worden.

Erste Schritte

Im Handbuch [59] von ExtendSim findet sich gleich zu Beginn ein Kapitel „Building a Model“, mit dem ein Wasserspeicher simuliert werden soll. Folgende Schritte sind dafür, wie auch generell zur Erzeugung von Simulationen in ExtendSim, notwendig:

- Einstellung der Parameter der Simulation festlegen
- Konstruktion des Modells durch die Verwendung von vordefinierten Blöcken
- Eigenschaften der Blöcke definieren
- Dialogparameter festlegen

Als erstes werden in einem neuen leeren Projekt die globalen Variablen der Simulation definiert. Diese sind zum Beispiel die Anzahl der Zeitschritte, die die Simulation durchlaufen soll, sowie welche Zeiteinheit (z.B. Sekunden, Jahre) verwendet werden soll. Im nächsten Schritt werden die für das Modell benötigten Blöcke erstellt und miteinander verbunden. Die erste Quelle des Modells ist der Regen. In den Einstellungen wird der Niederschlag pro Monat eingetragen. Als zweite Quelle wird ein Bach definiert, der einen zusätzlichen zufälligen Zufluss zum Wasserspeicher erzeugt. Durch den Additionsblock werden die beiden Quel-

len zusammengefasst und als Input an den Wasserspeicher weitergegeben. Schließlich werden noch die einzelnen Blocks mit dem Grafikblock verbunden. Abbildung 5.13 zeigt das fertige Modell des Wasserspeichers auf der linken Seite sowie eine grafische Darstellung der Simulation mit den zwei Wasserquellen und dem anwachsenden Wasserspeicher auf der rechten Seite. Zur grafischen Darstellung wurden zwei verschiedene Skalen auf der y-Achse verwendet.

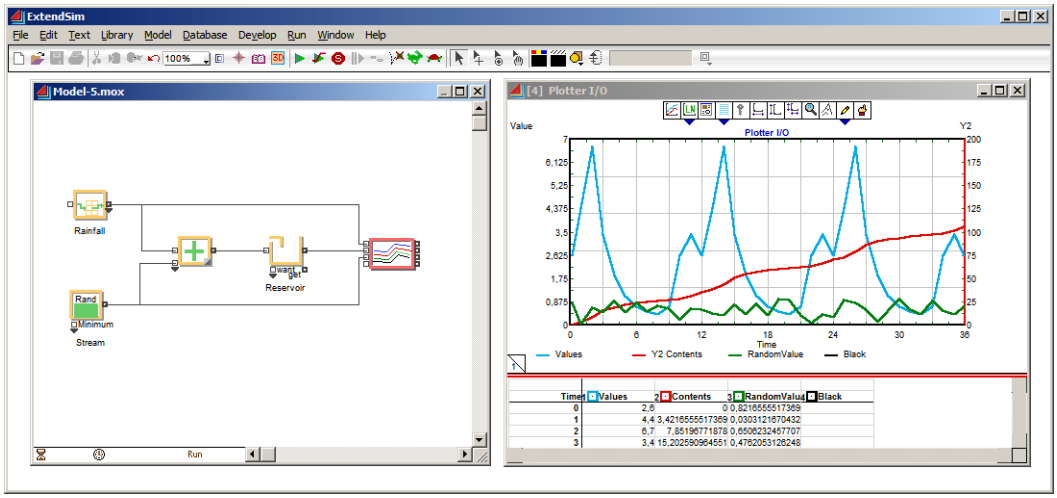

Abbildung 5.13: Wasserspeicher Modell in ExtendSim

Dokumentation

Das „ExtendSim User Guide and Developer Reference“ [59] findet sich auf der Homepage und besteht aus über 800 Seiten. Das Handbuch bietet einen guten Einstieg in die Software und zusätzlich eine Einführung in ereignisbasierte Simulationen. Einziger Nachteil der Dokumentation auf der Homepage ist, dass als Zugangsvoraussetzung eine erneute Registrierung notwendig ist. Die gesamte Dokumentation ist jedoch auch Teil der Installation und kann über das Windows-Startmenü erreicht werden. Zusätzlich zum Handbuch findet sich noch ein Verzeichnis sämtlicher Funktionen von ExtendSim als PDF.

Beispielbibliothek

Es finden sich nur vier Beispiele als Teil der Installation. Diese können beim Start von ExtendSim geöffnet werden. Auf der Projektseite finden sich keine weiteren

Beispiele, jedoch einer Verlinkung von Dutzenden wissenschaftlichen Publikationen und Projekten, die mit Hilfe von ExtendSim erstellt worden sind.

Motivation, Erfolgserlebnisse

Das Beispielprogramm aus der Dokumentation konnte ohne Probleme konstruiert werden. Das Erstellen des Modells erfolgt sehr grafikorientiert und intuitiv. Die Simulationsdurchläufe und die Adaptierungen des Modells verliefen ohne Fehler. Generell kann festgehalten werden, dass die Konstruktion von ereignisbasierten Modellen in ExtendSim ein einfach wirkender Vorgang ist. Die Umgebung wirkt für BenutzerInnen des Windows-Office-Pakets sehr vertraut, was positiv für die Motivation ist. Erfolgserlebnisse durch das Ablaufen einer Simulation stellen sich im Vergleich zu anderen Programmen schnell ein.

Gesamteindruck

ExtendSim ist ein benutzerfreundliches Programm, das es einfach ermöglicht, Prozessabläufe im Computer zu erstellen. Dieses Erstellen kann fast ausschließlich mit der Maus durchgeführt werden. Für EinsteigerInnen stellt dies eine maximale Einfachheit dar, da keine manuellen Eingaben von Programmcode oder mathematischer Funktionen notwendig sind. Dass sogar einfache Funktionen, wie das Zusammenzählen zweier Werte, durch grafische Objekte gelöst werden muss, kann jedoch für versierte BenutzerInnen von Simulationsumgebungen als umständlich empfunden werden.

Eignung für den Unterricht

Die hohen Kosten sind ein wesentlicher Grund, dass ExtendSim für einen ernsthaften Einsatz im didaktischen Kontext nicht geeignet ist. Dennoch ist diese Software für das Kennenlernen von ereignisorientierten Systemen gut geeignet, da von den BenutzerInnen keinerlei Vorwissen verlangt wird und auch nur eine kurze Einschulungsphase erforderlich ist. Als Zielgruppe sind Studierende aller Fächer geeignet. ExtendSim kann wahrscheinlich auch im schulischen Kontext eingesetzt werden.

5.7 Übersicht der Simulationsumgebungen

	StarLogo/NetLogo	**Repast**	**AnyLogic**
Art	agentenbasiert	agentenbasiert	System Dynamics, agentenbasiert, ereignisorientiert
Quelle	http://education.mit.edu /starlogo	http://repast. sourceforge.net	http://www.xjtek.com /anylogic
Zugang, Kosten	kostenloser Download von der Projekthomepage, problemlose Installation	kostenloser Download bei Sourceforge, opensource, problemlose Installation	kostenpflichtig (Educational Version $ 330,-), 15 Tage gratis Testversion, problemlose Installation
Erste Schritte	gute Einführung, sehr einfach funktionierende Beispiele	Beispielprojekt für AnfängerInnen, dennoch sehr aufwändig	gut nachvollziehbar, trotz kompliziertem Programm
Dokumentation	online und lokal installiert vorhanden	vorhanden, auch FAQ Sammlung vorhanden	online und als Teil des Programms
Modellbibliothek	Beispiele aus verschiedenen Bereichen werden lokal installiert, leicht zu starten	Beispielprojekte vorhanden aber schwer zu finden, Installation erfordert Eclipse Kenntnisse	ca. 50 Modellbeispiele in Programm über Startmenü, gute Beschreibung, leicht zu starten
Motivation, Erfolgserlebnisse	schnelle Erfolgserlebnisse, motiviert zum ausprobieren	für UserInnen ohne Erfahrungen mit Eclipse-Umgebung demotivierend	viele Schritte vor eigentlicher Simulation notwendig, dennoch erfolgreiche Umsetzung möglich
Gesamteindruck	einfach gehaltenes Tool, nicht für sehr große oder komplexe Simulationen geeignet	sehr aufwändiges professionelles Tool, das nicht für den schnellen Einstieg geeignet ist	anspruchsvolles und professionelles Tool, keine Transaktionskosten, eigene umfassende Simulationswelt
Eignung für den Unterricht	sehr gut geeignet für alle Studienrichtungen, auch für schulischen Einsatz geeignet	vor allem für technische und naturwissenschaftliche Studierende	eventuell für technische Studienrichtungen, jedoch sehr hohe Kosten

Tabelle 5.1: Zusammenfassung der Simulations-Tools, Teil 1

	Stella	**Vensim**	**ExtendSim**
Art	System Dynamics	System Dynamics, Kausaldiagramme	ereignisorientiert
Quelle	http://www.iseesystems.com	http://www.vensim.com	http://www.extendsim.com/
Zugang, Kosten	kostenpflichtig (z.B. Studierenden-Lizenz $ 129,-), gratis 30 Tage Save-Disabled Testversion, problemlose Installation	kommerzielle Version bis zu $ 1.995,-, gratis zeitlich unbeschränkte PLE Version für Testzwecke und für didaktischen Einsatz, problemlose Installation	kostenpflichtig (Standard $ 995,-. Bildungs-Version $ 495,-), gratis zeitlich unbegrenzte aber eingeschränkte Version, problemlose Installation
Erste Schritte	Einführung auf Projektseite, gute Einführung in „Road Maps“ (z.B. [31])	wirkt kompliziert, erweist sich aber als intuitiv, Einführung aus Dokumentation [29] ist nachvollziehbar	gut in Dokumentation integriert, Einführung ist nachvollziehbar
Dokumentation	online verfügbar, unzählige weitere Dokumentationen im Web	online und als Teil der Installation, Erweiterung der Road Maps [57] für Vensim	online und als Teil der Installation, sehr umfangreich, zusätzlich ein Funktionsverzeichnis
Modellbibliothek	ca. 20 Beispiele online, detaillierte Beschreibung dieser innerhalb der Stella-Datei	14 Beispiele als Teil der Installation, keine Beschreibung	nur 4 Beispiele als Teil der Installation, online viele Referenzen zu Projekte mit ExtendSim
Motivation, Erfolgserlebnisse	leichter Einstieg, Erfolgserlebnisse innerhalb von wenigen Minuten, motivierend	leichter Einstieg mit Dokumentation, intuitive Anwendung, schnelle Erfolgserlebnisse	leichter Einstieg, intuitive grafische Anwendung, schnelle Erfolgserlebnisse
Gesamteindruck	wirkt überschaubar, Designelemente sind klar, neue Versionen wirken komplexer wegen Interface-Elementen	komplexer als Stella, intuitive Anwendung, inkludierte Tools laden zum Experimentieren ein, größter Vorteil: gratis	benutzerfreundliches Programm, sehr grafikorientiert, auch inder Demo-Version geeignet für einfache Konstruktionen
Eignung für den Unterricht	gut geeignet für Studierende aller Fächer, Problem: hohe Anschaffungskosten	gut geeignet für Studierende aller Fächer, Kausaldiagramme auch im schulischen Kontext einsetzbar	gut geeignet für Studierende aller Fächer, auch in Schulen, Problem: hohe Anschaffungskosten

Tabelle 5.2: Zusammenfassung der Simulations-Tools, Teil 2

6 Zusammenfassung

… in allen Gemeinden, Städten und Dörfern irgendeines christlichen Reiches solche Schulen zu errichten, … die gesamte Jugend beiderlei Geschlechts ohne Ausnahme in den Wissenschaften unterwiesen … und zwar kurz, angenehm und gründlich … Unser Didaktik Stern und Steuer soll sein ein Verfahren zu suchen und zu finden, dass die Lehrenden weniger zu lehren brauchen, die Lernenden aber mehr lernen; dass es in den Schulen weniger Lärm und vergebliche Mühsal, aber mehr Ruhe, Lust und festen Erfolg gibt …

Jan Amos Comenius, 1657

Jan Amos Comenius (1592-1670) fordert in seiner „Didactica Magna“ 1657 „Alles Allen zu lehren“ [3]. Der Einsatz des computerunterstützten Lernens, vor allem des Instruments des Distanzlernens, nährt die Phantasien, dass diese Forderung heute erfüllbar ist. Lewis J. Perelman zum Beispiel skizziert, dass „Hyperlearning“ [60], wie er das durch neue Medien unterstützte Lernen nennt, wesentliche Probleme der Menschheit lösen und gleichzeitig das bisherige Bildungssystem ersetzen wird (Substitutionsthese). Dagegen hält Werner Sacher [61] in der Diskussion über computerunterstütztes Lernen den „flachen Positivismus“ für ein zentrales Problem. Damit ist der Glaube gemeint, dass der bloße Einsatz von Computern quasi automatisch die Probleme der Welt im Allgemeinen und im Speziellen die der Bildung lösen könnte.

Doch Medien, gleich wie egalitär deren Inhalt und Zugang gestaltet ist, transportieren kein Wissen, sondern Informationen. Wissen wird durch Lernen des Menschen im Menschen erzeugt. Dieses Lernen erfolgt immer im Kontext des bereits bestehenden Wissens (vgl. Abschnitt 2.2.2 Kognitivismus) [62]. Die Fähigkeit, neues Wissen zu Generieren, steht demnach in direkter Abhängigkeit vom bisher Gelerntem und Erfahrenem. Ein bloßes Anbieten von Informationen und die Offenheit des Zugangs dazu können also Benachteiligungen aufgrund von Herkunft und damit einhergehender schlechter Vorbildung nicht ausgleichen. Der Zugang zu Information und Informationstechnologien ist zwar eine wesentliche Voraussetzung für die Möglichkeiten des Lernens, löst jedoch noch nicht die Fragen der gleichen Chancen auf gleiches Wissen.

6.1 Didaktik und Simulationsumgebungen

Die Einsatzfelder von Computerprogrammen sind mittlerweise fast grenzenlos. Dennoch basieren die aktuell gängigen Tools für computerunterstütztes Lernen zentral auf Annahmen des behavioristischen Lernens. Wie bei den ersten Lernmaschinen vor fast 150 Jahren wird das zu vermittelnde Wissen in kleinst mögliche Stücke zerteilt und den Lernenden präsentiert, die dann auf das richtige Antwortverhalten konditioniert werden. Diese „Drill & Practice-Systeme“ nutzen die Potentiale des Computereinsatzes nicht aus. Individuelles Lernen und Erfahren ist in diesen Systemen nur sehr begrenzt möglich. Baumgartner und Payr [4] sprechen auch von der „Software als autoritärer Lehrer“ [4].

Der Einsatz von Simulationen und die Verwendung von Simulationsumgebungen im didaktischen Kontext unterscheiden sich in vielen Punkten von herkömmlichen computerunterstützten Lernumgebungen. Den zentralen Unterschied stellt die zugrunde liegende Lerntheorie dar, auf die Designüberlegungen explizit oder implizit basieren. Simulationen sind ein sehr passendes Instrument für den Unterricht nach konstruktivistischen Grundsätzen und werden auch explizit als idealer Softwaretypus dafür bezeichnet [4]. Mit einer Situation operieren und komplexe Situationen bewältigen zu können wird als Wissen und Lernziel der

konstruktivistischen Lerntheorie bezeichnet [3]. Komplexe Systeme in einem Computermodell abzubilden und damit Simulationen durchzuführen entspricht demnach dem konstruktivistischen Lernziel.

Das Ziel des Lernens mit Simulationen ist „die Bewältigung komplexer Situationen auf dem Niveau von Gewandtheit oder Expertentum“ [4]. Die Konstruktion realer Systeme in Simulationsumgebungen und das Experimentieren mit den so erzeugten Mikrowelten ist eine optimale computerunterstützte Umsetzung erfahrungsbasierten Lernens (vgl. dazu auch den Lernzyklus von David Kolb im Abschnitt 2.2.6). Vor allem bei der Abbildung von Systemen, deren reale Beobachtung schwierig oder unmöglich ist, kann Simulation der einzige Ansatz sein, erfahrungsbasiertes Lernen überhaupt zu ermöglichen.

Kognitivistische Ansätze spielen im Zusammenhang mit der Verwendung von Simulationsumgebungen ebenfalls eine Rolle. Durch den Einsatz von Bildern und visuellen Interaktionen kann computerunterstütztes Lernen aufgrund der Dual Coding Theorie (vgl. Abschnitt 2.2.3) Vorteile gegenüber dem klassischen Lernen geltend machen. Im Besonderen für die Erstellung von Simulationen soll auf das Konzept der mentalen Modelle hingewiesen werden, da die Abbildung eines mentalen Modells in ein zu simulierendes Modell in einer Simulationsumgebung eine Adaption des mentalen Modells bewirken kann. Die Abstraktion eines realen Systems in eine Simulation schärft zudem den Blick auf die essentiellen Eigenschaften des Systems.

6.2 Der Einsatz von Simulationsumgebungen

Im Folgenden finden sich Hinweise für Lehrende, die in ihrem Unterricht Simulationsumgebungen verwenden wollen. Diese Punkte sind eine Zusammenfassung der Erkenntnisse, die dem Autor aus der Erstellung der vorliegenden Arbeit erwachsen sind.

1. Es ist (fast) nie zu früh für den Einsatz von Simulationsumgebungen

Nach den Stadien der kognitiven Entwicklung bei Piaget [1] entwickelt das Kind ab dem Alter von elf Jahren die Fähigkeit zu „abstrakten Schlussfolgerungen und hypothetischem Denken“. Ab diesem Zeitpunkt sind Kinder in der Lage, durch kontrolliertes Experimentieren Systeme auszukundschaften und Schlussfolgerungen daraus zu ziehen. Die AutorInnen von StarLogo (vgl. Abschnitt 5.1) haben ein spezielles Lehrbuch für den Einsatz an Schulen geschrieben [41], das es auch Lehrenden ohne Erfahrungen mit Simulationen und Programmierung ermöglicht, in diese Thematiken einzusteigen.

2. Simulationsumgebungen fördern das Verständnis komplexer Systeme

Mit Hilfe von Simulationsumgebungen ist es möglich, unterschiedlichste komplexe Systeme zu erforschen. Bei der Konstruktion von Modellen in Simulationsumgebungen ist eine detaillierte Auseinandersetzung mit dem realen System und eine Fokussierung auf dessen essentielle Eigenschaften erforderlich.

3. Erfahrung ohne Reflexion vermindert den Lernerfolg

Rieber [9] zeigt, dass es nicht reicht, Lernende nach einer Fragestellung mit Simulationen experimentieren zu lassen. Vor allem, wenn das Lösen der Simulationsaufgabe mit einem Spiel vergleichbar wird, schalten viele BenutzerInnen in einen sogenannten „twitch“-Modus. Dieser besagt, dass BenutzerInnen an der Optimierung der Beherrschung des User-Interfaces arbeiten und nicht mehr an der eigentlichen Fragestellung. SchülerInnen, die für das schnelle Lösen von Simulationsaufgaben Punkte bekommen haben, haben im darauffolgenden Test bedeutend schlechter abgeschnitten als jene, deren Simulationsexperimente ohne spielerische Anreize konstruiert waren (vgl. dazu auch den Unterschied zwischen „Game“ und „Play“ in Kapitel 2.5). Aus diesen Beobachtungen leiten sich zwei Erkenntnisse für Lehrende ab. Erstens, die richtige Information zum richtigen Zeitpunkt kann zentral für Reflexion und damit Erkenntnis sein. Zweitens, die Reflexion ist eine Voraussetzung für das Lernen und für das Generieren oder Adaptieren mentaler Modelle.

4. Die Rolle des Lehrenden als Coach

Eine zentrale Eigenschaft der konstruktivistischen Lerntheorie ist die Rolle der Lehrenden als Coach und nicht als autoritäre WissensvermittlerInnen (vgl. Übersicht auf Seite 14). Das bedeutet im Zusammenhang mit simulationsbasiertem Unterricht, dass die Lehrenden nicht mehr wissen müssen als die Lernenden. Es macht also nichts aus, wenn einzelne SchülerInnen oder Studierende in einem bestimmten Teilbereich über höheres Wissen verfügen als die Lehrenden. Die Aufgabe der Lehrenden ist es, den Lernenden zur Seite zu stehen, ihnen Anregungen zu geben und Reflexionsprozesse in Gang zu setzen.

5. Transaktionskosten niedrig halten

Der Einsatz von Software ist immer mit technischen Hürden verbunden. Zu den Transaktionskosten zählt dabei vor allem der zeitliche Aufwand, der betrieben werden muss, bevor die Software beherrschbar ist. Kapitel 5 zeigt eine Einführung in sechs Simulationsumgebungen. Tatsächlich gibt es unzählige verschiedene Tools, die für unterschiedlichen Einsatz unterschiedlich gut geeignet sind. Für den jeweiligen Einsatz sollten Tools gewählt werden, die für die Zielgruppe und die Vorhaben am besten geeignet und ohne großen Aufwand anwendbar sind. Vor allem im didaktischen Kontext ist ein unkomplizierter Einsatz von zentraler Bedeutung, denn nicht das technische Beherrschen einer Simulationsumgebung ist das Lernziel, sondern das Arbeiten und Erfahrung sammeln damit.

6. Nicht alles was gut ist muss teuer und aufwändig sein

Eine wesentliche Begrenzung der Möglichkeiten des computerunterstützten Lernens stellen die damit verbundenen Kosten dar. Erfahrene BenutzerInnen von Computerprogrammen sind hohe grafische Standards gewohnt. Für regelmäßige BenutzerInnen von Computerspielen liegt der Maßstab, an dem sie computerbasiertes Lernen messen, noch um einiges höher. Ein Vergleich: Die aktuell neueste Version IV des Computerspiels „Grand Theft Auto“ verursachte Entwicklungskosten von 100 Millionen US-Dollar und gilt als das bis dato teuerste Spiel aller Zeiten. Ein einziges Spiel verursacht also zum Teil Kosten im Ausmaß eines auf-

wändigen Hollywood-Filmes. Die Erzeugung von computerunterstützten Lernumgebungen kann da naturgemäß nicht mithalten. Simulationsumgebungen sind vielfach gratis und dennoch professionell. Gerade die Einfachheit mancher Tools ermöglicht einen klareren Blick auf das Modell und die Simulation, da aufwändig inszenierte Programmoberflächen das Wesentliche verdecken können, das Experimentieren, Erfahren und Lernen.

7. System Dynamics ist auch ohne fortgeschrittene Mathematik möglich

System Dynamics funktioniert mit nichtlinearen Differenzengleichungen und nähert Differentialgleichungen an. Dennoch ist es vor allem für EinsteigerInnen in der Modellkonstruktion und Simulation mit System Dynamics Simulationsumgebungen nicht notwendig, diese Gebiete der Mathematik zu beherrschen. Sämtliche in Kapitel 5 vorgestellten Simulationsumgebungen, die sich mit System Dynamics beschäftigen (vgl. die Übersicht ab Seite 84), können mit dem gängigen Schulwissen aus Mathematik bedient werden.

8. Simulationen sind skalierbar und beliebig erweiterbar

Ein wesentlicher Aspekt bei der Konstruktion von Simulationen ist es, mit einfachen Modellen zu beginnen. Einige Simulationsumgebungen ermöglichen zudem, dass diese einfachen Modelle innerhalb kurzer Zeit konstruiert werden können. Darauf aufbauend sind die Modelle aber beliebig erweiterbar. Mit den vier Elementen des System Dynamics sind Modelle konstruierbar, die ökonomische und ökologische Abläufe auf globaler Ebene darstellen (vgl. Abschnitt 3.4). Mit anfangs einfachem Verhalten von Agenten in agentenbasierten Simulationen sind komplexe Populationen simulierbar. Diese Möglichkeiten der modularen Erweiterbarkeit regen die Phantasie der BenutzerInnen an und laden zum Experimentieren ein.

9. Alles kann simuliert werden

Die Modellkonstruktion auf der Grundlage von realen Systemen und die darauffolgenden Simulationen sind nicht an naturwissenschaftliche Phänomene

gebunden. In jedem Fachgebiet können Aspekte des Lehrstoffs mit zumindest einem der drei Paradigmen, die in Kapitel 3 vorgestellt wurden (ereignisorientierte Simulation, System Dynamics, agentenbasierte Simulation) modelliert werden.

7 Literaturverzeichnis

[1] P. G. Zimbardo, R. J. Gerrig, Psychologie, 16. Auflage, Pearson Education Deutschland, München, 2004.

[2] E. Minass, Dimensionen des E-Learning, Neue Blickwinkel und Hintergründe für das Lernen mit dem Computer, SmartBooks Publishing AG, Kilchberg, 2002.

[3] J. Raithel et al., Einführung Pädagogik, 2. Auflage, VS Verlag, Wiesbaden, 2007.

[4] P. Baumgartner, S. Payr, Lernen mit Software, Österreichischer Studien-Verlag, Innsbruck, 1994.

[5] R. Kammerl, Computerunterstütztes Lernen – Eine Einführung, In: R. Kammerl, Computerunterstütztes Lernen, Oldenbourg Verlag, München, 2000.

[6] M. Pohl, Erläuterungen zu den Folien zur Vorlesung „Vernetztes Lernen“, Institut für Gestaltungs- und Wirkungsforschung, Technische Universität Wien, 2008.

[7] L. Roth (Hrsg.), Pädagogik, Handbuch für Studium und Praxis, Ehrenwirt Verlag, München, 1991.

[8] M. R. Quillian, Word concepts. A theory and simulation of some basic semantic capabilities, In: Behavioral Science, Nr 12, S. 410-430, 1967.

[9] L. P. Rieber, Multimedia Learning in Games, Simulations, and Microworlds, In: The Cambridge Handbook of Multimedia Learning, Cambridge University Press, Cambridge, 2005.

[10] G. Bodenann et al., Klassische Lerntheorien, Grundlagen und Anwendungen in Erziehung und Psychotherapie, Verlag Hans Huber, Bern, 2004.

[11] F. Collin, Konstruktivismus für Einsteiger, Wilhelm Fink GmbH & Co. Verlags-KG, Paderborn, 2008.

[12] H. v. Foerster, E. v. Glasersfeld, P. M. Hejl, S. J. Schmidt, P. Watzlawick, Einführung in den Konstruktivismus – Beiträge, 10. Auflage, Piper Verlag, München, 2008.

[13] D. E. Rumelhart, J. L. McClelland (Hrsg.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 2: Psychological and Biological Models. MIT Press, Cambridge, 1986.

[14] G. Siemens, Connectivism: A Learning Theory for the Digital Age, Quelle: http://www.elearnspace.org/Articles/connectivism.htm [19. 10. 2008].

[15] T. Bernhard, M. Kircher, E-Learning 2.0 im Einsatz, Verlag Werner Hülsbusch, Boizenburg, 2007.

[16] B. Kerr, Sunday, Which radical discontinuity?, Webblog Eintrag vom 11. Februar 2007, Quelle: http://billkerr2.blogspot.com/2007/02/which-radical-discontinuity.html [19. 10. 2008].

[17] H. M. Niegemann et al., Kompendium multimediales Lernen, Springer-Verlag, Berlin, 2008.

[18] F. Gertsch, Das Moodle 1.8 Praxisbuch, Addison-Wesley, München, 2007.

[19] Programm Moodle, Version 1.9.3, Quelle: http://moodle.org/ [19. 10. 2008].

[20] I. M. Breinbauer, Blended Learning, In: A. Dzierzbicka, A. Schirlbauer (Hrsg.), Pädagogisches Glossar der Gegenwart, Löcker Verlag, Wien, S. 39-49, 2006.

[21] J. Dron, Control and Constraint in E-Learning, Idea Group Publishing, Hershey, 2007.

[22] R. v. d. Kooij, Pädagogik und Spiel, In: L. Roth (Hrsg.), Pädagogik, Handbuch für Studium und Praxis, Ehrenwirt Verlag, München, S. 241-255, 1991.

[23] VDI-Gesellschaft Fördertechnik Materialfluss Logistik (Hrsg.), VDI-Richtlinie: VDI 3633, Simulation von Logistik-, Materialfluß- und Produktionssystemen – Begriffsdefinitionen, 1996.

[24] N. Gilbert, K. G. Troitzsch, Simulation for the Social Scientist, 2nd. Edition. Open University Press. 2005.

[25] H. Stachowiak, Allgemeine Modelltheorie, Springer-Verlag, Wien, 1973.

[26] N. Gilbert, Agent-Based Models, Series: Quantitative Applications in the Social Sciences, Sage Pubn Inc, Thousand Oaks, 2007.

[27] H. Bossel, Systeme Dynamik Simulation, Modellbildung, Analyse und Simulation komplexer Systeme, Books on Demand Gmbh, Norderstedt, 2004.

[28] B. Baumgartner, Petri-Netze: Grundlagen und Anwendungen, 2. Auflage, Spektrum Akademischer Verlag, Heidelberg, 1996.

[29] Vensim User's Guide Version 5, Stand: 4. Juli 2007: http://www.vensim.com/ffiles/VensimUsersGuide.zip [04.11.2008]

[30] P. Fleissner, Einführung in die Sozialkybernetik (Erster Teil der Unterlagen zum Seminar Mathematische Modellbildung und Simulation Sommersemester 2005, 2005.

[31] Leslie A. Martin, The First Step, In: J. W. Forrester, Road Maps: A Guide to Learning System Dynamics, Massachusetts Institute of Technology, 1997.

[32] J. W. Forrester, Industrial Dynamics, Pegasus Communications Waltham, 1961.

[33] J. W. Forrester, Urban Dynamics. Pegasus Communications Waltham, 1969.

[34] J. W. Forrester, World Dynamics. Pegasus Communications Waltham, 1973.

[35] D. H. Meadows, D. Meadows, E. Zahn, P. Millig, Limits to Growth, University Books, New York, 1972.

[36] M. Drmota, B. Gittenberger, G. Karigl, A. Panholzer, Mathematik für Informatiker, Heldermann Verlag, Lemgo, 2007.

[37] M. Resnick, Turtles, Termites, and Traffic Jams, 7th Edition, The MIT Press, Cambridge, 2001.

[38] S. Johnson, Emergence, The Penguin Press, London, 2001.

[39] D. Gordon, Ants at Work, How an Insect Society is Organiszed, W. W. Norton & Company Ldt, New York, 1999.

[40] J. H. Miller, S. E. Pager, Complex Adaptive System – An Introduction to Computational Models of Social Life, Princeton University Press, Princeton, 2007.

[41] V.S. Colella, E. Klopfer, M. Resnick, Adventures in Modeling, Exploring Complex, Dynamic Systems with StarLogo, Teachers College Press, New York, 2001.

[42] St. Kühl, Experiment, In: St. Kühl, P. Strodtholz, A. Taffertshofer (Hrsg.), Quantitative Methoden der Organisationsforschung, pp. 190-209, VS Verlag, Wiesbaden, 2005.

[43] L. Kirkup, Experimental Methods – An Introduction to the Analysis and Representation of Data, John Wiley & Sons, Brisbane, 1994.

[44] C. Dondi et al., Why choose a game for improving learning and teaching processes?, In: M. Pivec et al. (Hrsg), Guidelines for Game-Based Learning, Pabst Science Publishers, Lengerich, 2004.

[45] Programm „StarLogo“, Version 2.21, 2008; Quelle: http://education.mit.edu/starlogo/ [10. 10. 2008].

[46] Programm „MSW Logo“, Version 6.5b, 2002; Quelle: http://www.softronix.com/logo.html [10. 10. 2008].

[47] Programm „Repast“, Version 1.1.0, 2008, Quelle: http://repast.sourceforge.net/index.html [13. 10. 2008].

[48] Jung Bibliothek für soziale Netzwerkanalyse, Version 2.0 beta 1, 25. Juli 2008, Quelle: http://jung.sourceforge.net/ [13. 10. 2008].

[49] Programm: UCINET, Version 6.205, Quelle: http://www.analytictech.com/downloaduc6.htm [15. 10. 2008].

[50] Dokumentation der Software „Repast“, Stand 29. 11. 2007, Quelle: http://repast.sourceforge.net/docs/Getting%20Started.pdf [13. 10. 2008].

[51] M.J North, et. al., Visual Agent-based Model Development with Repast Simphony, In: Proceedings of the Agent 2007 Conference on Complex Interaction and Social Emergence, Argonne, USA, 2007.

[52] Programm „AnyLogic“, Version 6, Quelle: http://www.xjtek.com/anylogic/ [21. 10. 2008]

[53] AnyLogic 6, Agent Based Modeling Tutorial: http://www.xjtek.com/file/140 [05. 11. 2008]

[54] Programm „Stella“, Version 9.1, Quelle: http://www.iseesystems.com/ [21. 10. 2008].

[55] J. W. Forrester, System Dynamics and K-12 Teachers, In: J. W. Forrester, Road Maps: A Guide to Learning System Dynamics, Massachusetts Institute of Technology, 1997.

[56] Programm „Vensim PLE“, Version 8.8b, Quelle: http://www.vensim.com/ [21. 10. 2008].

[57] N. Repenning, Formulating Models of Simple Systems using Vensim PLE, In: J. W. Forrester, Road Maps: A Guide to Learning System Dynamics, Massachusetts Institute of Technology, 1998.

[58] Programm „ExtendSim“, Version 7.0.4, Quelle: http://www.extendsim.com/index.html [10. 11. 2008]

[59] C. Sackett, P. Diamond, K. Hansen, ExtendSim User Guide, Imagine That Inc, San Jose, 2007.

[60] L. J. Perelman, School's Out: Hyperlearning, the New Technology, and the End of Education, 1. Auflage, William Morrow & Co, New York, 1992.

[61] W. Sacher, Computer und die Krise des Lernens, Verlag Julius Klinkhardt, Bad Heilbrunn, 1989.

[62] G. Pollak, R. Kammerl, To know or not to know - Erziehungswissenschaftliche Bemerkungen zur Wissensgesellschaft, In: R. Kammerl, Computerunterstütztes Lernen, Oldenbourg Verlag, München, 2000.

8 Linksammlung von Materialen

8.1 Allgemein

Road Maps: A Guide to Learning System Dynamics, System Dynamics in Education Project von Jay W. Forrester,:
http://sysdyn.clexchange.org/road-maps/rm-toc.html [03. 11. 2008]

8.2 Getestete Simulationsumgebungen

StarLogo

StarLogo, Version 2.21;
http://education.mit.edu/starlogo/ [10. 10. 2008]

Einführung in StarLogo:
http://education.mit.edu/starlogo/tutorial/tutorial.html [11. 11. 2008]

Repast

Repast, Version 1.1.0:
http://repast.sourceforge.net/index.html [13. 10. 2008]

Dokumentation der Software Repast:
http://repast.sourceforge.net/docs/Getting%20Started.pdf [13. 10. 2008]

AnyLogic

AnyLogic, Version 6:
http://www.xjtek.com/anylogic/ [21. 10. 2008]

AnyLogic 6, System Dynamics Modeling Tutorial:
http://www.xjtek.com/file/107 [05. 11. 2008]

AnyLogic 6, Agent Based Modeling Tutorial:
http://www.xjtek.com/file/140 [05. 11. 2008]

Stella

Stella, Version 9.1:
http://www.iseesystems.com/ [21. 10. 2008].

Dokumentation der Software Stella:
http://www.iseesystems.com/community/downloads/tutorials/ModelBuilding.aspx
[21. 10. 2008].

Vensim

Vensim PLE, Version 8.8b:
http://www.vensim.com/ [21. 10. 2008]

Vensim User's Guide Version 5, Stand: 4. Juli 2007:
http://www.vensim.com/ffiles/VensimUsersGuide.zip [04.11.2008]

Einführung in Vensim als Erweiterung zu den Road Maps von Forrester:
http://sysdyn.clexchange.org/sdep/Roadmaps/RM2/D-4697-2.pdf [05. 11. 2008]

ExtendSim

ExtendSim, Version 7.0.4, Quelle:
http://www.extendsim.com/index.html [10. 11. 2008]

Benutzerhandbuch für ExtendSim:
http://www.extendsim.com/support_manuals.html [11. 11. 2008]

9 Index

L

M

N

P

Q

R

S

T

U

V

W

Z