Das Referenzmodell PECS

Agentenbasierte Modellierung menschlichen Handelns, Entscheidens und Verhaltens

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)

Fakultät für Mathematik und Informatik
Universität Passau

Dipl.-Inf. Christoph Urban

April 2004

INHALT

1 EINLEITUNG UND ÜBERBLICK

Der Begriff *Agent* hat sich in den letzten Jahren zu einem der zentralen Begriffe in der Verteilten Künstlichen Intelligenz, sowie in angrenzenden Gebieten entwickelt. Sprachlich stammt der Begriff *Agent* vom lateinischen Wort *agens* (der / die / das Handelnde) ab. Eine allgemein akzeptierte, theoretisch oder praktisch fundierte Agentendefinition existiert bislang nicht. In der Regel werden Agenten jedoch die folgenden drei Grundfunktionen zugeordnet (Müller, 1996):

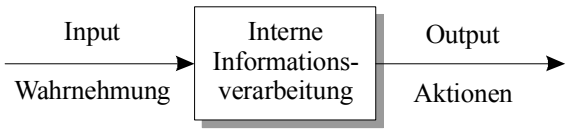

Abbildung 1.1: Minimalistisches Agentenmodell

- Wahrnehmung der Umwelt

  Agenten sind sehr häufig in komplexe, dynamische Umgebungen eingebettet, die Randbedingungen für ihr Verhalten vorgeben. Um adäquat handeln zu können, verfügen Agenten über die Möglichkeit, ihre Umwelt wahrzunehmen. Sie nehmen Sinneseindrücke und Reize aus ihrer Umgebung auf und führen diese einer internen Weiterverarbeitung zu.

- Interne Verarbeitung der wahrgenommenen Informationen

  Aufgrund der wahrgenommenen Umweltinformationen legen Agenten intern ihr Verhalten fest, wobei mehr oder weniger komplexe Steuerungsmechanismen zum Einsatz kommen. Bei einfach strukturierten Agenten erfolgt die Verhaltenssteuerung auf der Grundlage vorgegebener Regeln, die bestimmen, in welcher Weise sich ein Agent in einer gegebenen Situation zu verhalten hat. Komplexere Agenten können zudem auf kognitive Fähigkeiten zurückgreifen, die es ihnen beispielsweise erlauben, Entscheidungen zu treffen, Handlungspläne zu entwerfen oder Probleme zu lösen.

- Ausführung von Aktionen

  Agenten können im allgemeinen ihre Umwelt nicht nur wahrnehmen, sondern sie verfügen auch über Möglichkeiten, modifizierend auf sie einzuwirken.

Dies geschieht mit Hilfe von extern wirksamen Aktionen, die als Ergebnis interner Verhaltenssteuerungsmechanismen ausgelöst werden.

Zusätzlich werden Agenten oftmals mit Attributen versehen, um deren besondere Charakteristika in den jeweiligen Anwendungsfällen herauszustellen. Einige häufig vorkommende Attribute sind in der folgenden Liste aufgeführt (Wooldridge & Jennings, 1995):

- Autonomie

  Agenten handeln ohne direkten Eingriff von außen und kontrollieren selbständig ihre Aktionen, sowie ihren internen Zustand.

- Soziale Fähigkeiten

  Agenten sind oftmals Bestandteil eines größeren Agentensystems. In diesen Fällen ist es erforderlich, daß die einzelnen Agenten soziale Fähigkeiten aufweisen. Dazu zählen beispielsweise Kommunikationsfähigkeiten (die Fähigkeit, Informationen mit anderen Agenten auszutauschen), sowie Kooperationsfähigkeiten (die Fähigkeit, Ziele zu verfolgen oder Präferenzen zu berücksichtigen, die für die Erreichung eines übergeordneten Gesamtziels nützlich sind). Im Rahmen der Kommunikation wird erwartet, daß Agenten aufrichtig sind, d. h. daß sie nicht wissentlich falsche Informationen verbreiten. Eine Voraussetzung für die Kooperation zwischen Agenten besteht in deren Wohlwollen, d. h. in deren Bereitschaft, die übertragenen Aufgaben zu erfüllen.

- Reaktivität

  Agenten nehmen ihre Umwelt wahr und reagieren, sobald Veränderungen in der Umwelt auftreten, die für die Agenten von Bedeutung sind.

- Proaktivität

  Das Verhaltenspotential von Agenten ist im allgemeinen nicht auf Reaktionen beschränkt. Agenten können auch ohne externen Stimulus die Initiative ergreifen und zielorientiert handeln.

- Mobilität

  Mobilität meint die Fähigkeit eines Agenten, den physischen Aufenthaltsort in der Umwelt zu verändern.

- Rationalität

  Der Begriff *Rationalität* wird in der Literatur mit sehr unterschiedlichen Bedeutungen belegt. Wooldridge und Jennings (1995) bezeichnen mit Rationalität die Annahme, daß Agenten in einer derartigen Weise handeln, daß ihre Ziele nach Möglichkeit erreicht werden können.

- Intelligentes Verhalten

  Ebenso wie der Rationalitätsbegriff wird auch der Intelligenzbegriff in der Literatur nicht einheitlich behandelt (Sundermeyer, 1993). Agenten werden

oftmals als *intelligent* bezeichnet, sofern sie eine gewisse Lernfähigkeit oder besondere Anpassungsfähigkeiten an veränderliche Umweltbedingungen aufweisen.

Moffat und Frijda (1995) fügen dieser Liste weitere Attribute hinzu, die besondere Bedeutung erlangen, wenn die Umwelt der Agenten sehr dynamisch ist:

- Flexibilität

  Agenten sollen nicht starr auf ihren Verhaltensweisen beruhen, sondern in flexibler Weise auch auf unerwartete Ereignisse in der Umwelt reagieren können.

- Planung

  Agenten sollen in der Lage sein, ihre zukünftigen Handlungen bei Bedarf zu planen. Dies setzt voraus, daß Agenten die Konsequenzen ihrer Aktionen abschätzen können. Der Vorteil dieser Fähigkeit besteht darin, daß sich Agenten auf bevorstehende Schwierigkeiten vorbereiten bzw. diese sogar vermeiden können, bevor sie tatsächlich eintreten.

- Selektivität

  Agenten müssen aus der Fülle an Informationen, die von ihrer Umwelt geliefert werden, diejenigen herausfiltern, die für ihre aktuelle Situation von besonderer Bedeutung sind. Die Aufmerksamkeit von Agenten muß also auf wichtige Informationen gerichtet werden.

- Robustheit

  Agenten sollen robust gegenüber Fehlern sein und ihre Arbeit auch in kritischen Situationen fortsetzen können.

1.1 Agenten und ihre Anwendungsfelder

Ausgehend von einem kurzen Streifzug durch ihre historische Entwicklung soll im folgenden Abschnitt ein Überblick über die derzeit wichtigsten Anwendungsgebiete der Agententechnologie gegeben werden. Dabei werden die grundlegenden Bereiche *Agenten in technischen Anwendungen*, *Agenten in empirischen Wissenschaften* und *Agenten im Kontext der Mensch-Maschine-Interaktion* voneinander unterschieden. Dies erscheint sinnvoll, da sich die Anforderungen, die in diesen beiden Anwendungsbereichen an Agenten gestellt werden, sehr unterschiedlich darstellen.

Die Künstliche Intelligenz (KI) versteht sich nach Görz (1995) als eine wissenschaftliche Disziplin, in der das Ziel verfolgt wird, informationsverarbeitende, technische Systeme verfügbar zu machen, die nach den Grundprinzipien menschlicher Wahrnehmungs- und Verstandesleistungen gestaltet sind. Einen Kernbereich, mit dem sich die KI beschäftigt, stellt die Modellierung höherer Intelligenz-

funktionen dar. Hier gilt es vor allem zu erforschen, in welcher Weise Wissen in künstlichen Systemen repräsentiert und verarbeitet werden kann. Zu speziellen Forschungsthemen zählen dabei etwa Wissensrepräsentation, maschinelles Lernen, Planen und Problemlösen, Bildverarbeitung oder auch künstliche neuronale Netze (Freksa, 1996).

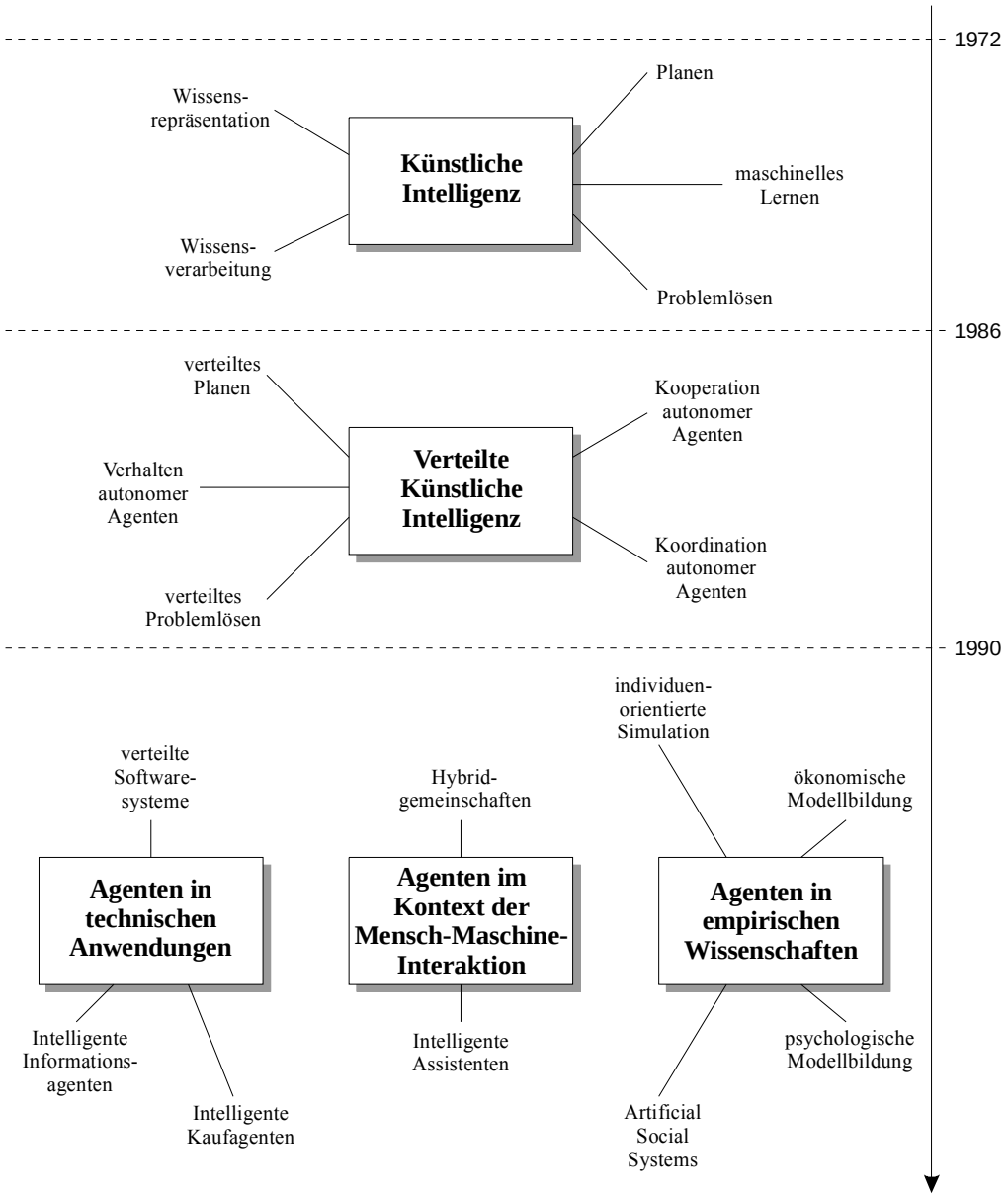

Abbildung 1.2: Agenten und ihre Anwendungsfelder

Als zentrales Paradigma der symbolischen KI wurde Anfang der siebziger Jahre von Newell und Simon (1972) der Begriff des *general intelligent agent* geprägt. Ein derartiger intelligenter Agent verfügt über Sensoren, mit deren Hilfe In-

formationen aus der Umgebung aufgenommen werden können, über Aktuatoren, mit denen der Agent seine Umwelt modifizieren kann, und über Wissen. Das Wissen setzt sich aus einer Art Gedächtnis zusammen und der spezifischen Fähigkeit, beim Handeln in der Umwelt auf die gespeicherten Informationen zuzugreifen. Auf diese Weise erhält ein Agent die Möglichkeit des Probehandelns, d. h. durch Modifikation von internen Repräsentationen können Konsequenzen von Handlungen abgeschätzt werden, bevor diese irreversibel in der Umwelt ausgeführt werden. Auf der Grundlage von *single-agent*-Architekturen wurden im Laufe der Zeit immer größere und komplexere wissensverarbeitende Systeme realisiert und dabei auch Nachteile und Grenzen dieses Ansatzes aufgezeigt (Adler, Durfee, Huhns, Punch & Simoudis, 1992). Diese Erfahrungen beflügelten in der Folgezeit die Konzipierung einer *Verteilten Künstlichen Intelligenz (VKI)*.

Mitte der achtziger Jahre brachte Minsky (1986) eine neue Perspektive in die KI-Forschung ein. Minsky schlug anstelle zentraler Ansätze einen dezentralen Ansatz vor, bei dem intelligentes Verhalten durch die Kooperation einer Vielzahl einfach strukturierter, unabhängiger Einheiten (autonome Agenten) entsteht. Keiner dieser Agenten benötigt mehr ein vollständiges Wissen über den gesamten Problemlöseprozeß, sondern nur noch eine überschaubare Menge an spezifischem Wissen. Lösungen für gegebene Probleme werden durch Kooperation mehrerer Agenten erarbeitet, wobei jeder einzelne Agent seine spezifischen Fähigkeiten in den Problemlöseprozeß einbringt.

Die VKI wirft eine Reihe neuer Fragestellungen auf, die über die Probleme der klassischen KI hinausgehen. Im Vergleich zur klassischen KI tritt in der Verteilten KI der Aspekt der Verhaltenssteuerung autonomer Systeme gegenüber dem Aspekt der Wissensmodellierung wesentlich in den Vordergrund. Es geht hier bevorzugt um die Frage, wie das Verhalten autonomer Systeme gesteuert werden kann. Eine Vielzahl von Ansätzen orientiert sich dabei an Mechanismen, die in natürlichen Systemen, wie beispielsweise bei Tieren oder beim Menschen, vorzufinden sind. Dem Problem der Verhaltenssteuerung treten weitere Schwierigkeiten hinzu, die sich aus der Notwendigkeit der Koordination der Teilsysteme ergeben. Die Aktivitäten der Agenten müssen in vielen Fällen in koordinierter Weise ausgeführt werden, damit die Globalziele der Agentengesellschaft erreicht werden können. Auch hier liefern die reale Welt und insbesondere menschliche Sozialsysteme Grundprinzipien, nach denen künstliche Systeme gestaltet werden können.

Seit Beginn der 90er Jahre nehmen die im Rahmen der VKI-Forschung entwickelten Konzepte zunehmend Einfluß auf andere Forschungs- und Anwendungsdomänen (Brenner, Zarnekow & Wittig, 1998). Die Attraktivität des Agentenansatzes liegt dabei vor allem in der natürlichen Sichtweise auf große, komplexe Systeme (Huhns, 1994), wie sie in vielen Forschungs- und Anwendungsgebieten vorzufinden sind. Der Agentenbegriff beschränkt sich also nicht mehr nur auf intelligente, problemlösende Einheiten. Vielmehr dehnen sich die Einsatzmöglichkeiten der Agententechnologie immer weiter aus, so daß sich in diesem Zusammenhang ein neues, interdisziplinär angelegtes Forschungsfeld entwickelt. Im folgenden soll zwischen drei übergeordneten Bereichen unterschieden werden, in

denen agentenorientierte Ansätze eine wichtige Rolle spielen: *Agenten in technischen Anwendungen*, *Agenten in empirischen Wissenschaften* und *Agenten im Kontext der Mensch-Maschine-Interaktion*.

1.1.1 Agenten in technischen Anwendungen

Technische Disziplinen verfolgen das Ziel, neue, technische Systeme zu entwickeln und zu realisieren. Im Vordergrund stehen dabei die Funktionalität und Leistungsfähigkeit der Systeme, sowie Kostenaspekte. Auch Software- oder Hardwaresysteme, wie z. B. autonome Roboter, sind in dieser Hinsicht als technische Systeme einzustufen.

Agenten sind in technischen Anwendungen immer dann von Bedeutung, wenn bestimmte Aufgaben in autonomer Weise zu erledigen sind. Maes (1995) spricht in diesem Zusammenhang von Agenten als artifizielle Systeme, die in komplexen, dynamischen Umgebungen existieren, ihre Umgebungen mit Hilfe von Sensoren wahrnehmen, autonom in diesen Umgebungen handeln und dabei Ziele verfolgen, für deren Erreichung sie entwickelt worden sind. Im Bereich der Softwaretechnologie werden Agenten häufig als fundamentale architektonische Grundbausteine komplexer (verteilter) Softwaresysteme eingesetzt. Im Hardwarekontext dienen Agenten oftmals als abstraktes Konzept zur Steuerung autonomer Roboter.

Ein ideales Einsatzfeld für die Agententechnologie, das in den letzten Jahren einer rasanten Entwicklung unterliegt, bietet das Internet. Als komplexes, verteiltes System weist das Internet charakteristische Eigenschaften auf, die den Einsatz autonomer Agenten sinnvoll erscheinen lassen. So werden Softwareagenten beispielsweise als intelligente Informationssammler (Klusch, 1998) genutzt, um Anwender von zeitintensiven Suchvorgängen im Internet zu befreien, oder als Vertreter realer Käufer und Verkäufer in Electronic Commerce-Anwendungen (Brenner, Zarnekow & Wittig, 1998).

Die Anforderungen an die Agenten, die sich in technischen Anwendungen in bezug auf deren Aufbau, Eigenschaften und Funktionalität ergeben, resultieren ausschließlich aus der geforderten Funktionalität und Leistungsfähigkeit des Gesamtsystems, in das die Agenten eingebettet sind. Die Agenten sind damit derartig zu konstruieren und zu realisieren, daß die geforderten Leistungsmerkmale so gut wie möglich erreicht werden und möglichst einfache und klar strukturierte Systeme entstehen. Um diese Anforderungen zu erreichen, sind der Kreativität des Entwicklers keine Grenzen gesetzt.

1.1.2 Agenten in empirischen Wissenschaften

Empirische Wissenschaften beschäftigen sich mit der Untersuchung realer Systeme. Sie greifen abgrenzbare Ausschnitte aus der realen Welt heraus und versu-

chen, über diese Realitätsausschnitte neue Erkenntnisse zu gewinnen. Sehr häufig werden in diesem Zusammenhang Methoden der Modellbildung und Simulation eingesetzt.

Nach einer klassischen analytischen Vorgehensweise werden zunächst einmal im Rahmen einer Systemanalyse die relevanten Strukturen, Eigenschaften und Daten des realen Systems festgestellt. Die gesammelten Informationen werden in einem nächsten Schritt dazu verwendet, ein Modell des Gegenstandsbereiches aufzubauen, das im Rahmen von Modellexperimenten einen vertieften Erkenntnisgewinn ermöglichen soll (Schmidt, 2000).

Besonders in den Sozialwissenschaften gestaltet sich die Systemanalyse oft sehr schwierig und auch aufwendig. Aus diesem Grund hat sich in den letzten Jahren neben der analytischen Vorgehensweise auch eine synthetische Vorgehensweise etabliert (Epstein & Axtell, 1996). Dabei tritt der Arbeitsschritt der Systemanalyse in den Hintergrund und es werden basierend auf Hypothesen über die reale Welt in experimenteller Weise Modelle künstlicher Sozialsysteme erstellt.

Während bei einer analytischen Vorgehensweise die Erstellung valider und damit möglichst realitätsnaher Modelle im Vordergrund steht, besteht bei einer synthetischen Vorgehensweise eher Interesse an der Untersuchung der Entstehung emergenten Verhaltens. Von *emergentem Verhalten* kann man sprechen, wenn auf einer höheren Abstraktionsebene eines Systems neue, in keinem Subsystem vordefinierte Eigenschaften entstehen, die sich aus dem autonomen Verhalten der Subsysteme auf einer niedrigeren Abstraktionsebene, sowie aus deren Interaktion ergeben (Schmidt, 2000a; Strube 1996).

Agenten spielen auch im Bereich der Modellbildung in empirischen Wissenschaften eine große Rolle. Die zu untersuchenden Systeme oder Teile davon werden oftmals auf der Grundlage von Agenten modelliert. Insbesondere werden Agenten als Modellierungsparadigma dann genutzt, wenn Lebewesen und speziell auch Menschen modelliert werden sollen. Agenten dienen in dieser Hinsicht als virtuelle Repräsentanten realer Entitäten. Als eines der wohl prominentesten Beispiele, in denen Agenten zum Aufbau künstlicher Sozialsysteme genutzt werden und individuelles menschliches Verhalten nachbilden, gelten die Sugarscape-Experimente von Epstein und Axtell (1996). Weitere Anwendungen der agentenbasierten Modellbildung finden sich beispielsweise auch in der Psychologie, der Ökonomie oder der Ökologie (s. Kapitel 2.1).

Im Gegensatz zum Einsatz der Agententechnologie in technischen Anwendungsfeldern können der Aufbau, die Eigenschaften und das Verhalten von Agenten nicht frei gewählt werden, sofern diese im Rahmen der Modellbildung eingesetzt werden. Vielmehr besteht hier die Forderung nach Struktur- und Verhaltensähnlichkeit, d. h. die Agenten müssen so konstruiert werden, daß sie im Hinblick auf ihren Aufbau und ihr Verhalten den realen Vorbildern möglichst ähnlich sind. Wird ein Agent also beispielsweise eingesetzt, um einen Menschen im Modell nachzubilden, muß der Agent über alle Eigenschaften und Verhaltensweisen des realen Menschen verfügen, die in der zu untersuchenden Situation als wesentlich erscheinen.

1.1.3 Agenten im Kontext der Mensch-Maschine-Interaktion

Die beiden vorangegangenen Abschnitte zeigen deutlich auf, daß agentenorientierte Ansätze gewinnbringend in unterschiedlichen Anwendungsgebieten eingesetzt werden können. Es stellt sich jedoch ebenso heraus, daß sich die Anforderungen, die im jeweiligen Kontext an Agenten gestellt werden, grundlegend voneinander unterscheiden.

Es gibt nun aber auch Bereiche, in denen sowohl technische Aspekte, als auch Modellierungsaspekte gleichermaßen eine Rolle spielen. Dies ist beispielsweise in neuen Ansätzen zur Mensch-Maschine-Interaktion der Fall, in denen Anwendern der Umgang mit Softwaresystemen erleichtert werden soll. In sogenannten Hybridgemeinschaften (Rammert, 1998) sollen den menschlichen Anwendern künstliche Agenten zur Seite gestellt werden, die die Anwender bei der Durchführung ihrer Arbeiten am Rechner unterstützen. Ein konkretes Beispiel wäre hier ein Tutorsystem, in dem ein künstlicher Agent die Rolle eines menschlichen Tutors übernimmt und einen Schüler bei der Erarbeitung seines Lernstoffes unterstützt (Harrer, 1999). Ein derartiger Agent sollte technischen Anforderungen genügen, z. B. sollte er in der Lage sein, die Antworten, die der Schüler dem Tutorsystem gibt, zu interpretieren, damit er adäquat auf die jeweilige Lernsituation reagieren kann. Zudem sollte der Agent aber auch gewisse Verhaltensregeln beachten, um einen glaubwürdigen Umgang mit dem menschlichen Interaktionspartner zu ermöglichen, oder auch ein ansprechendes Erscheinungsbild haben. In dieser Hinsicht können die Eigenschaften eines derartigen Agenten an die Eigenschaften realer Menschen angelehnt werden, wodurch bei dieser Anwendung auch der Modellierungsaspekt zum tragen kommt.

Es zeigt sich also, daß für Agenten in technischen Anwendungen und im Bereich der Modellbildung zwar unterschiedliche Anforderungen existieren, aber durchaus auch Synergien zwischen diesen beiden Bereichen existieren. Diese Einsicht untermauert den interdisziplinären Charakter vieler Forschungsthemen im Bereich der Agententechnologie.

1.2 Motivation und Ziele dieser Arbeit

Dieser Abschnitt soll in kurzer und knapper Form einen Überblick über die Motivation und die Ziele dieser Arbeit geben. Auf eine detaillierte Diskussion der auftretenden Begriffe wird an dieser Stelle verzichtet. Genauere Ausführungen dazu finden sich in den nachfolgenden Kapiteln.

Diese Arbeit beschäftigt sich mit dem Einsatz der Agententechnologie im Rahmen der Modellbildung. Agenten sollen also, wie in Kapitel 1.1.2 beschrieben wurde, eingesetzt werden, um Entitäten der realen Welt und insbesondere Menschen in Simulationsmodellen abzubilden.

In einer Vielzahl von Fallstudien und Projekten wurde in den letzten Jahren gezeigt, daß sich agentenbasierte Methoden sehr gut eignen, um Simulationsmodelle aufzubauen, in denen der Faktor Mensch eine entscheidende Rolle spielt (s. Kapitel 2.1). Charakteristisch für derartige Modelle ist, daß menschliches Verhalten in der Regel auf rein kognitive Fähigkeiten reduziert wird, d. h. der Mensch wird im wesentlichen als rationaler Entscheider verstanden. Darüber hinaus gehen viele Modellierungsansätze davon aus, daß Interaktionen zwischen Akteuren als strategische Interdependenzen im Sinne der Spieltheorie modelliert werden können.

Diese „klassischen" Ansätze geraten zunehmend in Kritik, da sich die Sichtweise des Menschen als rein rationaler Entscheider, der perfekt informiert ist, über uneingeschränkte Rechenkapazität verfügt und stur eine exogen vorgegebene und unveränderliche Nutzenfunktion zu maximieren versucht (Epstein & Axtell, 1996), als zu restriktiv erweist. Parallel dazu gewinnen in der Psychologie komplexere Handlungstheorien, wie sie beispielsweise von Dörner (1999) im Rahmen seiner Ψ–Theorie vorgestellt werden, an Bedeutung. Derartige Theorien berücksichtigen in bezug auf die menschliche Handlungsregulation nicht nur kognitive Fähigkeiten, sondern beziehen auch physische oder emotionale Befindlichkeiten, sowie Wechselwirkungen mit dem sozialen Umfeld ein.

Mit dieser komplexeren Betrachtungsweise des menschlichen Verhaltens steigen auch die Anforderungen, die an die Entwurfsmethodik agentenbasierter Modelle gestellt werden. Es entsteht ein Bedarf an Agenten, die eine Modellierung ausgeprägter interner Zustände, sowie von Wechselwirkungen psychischer und physischer Prozesse ermöglichen. Allerdings sind derzeit noch keine Modellierungskonzepte verfügbar, die in geeigneter Weise Hilfestellungen für den Entwurf derartiger agentenbasierter Modelle bieten.

Das Ziel dieser Arbeit besteht also darin, den Entwurfsprozeß agentenbasierter Modelle, in denen

- individuelles menschliches Handeln, Entscheiden und Verhalten,
- die Interaktion von Individuen und
- die Interaktion von Individuen mit ihrer Umwelt

von ausschlaggebender Bedeutung sind, auf konzeptioneller Ebene zu unterstützen. Zu diesem Zweck wird im Rahmen dieser Arbeit das Referenzmodell *PECS* (Physis, Emotion, Cognition, Social Characteristics) vorgestellt, das Konzepte für den Aufbau von Agenten, die Interaktion zwischen Agenten und die Modellierung der Umwelt der Agenten anbietet. Im Hinblick auf den Entwurf der Agenten geht *PECS* von der Annahme aus, daß eine Agentenarchitektur eine integrative Beschreibung physischer, emotionaler, kognitiver und sozialer Einflüsse ermöglichen muß, um eine solide Plattform für die Modellierung menschlichen Verhaltens bereitzustellen. Darüber hinaus ist *PECS* domänenunabhängig angelegt, d. h. es wird ein allgemeines, methodologisch begründetes Konstruktionsschema (Klinger, 1999) vorgeschlagen, das in verschiedenen Anwendungsgebieten eingesetzt werden kann und dazu domänenspezifisch mit Inhalt gefüllt werden

muß. Der praktische Einsatz von *PECS* und der Gesichtspunkt der Domänenunabhängigkeit werden anhand von vier Fallstudien aus den Bereichen Soziologie, Ökonomie, Sozialpsychologie und Psychologie demonstriert.

1.3 Inhalt und Aufbau dieser Arbeit

Den Schwerpunkt dieser Arbeit bildet die Beschreibung des Referenzmodells *PECS*, das ein domänenunabhängiges Konstruktionsschema für agentenbasierte Simulationsmodelle vorgibt, in denen menschliches Entscheiden, Handeln und Verhalten von ausschlaggebender Bedeutung sind. In Kapitel 4 werden im Sinne einer Anforderungsanalyse zunächst wesentliche Anforderungen zusammengetragen, die den Entwurf eines derartigen Referenzmodells beeinflussen. Das Kapitel 5 geht auf den systemtheoretischen Ansatz sowie den komponentenorientierten und hierarchischen Modellaufbau ein. Diese beiden Konzepte bilden den Ausgangspunkt für die Beschreibung des Referenzmodells PECS. Der Aufbau und die Eigenschaften des Referenzmodells werden in Kapitel 6 dargestellt. Kapitel 7 gibt einen Überblick über das Zusammenspiel der *PECS*-Komponenten bei der Abbildung ausgewählter menschlicher Verhaltenssteuerungsmechanismen.

Wie bereits in den einführenden Abschnitten angedeutet wurde, ist das Gebiet der agentenbasierten Modellbildung durch ein hohes Maß an Interdisziplinarität geprägt. Aus dem Ziel, ein Referenzmodell zu schaffen, das die Modellierung menschlichen Verhaltens ermöglicht, resultiert ganz automatisch die Notwendigkeit, sich mit einer Reihe von Fachgebieten auseinanderzusetzen, die über die Grenzen der Informatik hinausgehen. Aus diesem Grund beschäftigt sich das Kapitel 2 mit dem Stand der Forschung, und zwar nicht nur in bezug auf das Gebiet der Verteilten Künstlichen Intelligenz, sondern auch in bezug auf relevante Ansätze im Bereich der Sozialwissenschaften, wie beispielsweise der Soziologie oder der Psychologie.

Das Kapitel 3 befaßt sich mit Referenzmodellen in Modellbildung und Simulation sowie mit softwaretechnischen Architekturmustern. Ein besonderes Augenmerk liegt in diesem Abschnitt auf der Definition des Begriffes *Referenzmodell*, dem Einsatz von Referenzmodellen im Rahmen der Modellbildung, sowie auf Beschreibungsmöglichkeiten für Referenzmodelle.

Im Anschluß an die Beschreibung des Referenzmodells werden vier Fallstudien diskutiert, die den Einsatz des Referenzmodells in verschiedenen Anwendungsgebieten zeigen und damit den Anspruch der Domänenunabhängigkeit untermauern.

Die erste Fallstudie beschäftigt sich mit einer Themenstellung aus der Soziologie. Sie behandelt die Evolution von Solidarnetzwerken in einer Gesellschaft egoistischer, rationaler Agenten.

Die zweite Fallstudie entstammt der Ökonomie und befaßt sich mit der Modellierung strategischer Interaktionen zwischen Akteuren bei der Preisfestlegung auf oligopolistischen Märkten.

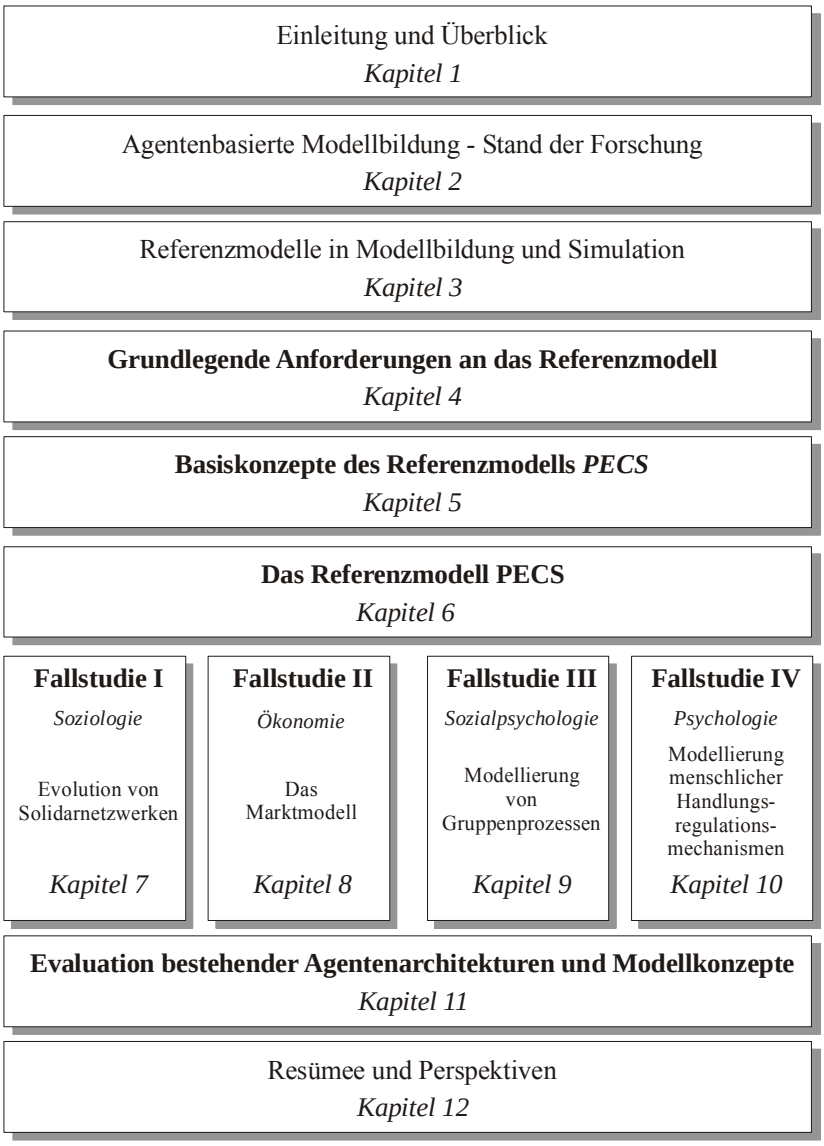

Abbildung 1.3 Inhalt und Aufbau dieser Arbeit

Ein agentenbasierter Ansatz für die Modellierung elementarer, sozialpsychologischer Prozesse wird im Rahmen der dritten Fallstudie beschrieben. Am Beispiel einer Lerngruppe wird ein Modellierungsansatz für die Prozesse Gruppenformation, Gruppenarbeit und Gruppendisintegration untersucht.

Im Rahmen der vierten Fallstudie wird das Modell *Adam* vorgestellt. Im Vordergrund steht hier die Modellierung grundlegender Mechanismen der menschli-

chen Handlungsregulation unter Einbeziehung physischer, emotionaler und kognitiver Prozesse.

Im Kapitel 11 wird ein Vergleich des im Rahmen dieser Arbeit entwickelten Referenzmodells *PECS* mit existierenden Arbeiten, die vorwiegend dem Grenzgebiet zwischen Künstlicher Intelligenz, Kognitionspsychologie, Emotionspsychologie und Sozialwissenschaften entstammen, angestellt.

Den Abschluß bildet schließlich eine kurze Zusammenfassung der wichtigsten Ergebnisse dieser Arbeit, sowie ein Ausblick auf weiterführende Forschungsmöglichkeiten.

2 AGENTENBASIERTE MODELLBILDUNG - STAND DER FORSCHUNG

Aus dem Ziel, ein Referenzmodell zu entwickeln, das als domänenunabhängiges Strukturierungskonzept für agentenbasierte Simulationsmodelle dient, in denen menschliches Handeln, Entscheiden und Verhalten untersucht werden sollen, resultiert in unmittelbarer Weise die Notwendigkeit der Auseinandersetzung mit Forschungsansätzen aus unterschiedlichen Fachgebieten. Die folgenden Abschnitte sollen deshalb einen Überblick über den gegenwärtigen Stand der Forschung in denjenigen Arbeitsgebieten geben, die für diese Arbeit von besonderer Bedeutung sind.

2.1 Anwendungsbeispiele für die agentenbasierte Modellbildung

Agentenbasierte Systeme, die Auswirkungen auf den Bereich der Modellbildung und Simulation haben, finden sich seit Mitte der achtziger Jahre. Im folgenden Abschnitt werden die wichtigsten Anwendungen der agentenbasierten Modellbildung, die in den letzten Jahren bekannt geworden sind, sowie aktuelle Forschungsschwerpunkte im Überblick dargestellt.

2.1.1 Pengi

Eine der ersten und zugleich einfachsten Anwendungen, die in diesem Zusammenhang von Bedeutung sind, ist das System *Pengi* (Agre & Chapman, 1987). Hauptdarsteller in dieser Simulation ist ein Pinguin, der in einer zweidimensionalen Welt Diamanten einsammeln soll und dabei durch Eiswürfel und Bienen behindert wird. Die Verhaltenssteuerung des Pinguin ist geprägt durch das Streben nach Zufriedenheit, die sich einstellt, wenn ein bestimmter Zielzustand erreicht wurde. Zudem versucht der Pinguin seine Bewegungsfreiheit, sowie die Möglichkeit des freien Agierens aufrecht zu erhalten und mögliche Konflikte zu vermei-

den. *Pengi* basiert auf einem rein reaktiven Ansatz, der zu diesem Zeitpunkt als Alternative zu klassischen Planungsparadigmen der KI gedacht war.

2.1.2 MANTA – Modelling an Anthill Activity

Eine schon wesentlich komplexere Anwendung der agentenbasierten Modellbildung stellt das Projekt *MANTA – Modelling an Anthill Activity* (Drogoul & Ferber, 1994; Drogoul, Corbara & Lalande, 1995) dar. Das Ziel besteht hier in der Modellierung der Soziogenese eines künstlichen Ameisenstaates, d. h. der Entwicklung einer Ameisenkolonie mit ihrer zugehörigen Sozialstruktur, beginnend nur mit einer Königin.

Die Ameisen werden mit Hilfe reaktiver Agenten abgebildet. Die Agenten verfügen über die Fähigkeit, sich in ihrer Umwelt, dem simulierten Ameisennest, frei zu bewegen und müssen bestimmte Aufgaben erfüllen, um die Ameisenkolonie am Leben zu erhalten. Den Agenten stehen verschiedene, vordefinierte Aktionen zur Auswahl, die sowohl einzeln als auch im Verbund in Form von Aktionsfolgen ausgeführt werden können. Aufgrund des reaktiven Ansatzes ist die Ausführung von Aktionen immer an das Auftreten eines internen oder externen Stimulus geknüpft. Zwischen den einzelnen Agenten ist keine verbale Kommunikation möglich. Eine Interaktion erfolgt ausschließlich über absichtlich oder auch unabsichtlich in der Umwelt platzierte Stimuli.

Je öfter ein Agent eine bestimmte Aktion bzw. Aktionsfolge ausführt, um so höher wird deren dynamische Gewichtung. Dies hat zur Folge, daß ein Agent bei der nächsten Auswahlentscheidung eher dazu bereit ist, diese Aktion bzw. Aktionsfolge erneut auszuführen. Der angegebene Mechanismus bewirkt eine Spezialisierung und damit auch eine soziale Differenzierung innerhalb der Ameisenkolonie, d. h. Ameisen mit hohen Gewichten für bestimmte Aufgaben werden als Spezialisten für diese Aufgaben angesehen.

Wie oben bereits angedeutet, basiert dieses Modell auf rein reaktiven Agenten. Die Agenten, die die Ameisen modellieren, verfügen also über keinerlei kognitive oder symbolverarbeitende Prozesse. Insbesondere haben die Agenten keine mentale Repräsentation ihrer sozialen Umgebung. Aber trotz des sehr einfachen Modellentwurfs erzeugt dieses Modell durchaus Phänomene, die sich auch in der beobachteten, natürlichen Soziogenese in Ameisenkolonien ablesen lassen.

2.1.3 Sugarscape

Einer der wohl bekanntesten Ansätze der agentenbasierten Modellbildung trägt den Namen *Sugarscape* (Epstein & Axtell, 1996). In dieser Multi-Agenten-Simulation, die aufgrund der einfachen Strukturierung der Agenten eine enge Verwandtschaft zu Modellen auf der Grundlage von zellulären Automaten zeigt,

wird die Entstehung sozialer Netzwerke, von Handel, von Märkten oder auch von kultureller Differenzierung untersucht.

Als Bezugssystem dient ein diskretisiertes, zweidimensionales Spielfeld in der Größe von 50×50 Feldern. In der Basisversion stellt jedes der Felder eine gewisse Menge an Zucker zur Verfügung, die von Agenten konsumiert werden kann, im Laufe der Zeit aber auch wieder regeneriert wird. Den einzelnen Feldern kann eine unterschiedliche Maximalmenge an Zucker zugeordnet werden. Dadurch können in bezug auf die Umweltbedingungen günstigere und ungünstigere Regionen der Umwelt modelliert werden.

Die Agenten müssen, um zu überleben, Zucker konsumieren und können sich nach vorgegebenen Regeln in ihrer Umwelt bewegen. Zudem sind Agenten charakterisiert durch eine individuelle Sichtweite und einen individuellen Stoffwechsel. Die Sichtweite bestimmt die Fähigkeit der Agenten, ihre Umgebung auszuspähen, und damit auch deren Geschick beim Erschließen günstiger Regionen. Der Stoffwechsel determiniert die Verbrauchsrate an Zucker. Aufgrund dieser beiden Eigenschaften können Agenten initialisiert werden, die in unterschiedlicher Weise für ein Leben in der gegebenen Umwelt geeignet sind.

Aufbauend auf diesem Szenario wird eine Vielzahl von Erweiterungen des Grundmodells vorgestellt, die unterschiedliche gesellschaftliche Phänomene illustrieren sollen. Die im Rahmen von *Sugarscape* beschriebenen Modellierungsansätze finden sich derzeit maßgeblich im Bereich der ökonomischen Modellbildung, der Sozialsimulation, sowie in Modellen für künstliche Sozialsysteme wieder.

2.1.4 EOS – Evolution of Organized Societies

Am Ende dieses Überblicks soll schließlich noch eine Studie stehen, in der Agenten mit etwas komplexeren kognitiven Fähigkeiten Verwendung finden. Es handelt sich hierbei um das sogenannte *EOS* (Evolution of Organized Societies)-Projekt (Doran, Palmer, Gilbert & Mellars, 1994). Im Rahmen dieses Projektes wurde die Entstehung komplexer sozialer Beziehungen in einer Population im Süd-Westen von Frankreich zur Zeit des Paläolithikum untersucht. Man nimmt an, daß sich zu dieser Zeit aufgrund eines gewissen Anpassungsdruckes an Umweltgegebenheiten relativ kleine, autonome Gruppen von Jägern und Sammlern zu wesentlich größeren sozialen Verbänden zusammenschlossen. Ebenso datiert man die Entwicklung eines sozialen Status und unterschiedlicher Rollen innerhalb menschlicher Gesellschaften auf diese Zeit (Mellars, 1985).

Im Rahmen des *EOS*-Projektes wurde auf der Grundlage eines agentenbasierten Simulationsmodells untersucht, wie sich unterschiedliche Umweltbedingungen auf die Interaktionsmuster zwischen Agenten auswirken. Das Modell besteht aus einer Umwelt, einer Population mobiler Agenten und Ressourcen, die Energie für die Agenten liefern. Die Agenten sind dabei als Produktionssysteme realisiert. Sie verfügen über kognitive Fähigkeiten wie Wahrnehmung der Umwelt, Formu-

lierung von Annahmen über die Umwelt, Entwurf von Plänen, Entscheiden über auszuführende Aktionen, sowie Beobachtung der Konsequenzen des eigenen Handelns.

2.1.5 Aktuelle Forschungsansätze

Die Liste der Anwendungsbeispiele agentenbasierter Modellbildung könnte an dieser Stelle durch eine Vielzahl ähnlicher Projekte ergänzt werden. Als Fazit kann man sagen, daß die bisher realisierten Projekte die agentenbasierte Modellbildung als wirkungsvolles Mittel zur Untersuchung komplexer Systeme, in denen insbesondere menschliches Verhalten von Bedeutung ist, herausstellen. Auf der Grundlage dieser Einsicht entstehen derzeit in zunehmendem Maße Forschungsaktivitäten, die sich auf die Anwendung und Weiterentwicklung der Agententechnologie im Zusammenhang mit Modellbildung und Simulation konzentrieren.

Ein Beispiel dafür manifestiert sich im Rahmen der *Sozionik*-Forschung (Malsch, 1997). Hier wird untersucht, in welcher Weise sich sozialwissenschaftliche Forschung auf der Grundlage der Agententechnologie mit Forschung im Bereich der Verteilten Künstlichen Intelligenz verknüpfen läßt, so daß für beide Seiten ein Erkenntnisgewinn möglich wird.

Ein Schwerpunkt liegt dabei auf der Erforschung und Modellierung künstlicher Sozialität. Es besteht die Hoffnung, dem sogenannten Mikro-Makro-Problem der Sozialforschung durch die Entwicklung agentenbasierter Simulationsmodelle näherzukommen. Das Mikro-Makro-Problem beschreibt die Frage, ob sich gesellschaftliche Strukturen als Resultat individueller Handlungsstrategien ergeben, umgekehrt individuelles Handeln durch gesellschaftliche Strukturen determiniert ist, oder Handlungen und Strukturen in einem gegenseitigen Bedingungs- und Ermöglichungsverhältnis stehen (Alexander, 1987). Die agentenbasierte Modellbildung und Simulation kann hier für einen methodischen Theorienvergleich genutzt werden. Darüber hinaus soll die agentenbasierte Modellbildung und Simulation dazu beitragen, sozialwissenschaftliche Theorien, die in der Regel nicht auf eine formale oder algorithmische Ebene absteigen, zu präzisieren und zu validieren (Brauer, Malsch & Rammert, 1999).

Ein weiteres reales Einsatzgebiet der agentenbasierten Modellbildung und Simulation eröffnet sich derzeit auch durch die Betrachtung betriebswirtschaftlicher Anwendungsszenarien (Herzog, Kirn, Krallmann, Spaniol & Zelewski, 1999). Das Hauptinteresse liegt hier vorrangig in der Erforschung und Weiterentwicklung der Agententechnologie im Hinblick auf deren Einsatz in realitätsnahen betriebswirtschaftlichen Anwendungsszenarien. Gemessen an den Anforderungen, die in betriebswirtschaftlichen Anwendungen entstehen, sollen wesentliche Fortschritte in den Arbeitsgebieten Agentenmodellierung und Agentenarchitekturen erzielt werden. Darüber hinaus sollen domänenunabhängige Standardbausteine für Agenten erarbeitet, sowie Vorschläge zur Standardisierung der Schnittstellen derartiger Bausteine gemacht werden.

In aktuellen Forschungsansätzen ist also eine klare Tendenz erkennbar, die Methoden der agentenbasierten Modellbildung in zunehmend komplexeren Anwendungsszenarien einzusetzen und die zugrundeliegenden Modellierungskonzepte in einer Weise zu verfeinern und zu erweitern, daß sie den hohen Anforderungen realer Anwendungsfälle gerecht werden können.

2.2 Sozialwissenschaftliche Grundlagen

Die Sozialwissenschaften liefern eine Reihe von Theorien, die herangezogen werden, um menschliches Handeln, Entscheiden und Verhalten zu beschreiben, zu erklären und zu prognostizieren. In klassischen Ansätzen wird häufig unterstellt, daß handlungsleitende Regeln Ausdruck rationaler Kalküle sind, d. h. daß Individuen diese Regeln bewußt verfolgen, um gesteckte Ziele zu erreichen. Diese Sichtweise des Menschen wird vorwiegend im Rahmen der rationalen Entscheidungstheorie und der Spieltheorie vertreten. Daneben existieren aber auch komplexere psychologische Theorien, die das menschliche Verhalten nicht nur aus dem Blickwinkel des rationalen Entscheiders betrachten, sondern auch andere Einflüsse, wie beispielsweise Emotionen, einbeziehen.

Wegen ihrer enormen Bedeutung für agentenbasierte Modellierungsansätze sollen in den folgenden Abschnitten wesentliche Gesichtspunkte der rationalen Entscheidungstheorie, der Spieltheorie, der Kognitionswissenschaft, sowie psychologischer Handlungstheorien im Überblick dargestellt werden.

2.2.1 Rationale Entscheidungstheorie

Gegenstand der Entscheidungstheorie sind im allgemeinen Situationen, in denen sich ein Akteur durch Auswahl einer Aktion in einen von mindestens zwei voneinander unterscheidbaren, neuen Zuständen versetzen kann (Tack, 1996). Für eine Entscheidung werden zwei wesentliche Einflussfaktoren als relevant erachtet. Zum einen gehen die erwarteten Konsequenzen einer Entscheidung als Einflußgröße in den Entscheidungsprozeß ein. Zum anderen finden auch individuelle Präferenzen auf der Menge der möglichen Konsequenzen Berücksichtigung.

Das Grundprinzip der Entscheidungstheorie beruht auf der Annahme, daß sich ein Entscheidungsproblem grundsätzlich auf ein einfaches Maximierungsproblem zurückführen läßt. Mit jeder Entscheidung wird zunächst eine erwartete Konsequenz in Verbindung gebracht. Alle Konsequenzen werden dann mit einem individuellen Wert versehen, der die Präferenz des Akteurs für die jeweilige Konsequenz zum Ausdruck bringt. Dadurch entsteht eine sogenannte Nutzenfunktion, die es durch Auswahl einer geeigneten Alternative zu maximieren gilt. Unter der Voraussetzung, daß sich auf der Menge aller Präferenzen eine Ordnung definieren läßt, wird also die Nutzenfunktion dadurch maximiert, daß einfach diejenige Al-

ternative ausgewählt und zur Ausführung gebracht wird, die mit der höchsten Präferenz versehen ist.

Das historisch älteste Beispiel einer Theorie menschlicher Entscheidungsprozesse ist die Theorie der Maximierung des subjektiv erwarteten Nutzens, die sogenannte *SEU-Theorie* (Edwards, 1954). Dieser Ansatz modelliert Entscheidungen unter Unsicherheit, was meint, daß die Konsequenzen der Entscheidungen erst im Nachhinein eintreten und der Entscheider deshalb nie absolut sicher sein kann, ob die getroffene Entscheidung die erwarteten Konsequenzen wirklich nach sich ziehen wird. Der subjektive Wert einer Konsequenz einer Entscheidung wird in der Entscheidungstheorie auch als *utility* bezeichnet. Bei Vorliegen von mindestens zwei möglichen Konsequenzen einer Entscheidung wird der relative Nutzen miteinander verglichen und der Entscheider gibt ein Präferenzurteil für eine der beiden Alternativen ab. Da es sich um Entscheidungen mit unsicheren Konsequenzen handelt, bedeutet Präferenz im einfachsten Fall, eine Option mit unsicheren Konsequenzen einer anderen Option mit ebenfalls unsicheren Konsequenzen vorzuziehen.

Die Aussage der *SEU-Theorie* ist schlicht, daß ein Entscheider immer diejenige Option wählen wird, die den höchsten subjektiv erwarteten Nutzen besitzt. Formal läßt sich dies wie folgt ausdrücken:

$$SEU_i = \sum_{j=1}^{n} p_j u_j \qquad \text{(Gl. 2.1)}$$

SEU_i ist dabei der Gesamtnutzen der Option i über alle Konsequenzen $j=1..n$ der Option i, wobei p_j die subjektive Wahrscheinlichkeit und u_j den subjektiven Nutzen einer Konsequenz j darstellt. Sowohl der Nutzen einer Konsequenz als auch ihre Unsicherheit sind in der *SEU-Theorie* als subjektive Größen konzipiert.

Laborversuche mit Testpersonen liefern empirische Belege dafür, daß sich Menschen in realen Entscheidungssituationen oftmals anders verhalten als es entscheidungstheoretische Ansätze vorhersagen. Beispielsweise zeigt sich, daß menschliche Entscheider oft Schwierigkeiten damit haben, die in die Entscheidung einfließenden Optionen miteinander zu vergleichen (Jungermann, Pfister & Fischer, 1998). Legt man Probanden mehrfach unterschiedliche Optionspaare aus Optionsmengen mit eindeutiger Rangfolge des theoretischen Gesamtnutzens vor, so zeigt ein bestimmter Prozentsatz der Probanden intransitive Präferenzen, was gegen das Transitivitätsaxiom der *SEU-Theorie* verstößt. Ebenfalls gegen eines der Axiome der *SEU-Theorie* (Unabhängigkeitsaxiom) verstößt die Tatsache, daß Probanden durch die Hinzunahme einer Konsequenz, die bei beiden Optionen gleich ist, zu einer Umkehrung ihrer Präferenzwahlen geführt werden. Des weiteren werden menschliche Entscheider von unterschiedlichen Beschreibungen von Optionen beeinflußt, die jedoch formal äquivalent sind. Solche Befunde führen zu der Einschätzung, daß entscheidungstheoretische Ansätze nur unter sehr spezifischen Bedingungen für eine adäquate Modellierung menschlicher Entscheidungsprozesse geeignet sind.

Um eine bessere Vorhersagbarkeit menschlicher Entscheidungen zu erreichen, wurden unter anderem emotionale Faktoren in die SEU-Grundkonzeption integriert (Jungermann, Pfister & Fischer, 1998). So erhöht z. B. die Hinzunahme antizipierter negativer Emotionen für den Fall des Auftretens einer negativen Konsequenz die Vorhersagegüte menschlicher Entscheidungsprozesse.

Im Rahmen der Entscheidungstheorie entwickelte Verfahren werden seit den Anfängen der Agententechnologie für die Modellierung der Verhaltenssteuerung autonomer Agenten eingesetzt. Das Rationalitätsprinzip wird dabei herangezogen, um in Situationen, in denen Agenten zwischen mehreren möglichen Aktionen oder Verhaltensweisen wählen können, möglichst optimale Entscheidungen zu treffen. Die oben angeführten Untersuchungen zeigen jedoch, daß Ansätze der rationalen Entscheidungstheorie im allgemeinen nicht ausreichen, um menschliches Verhalten angemessen zu modellieren. Es scheint sinnvoll, neben rationalen Aspekten zumindest auch emotionale Einflüsse nicht gänzlich außer Acht zu lassen.

2.2.2 Spieltheorie

Die Entscheidungstheorie beschäftigt sich mit Situationen, in denen das Verhalten eines einzelnen Agenten zu abschätzbaren Konsequenzen führt. Im Rahmen der Spieltheorie werden Situationen betrachtet, in denen mindestens zwei Akteure beteiligt sind. Die Entscheidung eines einzelnen Akteurs hängt dabei nicht nur von dessen individuellen Präferenzen ab, sondern auch davon, wie sich alle anderen Beteiligten verhalten. Durch die Einbeziehung und individuelle Bewertung der erwarteten Entscheidungen anderer Akteure entstehen also strategische Interdependenzen zwischen den Verhaltensweisen von Akteuren. Von Neumann und Morgenstern (1944) identifizierten dieses Szenario als grundlegend für viele Situationen des täglichen Lebens und für viele Gesellschaftsspiele.

Grundsätzlich unterscheidet man 2-Personen-Spiele von n-Personen-Spielen. Bei 2-Personen-Spielen besteht die Möglichkeit der Modellierung streng gegenläufiger Interessen. Man spricht hier von sog. Nullsummen-Spielen. Bei n-Personen-Spielen tritt oftmals der Effekt der Koalitionsbildung ein, da bei einigen Spielern partiell gleichgerichtete Interessen bzw. Präferenzen existieren. Diese Strukturbildungsprozesse setzen eine Interaktionsmöglichkeit zwischen den einzelnen Spielern zwingend voraus.

Als klassisches Beispiel einer spieltheoretischen Situation wird oftmals das sogenannte *Gefangenendilemma* angeführt (Rapaport & Chammah, 1965). Zwei Personen, sagen wir A und B, wurden wegen bewaffneten Raubüberfalls verhaftet. Die Polizei verfügt jedoch nur über Beweise, die gerade ausreichen, um den Tatbestand des Einbruchs nachzuweisen. Deshalb bittet die Polizei die beiden Männer in getrennte Räume und bietet ihnen das folgende „Geschäft“ an: Wenn sie zustimmen, gegen ihren Partner auszusagen, kommen sie ohne jegliche Gefängnisstrafe davon. Das Problem besteht nun darin, zu entscheiden, ob eine Aus-

sage gemacht werden soll oder nicht. Die Randbedingungen der Entscheidung sind in der folgenden Tabelle festgehalten.

	B sagt nicht aus		**B sagt aus**	
	Strafe für A	Strafe für B	Strafe für A	Strafe für B
A sagt nicht aus	3 Jahre	3 Jahre	30 Jahre	Freispruch
A sagt aus	Freispruch	30 Jahre	10 Jahre	10 Jahre

Abbildung 2.1: Payout-Matrix im Gefangenendilemma

Wenn beide schweigen, werden sie wegen Einbruchs verurteilt und erhalten eine Strafe von drei Jahren. In diesem Fall kommen beide sozusagen „mit einem blauen Auge“ davon. Wenn A aussagt, B jedoch schweigt, wird A freigesprochen, aber B erhält eine Freiheitsstrafe von 30 Jahren. Wenn alle beide aussagen, besteht keine Rechtfertigung mehr für eine Kronzeugenregelung, so daß beide dann zu einer Freiheitsstrafe von 10 Jahren verurteilt werden.

Spiele dieser Art werden in der Sozialpsychologie, der Soziologie, der Ökonomie, sowie in anderen Anwendungsgebieten häufig als Grundlage für empirische Studien verwendet. In vielen Fällen werden diese empirischen Studien jedoch auch durch Computersimulationen ersetzt, in denen autonome Agenten gegeneinander antreten, die mit unterschiedlichen Verhaltensstrategien ausgestattet sind. Das Ziel besteht oftmals darin, die Entstehung emergenter Phänomene, wie z. B. die Evolution stabiler Strukturen in einer ursprünglich ungeordneten Welt, zu untersuchen. Die Agenten und deren Interaktionen dienen dabei als mikroskopische Triebfeder für dynamische Prozesse auf einer makroskopischen Betrachtungsebene.

In dieser Hinsicht erscheinen spieltheoretische Ansätze als leistungsfähiges Vehikel, um komplexe Phänomene auf der Grundlage einfacher, autonomer Systeme künstlich zu erzeugen. Für die Modellierung realer Interaktions- und Austauschprozesse zwischen Menschen eignet sich die Spieltheorie jedoch nur bedingt. Durch die Beschränkung auf „unidimensionale“ Verhaltensweisen abstrahieren spieltheoretische Ansätze beispielsweise von Einflussfaktoren wie kognitiven Zuständen, Einstellungen, Emotionen, sozialen Rollen, Attributionsprozessen oder individueller Personenwahrnehmung, die das Verhalten realer Akteure in offensichtlicher Weise mitbestimmen (Mogel, Ohler & Urban, 2000).

2.2.3 Kognitionswissenschaften

Im Bereich der Kognitionswissenschaft wendet man sich der Modellierung kognitiver Prozesse von Lebewesen zu. Damit ergibt sich ein ideales Interaktionsfeld für Simulationsexperten, Psychologen, Kognitionswissenschaftler, Neurowissenschaftler, Soziologen, Philosophen etc. In diesem interdisziplinären Forschungsfeld geht es im wesentlichen darum, auf der Grundlage von empirischen Studien

und Computermodellen Erkenntnisse über menschliche Informationsverarbeitungsprozesse sowie auch in eingeschränkter Form über menschliches Handeln und die Dynamik seines Verhaltens zu gewinnen.

Legitimieren läßt sich diese Vorgehensweise durch die zentrale Annahme der Kognitionswissenschaft, daß Menschen und Computer in vergleichbarer Weise Informationen speichern und verarbeiten (jedenfalls dann, wenn man beide auf der für die Erforschung der kognitiven Prozesse relevanten Abstraktionsebene betrachtet). Damit sind Computermodelle in der Lage, Aufschluß über Aspekte der menschlichen Informationsverarbeitung zu geben und die Erforschung grundlegender Prinzipien des Geistes zu unterstützen, die für unterschiedlichste kognitive Vorgänge sowie für Mensch, Tier oder Maschine gleichermaßen gelten (Görz, 1995).

Die Computermodelle sollen grundsätzlich so gestaltet werden, daß sie den Gegebenheiten in der Realität möglichst nahe kommen. Es ergibt sich also die Aufgabe, Methoden zur Verfügung zu stellen, die es ermöglichen, menschliche Kognition, sowie menschliches Handeln, Entscheiden und Verhalten, das durch Sozialwissenschaftler empirisch erforscht wird, in adäquater Weise in Simulationsmodelle abzubilden. Als Ansatz für die Modellierung kognitiver Systeme setzt sich immer mehr das Paradigma autonomer Agenten durch und bildet dadurch eine gemeinsame Klammer zwischen Psychologie und Informatik (Mogel, Ohler & Urban, 2000).

Von den psychischen Kräften und Funktionen des Menschen befaßt sich die Kognitionswissenschaft primär mit den kognitiven Funktionen und blendet Motive und Emotionen noch weitgehend aus. Aber das kognitive Funktionieren des Menschen findet in sozialen Situationen mit den damit verbundenen Motiven und Emotionen statt. Kognitionen sind nicht Selbstzweck, sondern münden in Handlungen. Die Psychologie geht deshalb den Weg, den größeren Rahmen des menschlichen Handelns zur Grundlage ihrer Modellbildung zu machen. Es entstehen die sogenannten Handlungstheorien. Diese stellen ein System von Begriffen und konzeptuellen Annahmen bereit, um das Handeln meist einzelner Akteure systematisch zu beschreiben und zu erklären (Mogel, Ohler & Urban, 2000).

2.2.4 Psychologische Handlungstheorien

Die in den beiden vorangegangenen Abschnitten vorgestellten Ansätze stellen Modellierungsansätze für bestimmte Teilbereiche menschlichen Denkens, Entscheidens, Handelns und Verhaltens bereit. Also beschreiben und erklären sie auch nur diese Teilbereiche seines Handelns, Entscheidens und Verhaltens. Aber bereits in den experimentell überprüften Geltungsbereichen zeigen sich ihre Beschränkungen zur Modellierung menschlichen Verhaltens in individuellen und sozialen Situationen. So verwundert es nicht, daß sie für sich betrachtet auch nicht hinreichen, komplexere psychische Prozesse zu modellieren. An dieser Stelle setzen psychologische Handlungstheorien an.

Der Begriff der Handlung wird in der Psychologie meist in Abgrenzung zum Verhalten eingeführt. Während Verhalten nur auf die wenn auch oft komplexen beobachtbaren Aktionen und Reaktionen von Organismen abzielt, ist allen Handlungstheorien der Rekurs auf innere Zustände der Akteure gemeinsam. Handlungen sind intentional, also auf ein Ziel, das der Akteur verfolgt, ausgerichtet. Es wird angenommen, daß für die Zielgerichtetheit eines Akteurs neben kognitiven Einflüssen in gleichem Maße auch Motive und Emotionen von Bedeutung sind. Während viele Prozesse in der menschlichen Informationsverarbeitung automatisch ablaufen, gelten Handlungen gemeinhin als bewußt oder zumindest als bewußtseinsfähig. Das Handlungsziel ist während der Handlungsausführung im Arbeitsgedächtnis gespeichert, und es kann darauf zugegriffen werden. Weitere Bestimmungsstücke sind die hierarchisch-sequentielle Organisation von Handlungen und ihre Situationsbezogenheit (Dörner & Selg, 1996).

Die Motive sind für die Bildung von Handlungszielen verantwortlich. Oft werden Varianten des in Kap. 2.2.1 eingeführten SEU-Ansatzes herangezogen, um die motivationale Komponente der Zielauswahl zu modellieren. Diese Modelle bilden im Rahmen von Handlungstheorien jedoch nur eine Teilkomponente, die mit anderen psychischen Modulen dynamisch zusammenwirkt. Die Aufmerksamkeit des Handelnden richtet sich nach der Etablierung des Handlungsziels auf dessen Erreichung, wobei Barrieren zu überwinden sind, die oft die Formulierung von untergeordneten Teilzielen notwendig machen. Unter Umständen muß das Handlungsziel jedoch auch aufgegeben werden, wenn sich die Barrieren als unüberwindlich herausstellen.

In der kognitiven Komponente handlungstheoretischer Ansätze findet vor allem die Bildung eines Handlungsplanes zur Zielerreichung statt. Die Handlungsplanung erfolgt jedoch nicht bis ins Detail. Diejenigen Schritte, die zuerst zur Ausführung kommen sollen, sind detaillierter repräsentiert als spätere Handlungsphasen. Die Flexibilität menschlichen Handelns verdankt sich unter anderem dieser partiellen Offenheit von Handlungsplänen. Während der Umgang mit übergeordneten Handlungszielen im kognitiven System sich als Denk- oder Problemlöseprozeß gestalten kann, stehen für die Erreichung untergeordneter Handlungsziele bereits Routinen im Langzeitgedächtnis zur Verfügung, die nur noch an die jeweilige Situation angepaßt werden müssen. Das kognitive System reguliert zudem die Koordination der Aktionen. Damit wird gewährleistet, daß die Planbestandteile sequentiell in der richtigen Ordnung, zum richtigen Zeitpunkt und kontextadäquat zur Ausführung kommen. Das Wahrnehmungssystem des Akteurs gewährleistet, daß sein kognitives System mit Rückmeldungen, ob die angestrebten Handlungsteilziele erreicht wurden, versorgt wird.

Im Verlauf einer Handlung sind die kognitiven Funktionen und die Reaktionen des emotionalen Systems vollständig miteinander verzahnt. Sachverhalte und Ereignisse werden im Hinblick darauf bewertet, wie förderlich respektive hinderlich sie für die übergeordnete Zielsetzung sind. Diese Bewertungen sind mit emotionalen Reaktionen wie Hoffnung oder Furcht verbunden, was zu einer selektiven Steuerung der Aufmerksamkeit des Akteurs führt und auch seine Gedächtnisrep-

räsentationen in bezug auf späteres Handeln in vergleichbaren Situationen beeinflußt. Konkrete Konsequenzen von Aktionen lösen Emotionen aus, die wiederum handlungsleitend für neue Aktionen sein können. Hier besitzen Emotionen motivationalen Charakter (Mogel, Ohler & Urban, 2000).

Handlungstheoretische Ansätze besitzen den Vorteil, daß sie im Gegensatz zu den vorher genannten Modellen eine Integration der unterschiedlichen psychischen Kräfte und Funktionen anstreben. Dieser Vorteil wird jedoch oft auf Kosten einer fehlenden Präzision der entsprechenden Modelle erkauft. Die Modellkonstrukteure versäumen, das Zusammenspiel der beteiligten Komponenten formal hinreichend zu spezifizieren. Diese formale Spezifikation der Wechselwirkungen der Modellkomponenten ist jedoch bei der Erstellung von Simulationsmodellen zwingend notwendig und führt damit zu einer höheren Präzision der psychologischen Modelle. Zur Unterstützung der psychologischen Modellbildung bietet sich ein architekturbasiertes Vorgehen an, d. h. die Schaffung eines allgemeinen Rahmens, der eine integrative Modellierung der verschiedenen Einflußbereiche menschlichen Handelns, Entscheidens und Verhaltens ermöglicht. Fallbasiert kann diese Architektur dann mit Dynamik gefüllt werden. Was sich an psychologischen Inhalten in dieser Architektur darstellen läßt, hat die Gewähr, daß es formal präzise dargestellt ist. Diese Ausführungen zeigen also deutlich auf, daß auch im Zusammenhang mit psychologischen Fragestellungen Methoden der agentenbasierten Modellbildung und Simulation gewinnbringend eingesetzt werden können, sofern geeignete Modellierungsansätze zur Verfügung gestellt werden.

2.3 Softwarewerkzeuge für die agentenbasierte Simulation

Die Implementierung agentenbasierter Simulationsmodelle stellt sehr hohe technische Anforderungen an den Modellentwickler. Ein wesentliches Problem, das sich im Zusammenhang mit der Erstellung von Simulationsmodellen und insbesondere auch von agentenbasierten Simulationsmodellen sehr häufig beobachten läßt, ist, daß für die Modellrealisierung nicht Simulationsexperten zu Rate gezogen werden, sondern diese Aufgabe häufig von Wissenschaftlern in den jeweiligen informatikfremden Fachgebieten in mühevoller und zeitaufwendiger Arbeit selbst übernommen wird. Langton et. al. (1996) beschreiben diesen Sachverhalt sehr treffend mit dem Satz *„computer modeling frequently turns good scientists into bad programmers“*.

Viele Wissenschaftler haben nur sehr rudimentäre Kenntnisse in den Bereichen Modellbildung, Simulation und Softwareengineering. Aus diesem Grund ist es nicht verwunderlich, daß für die Realisierung agentenbasierter Simulationsmodelle in vielen Fällen prozedurale Sprachen wie Pascal oder C bzw. in neueren Ansätzen objektorientierte Sprachen wie C++ oder Java ohne weitere simulationsspezifische Unterstützung eingesetzt werden. Derartig entwickelte Modelle sind

aus Sicht des Simulationsexperten oder Software-Ingenieurs auch häufig schlecht entworfen und strukturiert, so daß sich im Laufe der Zeit ernsthafte Probleme im Hinblick auf deren Verständlichkeit, Vergleichbarkeit, Erweiterbarkeit und Wartbarkeit ergeben.

Um diesen Problemen entgegenzuwirken, sind in den letzten Jahren Softwarewerkzeuge entwickelt worden, die die Implementierung agentenbasierter Simulationsmodelle unterstützen und standardisieren sollen. Im Hinblick auf die zugrundeliegenden Implementierungsparadigmen kann hier im wesentlichen zwischen objektorientierten und logikorientierten Ansätzen unterschieden werden. In den beiden folgenden Abschnitten soll nun jeweils ein charakteristischer Vertreter dieser beiden Klassen von Systemen im Überblick dargestellt werden.

2.3.1 Swarm

Swarm (Langton, Minar, Burkhart & Askenazi, 1996) ist eine aus dem Bereich Artificial Life stammende und mittlerweile sehr weit verbreitete Softwareplattform zur Erstellung agentenbasierter Simulationsmodelle. Die Entwicklung von *Swarm* ist durch den Grundgedanken motiviert, ein standardisiertes und wohlstrukturiertes Software-Labor zu schaffen, das im Rahmen der agentenbasierten Simulation in verschiedenen Anwendungsgebieten eingesetzt werden kann.

Die Funktionalität von *Swarm* wird dem Anwender in Form von objektorientierten Klassenbibliotheken zur Verfügung gestellt. Die Bibliotheken umfassen dabei wiederverwendbare Softwarebausteine in Objective C, die den Modellaufbau, die Sammlung, Analyse und Anzeige von Simulationsdaten, sowie die Kontrolle von Simulationsexperimenten unterstützen.

Swarm-Modelle arbeiten rein diskret und ereignisgesteuert. Das Basiselement eines *Swarm*-Modells ist ein *Agent*. Als *Agent* wird im Rahmen von *Swarm* jeder beliebige Akteur in einem System bezeichnet. Technisch betrachtet ist ein Agent eine Entität, die Ereignisse generieren kann. Diese Ereignisse können sowohl den Agenten selbst als auch andere Agenten beeinflussen.

Agenten können auf der nächst höheren Abstraktionsebene zu sogenannten *Swarms* zusammengeschlossen werden. Ein *Swarm* besteht aus einer Menge zugeordneter Agenten, sowie einer Ereignisliste, die den Fortschritt der Simulation kontrolliert. *Swarms* können zudem ihrerseits wieder Agenten sein. Auf der Grundlage von *Swarms* wird also ein Hierarchiekonzept realisiert, das den Aufbau komplexer Modelle als Aggregat, das sich über mehrere Hierarchiestufen erstrecken kann, ermöglicht.

Swarms können dynamisch zur Laufzeit eines Modells generiert und vernichtet werden. Dieser Mechanismus erlaubt die Erstellung von Modellen mit über die Laufzeit variabler Struktur. Darüber hinaus kann dieser Mechanismus genutzt werden, um Agenten mit mentalen Modellen auszustatten, die sie nutzen können, um Erkenntnisse über das eigene Verhalten oder auch das Verhalten ihrer Umgebung, sowie anderer Agenten zu gewinnen.

Agenten werden in *Swarm* als Objekte modelliert. Typen von Agenten werden mit Hilfe von Klassen beschrieben, die bei der Implementierung von Basisklassen der *Swarm*-Bibliotheken abgeleitet und für die Anforderungen der jeweiligen Fallstudie spezialisiert werden können. Spezifische Agenten werden als Instanzen bestimmter Klassen erzeugt. Jedes Objekt trägt eine Menge von Zustandsvariablen, die den aktuellen Zustand des korrespondierenden Agenten beschreiben. Die Methoden des Objekts spezifizieren das Verhalten des Agenten.

Ein *Swarm*-Modell besteht auf der höchsten Abstraktionsebene immer aus einem *Swarm*, der eine Gruppe von Agenten umfaßt, sowie einer zugehörigen Ereignisliste. Eine Besonderheit bei *Swarm* besteht darin, daß auch die Umwelt der Agenten als Menge von Agenten modelliert wird. *Swarm* arbeitet also rein agentenbasiert und betrachtet die Umwelt, sowie andere Klassen zur Steuerung der Simulation und der Simulationsexperimente lediglich als spezielle Typen von Agenten.

Aufgrund der Flexibilität des Systems und der Vielzahl bereits realisierter Anwendungsprojekte (vgl. dazu auch http://www.swarm.org) hat sich *Swarm* in den letzten Jahren als das wohl führende System im Bereich der agentenbasierten Simulation herauskristallisiert.

2.3.2 SDML

SDML (strictly declarative modelling language) ist eine Programmiersprache, die für die Modellierung von Interaktionen zwischen Agenten innerhalb sozialer Strukturen ausgelegt ist (Moss, Gaylard, Wallis & Edmonds, 2001).

SDML verfügt über logikorientierte, sowie elementare objektorientierte Eigenschaften, ermöglicht einen modularen Modellaufbau und legt großen Wert auf Wiederverwendbarkeit von Programmcode. Der Fokus der Sprache liegt auf Konzepten zur Modellierung individueller, kognitiver Eigenschaften von Agenten und zur adäquaten Repräsentation organisatorischer Strukturen.

Die Grundelemente eines *SDML*-Modells sind, wie auch in *Swarm*, Agenten. *SDML* unterscheidet jedoch im Gegensatz zu *Swarm* zwischen herkömmlichen Objekten und Agenten. Agenten sind im Vergleich zu Objekten zusätzlich mit einer Menge von Regelbasen ausgestattet, die das Verhalten der Agenten zu unterschiedlichen Zeitpunkten beschreiben. Die Grundelemente der Regelbasen sind logische Regeln, die aus einem Bedingungsteil und einer Menge von Konsequenzen bestehen.

SDML unterstützt das Konzept der mehrfachen Vererbung und bietet verschiedene, vordefinierte Klassen von Agenten an, die für die Erstellung individueller Agentenklassen genutzt werden können. Die Klasse *CompositeAgent* unterstützt beispielsweise die Beschreibung von Multi-Agenten-Strukturen. Die Klasse *LoopingAgent* stellt Funktionalität für die Ablaufsteuerung der Simulation bereit und ermöglicht die Iteration über die verschiedenen Zeitpunkte, die für das Fortschreiten der Simulation von Bedeutung sind. Im Gegensatz zu klassischen logik-

orientierten Ansätzen existiert in *SDML* eine explizite Repräsentation der Zeit. Dadurch besteht in *SDML* die Möglichkeit, zeitabhängig verschiedene Verhaltensbeschreibungen von Agenten zu aktivieren.

Für die Repräsentation organisatorischer Strukturen stellt *SDML* eine sogenannte *Container*-Hierarchie bereit, die eine Beschreibung hierarchischer Strukturen mit beliebiger Tiefe erlaubt. Der Typ des Containers legt fest, in welcher Weise die Verhaltensbeschreibungen der im Container eingeschlossenen Agenten abgearbeitet werden. Vorgesehen ist dabei eine sequentielle, verschränkte oder parallele Ausführung der einzelnen Agenten.

Die Kommunikation zwischen Agenten wird in *SDML* über Datenbasen organisiert. Informationen werden durch die Resultate gefeuerter Regeln beschrieben. Agenten, die Informationen produzieren, stellen die Resultate zum Zwecke der Kommunikation entweder unter Beachtung definierter Zugriffsrechte direkt in die Datenbasen der Empfänger oder in gemeinsam lesbare Datenbasen. Von diesen Stellen aus können die Informationen dann von den Empfängern abgeholt und weiterverarbeitet werden.

Die Stärke von *SDML* liegt im Bereich der Modellierung kognitiver Fähigkeiten von Agenten und organisatorischer Strukturen. Die Sprache profitiert dabei von der Eleganz des deklarativen Charakters logikorientierter Sprachen. Allerdings muß die zukünftige Entwicklung zeigen, ob sich derartige Ansätze angesichts des ständig wachsenden Einflusses rein objektorientierter Sprachen behaupten werden.

2.4 Zusammenfassung und Fazit

Die Projekte der letzten Jahre haben gezeigt, daß die agentenbasierte Modellbildung und Simulation ein wirkungsvolles Mittel zur Untersuchung komplexer Systeme ist, insbesondere dann, wenn menschliches Handeln, Entscheiden und Verhalten maßgeblich die Funktionsweise des Systems beeinflussen. Auf der Grundlage dieser Erkenntnis entstehen in zunehmendem Maße Forschungsaktivitäten, die sich zum Ziel setzen, die agentenbasierte Modellbildung im Rahmen komplexerer und realitätsnäherer Anwendungsszenarien einzusetzen (s. z. B. Mosler, 2002). Großen Nutzen verspricht man sich hier zum Beispiel für die Weiterentwicklung sozialwissenschaftlicher Theorien, da sich mit agentenbasierten Modellen wesentlich komplexere soziale Zusammenhänge modellieren lassen als mit herkömmlichen Methoden (Malsch, 1997). Zudem können agentenbasierte Modelle für einen methodischen Theorienvergleich genutzt werden und dazu beitragen, sozialwissenschaftliche Theorien, die in der Regel nicht auf eine formale oder algorithmische Ebene absteigen, zu präzisieren und zu validieren (Brauer, Malsch & Rammert, 1999).

Als Konsequenz dieser Forschungstendenzen ergibt sich jedoch auch die Notwendigkeit, die methodologischen Aspekte der agentenbasierten Modellbil-

dung weiter voranzutreiben. Es reicht für komplexere Anwendungsfälle nicht mehr aus, das Verhalten von Agenten auf der Grundlage einfacher, rationaler Kalküle zu beschreiben, sowie deren Interaktion auf spieltheoretische Zusammenhänge zu beschränken. Vielmehr müssen Modellierungskonzepte entwickelt werden, die den hohen Anforderungen realer Anwendungsfälle gerecht werden können und insbesondere eine komplexere Sichtweise auf die Modellierung menschlichen Handelns, Entscheidens und Verhaltens eröffnen.

Es stellt sich also heraus, daß die Konstruktion agentenbasierter Modelle eine ganz grundlegende und in hohem Maße komplexe Aufgabe ist, bei der eine Reihe von Problemen sowohl hinsichtlich des Entwurfes als auch der Implementierung zu Tage treten (Gilbert & Troitzsch, 1999). Diese Arbeit hat das Ziel, sowohl für die Konzeption als auch die Realisierung agentenbasierter Simulationsmodelle, in denen der Faktor Mensch von ausschlaggebender Bedeutung ist, geeignete Methoden zur Verfügung zu stellen. Es wird das domänenunabhängige Referenzmodell *PECS* vorgestellt, das einen wohlstrukturierten Entwurf komplexer, agentenbasierter Modelle ermöglicht. Zudem wird anhand mehrerer Fallstudien aufgezeigt, wie *PECS* im Zusammenspiel mit den Konzepten der objektorientierten Modellspezifikation wirkungsvoll für die Implementierung agentenbasierter Modelle eingesetzt werden kann. Der in dieser Arbeit vorgestellte Ansatz soll maßgeblich dazu beitragen, die Komplexität der Entwurfsaufgabe, sowie den daraus resultierenden Zeit- und Arbeitsaufwand zu verringern, der bei der Entwicklung derartiger agentenbasierter Simulationsmodelle entsteht.

3 REFERENZMODELLE IN MODELLBILDUNG UND SIMULATION

Der Schwerpunkt dieser Arbeit liegt auf der Bereitstellung eines domänenunabhängigen Referenzmodells, das den Entwurf agentenbasierter Simulationsmodelle, in denen menschliches Verhalten von zentraler Bedeutung ist, unterstützt.

Bevor in den nachfolgenden Kapiteln wesentliche Anforderungen und der Aufbau des Referenzmodells *PECS* vorgestellt werden, sollen in diesem Kapitel zunächst einige grundlegende Begriffe eingeführt, sowie das Konzept der Referenzmodelle im Bereich der Modellbildung und Simulation genauer diskutiert werden.

3.1 Der grundsätzliche Ablauf von Modelluntersuchungen

Als *Simulation* bezeichnet man ein Verfahren zur Nachbildung eines Systems mit seinen dynamischen Prozessen in einem experimentierbaren Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind (VDI, 1997).

Modelluntersuchungen mit Hilfe der Simulation tauchen in vielen unterschiedlichen Anwendungsfeldern auf. Trotz der Verschiedenheit der Anwendungen folgen Modelluntersuchungen in der Regel einem einheitlichen Grundschema. Schmidt (2000) beschreibt diesen grundsätzlichen Ablauf in der folgenden Weise:

Ausgangspunkt einer Modelluntersuchung ist ein reales System. Als *reales System* soll ein Ausschnitt aus der realen Welt bezeichnet werden, den sich der Mensch herausgreift, um ihn einer genaueren Untersuchung zu unterziehen und vertiefte Erkenntnisse über ihn zu erlangen.

Ein Hauptproblem, das in diesem Zusammenhang zu Tage tritt, besteht darin, daß der interne Wirkmechanismus, der dem realen System zugrunde liegt, zunächst verborgen ist. Um trotzdem Informationen über das Verhalten des realen Systems zu erhalten, können Experimente mit dem System durchgeführt, Messungen vorgenommen, Erfahrungen festgehalten, sowie Beobachtungsdaten gesammelt werden.

Im Rahmen der sich anschließenden *Systemanalyse* wird auf der Grundlage der vorhandenen Informationen ein *abstraktes (gedankliches) Modell* entwickelt. Das abstrakte Modell enthält dabei die wesentlichen Modellelemente, deren Wechselwirkungen, Ersatzdarstellungen für relevante Einflüsse aus der Umgebung des Systems, sowie Systemdaten. Wichtig ist an dieser Stelle die Einsicht, daß bei der Entwicklung des abstrakten Modells die Idee der Abstraktion und Idealisierung von entscheidender Bedeutung ist. *Abstraktion* bedeutet dabei, daß nicht alle Eigenschaften und Verhaltensweisen des realen Systems im Modell repräsentiert werden, sondern nur solche, die im Hinblick auf die gegebene Fragestellung als relevant erachtet werden. *Idealisierung* meint, daß reale Gegebenheiten im Modell in idealisierter Weise dargestellt werden. Ein abstraktes Modell enthält also eine auf wesentliche Eigenschaften und Verhaltensbeschreibungen verkürzte und idealisierte Repräsentation eines realen Systems.

Diese Einsicht beeinflußt auch in entscheidendem Maße das Vorhaben, menschliches Handeln, Entscheiden und Verhalten auf der Grundlage autonomer Agenten zu modellieren. In der realen Welt entsteht menschliches Verhalten auf der Grundlage eines komplexen Gefüges von Einflüssen und Wirkmechanismen. Dabei kommen z. B. die individuelle Erfahrung eines Menschen, soziale Rollenerwartungen, konkurrierende Motive, Stimmungen, sowie eine Vielzahl anderer Einflüsse zum tragen. Es mag also aufgrund der enormen Komplexität im ersten Moment aussichtslos erscheinen, menschliches Verhalten in Simulationsmodellen abbilden zu wollen. Nimmt man jedoch zur Kenntnis, daß Modelle keine Replikate, sondern verkürzte, durch Abstraktion und Idealisierung hervorgegangene Abbildungen von Systemen sind, stellt sich dieses Vorhaben nicht mehr zwangsläufig als unüberwindlich dar. Dieser grundlegende Sachverhalt befreit den Modellentwickler von der Notwendigkeit, in ein Modell alle Eigenschaften und Verhaltensweisen aufzunehmen, die den Menschen in irgendeiner Weise auszeichnen. Vielmehr kann eine Beschränkung auf die Menge derjenigen Einflüsse erfolgen, deren Einbeziehung für die gegebene Fragestellung als unerläßlich erachtet wird. Durch diese Einschränkung kommt man zu vereinfachten und handhabbaren Modellen für menschliches Handeln, Entscheiden und Verhalten, die aber nichtsdestotrotz wertvolle Einsichten über das zu untersuchende System liefern können.

Im nächsten übergeordneten Schritt einer Modelluntersuchung wird das abstrakte Modell auf der Grundlage einer formalen Beschreibungsmethode, z. B. einer Programmiersprache oder Modellspezifikationssprache, in ein formales, experimentierbares Modell, das auch *Simulationsmodell* genannt wird, überführt.
Mit dem Simulationsmodell werden dann *Experimente* durchgeführt, in denen Daten über das Verhalten des Modells gesammelt werden. Die auf diesem Wege gewonnenen Modelldaten dienen zum einen der *Validierung* des Modells. Die Validierung erfolgt durch Vergleich der Modelldaten mit den erhobenen Daten des realen Systems. Darüber hinaus erlauben die Informationen, die auf der Grundlage eines validen Modells generiert wurden, Rückschlüsse auf das Verhalten des zugrunde liegenden realen Systems. Die am Modell gewonnenen Informationen

werden also im letzten Schritt in einer geeigneten Weise interpretiert und auf das ursprüngliche System zurückübertragen.

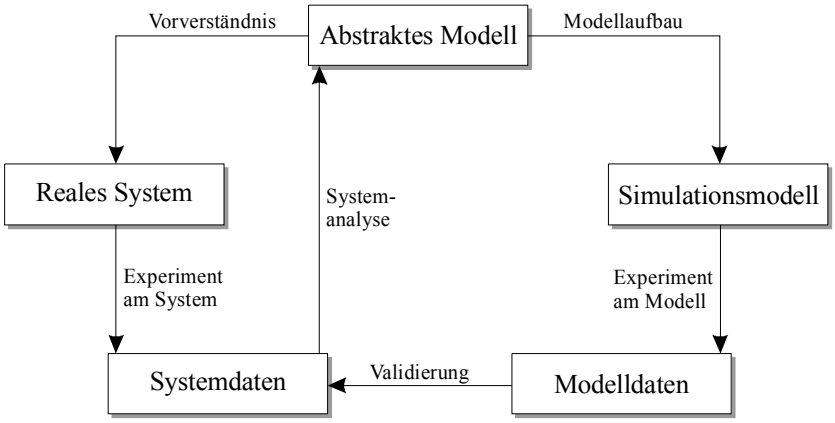

Abbildung 3.1: Der grundsätzliche Ablauf einer Modelluntersuchung in Anlehnung an Schmidt (2000)

Modellbildung und Simulation werden als Methode aber nicht nur dann eingesetzt, wenn es um die Untersuchung eines real existierenden Systems geht. Vielmehr finden sich auch Bereiche, wie beispielsweise Artificial Life oder Artificial Social Systems, in denen man anstelle der klassischen analytischen Vorgehensweise häufig ein synthetisches Vorgehen wählt. Dabei werden in Simulationsmodellen künstliche Systeme mit beliebigen Eigenschaften und Verhaltensbeschreibungen aufgebaut, für die nicht unbedingt eine Entsprechung in der realen Welt existieren muß. Auch hier werden mit Hilfe von Experimenten Informationen über das Verhalten des Modells gesammelt und Modelldaten erzeugt. Eine Validierung des Modells im herkömmlichen Sinne kann in diesem Fall jedoch nicht erfolgen, da die Vergleichsdaten aus dem realen System fehlen. Man zieht sich in diesem Fall sehr häufig auf Plausibilitätsbetrachtungen und Expertenwissen zurück und sucht nach Übereinstimmungen auf einer phänomenologischen Ebene. Lassen sich für die durch das Modell erzeugten Phänomene vergleichbare Phänomene in der realen Welt finden, so kann man annehmen, daß die dem Modell zugrunde gelegten Wirkmechanismen auch in der realen Welt von Bedeutung sind.

3.2 Der Begriff *Referenzmodell*

Ein Arbeitsgebiet, das im Bereich der Modellbildung und Simulation immer mehr an Bedeutung gewinnt, beschäftigt sich mit der Entwicklung von Referenzmodellen. Der Begriff *Referenzmodell* wird derzeit in der Literatur nicht einheitlich verwendet. Aus diesem Grund erscheint es sinnvoll, zunächst einmal die Bedeutung zu klären, die diesem Begriff im Rahmen dieser Arbeit zugeordnet wird.

Im Bereich der Simulation in Produktion und Logistik, einem Arbeitsgebiet, in dem Modellbildung und Simulation bereits einen sehr fortgeschrittenen Stand erreicht haben, beschäftigt man sich derzeit intensiv mit Referenzmodellen. Die Fachgruppe „Simulation in Produktion und Logistik“ der Arbeitsgemeinschaft Simulation (ASIM) schlägt in diesem Zusammenhang die folgende Definition vor (Klinger, 1999):

> *Ein Referenzmodell umfaßt eine systematische und allgemeingültige Beschreibung eines definierten Bereiches der Realität mit den für eine vorgegebene Aufgabenstellung relevanten charakteristischen Eigenschaften und legt das zugehörige Modellierungskonzept fest. Im Bereich der Simulation dienen Referenzmodelle als Konstruktionsschemata für den Entwurf von Simulationsmodellen.*

Ein Referenzmodell bezieht sich auf eine definierte Modellierungsaufgabe und stellt für diese ein geeignetes Modellierungskonzept zur Verfügung. Die Modellierungsaufgabe kann dabei auf unterschiedlichen Abstraktionsebenen angesiedelt sein. Referenzmodelle können sich auf einen sehr weit gefassten Anwendungsbereich beziehen und dadurch sehr allgemeinen Charakter haben. Ebenso kann es jedoch vorkommen, daß Referenzmodelle auf sehr eng begrenzte Anwendungsbereiche ausgerichtet sind und Anhaltspunkte für die Lösung darin auftretender Detailprobleme liefern. Das durch ein Referenzmodell bereitgestellte Modellierungskonzept umfaßt strukturelle Aspekte, sowie grundlegende Eigenschaften und Abläufe des spezifizierten Anwendungsbereiches. Referenzmodelle können als Schablonen bzw. Konstruktionsschemata betrachtet werden, die den Entwurf und die Konzeption von Simulationsmodellen unterstützen.

Die Beschreibung von Referenzmodellen muß in allgemeingültiger Form erfolgen. Die durch ein Referenzmodell vorgegebenen Konzepte müssen unabhängig von den Konstrukten eines speziellen Werkzeuges oder einer speziellen Sprache beschrieben werden. Im Rahmen der Beschreibung eines Referenzmodells sollen Angaben darüber gemacht werden, welche Anforderungen ein Simulationswerkzeug erfüllen muß, damit es für die Realisierung konkreter Modelle, die auf dem Referenzmodell basieren, eingesetzt werden kann. Die allgemeine Ausrichtung der Beschreibung soll den Einsatz von Referenzmodellen im Zusammenhang mit möglichst vielen geeigneten Werkzeugen ermöglichen.

Die Beschreibung von Referenzmodellen muß darüber hinaus auf systematische Weise erfolgen. Referenzmodelle müssen also klar strukturiert und methodologisch begründet sein. Eine klare und übersichtliche Strukturierung eines Referenzmodells trägt zu dessen Verständlichkeit und letztlich auch zu dessen Anwendbarkeit bei. Zudem vereinfacht eine klare Strukturierung die Erweiterung und Anpassung des Referenzmodells an die Anforderungen konkreter Aufgabenstellungen. Die Forderung nach methodologischer Begründung bedeutet, daß ein Referenzmodell die spezifischen Eigenschaften und Kenntnisse des zugehörigen Anwendungsbereiches berücksichtigt. Dies kann beispielsweise dadurch gesche-

hen, daß in die Entwicklung eines Referenzmodells entsprechendes Expertenwissen mit eingebracht wird.

3.3 Modelluntersuchungen auf der Grundlage von Referenzmodellen

Referenzmodelle leisten eine Vorarbeit für die Modellentwicklung, indem sie auf der Grundlage einer Analyse des zu untersuchenden Systems charakteristische Strukturen, Eigenschaften und Abläufe vorgeben. Sie machen Aussagen darüber, wie Modelle strukturiert werden können und in welcher Weise die Modelldynamik beschrieben werden kann.

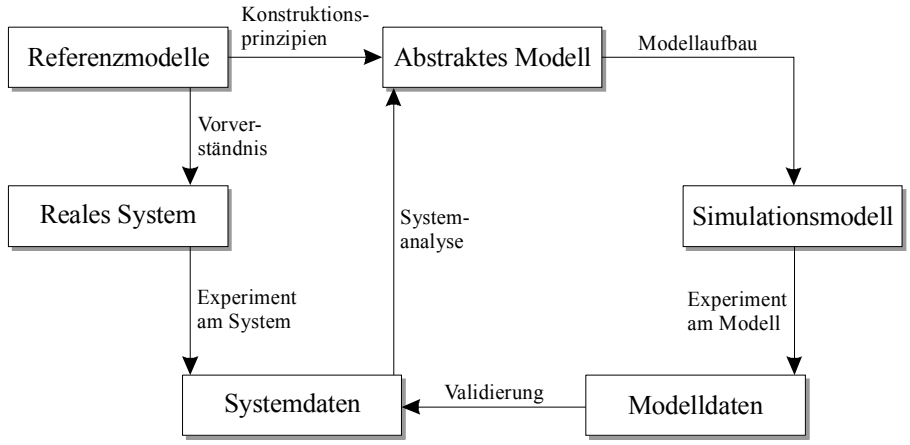

Abbildung 3.2: Der grundsätzliche Ablauf einer Modelluntersuchung bei Verwendung von Referenzmodellen

Die Abgrenzung des zu untersuchenden Systems und der Entwurf des abstrakten Modells sind nicht nur von den Gegebenheiten des Systems abhängig. Von entscheidender Bedeutung sind hier beispielsweise auch das Vorverständnis, die Betrachtungsweise, sowie die Erfahrung des Modellentwicklers. Die Sichtweise auf das reale System ist von Anfang an durch ein zunächst noch vorläufiges, ungenaues und provisorisches Modell geprägt, das sich der Modellentwickler vom realen System gebildet hat (Schmidt, 2000a). Die Gestaltung des abstrakten Modells wird neben den oben genannten Sachverhalten zusätzlich durch die Wahl der im nachfolgenden Schritt zu verwendenden Modellbeschreibungsmethode beeinflußt, da diese den Rahmen für die formale Beschreibung und die grundsätzliche Strukturierung des Modells festlegt. Die Entwicklung eines brauchbaren abstrakten Modells ist in vielen Fällen eine schwierige intellektuelle Leistung. Es gilt in diesem Schritt, den Wirkmechanismus, der dem zu untersuchenden System zugrunde liegt, zu identifizieren und die wesentlichen Aspekte in ein brauchbares abstraktes Modell zu integrieren.

Referenzmodelle können sowohl bei der Abgrenzung des realen Systems als auch bei der Entwicklung eines abstrakten Modells Hilfestellungen geben. Sie unterstützen über die Entwicklung eines gewissen Vorverständnisses den Schritt der Systemabgrenzung und stellen Konstruktionsprinzipien bereit, die im Rahmen der Systemanalyse für den Entwurf eines abstrakten Modells genutzt werden können. Derartige Konstruktionsprinzipien umfassen Vorgaben im Hinblick auf die Strukturierung, sowie auf wesentliche Eigenschaften und Abläufe in Modellen.

Ein Referenzmodell stellt im wesentlichen ein konzeptionelles Modellierungsgerüst bereit, das durch den Modellentwickler an die Gegebenheiten der zu bearbeitenden Fallstudie angepaßt werden muß. Dies kann durch Modifikationen der Modellstruktur erfolgen, beispielsweise indem weitere Modellelemente hinzugefügt werden, die durch das Referenzmodell nicht berücksichtigt wurden, oder auch dadurch, daß bestehende Modellelemente weggelassen werden, die für die gegebene Fallstudie keine Bedeutung haben. Neben den strukturellen Anpassungen müssen die einzelnen Modellelemente zudem anwendungsspezifisch mit Inhalt gefüllt werden. Dies umfaßt die Bereitstellung und Integration relevanter Eigenschaften, adäquater Verhaltensbeschreibungen, sowie geeigneter Modelldaten.

Der Einsatz von Referenzmodellen im Rahmen der Modellbildung und Simulation führt zu einer Reihe von Vorteilen, die im folgenden kurz erläutert werden sollen:

Referenzmodelle beschleunigen den Modellbildungsprozeß, da sie dem Modellentwickler Richtlinien in bezug auf die Strukturierung von Modellen, sowie die Beschreibung der Modelldynamik an die Hand geben.

Referenzmodelle können als Standardlösungen angesehen werden, die Expertenwissen aus der jeweiligen Problemdomäne enthalten. Werden diese Standardlösungen zur Kenntnis genommen und entsprechend eingesetzt, kann verhindert werden, daß für jede neue Fallstudie mit großer Mühe immer wieder individuelle Bastellösungen entworfen werden (Schmidt, 1996). Hier tritt der Aspekt der Wiederverwendung in den Vordergrund, der nicht nur im Bereich der Softwaretechnik eine große Rolle spielt, sondern mittlerweile auch im Bereich der Modellbildung und Simulation zunehmend an Bedeutung gewinnt.

Insgesamt gesehen tragen Referenzmodelle also dazu bei, den Aufwand, der mit der Entwicklung abstrakter Modelle verbunden ist, sowie die Komplexität der Entwurfsaufgabe deutlich zu reduzieren. Mit dieser Aufwandsreduzierung gehen nicht zuletzt auch Zeit- und Kosteneinsparungen einher, die sich positiv auf die Durchführung von Simulationsprojekten auswirken.

3.4 Beschreibungsmethoden für Referenzmodelle

Die Entwicklung von Referenzmodellen ist als Arbeitsgebiet im Bereich der Modellbildung und Simulation erst seit wenigen Jahren etabliert. Da sich dieses Arbeitsgebiet also noch in einem relativ frühen Entwicklungsstadium befindet, spie-

len Standardisierungsbemühungen bislang eine eher untergeordnete Rolle. Dies hat zur Folge, daß derzeit für die Beschreibung von Referenzmodellen noch keine einheitlichen Vorgehensweisen oder Standards existieren. Es besteht also die Notwendigkeit, in dieser Hinsicht auf die Beschreibungsmethoden anderer Fachgebiete zurückzugreifen und diese gegebenenfalls an die eigenen Anforderungen anzupassen.

Wählt man für ein Referenzmodell eine komponenten- bzw. objektorientierte Sichtweise, so eröffnet sich die Möglichkeit, für die Beschreibung von strukturellen Eigenschaften sowie dynamischen Abläufen Notationen einzusetzen, die bereits auf dem Gebiet des Softwareengineering etabliert sind. Besonders hervorzuheben ist in diesem Zusammenhang die *Unified Modeling Language* (UML) (Burkhardt, 1997; Quatrani, 1998), die sich in den letzten Jahren als Standard für die Beschreibung objektorientierter Systementwürfe durchgesetzt hat. UML stellt eine Menge von Diagrammtypen zur Verfügung, die es ermöglichen, unterschiedliche Gesichtspunkte objektorientierter Entwürfe in geeigneter Weise graphisch darzustellen.

Es ist nicht Ziel dieser Arbeit, die Einsatzmöglichkeiten von UML oder auch anderer Notationen für die Beschreibung von Referenzmodellen erschöpfend zu diskutieren. Im Zuge der Beschreibung des Referenzmodells *PECS* wird jedoch auf einige bestehende Beschreibungsmechanismen zurückgegriffen. Aus diesem Grund soll an dieser Stelle für eine Auswahl an Diagrammtypen und anderen Notationen zumindest angedeutet werden, in welcher Weise sie für die Beschreibung von Referenzmodellen dienlich sein können. Für eine detaillierte Einführung der genannten Notationen sei auf Balzert (2000) verwiesen.

Klassendiagramme beschreiben die statische Struktur objektorientierter Systeme. Sie lassen sich im Rahmen der Beschreibung von Referenzmodellen einsetzen, um deren Klassenstruktur, sowie Attribute und statische Aspekte des Verhaltens einzelner Klassen zu dokumentieren.

Interaktionsdiagramme veranschaulichen Beziehungen zwischen Objekten in speziellen Szenarien. In Form von Sequenzdiagrammen oder Kollaborationsdiagrammen eignen sie sich, um Abläufe, an denen mehrere Elemente von Referenzmodellen beteiligt sind, in übersichtlicher Weise darzustellen. Bei Sequenzdiagrammen steht die zeitliche Abfolge der Objektinteraktionen im Vordergrund, während der Schwerpunkt bei Kollaborationsdiagrammen eher auf strukturellen Aspekten liegt.

Datenflußdiagramme ermöglichen die Darstellung kontinuierlicher Signale, die als Flüsse zwischen unterschiedlichen Modellkomponenten auftreten. Daneben können in einem Modell aber auch diskrete Flüsse existieren. Diskrete Flüsse werden eingesetzt um beispielsweise Informationsflüsse oder Ereignisse, die das Modell steuern oder Aktivierungen auslösen, zu modellieren. Um beide Arten von Flüssen mit Hilfe eines einzigen Diagrammtyps darzustellen, kann man auf sogenannte Flußdiagramme zurückgreifen

Die bisher angegebenen Beschreibungsformen eignen sich im wesentlichen, um die strukturellen Aspekte eines Modellentwurfes zu dokumentieren. Darüber

hinaus existieren jedoch auch weitere Beschreibungsmöglichkeiten, die es erlauben, das dynamische Verhalten von Modellen in geeigneter Weise darzustellen.

Zustandsdiagramme ermöglichen beispielsweise die Analyse von Zuständen und diskreten Zustandsübergängen, die innerhalb von Modellkomponenten auftreten können.

Häufig werden für die Beschreibung von Referenzmodellen algorithmische Elemente oder Kontrollstrukturen benötigt. In diesem Zusammenhang können Struktogramme oder Programmablaufpläne als graphische Hilfsmittel eingesetzt werden. Als textuelle Beschreibungsform bietet sich hier aber auch eine Pseudocode-Darstellung an, die auf einer semiformalen Ebene angesiedelt ist.

Im Bereich der agentenbasierten Modellbildung kommt der Beschreibung von Regeln eine besondere Bedeutung zu. Regeln geben an, welche Aktionen bei der Erfüllung oder Nichterfüllung bestimmter Bedingungen ausgeführt werden sollen und werden häufig für die Beschreibung des Verhaltens von Agenten eingesetzt. Da Agenten häufig mit einer großen Zahl derartiger Regeln ausgestattet werden, muß bei deren Dokumentation großer Wert auf eine übersichtliche und klare Darstellung gelegt werden. Dieser Anforderung kann entsprochen werden, indem beispielsweise Entscheidungstabellen oder Entscheidungsbäume eingesetzt werden, die eine strukturierte Dokumentation von Wenn-Dann-Sätzen ermöglichen. Als Alternative dazu können derartige Regeln auch textuell mit Hilfe einer Pseudocode-Darstellung beschrieben werden, die sich einer Wenn-Dann-Syntax bedient. Bei sehr großen Regelmengen erscheint darüber hinaus eine Aufteilung in unterschiedliche Module sinnvoll, die getrennt und unabhängig voneinander beschrieben werden können.

Die hier angegebene Übersicht erhebt keinerlei Anspruch auf Vollständigkeit. Sie soll lediglich einen Anhaltspunkt dafür liefern, wie Konzepte, die im Rahmen des objektorientierten Softwareentwurfes etabliert sind, auch für die Beschreibung von komponenten- bzw. objektorientierten Referenzmodellen eingesetzt werden können. Die Entwicklung eines allgemeinen und standardisierten Instrumentariums zur Beschreibung von Referenzmodellen ist aktueller Forschungsgegenstand.

3.5 Referenzmodelle und softwaretechnische Muster

Wenn Referenzmodelle in der oben angegebenen Weise als allgemeine Konstruktionsschemata für abstrakte Modelle, die einer bestimmten Problemklasse angehören, definiert werden, so lassen sich für dieses Konzept gewisse Parallelen zum Konzept der Muster im Bereich der Softwaretechnik ziehen.

Muster in der Softwaretechnik stellen standardisierte, generische Lösungen für bestimmte, wiederkehrende Entwurfsprobleme zur Verfügung. Sie dokumentieren und erklären grundlegende Entwurfsvorschläge und tragen auf diese Weise dazu bei, den Entwurfsprozeß von Softwaresystemen zu beschleunigen. Darüber hinaus vereinfachen Entwurfsmuster die Kommunikation über objektorientierte

Systementwürfe, indem sie ein grundlegendes Vokabular und Verständnis im Zusammenhang mit Entwurfsaufgaben einführen (Balzert, 2000).

Nach Gamma (Gamma, Helm, Johnson & Vlissides, 1994) wird ein Muster im wesentlichen durch vier grundlegende Elemente beschrieben. Einem Muster wird ein *Name* zugeordnet, der Aufschluß über die Bedeutung des Musters geben sollte. Es enthält eine *Problembeschreibung*, die angibt, in welchen Situationen bzw. bei welchen Entwurfsproblemen das Muster zum Einsatz kommen kann. Der Kern eines Musters besteht in der Beschreibung der *Lösung für die angegebene Entwurfsaufgabe*. Diese Lösung spezifiziert einen konkreten objektorientierten Entwurf, individuelle Verantwortlichkeiten der einzelnen Elemente, die zwischen den Elementen auftretenden Beziehungen, sowie deren Interaktion. Abschließend werden sogenannte *Trade-Offs* bzw. Randbedingungen erläutert, die sich aus der Anwendung eines Entwurfsmusters ergeben. Hierzu zählen beispielsweise Angaben über die Effizienz hinsichtlich des Speicherbedarfes oder der Laufzeit, spezifische Eigenschaften der für die Implementierung des Muster zu verwendenden Sprache, sowie Aussagen über die Flexibilität bzw. Erweiterbarkeit des Musters.

Softwaretechnische Muster dienen also ebenso wie Referenzmodelle der Vereinfachung und Standardisierung von Entwurfsaufgaben und werden auch in ähnlicher Weise beschrieben. Eine weitere Analogie zwischen diesen beiden Konzepten besteht darin, daß Muster, ebenso wie Referenzmodelle, auf unterschiedlichen Abstraktionsebenen angesiedelt sein können. *Architekturmuster* beschreiben nach Buschmann (1996) ein strukturelles Organisationsschema für Softwaresysteme. Sie stellen eine Menge vordefinierter Subsysteme zur Verfügung, spezifizieren deren Zuständigkeiten und schließen auch Regeln und Richtlinien für die Organisation der Interaktion der Systemkomponenten mit ein. Architekturmuster liefern also Vorlagen für konkrete Softwarearchitekturen. *Entwurfsmuster* sind dagegen gemäß der Definition von Gamma (Gamma, Helm, Johnson & Vlissides, 1994) Beschreibungen interagierender Objekte und Klassen, die so konfiguriert sind, daß sie ein allgemeines Entwurfsproblem in einem speziellen softwaretechnischen Kontext lösen. Entwurfsmuster sind also auf einer niedrigeren Abstraktionsebene angesiedelt als Architekturmuster. An manchen Stellen wird auch noch eine dritte Kategorie von Mustern, sogenannte *Idiome*, eingeführt. Idiome sind Muster auf der niedrigsten Abstraktionsebene und berücksichtigen bereits spezifische Eigenschaften bestimmter Programmiersprachen. Idiome beschreiben, wie spezielle Aspekte innerhalb von Systemkomponenten unter Verwendung der gegebenen Eigenschaften einer bestimmten Programmiersprache realisiert werden können.

Referenzmodelle in der Simulation haben also sehr viele Gemeinsamkeiten mit Mustern in der Softwaretechnik. Ein grundlegender Unterschied zwischen diesen beiden Ansätzen existiert jedoch in der folgenden Hinsicht:

Die Qualität von Mustern wird daran gemessen, wie gut die bereitgestellten Konzepte die zugrunde liegenden Entwurfsprobleme lösen. Ihre Aufgabe besteht im wesentlichen darin, für gegebene technische Entwurfsprobleme möglichst einfache, wohlstrukturierte und elegante Lösungen zur Verfügung zu stellen. Bei Referenzmodellen treffen diese Kriterien in derselben Weise zu. Da Referenzmodel-

le jedoch grundlegende Entwurfsvorschläge für abstrakte Modelle realer Systeme liefern sollen, gilt hier zusätzlich die Forderung nach Strukturähnlichkeit. Referenzmodelle sollen also in einer derartigen Weise entworfen werden, daß die Grundstrukturen, durch die die betrachteten Systeme ausgezeichnet sind, auch in den zugehörigen Modellierungsansätzen adäquat repräsentiert werden. Diese zusätzlich gestellte Anforderung ermöglicht ein intuitives Verständnis von Referenzmodellen, schränkt aber gleichzeitig den Gestaltungsfreiraum, der für Referenzmodelle zur Verfügung steht, im Vergleich zu Mustern in der Softwaretechnik enorm ein.

4 GRUNDLEGENDE ANFORDERUNGEN AN DAS REFERENZMODELL

Das Hauptziel dieser Arbeit besteht darin, ein domänenunabhängiges Referenzmodell zur Verfügung zu stellen, das den Entwurfsprozess agentenbasierter Modelle, in denen menschliches Handeln, Entscheiden und Verhalten von ausschlaggebender Bedeutung sind, unterstützt.

Referenzmodelle, die auf einer höheren Abstraktionsebene angesiedelt sind, beschreiben Konstruktionsprinzipien für den Entwurf von Modellen. Es werden ganz allgemein vielfältige Anforderungen sowohl an die Beschreibung als auch an den Entwurf von Referenzmodellen gestellt. Die wichtigsten Anforderungen, die im wesentlichen bereits im vorangegangenen Kapitel zusammengetragen wurden, sollen an dieser Stelle noch einmal ins Gedächtnis gerufen und im Zusammenhang mit der Bereitstellung eines Referenzmodells für die Modellierung menschlichen Handelns, Entscheidens und Verhaltens diskutiert werden.

Im Sinne einer funktionalen Dekomposition des Gegenstandsbereiches sollen im weiteren Verlauf dieses Kapitels elementare Einflüsse identifiziert und charakterisiert werden, die den Entwurf eines derartigen Referenzmodells leiten. Die Grundlage dieser Anforderungsanalyse bilden zum einen disziplinübergreifende Erkenntnisse über menschliches Handeln, Entscheiden und Verhalten, sowie bestehende Modellierungsansätze aus dem Grenzgebiet zwischen Künstlicher Intelligenz und Sozialwissenschaften.

Der Entwurf agentenbasierter Modelle erfordert im allgemeinen die Auseinandersetzung mit drei übergeordneten Aufgabenstellungen:

Die wichtigste und gleichzeitig komplexeste Aufgabe betrifft den Entwurf und die Konzeption der Agenten. Werden Agenten eingesetzt, um menschliche Akteure im Modell abzubilden, besteht die Notwendigkeit, verschiedene Funktionalitäten in einer Gesamtarchitektur ein Einklang zu bringen. Es spielen mehrere nebenläufige, zum Teil voneinander unabhängige, zum Teil voneinander abhängige, zum Teil sich begünstigende, zum Teil konkurrierende Prozesse eine Rolle, die verschiedene Aufgaben erfüllen. Ein Architekturansatz, der eine integrative Betrachtung unterschiedlicher Eigenschaften und Prozesse ermöglichen soll, muß in diesem Sinne sehr allgemein ausgerichtet und auf einer hohen Abstraktionsebene angesiedelt sein.

Betrachtet man Szenarien, in denen nicht nur ein Agent, sondern mehrere Agenten oder größere Agentengesellschaften das Geschehen bestimmen, so stellt sich heraus, daß Kommunikation als eine wesentliche Grundlage für die Interaktion von Agenten anzusehen ist. Kommunikation dient in erster Linie dazu, Informationen zwischen Agenten auszutauschen. Darüber hinaus ermöglicht Kommunikation jedoch auch die Koordination und Kooperation von Agenten. Kommunikationsmechanismen müssen daher als essentieller Bestandteil beim Entwurf agentenbasierter Modelle berücksichtigt werden.

Nicht zuletzt spielt in agentenbasierten Modellen ein externes Bezugssystem eine entscheidende Rolle. Dieses Bezugssystem beschreibt die Umwelt oder Umgebung, in die Agenten eingebettet sind. Zwischen den Agenten und ihrer Umgebung besteht in der Regel eine wechselseitige Beziehung. Zum einen beeinflussen die Agenten den Zustand ihrer Umwelt durch Ausführung von Aktionen. Andererseits gibt die Umgebung aber auch Rahmen- und Randbedingungen vor, die das Handeln und Verhalten der Agenten in entscheidender Weise beeinflussen können.

Die folgenden Abschnitte werden sich nun etwas ausführlicher mit diesen Problemstellungen und Anforderungen beim Entwurf eines agentenbasierten Referenzmodells für menschliches Handeln, Entscheiden und Verhalten auseinandersetzen.

4.1 Strukturelle Anforderungen an das Referenzmodell

Wie in Kapitel 3 bereits erläutert wurde, muß die Beschreibung von Referenzmodellen auf systematische Weise erfolgen. Ebenso sollen Referenzmodelle klar strukturiert sein und methodologische Aspekte des jeweiligen Einsatzgebietes adäquat widerspiegeln.

Diese Anforderungen in bezug auf den Entwurf des Referenzmodells sollen auch im Rahmen dieser Arbeit berücksichtigt werden. Eine klare und übersichtliche Strukturierung trägt nicht nur maßgeblich zur Verständlichkeit von Referenzmodellen bei. Sie ist insbesondere auch eine Grundvoraussetzung für die Anpaßbarkeit des Referenzmodells an Anforderungen individueller Anwendungsszenarien.

Um eine klare Strukturierung zu erreichen, soll für den Entwurf des Referenzmodells ein komponentenorientierter Ansatz gewählt werden. Das Referenzmodell soll also in eine Menge von Komponenten mit individuellen Zuständigkeiten und Funktionalitäten zerfallen, die miteinander über definierte Mechanismen interagieren können.

Aus Sicht der Sozialwissenschaften ließe sich an dieser Stelle besonders im Hinblick auf die agentenbasierte Modellierung menschlichen Handelns, Entscheidens und Verhaltens entgegnen, daß die vielschichtigen Interaktionen psychischer Funktionen und Prozesse beim Menschen eine klare Zerlegung einer Agentenar-

chitektur in miteinander wechselwirkende Module aussichtslos erscheinen lassen. Aus Sicht des Modellentwicklers bietet es sich jedoch an, agentenbasierte Modelle im Sinne einer funktionalen Dekomposition in einzelne Komponenten zu zerlegen, um dadurch die Komplexität, die mit einer derartigen Entwurfsaufgabe verbunden ist, zu reduzieren. Die Kunst des Modellentwurfes besteht an dieser Stelle im wesentlichen darin, die Grenzen der einzelnen Modellkomponenten sowie deren Interaktion in einer derartigen Weise einzurichten, daß die komplexen Prozesse, die in der Realität auftreten, im Modell in angemessener Weise beschrieben werden können.

4.2 Anforderungen an die Modellierung der Agenten

Ein Referenzmodell, das die Modellierung menschlichen Handelns, Entscheidens und Verhaltens unterstützen soll, muß eine Agentenarchitektur zur Verfügung stellen, die es ermöglicht, menschliche Akteure in adäquater Weise in Modellen abzubilden.

Für die Entwicklung individueller Modelle geht man in der Regel von spezifischen Anforderungen der jeweiligen Anwendungsdomäne aus. Im Rahmen dieser Arbeit soll jedoch ein domänenunabhängiges Referenzmodell entwickelt werden. Aus diesem Grund genügt es an dieser Stelle nicht, die Anforderungsanalyse auf eine einzelne Anwendungsdomäne zu beschränken. Vielmehr muß für eine Zusammenstellung von Anforderungen im Kontext dieser Arbeit wesentlich allgemeiner angesetzt werden. Es müssen Grundmechanismen und elementare Einflüsse auf das menschliche Handeln, Verhalten und Entscheiden identifiziert und für den Entwurf der Agentenarchitektur berücksichtigt werden.

Das Wissen, das der Mensch über die Menge an internen Prozessen, die in ihm stattfinden und die sein Verhalten determinieren, bis zum gegenwärtigen Zeitpunkt erworben hat, ist noch sehr begrenzt. In verschiedenen Bereichen der Wissenschaft existieren zwar bereits sehr aussagekräftige und leistungsfähige Modelle für die Funktionsweise ausgewählter menschlicher Fähigkeiten. Umfassende und integrative Betrachtungen des menschlichen Verhaltens, wie sie für die Konzeption einer derartigen Agentenarchitektur erforderlich sind, existieren bislang jedoch nicht.

Um die domänenunabhängige Anwendbarkeit des Referenzmodells zu erreichen, soll für die Konzeption des Referenzmodells ein sogenannter *Broad and Shallow*-Ansatz (Bates, Loyall & Reilly, 1991) gewählt werden. Dabei soll eine funktionale Agentenarchitektur entwickelt werden, die eine globale Organisation einer Menge interagierender Fähigkeiten vorgibt, wobei jede einzelne Fähigkeit (zumindest im Rahmen dieser Arbeit) noch auf einer relativ hohen Abstraktionsebene beschrieben wird. Der Schwerpunkt soll also auf der Integration unterschiedlicher Funktionalitäten liegen und weniger auf einer detaillierten Betrachtung einzelner, spezieller Fähigkeiten. Aus diesem Grund wird sich die anschlie-

ßende Systemanalyse aus Sicht der einzelnen, gestreiften Fachgebiete auf einer relativ hohen Abstraktionsebene bewegen und versuchen, nach dem derzeitigen Stand der Forschung relevante Einsichten über die Funktionsweise des menschlichen Handelns, Verhaltens und Entscheidens zusammenzutragen.

Die folgenden Abschnitte werden sich nun mit relevanten Eigenschaften, Kräften und Funktionen beschäftigen, die den Menschen in seinem Verhalten beeinflussen, sowie unterschiedliche Ebenen der Verhaltenssteuerung voneinander unterscheiden. Aus dieser Betrachtung werden zugleich Anforderungen abgeleitet, die an eine Agentenarchitektur gestellt werden, wenn sie für eine domänenunabhängige Modellierung menschlichen Handelns, Entscheidens und Verhaltens eingesetzt werden soll. Die folgenden Abschnitte beruhen im wesentlichen auf Erkenntnissen und Annahmen aus den Bereichen Physiologie, Psychologie, Kognitionswissenschaften, Sozialpsychologie, Soziologie, sowie Künstliche Intelligenz.

4.2.1 Informationsaufnahme und Wahrnehmung

Der Mensch verfügt über sensorische Fähigkeiten, die ihm die Möglichkeit geben, Informationen aus seiner Umgebung zu registrieren. Diese Informationen stehen in Form von physikalischen Reizen zur Verfügung und werden im Rahmen des Wahrnehmungsprozesses zunächst in neuronal kodierte Informationen überführt. Im nächsten Schritt, der sogenannten perzeptuellen Organisation, wird eine interne Repräsentation der aufgenommenen Informationen erstellt. Ein derartiges Perzept beinhaltet eine Beschreibung der äußeren Reizsituation. Hier findet bereits eine Integration des äußeren Reizes mit bestehendem Wissen des Menschen statt. So werden beispielsweise einzelne akustische oder visuelle Informationen zu wiedererkannten Objekten integriert. Diese Vorgänge erfolgen weitgehend automatisch und unbewußt. Im letzten Schritt werden den erstellten Perzepten schließlich Bedeutungen zugewiesen. Die wahrgenommenen Objekte werden identifiziert und ihrer Bedeutung entsprechend eingeordnet. Dabei spielen höhere kognitive Prozesse wie beispielsweise das Erinnern von Informationen, aber auch Wertvorstellungen oder Einstellungen des Menschen gegenüber dem wahrgenommenen Objekt eine Rolle (Zimbardo & Gerrig, 1996).

Im Rahmen der agentenbasierten Modellbildung wird in der Regel auf die Abbildung physikalischer Reize verzichtet. Externe Informationen, die für Agenten von Bedeutung sind, liegen im Normalfall bereits in symbolisch kodierter Form vor. Dieses Vorgehen befreit von der Notwendigkeit, aufwendige Prozesse wie beispielsweise Sprach- oder Mustererkennung, die in den meisten Fällen für die zentrale Fragestellung von Modellen ohne Bedeutung sind, in die Modelle aufzunehmen.

Informationen können im wesentlichen in Umweltinformationen und Nachrichten, die von anderen Agenten abgesetzt wurden, unterschieden werden. Umweltinformationen geben Aufschluß über aktuelle Zustände in der Umgebung des Agenten und dienen dem Agenten beispielsweise zur Orientierung oder zur Kon-

trolle des Fortschritts und Erfolgs von ausgeführten Aktionen. Nachrichten werden genutzt, um Kommunikation zwischen Agenten zu ermöglichen und Informationen unterschiedlicher Art auszutauschen.

Die Hauptaufgabe, die der Wahrnehmung im Zusammenhang mit der agentenbasierten Simulation zukommt, liegt am ehesten in der Modellierung grundlegender Prozesse der Informationsfilterung. Aus der Menge der für einen Agenten zu einem bestimmten Zeitpunkt verfügbaren Informationen werden diejenigen Informationen ausgewählt und für eine weitere Verarbeitung im Rahmen kognitiver Prozesse zur Verfügung gestellt, die in Anbetracht seines aktuellen internen Zustands als relevant erachtet werden.

4.2.2 Aktionen

Menschen verfügen nicht nur über die Fähigkeit, Informationen über ihre Umgebung und andere Menschen aufzunehmen. Sie sind auch in der Lage, durch Ausführung von Aktionen sowohl ihren eigenen Zustand als auch ihre Umwelt zu modifizieren.

Aktionen, die der Mensch ausführen kann, lassen sich ganz grob in zwei unterschiedliche Kategorien einordnen. Zum einen existieren Aktionen, mit denen der Mensch den Zustand seiner Umgebung verändern kann. Diese Aktionen sollen als *externe Aktionen* bezeichnet werden. Eine derartige Veränderung kann beispielsweise darin bestehen, daß der aktuelle Zustand eines Objektes in der Umgebung modifiziert wird oder daß Relationen zwischen mehreren Objekten in der Umgebung verändert werden. Ebenfalls als externe Aktionen kann man Aktionen im Zusammenhang mit dem Informationsaustausch mit anderen Menschen einstufen. Hier werden Informationen in unterschiedlichen Erscheinungsformen ausgetauscht, um mit anderen Menschen in Kontakt zu treten und zu kommunizieren.

Die zweite Klasse von Aktionen könnte man mit dem Begriff *interne Aktionen* umschreiben. Zu dieser Klasse sollen derartige Aktionen gehören, die den internen Zustand des Menschen verändern. Hier sind zum einen Aktionen zu nennen, die Denk- oder Planungsprozesse anstoßen und damit den kognitiven Zustand des Menschen verändern. In diesem Zusammenhang spricht man häufig auch von internem Probehandeln, da die Konsequenzen von Aktionen nicht in der Realität erkundet werden, sondern durch Modifikation interner, mentaler Repräsentationen erschlossen werden sollen. Auch die Ausrichtung der Sinne zum Zwecke der Orientierung in der Umwelt kann der Klasse der internen Aktionen zugerechnet werden.

Eine Agentenarchitektur, die eine flexible Modellierung menschlichen Handelns, Verhaltens und Entscheidens ermöglichen soll, muß also zum einen Mechanismen zur Verfügung stellen, mit denen ein Agent Einfluß auf den Zustand seiner Umgebung nehmen kann. Zum anderen muß die Möglichkeit vorgesehen werden, Prozesse in Gang zu setzen, die den internen Zustand des Agenten modifizieren.

4.2.3 Grundmechanismen der Verhaltenssteuerung

Der Begriff *Verhalten* findet in der Umgangssprache sehr allgemeine Verwendung und subsumiert im wesentlichen alle inneren und auch äußeren Aktivitäten eines Organismus (Jäger, 1996). Die Sozialwissenschaften und auch die Künstliche Intelligenz gehen allerdings von einer detaillierteren Betrachtungsweise aus und unterscheiden in bezug auf die menschliche Verhaltenssteuerung zwischen unterschiedlich komplexen Grundmechanismen, die in den nachfolgenden Abschnitten etwas genauer betrachtet werden sollen.

4.2.3.1 Automatisches bzw. reaktives Verhalten

Die einfachste Form der menschlichen Handlungsregulation ist das automatische bzw. reaktive Verhalten. Reaktives Verhalten folgt instinktiv vorprogrammierten oder erlernten Regeln, die im Gedächtnis hinterlegt sind. Es läuft in der Regel automatisch und unbewußt ab. Dies bedeutet, daß Ziele, die mit spezifischen Verhaltensweisen erreicht werden sollen, dem Akteur im Falle des automatischen Verhaltens nicht unbedingt bewußt sein müssen.

Reaktive Verhaltensweisen werden automatisch abgerufen und gegebenenfalls situationsspezifisch modifiziert. Handlungstheorien gehen in der Regel davon aus, daß reaktives Verhalten Reiz-Zustands-Reaktions-Mustern folgt. In einer gegebenen Situation, zu der sowohl ein externer Reiz, als auch ein interner, organismischer Zustand gehört, führt ein Akteur eine entsprechende, für die Situation spezifische Reaktion bzw. Aktion aus. Darüber hinaus verbindet der Akteur, wenn auch in den meisten Fällen unbewußt, mit der Ausführung einer Aktion eine gewisse Erfolgserwartung, d. h. es entsteht eine Vorstellungen von den Konsequenzen, die sich durch Ausführung einer bestimmten Aktion einstellen sollen. Bewußt wird diese Erfolgserwartung in den meisten Fällen erst dann, wenn die tatsächliche Konsequenz einer Verhaltensweise von der erwarteten Konsequenz abweicht und der Automatismus dadurch unterbrochen wird (Dörner, 1996).

Auch diese verhältnismäßig einfache Form der menschlichen Handlungsregulation erfordert bereits recht komplexe Gedächtnisstrukturen. So bezeichnet Tolman (1948) beispielsweise eine Repräsentation der Umgebung sowie der darin ausführbaren Handlungsmöglichkeiten als *cognitive map*. Diese Sichtweise vernachlässigt allerdings die Bedeutung des internen Zustands eines Akteurs für dessen Verhaltensauswahl. Dörner (1996) beschreibt automatische Verhaltensweisen mit Hilfe sogenannter *Aktionsschemata*.

Ein Aktionsschema besteht aus einem Inputschema, einem Outputschema, sowie einem Erwartungsschema. Das Inputschema stellt eine Art Hohlform zur Verfügung, in der Informationen über Input-Bedingungen einer Aktion abgelegt werden. Dazu gehören im wesentlichen Informationen über Merkmale von Situationen, in denen die Aktion angewendet werden kann. Das Outputschema gibt an, welche Operationen in der gegebenen Situation ausgeführt werden müssen, damit

das gewünschte Resultat erzielt werden kann. Die Erwartungen in bezug auf die Konsequenzen der ausgeführten Aktionen werden im Erwartungsschema zusammengefasst.

Ein einzelnes Aktionsschema kann als das kleinste mögliche Handlungsprogramm betrachtet werden. Durch Hintereinanderschaltung mehrerer Aktionsschemata können bereits sehr komplizierte, automatische Verhaltensweisen erreicht werden. Es werden solche Aktionsschemata verkettet, deren Erwartungsschemata und Inputschemata übereinstimmen. Die Kette wird so lange verlängert, bis das Erwartungsschema des letzten Aktionsschemas der angestrebten Situation entspricht. Wichtig ist dabei der Sachverhalt, daß derartige Prozesse automatisch und unbewußt bzw. allenfalls „mitbewußt" ablaufen (Dörner, 1996).

Automatisches Verhalten ist so lange erfolgreich, wie für die Situationen, in die man im Laufe der Zeit gerät, geeignete Aktionsschemata bzw. Aktionssequenzen existieren. Rein reaktives Verhalten stößt jedoch an Grenzen, wenn ein Akteur Situationen begegnet, für die keine festen Verhaltensprogramme existieren. Dies kann der Fall sein, wenn ganz neue Situationen eintreten, denen der Akteur noch nicht gegenüberstand, oder auch dann, wenn sich Erwartungen, die mit der Ausführung von Aktionen verbunden sind, in der Realität nicht erfüllen. Sind für derartige Situationen keine automatischen Verhaltensweisen vorgesehen, so ist der Akteur in dieser Situation handlungsunfähig. Der Mensch ist in derartigen Situationen in der Lage, seine Handlungen bewußt zu organisieren.

4.2.3.2 Handeln, deliberatives Verhalten und Entscheiden

Handlungstheorien, die verschiedenen Wissenschaftsdisziplinen wie der Philosophie, der Anthropologie, der Psychologie oder der Soziologie entstammen, beschäftigen sich aus verschiedenen Perspektiven mit der Frage, wie der Mensch seine Handlungen steuert. Allen Ansätzen ist trotz vielfältiger Unterschiede die Grundannahme gemein, daß der Mensch im Gegensatz zum reaktiven Verhalten, das weitgehend automatisch und unbewußt abläuft, auch über die Fähigkeit verfügt, Handlungen bewußt zu lenken. Menschliches Handeln unterscheidet sich von einfachem Verhalten im wesentlichen dadurch, daß es zielgerichtet und bewußt ist (Zapf, 1996). Unter Bewußtheit versteht man dabei, daß der Handelnde bewußt ein Ziel zu erreichen versucht, das in seinem Arbeitsgedächtnis abgelegt ist und über das er auch Auskunft geben könnte. Ein Ziel beschreibt nach Opwis (1996) einen anzustrebenden Zustand und kann sowohl eine Veränderung der Umwelt als auch eine Veränderung des internen Zustands eines Individuums betreffen (Häcker & Stapf, 1998).

In Zusammenhang mit Handlungszielen stellen sich zwei wesentliche Fragen. Zunächst muß geklärt werden, in welcher Weise Handlungsziele überhaupt entstehen. Hier spielen motivationale Prozesse eine wesentliche Rolle. Zudem gilt es zu beschreiben, welche Mechanismen zur Verfügung stehen, um die Handlungsziele, die im ersten Schritt festgelegt wurden, später auch zu erreichen.

Einflüsse auf die Genese von Handlungszielen

Die Entwicklung von Erklärungsansätzen für die Entstehung und Auswahl von Handlungszielen ist aktueller Forschungsgegenstand in unterschiedlichen Wissenschaftsdisziplinen. An dieser Stelle ist es daher nicht möglich, eine allgemeingültige Theorie anzuführen. Vielmehr können nur Einflußfaktoren zusammengetragen werden, die nach dem gegenwärtigen Stand der Forschung für die Genese von Handlungszielen als relevant erachtet werden.

Traditionelle Erwartung-Wert-Modelle gehen von der Annahme aus, daß ein Mensch bei der Auswahl seiner Handlungsziele rationale Entscheidungen trifft. Aus einer Menge existierender Handlungsziele wählt ein Mensch konkrete Handlungsziele gemäß ihres subjektiven Wertes und ihrer Realisierungswahrscheinlichkeit aus (Brandstätter & Gollwitzer, 1996). Je höher der subjektive Wert eines Zieles sowie dessen Realisierungswahrscheinlichkeit von einem Individuum eingeschätzt wird, um so eher wird das Ziel ausgewählt. In den Entscheidungsprozeß gehen also sowohl interne, den Entscheider betreffende Einflüsse, als auch externe, situationsbezogene Einflüsse ein. Der Wert einen Zieles wird oftmals auf Grundlage der mit der Zielverwirklichung verbundenen Motivbefriedigung (Brandstätter & Gollwitzer, 1996) oder der Stärke des dem Ziel zugrunde liegenden Motivs (Dörner, 1999) gemessen.

Nach Madsen (1974) sind Motive psychische Kräfte, die ein Handeln in Gang setzen, auf ein bestimmtes Ziel hin ausrichten und (im Erfolgsfall) bis zur Zielerreichung aufrecht erhalten. Motive werden als notwendige Voraussetzung für zielgerichtetes Handeln eingeschätzt und beeinflussen im wesentlichen die Auswahl von Handlungszielen.

Kognitivistische Ansätze machen höhere kognitive Prozesse für die Entstehung von Motiven verantwortlich. So kann beispielsweise die Antizipation zukünftiger Ereignisse oder Zustände motivierend wirken. In diesem Fall werden Handlungen derart ausgerichtet, daß die Diskrepanz zwischen der gewünschten Situation in der Zukunft und der aktuell vorliegenden Situation verringert wird (Festinger, 1957).

Nach Bergius (1998) können Motive ebenso sowohl von Umweltreizen (Stimuli) als auch von inneren organismischen Zuständen abhängig sein. Zu derartigen Zuständen gehören beispielsweise physiologische Bedürfnisse, die durch Störungen der Homöostase, d. h. des Gleichgewichtszustandes des inneren Milieus im Körper (Pschyrembel, 1985), entstehen.

Einen etwas weiter gefaßten Bedürfnis-Begriff führt Maslow (1955) ein. Maslow schreibt dem Menschen angeborene Bedürfnisse zu, die er in hierarchischer Weise strukturiert. Diesem Ansatz liegt die Annahme zugrunde, daß Bedürfnisse auf niedrigeren Hierarchiestufen befriedigt sein müssen, damit Bedürfnisse auf höher angesiedelten Hierarchiestufen zum Zuge kommen können. Grundsätzlich unterscheidet Maslow zwischen fundamentalen physiologischen Bedürfnissen, Sicherheitsbedürfnissen, sozialen Bedürfnissen, Selbstwert- und kognitiven Bedürfnissen, sowie einem Bedürfnis nach Selbstverwirklichung und Transzendenz. In vielen Fällen sind derartige Bedürfnisse an innere Zustände gekoppelt, die den

Menschen in bezug auf seine physiologische Konstitution, kognitive Aspekte und auch seine Einbindung in das soziale Umfeld kennzeichnen.

Ansätze zur Realisierung von Handlungszielen

Nachdem im vorhergehenden Abschnitt geschildert wurde, welche Einflüsse bei der Auswahl von Handlungszielen eine Rolle spielen, soll in diesem Abschnitt nun betrachtet werden, welche Möglichkeiten dem Menschen im wesentlichen zur Verfügung stehen, um festgelegte Handlungsziele auch tatsächlich zu erreichen.

Häufig liegen für die Erreichung von Handlungszielen reaktive Verhaltensweisen vor, die aktiviert werden können und weitgehend selbständig ablaufen. Dies ist jedoch nicht immer der Fall. Es können ebenso Situationen eintreten, in denen keine zielführenden Automatismen existieren und daher adäquate Verhaltensweisen erst neu aufgebaut werden müssen.

Die elementarste Mechanismus, der hier zur Anwendung kommen kann, ist das Versuch-und-Irrtum-Prinzip (Dörner, 1996). Hier werden vorhandene Aktionsschemata willkürlich miteinander verkettet und sofort in der Realität erprobt. Im Vergleich zu rein reaktiven Mechanismen der Verhaltenssteuerung steigt zwar dadurch die Verhaltensvariabilität eines Individuums deutlich an. Ein Akteur wird in die Lage versetzt, auch mit neuartigen Situationen fertig zu werden, für die bislang noch keine passenden Aktionsketten vorrätig waren. Der Raum von Aktionskombinationen ist jedoch enorm groß. Sind beispielsweise aus einer Menge von n unterschiedlichen Aktionen k Aktionen auszuwählen, so gibt es dafür insgesamt n^k Möglichkeiten. Im ungünstigsten Fall müssen alle Kombinationen ausprobiert werden, um zum Ziel zu gelangen. Reines Versuch-und-Irrtum-Verhalten erscheint aus diesem Grund nicht sehr effizient. Zudem bringt es auch gewisse Gefahren für das Individuum und auch für die Umwelt mit sich, da oftmals die Konsequenzen von Aktionen nicht absehbar sind und ungewollte, irreversible Änderungen die Folge sein können.

Alle Aktionskombinationen in der Realität zu erproben, ist im allgemeinen Fall also nahezu unmöglich. Viel günstiger ist es, die Suche nach einer geeigneten Aktionskette hypothetisch, d. h. auf der Grundlage einer mentalen Repräsentation aller relevanten Sachverhalte, durchführen zu können. Vor der tatsächlichen Ausführung von Aktionen in der Realität kann das Verhalten dadurch auf der Grundlage eines internen „Weltmodells“ erprobt und ausgefeilt werden. Dörner (1996) spricht hier von einem „inneren Probehandeln“ und sieht diesen Aspekt als den Kern des menschlichen Denkens. Verhalten, das auf internen Denkprozessen basiert, wird im Bereich der Agententechnologie auch als *deliberatives Verhalten* (Genesereth & Nilsson, 1987) bezeichnet.

Deliberatives Verhalten bedeutet die Synthese neuer, bislang unbekannter Verhaltensweisen und kann nach Aebli (1981) in die folgenden Schritte unterteilt werden: zunächst erfolgt eine Analyse der gegebenen Situation. Es muß eine ausreichend genaue Vorstellung über die Umwelt und auch den eigenen Zustand

entwickelt werden, die dann als Grundlage für die weiteren Überlegungen dienen kann. Daraufhin werden verschiedene Handlungsalternativen ausgearbeitet und schließlich eine davon für die Durchführung ausgewählt.

Bei der Konstruktion verschiedener Handlungsalternativen müssen Pläne entwickelt werden, wie die gegebene Situation durch Ausführung geeigneter Aktionen in den gewünschten Zielzustand überführt werden kann. Hierfür gibt es verschiedenste, häufig sehr spezifische Vorgehensweisen. Ein allgemeines Prinzip bei der Entwicklung von Handlungsplänen ist jedoch, den Lösungsweg zunächst ganz grob vorzustrukturieren. Dies kann beispielsweise dadurch geschehen, daß Zwischenziele eingeführt werden, an denen einzelne Etappen des Vorgehens festgemacht werden. Bei der Ausführung eines Handlungsplanes versucht der Akteur der Reihe nach alle Zwischenziele zu erreichen, bis schließlich am Ende das übergeordnete Handlungsziel realisiert werden kann. Auf einzelne Verfahren zur Generierung von Handlungsplänen soll an dieser Stelle nicht genauer eingegangen werden, da sie für das Ziel dieser Arbeit nur von untergeordneter Bedeutung sind. Essentiell ist an dieser Stelle allerdings die Erkenntnis, daß ein Agent, der menschliches Handeln modellieren soll, in der Lage sein muß, planerische Fähigkeiten abzubilden.

Liegen nach dem Planungsprozeß mehrere Pläne vor, die potentiell zur Ausführung kommen können, muß eine Auswahlentscheidung für einen dieser Pläne getroffen werden. Die Modellierung derartiger Situationen erfolgt in der Literatur in der Regel mit Hilfe von entscheidungstheoretischen Ansätzen, die auf Nutzenüberlegungen basieren (Dörner, 1996).

Nachdem ein Plan zur Ausführung ausgewählt ist, kann der Akteur mit dessen Umsetzung beginnen. Es müssen also Aktionen ausgeführt werden, die Schritt für Schritt die gewünschten Zielzustände hervorbringen. Für die Erreichung der einzelnen Zwischenziele kann die Notwendigkeit bestehen, nicht nur einzelne Aktionen, sondern längere Aktionsketten auszuführen. Diese Aktionsketten folgen dabei häufig reaktiven Verhaltensmustern. Ebenso kann der Fall eintreten, daß aufgrund veränderter Bedingungen der Umwelt der Handlungsplan nicht erfolgreich beendet werden kann. In diesem Fall muß der bestehende Handlungsplan verworfen werden und eine Neuplanung stattfinden. Nicht zuletzt kann auch infolge von veränderten Situationen eine Neuformulierung der gesteckten Ziele erforderlich sein, obwohl noch Aktionen im Gange sind, die der Realisierung eines anderen Zieles dienen.

Neben den oben geschilderten Einflüssen spielen im Zusammenhang mit deliberativem Verhalten auch Lernprozesse eine wichtige Rolle. Eine wiederholte Ausführung deliberativer Verhaltensweisen kann zu deren Automatisierung führen. Werden also Aktionsketten, die aus deliberativen Verhaltensweisen resultieren, häufig erfolgreich ausgeführt, so können diese in abstrahierter Form als reaktive Verhaltensmuster gespeichert werden und später ohne einen vorgelagerten Planungsprozeß aktiviert werden. Diese Automatisierung deliberativer Verhaltensweisen ermöglicht eine ökonomische Funktionsweise des menschlichen Verhaltens, da nur noch dann Denkprozesse angestoßen werden müssen, wenn keine

reaktiven Automatismen vorhanden sind, die in der vorliegenden Situation in geeigneter Weise ausgeführt werden können. Es werden geistige Kapazitäten frei für die Erledigung anderer Aufgaben.

4.2.3.3 Reflektives Verhalten

Der Mensch ist in der Lage, deliberatives Verhalten nicht nur auszuführen, sondern die damit in Verbindung stehenden Prozesse auch zu beobachten und zu beeinflussen. Sloman (1996) spricht in diesem Zusammenhang von *reflektivem Verhalten*.

Im Rahmen des reflektiven Verhaltens beobachtet ein Individuum seine mentalen Prozesse und versucht diese im Hinblick auf langfristig gesteckte Ziele zu bewerten und gegebenenfalls zu modifizieren. Hierzu zählen beispielsweise die Entwicklung und Argumentation neuer Handlungsziele, der Vergleich und die Bewertung bestehender Handlungsziele, die Bewertung und Modifikation von Planungs- und Problemlöseprozessen, sowie die Selbsteinschätzung der eigenen Fähigkeiten.

Im Vergleich zum deliberativen Verhalten, das eher konstruktiven Charakter besitzt, kommen beim reflektiven Verhalten also weitere Mechanismen hinzu, die beim Entwurf einer Agentenarchitektur für die Modellierung menschlichen Verhaltens berücksichtigt werden müssen. Auf einer Art „Metaebene“ (Sloman, 1996) können Menschen ihre geistigen Fähigkeiten überwachen und erhalten dadurch die Möglichkeit, diese zu modifizieren. Um dies zu bewerkstelligen, muß neben der mentalen Repräsentation der Umgebung auch eine mentale Repräsentation interner Zustände und Fähigkeiten vorliegen, die als Ausgangsbasis für derartige Verhaltensweisen dienen kann. Daneben müssen Prozesse vorhanden sein, die diese Modifikationen in Gang setzen.

Derzeit existieren weder im Bereich der Psychologie noch im Bereich der Künstlichen Intelligenz überzeugende Theorien für die Erklärung und Beschreibung reflektiver Verhaltenssteuerungsmechanismen. Darüber hinaus spielt Reflexion (zumindest derzeit) keine maßgebliche Rolle für den Großteil von Anwendungen im Bereich der agentenbasierten Modellbildung. Aus diesem Grund werden reflektive Verhaltensweisen im Rahmen dieser Arbeit nicht im Detail weiterverfolgt.

4.2.4 Kognitive Strukturen und Funktionen

Die in den vorhergehenden Abschnitten beschriebenen Mechanismen der menschlichen Verhaltensregulation setzen, um korrekt funktionieren zu können, eine gewisse Menge und Art an unterschiedlichen Informationen voraus. Um derartige Informationen zu speichern und zu verarbeiten, verfügt der Mensch über kognitive Fähigkeiten. Diese Fähigkeiten haben im Gegensatz zu Motiven, die als An-

triebsgeneratoren dienen, eher funktionalen Charakter und unterstützen das Handeln und Verhalten des Menschen.

Aus Sicht der Kognitionswissenschaften verfügt der Mensch über eine Menge an Informationen über seine Umwelt und auch seinen eigenen Zustand. Diese Informationen sind in verschiedenen Gedächtnisinstanzen wie Ultrakurzzeitgedächtnis, Arbeitsgedächtnis und Langzeitgedächtnis (Anderson, 1996) gespeichert, die sich im wesentlichen durch ihre Aufnahmekapazität, Latenzzeit der gespeicherten Informationen und ihre grundsätzliche Funktion voneinander unterscheiden.

Neben dieser strukturellen Sicht auf die menschliche Kognition existiert auch eine funktionale Sicht. Hier wird untersucht, welche kognitiven Funktionen existieren, die das Gedächtnis beeinflussen. Hierzu zählen beispielsweise die Integration wahrgenommener Reizsituationen oder interner Zustandsänderungen, das Erinnern und Vergessen von Informationen, Denken und auch Lernen. Diese Funktionen beschreiben ganz allgemein also Mechanismen, die neue Informationen an das Gedächtnis anfügen, bestehende Informationen aus dem Gedächtnis entfernen sowie Gedächtnisinhalte modifizieren.

Je nach Art der Verhaltenssteuerung werden unterschiedliche Anforderungen an die kognitiven Fähigkeiten eines Organismus gestellt.

Für das Funktionieren automatischer reaktiver Verhaltensweisen genügen relative einfache kognitive Fähigkeiten. Im wesentlichen müssen Zustände der Umwelt oder des internen Zustands des Individuums mental repräsentiert werden, damit die Inputschemata der reaktiven Aktionsschemata zur Auswahl geeigneter Aktionen mit den tatsächlichen Gegebenheiten und Situationen verglichen werden können.

Die Anforderungen an die kognitiven Fähigkeiten steigen im Falle des deliberativen Verhaltens deutlich an. Hier reicht es nicht mehr aus, Zustände nur statisch abzuspeichern, sondern vielmehr wird eine dynamische Repräsentation der Umwelt und der internen Zustände benötigt. Dieses mentale Modell dient als Grundlage für Denk-, Planungs- und Problemlösevorgänge, die als inneres Probehandeln dem eigentlichen realen Handeln vorgeschaltet sind. Als Ausgangspunkt für deliberatives Verhalten müssen Handlungsziele definiert und zugänglich gemacht werden. Resultate der handlungsorientierten Denkvorgänge wie z. B. Handlungspläne, müssen ebenso gespeichert und für die weiteren Verarbeitungsschritte zur Verfügung gestellt werden. Für deliberatives Verhalten müssen also sowohl die kognitiven Strukturen als auch die zugehörigen Funktionen im Vergleich zum reaktiven Verhalten erweitert werden.

Für die Ausführung reflektiver Verhaltensweisen scheint keine zusätzliche Erweiterung kognitiver Strukturen nötig zu sein. In bezug auf die Ausgestaltung mentaler Modelle rückt allerdings der Aspekt des Selbstbildes im Vergleich zum Umweltmodell in den Vordergrund, da im wesentlichen gesteckte Handlungsziele sowie interne Zustände, Prozesse und Fähigkeiten zum Gegenstand reflektiver Betrachtungen gemacht werden.

4.2.5 Emotionen

Neben Motiven und kognitiven Fähigkeiten spielen auch Emotionen als psychische Kräfte eine Rolle für das menschliche Handeln, Entscheiden und Verhalten. Emotionen sind alltägliche Phänomene, hängen oftmals mit persönlich bedeutsamen Ereignissen zusammen und stehen in enger Beziehung zum Handeln oder zumindest mit dem Impuls zu handeln (Meyer, Schützwohl & Reisenzein, 1993).

Emotionsforschung gewinnt nach einer gewissen Flaute, die im wesentlichen durch behavioristische Ansätze ausgelöst wurde, in den letzten Jahren wieder zunehmend an Bedeutung. Allerdings existiert derzeit keine allgemein anerkannte und umfassende Theorie der Emotionen. Vielmehr gibt es eine Vielzahl unterschiedlicher Ansätze, die sich zum Teil unterstützen, zum Teil widersprechen und oftmals auch nur mit bestimmten Teilaspekten beschäftigen. Ebenso wenig sind derzeit exakte Definitionen des Begriffes *Emotion* vorhanden. Meyer, Schützwohl und Reisenzein (1993) geben eine Arbeitsdefinition von Emotionen an. Dabei werden Emotionen als multidimensionale Zustände von Personen (Averill, 1968) definiert, die von unterschiedlicher Qualität und Intensität sein können, mit einem Erlebensaspekt in Verbindung stehen, mit physiologischen Veränderungen einhergehen und bestimmte Verhaltensweisen hervorrufen. Der Verhaltensaspekt von Emotionen umfaßt in einem engen Sinne den motorischen Ausdruck, d. h. den Gesichtsausdruck, Gestik, Körperhaltung und –orientierung. In einem weiter gefaßten Sinne zählen hierzu auch beobachtbare Handlungen oder Handlungstendenzen wie z. B. Fluchtverhalten bei Furcht oder Angriffsverhalten bei Wut (Öhman, 1987).

In der Literatur existiert eine Vielzahl unterschiedlicher Emotionstheorien, die die Entstehung, die Natur und die Auswirkungen von Emotionen auf das menschlichen Handeln in vielfältiger Weise zu erklären versuchen. Diese Theorien können nach Meyer, Schützwohl und Reisenzein (1993) in Verhaltenstheorien, mentalistische Theorien und Syndromtheorien klassifiziert werden. Verhaltenstheorien setzen Emotionen mit beobachtbarem Verhalten gleich und entspringen im wesentlichen der behavioristischen Tradition. Im Vergleich zu den beiden anderen Theoriekategorien spielen Verhaltenstheorien keine wesentliche Rolle mehr für die gegenwärtige Sozialforschung. Von größerer Bedeutung sind mentalistische Theorien, die Emotionen mit gewissen mentalen Zuständen gleichsetzen. Syndromtheorien bringen schließlich die beiden erst genannten Sichtweisen miteinander in Verbindung.

Für mentalistische Emotionstheorien bzw. Syndromtheorien ist charakteristisch, daß sie mentale Zustände, physiologische Veränderungen und spezifische Verhaltenstendenzen als wesentliche Aspekte im Zusammenhang mit Emotionen herausheben. Unterschiede finden sich hauptsächlich in der konkreten Definition von Emotionen und in den Beziehungen, die zwischen diesen Aspekten angenommen werden, d. h. welche Aspekte im einzelnen als Ursachen, Wirkungen oder auch Begleiterscheinungen von Emotionen betrachtet werden.

Eine weitere Gruppe von Theorien geht von der Annahme aus, daß eine geringe Anzahl an Grundemotionen wie z. B. Freude, Traurigkeit, Ärger, Furcht oder Ekel existiert und alle anderen Emotionen als Mischformen der Grundemotionen aufzufassen sind (z. B. Oatley & Johnson-Laird, 1987).

Im Rahmen dieser Arbeit ist es leider nicht möglich, konkrete Ansätze für derartige Theorien genauer darzustellen. Der interessierte Leser sei daher für eine weiterführende Lektüre an dieser Stelle beispielsweise auf die Theorien von Schachter und Singer (1962), Frijda (1986), Lazarus (1991) oder Plutchik (1994) verwiesen, die für die gegenwärtige Emotionsforschung von Bedeutung sind.

Agentenarchitekturen, die die Modellierung menschlichen Handelns, Entscheidens und Verhaltens unterstützen sollen, müssen grundsätzlich die Möglichkeit bieten, Emotionen zu beschreiben. Von Bedeutung sind hierbei vor allem die Aktualgenese von Emotionen, die Repräsentation von Emotionen als zusätzliche, eigenständige Zustände, sowie die Auswirkungen emotionaler Zustände auf andere interne Zustände, Prozesse und die Verhaltenssteuerung von Agenten. Je nach Art und Intensität können Emotionen das Verhalten von Agenten modulieren, d. h. die konkrete Durchführung von Handlungen beeinflussen. Starke Emotionen besitzen allerdings auch motivationale Eigenschaften und können als innere Antriebskräfte ein spezifisches Handeln in Gang setzen, leiten und aufrechterhalten.

4.2.6 Physiologische Einflüsse

Das Handeln, Entscheiden und Verhalten von Menschen hängt nicht nur von ihren kognitiven Fähigkeiten und emotionalen Befindlichkeiten ab. Einen wesentlichen Einfluß auf die Verhaltenssteuerung haben auch physiologische Zustände und Prozesse.

Der Mensch verfügt über physiologische Triebe, die zu einer bevorzugten Auswahl bestimmter Verhaltensweisen führen. Triebe werden als psychobiologische Prozesse angesehen und können nach Bierbaumer und Jänig (1997) in homöostatische und nichthomöostatische Triebe unterteilt werden.

Homöostatische Triebe weisen Abhängigkeiten zu körperinternen Sollwerten auf. Sie werden immer dann aktiviert, wenn der aktuelle physische Zustand von diesen Sollwerten abweicht, und sollen dazu führen, daß ein ausgeglichener Zustand wiederhergestellt wird. Beispiele für homöostatische Triebe sind Temperaturerhaltung, Hunger, Durst und Schlaf.

Exemplarisch für homöostatische Triebe sollen hier einige charakteristische Aspekte von Hunger und Durst erläutert werden. Hunger und Durst definieren zentrale Triebzustände, die eine gewisse Bereitschaft erzeugen, Nahrung bzw. trinkbare Flüssigkeit zu suchen und zu konsumieren. In der Physiologie finden sich relativ genaue Angaben darüber, welche Bedingungen für das Auftreten von Hunger- und Durstempfindungen verantwortlich sind. Durst ist im wesentlichen an den Wasserhaushalt gekoppelt und entsteht, wenn der Körper einen gewissen Prozentsatz seines Körpergewichtes an Wasser verloren hat. Hunger wird maß-

geblich durch den Glukosespiegel im Körper reguliert und ist an die Blutglukosekonzentration geknüpft. Hunger und Durst führen zu appetetiven Verhaltensweisen und letztlich zur Aufnahme von Flüssigkeit und Nahrung. Die Flüssigkeits- bzw. Nahrungsaufnahme wird dann nach einiger Zeit mit Hilfe von Sättigungsmechanismen gestoppt.

Neben homöostatischen Trieben existieren auch nichthomöostatische Triebe wie z. B. der Sexualtrieb, der mit Hilfe der geschlechtlichen Fortpflanzung für die Aufrechterhaltung der Population sorgen soll.

Die physische Verfassung hängt neben organismischen Eigenschaften auch von den Verhaltensweisen ab, die ein Mensch ausführt. Länger andauernde geistige und körperliche Beanspruchungen führen beispielsweise zu einer Ermüdung, die mit einer reversiblen Verminderung der Leistungsfähigkeit sowie der Fähigkeit, körperlich anstrengende Aktionen auszuführen, verbunden ist. Vor allem die Arbeits- und Sportphysiologie liefern hier empirische Untersuchungen, die etwa tagesrhythmische Einflüsse mit der Leistungsfähigkeit von Arbeitern in Beziehung setzen oder Aussagen über Menge und Dauer sinnvoller Arbeitspausen machen (Ulmer, 1997).

Auch krankhafte Ausprägungen physischer Eigenschaften können für die Gestaltung von Agenten eine Rolle spielen. Neuere Ansätze in der Gesundheitsökonomie wenden agentenbasierte Modellierungsansätze an, um unterschiedliche Organisationsformen und Strategien in medizinischen Institutionen zu untersuchen. Agenten werden dabei beispielsweise eingesetzt, um Patienten mit bestimmten Krankheitsbildern zu repräsentieren, die gemäß ihres physiologischen Zustands unterschiedliche Behandlungseinheiten durchlaufen müssen (Sibbel & Urban, 2000).

Auch an dieser Stelle kann eine Zusammenstellung physiologischer Einflüsse auf das menschliche Verhalten nicht erschöpfend sein. Die angeführten Aspekte sollen lediglich die Notwendigkeit vor Augen führen, im Rahmen der Erstellung einer Agentenarchitektur, die die Modellierung menschlichen Verhaltens in flexibler Weise ermöglichen soll, physiologische Eigenschaften zu berücksichtigen.

4.2.7 Soziale Einflüsse und Prozesse

Sozialpsychologen gehen von der Annahme aus, daß menschliches Verhalten maßgeblich durch die Gegebenheiten der jeweiligen sozialen Situation determiniert wird (Zimbardo & Gerrig, 1999; Mosler & Brucks, 2001). Für die Verhaltensweisen eines Menschen sind also nicht nur innere Zustände und Prozesse verantwortlich, sondern auch externe, das soziale Umfeld betreffende Faktoren von Bedeutung. Ganz grundsätzlich sollte also eine Agentenarchitektur, die eine umfassende Modellierung menschlichen Verhaltens unterstützen soll, auch die Modellierung derartiger Einflüsse ermöglichen, die die soziale Situation von Agenten betreffen. Im folgenden sollen zur Verdeutlichung der damit verbundenen Anforderungen an eine Modellierung einige sozialwissenschaftliche Phänomene und

Sachverhalte angerissen werden, die den Menschen als soziales Wesen beeinflussen und sein Verhalten in entscheidender Weise mitbestimmen.

Menschen nehmen in unterschiedlichen Situationen verschiedene soziale Rollen ein. Ein soziale Rolle beschreibt dabei eine Menge sozial definierter Verhaltensmuster, die in einer gegebenen Situation von einem Akteur erwartet werden und auf das Verhalten anderer Akteure abgestimmt sind (Häcker & Stapf, 1998). Rollen sind vor allem dann von Bedeutung, wenn mehrere Akteure im Rahmen sozialer Strukturen interagieren oder zusammenarbeiten.

Oftmals werden einzelnen Individuen unterschiedliche Positionen oder auch unterschiedlicher sozialer Status zugeordnet. Eine Position definiert die Stellung eines Individuums innerhalb einer sozialen Struktur. Sie kann in verschiedenen Situationen entscheidend dafür sein, wie hoch der Einfluß eines Individuums auf das Verhalten oder die Einstellungen anderer Individuen innerhalb der umgebenden Sozialstrukturen ist. Mit einer derartigen Position ist oftmals auch ein sozialer Status verbunden, der mit einem gewissen Ansehen oder auch bestimmten Rechten einhergeht, die die eigene Gruppe der jeweiligen Person zubilligt. Je nach Status und Position der einzelnen Mitglieder können sich Rangordnungen und Hierarchien in sozialen Strukturen ausbilden, die einen entscheidenden Einfluß auf deren Funktionsweise haben.

Um das Zusammenleben von Individuen in sozialen Strukturen zu erleichtern, haben sich im Laufe der Zeit soziale Normen und Regeln entwickelt. Diese Normen und Regeln geben anerkannte Handlungs- und Wertstandards vor und existieren zumeist in Form von ungeschriebenen Vorschriften (Bergius, 1998a). Die Übertretung bzw. Nichteinhaltung sozialer Normen wird häufig mit negativen Sanktionen bestraft. In der gegenwärtigen sozialwissenschaftlichen Forschung wird der Untersuchung von Entstehung und Auswirkungen sozialer Normen ein hohes Maß an Bedeutung beigemessen. Insbesondere die Verfügbarkeit agentenbasierter Modellierungsansätze fördert Forschungsaktivitäten auf diesem Gebiet.

Nicht zuletzt kennzeichnen den Menschen als soziales Wesen auch soziale Bedürfnisse. Er strebt nach Geselligkeit und die Integration in soziale Strukturen und Verbände. Man spricht in diesem Zusammenhang auch von Anschluß- oder Affiliationsmotiven (McClelland, 1965), die den Menschen dazu drängen, in Interaktion mit einer oder mehreren anderen Personen zu treten, ohne damit sofort einen bestimmten Zweck zu verfolgen.

4.3 Anforderungen an die Modellierung von Kommunikationsmechanismen

Das Verhalten von Menschen findet zu einem erheblichen Teil in einem sozialen Kontext statt. Menschen interagieren mit anderen Menschen und richten ihr Verhalten aufeinander aus. Ein zentrales Element im Zusammenhang mit sozialer Interaktion ist Kommunikation. Kommunikation bezeichnet den Informationsaus-

tausch zwischen Individuen, der durch das sinnlich wahrnehmbare Verhalten stattfindet (Ellgring, 1990).

Grundsätzlich stehen dem Menschen verschiedene Modalitäten zur Verfügung, um Informationen mit anderen Menschen auszutauschen. Hierbei kann im wesentlichen zwischen nonverbaler und verbaler Kommunikation unterschieden werden.

Im Rahmen der nonverbalen Kommunikation dienen etwa spezifische Verhaltensweisen, Mimik, Gestik, Körperhaltungen oder Modulationen der Stimme zur Enkodierung verschiedenartiger Informationen. Nonverbales Verhalten läuft beim Menschen zu einem großen Teil unbewußt ab und bringt eher den emotionalen Gehalt von Informationen zum Ausdruck als eigentliche Kommunikationsinhalte. Im Rahmen von agentenbasierten Modellen spielt nonverbale Kommunikation bislang keine Rolle. Aus diesem Grund sollen derartige Aspekte in dieser Arbeit nicht weiterverfolgt werden.

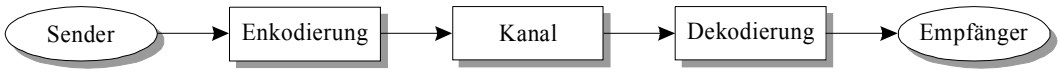

Abbildung 4.1: Vereinfachtes Modell direkter Kommunikation in Anlehnung an Neuburger (1972)

Bei der verbalen Kommunikation besteht das Ziel im wesentlichen in der Übermittlung konkreter Inhalte. Eine Nachricht wird dazu zunächst von einem Sender enkodiert. Dies bedeutet, daß der Inhalt einer Nachricht in konkrete Signale, z. B. einer bestimmten Sprache, transformiert wird. Die Nachricht wird dann im Falle einer direkten Kommunikation über einen Kanal zu einem oder einer Menge an Empfängern übermittelt. Der oder die Empfänger dekodieren dann die Nachricht, d. h. sie nehmen die Nachricht auf, interpretieren sie und erschließen damit deren Inhalt.

Abbildung 4.2: Vereinfachtes Modell indirekter Kommunikation in Anlehnung an Erman, Hayes-Roth, Lesser & Reddy (1980)

Zusätzlich existiert eine indirekte Form der Nachrichtenübermittlung. Hier wird eine Nachricht durch den Sender erarbeitet, enkodiert und daraufhin an einer für die geplanten Empfänger zugänglichen Stelle verfügbar gemacht. Die Empfänger können dann gemäß ihrer Aufgaben, Bedürfnisse und Ziele die so bereitgestellten Informationen anderer Akteure abholen und weiterverarbeiten. Diese Art der Kommunikation ermöglicht eine zeitlich entkoppelte Generierung und Verarbeitung von Nachrichten und bewirkt dadurch eine größere Unabhängigkeit zwischen Erzeugern und Verbrauchern.

Sowohl direkte als auch indirekte Kommunikation spielen im Rahmen von agentenbasierten Modellierungsansätzen eine wesentlichen Rolle. In der Verteilten Künstlichen Intelligenz wird direkte Kommunikation im wesentlichen mit Hilfe von Message Passing-Verfahren (Albayrak & Bussmann, 1993) modelliert, während für die Modellierung von indirekter Kommunikation in der Regel auf Blackboard-Systeme (Erman, Hayes-Roth, Lesser & Reddy, 1980) zurückgegriffen wird. Ein Referenzmodell, dessen Ziel darin besteht, die agentenbasierte Modellierung menschlichen Verhaltens auch im sozialen Kontext zu unterstützen, sollte grundsätzliche Modellierungsmechanismen sowohl für direkte Kommunikation als auch für indirekte Kommunikation zur Verfügung stellen.

4.4 Anforderungen an die Modellierung der Umwelt

Der Begriff *Umwelt* bezeichnet die Gesamtheit des Lebensraumes, der ein Lebewesen umgibt, beschränkt auf diejenigen Einflüsse, die für das Lebewesen tatsächlich bestimmend sind (Häcker & Stapf, 1998). Die Umwelt gibt einen äußeren Bezugsrahmen für das menschliche Handeln und Verhalten vor.

Von Bedeutung kann hier beispielsweise die Topologie der Umgebung sein. Die Umwelt stellt in diesem Zusammenhang Aufenthaltsorte für Individuen bereit, die mit unterschiedlichen Eigenschaften versehen sind und damit auch für das Individuum unterschiedliche Bedeutungen und Attraktivitäten haben können. Zudem können in der Umwelt verschiedenartige Ressourcen existieren, die durch die Individuen konsumiert, aber auch gepflegt und geschützt werden.

Grundsätzlich treten in der Umwelt von Individuen zwei Typen von dynamischen Veränderungen auf. Zum einen verfügt die Umwelt im allgemeinen über eine Eigendynamik. Hierbei handelt es sich um sowohl kontinuierliche als auch diskrete Übergänge zwischen Zuständen, die ohne weiteres Zutun von Akteuren selbständig in der Umgebung stattfinden. Als Beispiele für kontinuierliche Zustandsänderungen wären etwa Wachstums- oder Regenerationsprozesse von Ressourcen zu nennen. Diskrete Zustandsübergänge sind an auftretende Ereignisse gekoppelt und führen zu sprunghaften Veränderungen von Zuständen in der Umwelt. Derartige Ereignisse können beispielsweise durch Katastrophen hervorgerufen werden, bei denen sich der Bestand verschiedener Ressourcen plötzlich verändert.

Neben der Eigendynamik, die das Verhalten der Umwelt von Individuen bestimmt, nehmen auch die Individuen selbst Einfluß auf den Zustand ihrer Umwelt. Die Individuen induzieren durch ihre Verhaltensweisen Zustandsänderungen in der Umwelt. Dies kann beispielsweise geschehen, um Bedürfnisse zu befriedigen, oder auch, um die Umwelt nach den eigenen Wünschen und Vorstellungen zu verändern. Auch diese durch das individuelle Verhalten induzierten Zustandsübergänge können sowohl kontinuierlicher als auch diskreter Natur sein.

Zwischen dem Menschen und seiner Umwelt besteht also eine intensive Wechselwirkung (Mogel, 1996). Insbesondere spielt die Umwelt für das menschliche Verhalten eine ganz wesentliche Rolle und muß aus diesem Grund auch im Rahmen eines agentenbasierten Modellierungsansatzes in adäquater Weise berücksichtigt werden.

4.5 Zusammenfassung

Die agentenbasierte Modellierung menschlichen Handelns, Entscheidens und Verhaltens erfordert eine integrative Betrachtung einer Vielzahl unterschiedlicher Anforderungen. Grundsätzlich lassen sich diese Anforderungen in drei übergeordnete Kategorien einteilen. Die erste Kategorie betrifft Anforderungen an die Konzeption und den Entwurf einer Agentenarchitektur, die eine flexible Modellierung unterschiedlicher Einflüsse, die für das menschlichen Handeln, Entscheiden und Verhalten von Bedeutung sind, ermöglicht. Die zweite Kategorie umfaßt Anforderungen im Hinblick auf die Kommunikationsinfrastruktur in agentenbasierten Modellen und die dritte Kategorie beschäftigt sich schließlich mit der Modellierung der Umwelt von Agenten, sowie den Wechselwirkungen, die zwischen der Umwelt und den Agenten existieren.

Die entscheidende Entwurfsaufgabe besteht in der Konzeption einer Agentenarchitektur, die dem Modellentwickler die Möglichkeit bietet, verschiedenartige Funktionalitäten im Zusammenhang mit menschlichem Verhalten in Einklang zu bringen. Dazu ist eine Architektur zu entwickeln, die sehr allgemein ausgerichtet und auf einer hohen Abstraktionsebene angesiedelt ist. Auf diese Weise kann sichergestellt werden, daß die Agentenarchitektur in verschiedenen Anwendungsdomänen einsetzbar ist.

Zwei wesentliche Anforderungen an eine derartige Agentenarchitektur bestehen zunächst einmal in der Berücksichtigung sensorischer und aktuatorischer Fähigkeiten von Individuen. Sensorische Fähigkeiten ermöglichen in Zusammenarbeit mit Wahrnehmungsprozessen eine Informationsaufnahme aus der Umwelt, sowie die Verarbeitung von Nachrichten und Mitteilungen. Aktuatorische Fähigkeiten erlauben eine Modifikation sowohl des eigenen Zustandes als auch der Umgebung.

Aus der Betrachtung komplexer, sozialwissenschaftlicher Handlungstheorien resultiert die Einsicht, daß die menschliche Handlungsregulation in entscheidender Weise von kognitiven wie auch emotionalen, physiologischen und sozialen Zuständen und Prozessen geleitet wird. Dieser Erkenntnis muß beim Entwurf der Agentenarchitektur Rechnung getragen werden. Der interne Zustand von Agenten muß in einer derartigen Weise gestaltet werden, daß Zustände und Prozesse, die der Modellierung physischer, emotionaler, kognitiver und sozialer Eigenschaften realer Akteure dienen, gleichermaßen auf die Handlungsregulation von Agenten einwirken können. Um zudem die Entwicklung domänenspezifischer Modelle mit

unterschiedlichen Anforderungen an die interne Ausstattung der Agenten zu unterstützen, soll eine flexible Handhabung des internen Zustands von Agenten sichergestellt werden. Dieser Aspekt betrifft im wesentlichen die schrittweise Hinzunahme von einzelnen Kategorien interner Zustände im Laufe des Entwicklungsprozesses, sowie die Ausblendung von Kategorien, die etwa für die zu realisierende Fallstudie ohne Bedeutung sind.

Weitere Anforderungen entstehen durch die Tatsache, daß Menschen zu einem großen Teil in einem sozialen Umfeld agieren und dabei mit anderen Menschen kommunizieren. Um auch im Modell einen Informationsaustausch zwischen Agenten zu ermöglichen, muß das Referenzmodell neben einer Agentenarchitektur auch eine adäquate Infrastruktur vorsehen, die für die Organisation von Kommunikationsprozessen zuständig ist. Im wesentlichen gilt es dabei, eine direkte Kommunikationsmöglichkeit zwischen den Agenten im Sinne eines Message Passing zu schaffen, sowie einen indirekten Informationsaustausch zu ermöglichen, der mit Hilfe gemeinsam zugreifbarer Datenbereiche (Blackboards) realisiert werden kann.

Nicht zuletzt beeinflußt die Umgebung in hohem Maße das menschliche Verhalten. Die Modellierung von Umgebungseinflüssen gibt also eine weitere Anforderung für die Konzeption eines derartigen Referenzmodells vor. Das Umgebungsmodell ist dabei so auszulegen, daß sowohl die Umwelt durch das Verhalten der Agenten beeinflußt werden kann, als auch die Umwelt Rahmen- und Randbedingungen für das Verhalten der Agenten setzt.

Um die Anpaßbarkeit des Referenzmodells an die spezifischen Anforderungen verschiedener Problemdomänen zu gewährleisten, soll im Rahmen dieser Arbeit ein domänenunabhängiges Modellierungskonzept vorgestellt werden, das diese unterschiedlichen Anforderungen auf einer relativ hohen Abstraktionsebene integriert und im wesentlichen strukturelle Aspekte der Modellierung in den Vordergrund stellt.

5 BASISKONZEPTE DES REFERENZMODELLS *PECS*

Das Referenzmodell *PECS* (Physis, Emotion, Cognition, Social Characteristics) stellt ein domänenunabhängiges Konstruktionsschema für agentenbasierte Simulationsmodelle zur Verfügung, in denen menschliches Handeln, Entscheiden und Verhalten von ausschlaggebender Bedeutung sind. Es gibt einen konzeptionellen Rahmen vor, der anwendungsspezifisch mit Inhalt gefüllt werden kann. *PECS* soll dazu beitragen, die Komplexität der Entwurfsaufgabe, die mit der Entwicklung derartiger agentenbasierter Simulationsmodelle unumgänglich verbunden ist, zu reduzieren.

Bevor in den nachfolgenden Abschnitten die einzelnen Bestandteile des Referenzmodells *PECS* im Detail dargestellt werden, sollen nun zunächst die Grundideen vorgestellt werden, die den Entwurf des Referenzmodells im wesentlichen prägen.

Um eine adäquate Beschreibung sowie eine klare Strukturierung agentenbasierter Simulationsmodelle zu gewährleisten, werden für die Ausgestaltung des Referenzmodells *PECS* zwei übergeordnete Entwurfsprinzipien herangezogen. Es handelt sich dabei zum einen um die zustandsorientierte Beschreibung von Modellattributen und Modelldynamik auf der Grundlage eines klassischen systemtheoretischen Ansatzes und zum anderen um das Prinzip des komponentenorientierten, hierarchischen Modellaufbaus, der einen modularen und übersichtlichen Entwurf komplexer Modelle ermöglicht.

5.1 Der systemtheoretische Ansatz

Die Systemtheorie liefert wesentliche Beiträge für die Beschreibung und Modellierung sowohl natürlicher als auch künstlicher Systeme. Sie geht von der Grundüberzeugung aus, daß ein System durch einen internen Zustand gekennzeichnet ist, der im Laufe der Zeit sowohl eigendynamisch, als auch infolge von Umwelteinflüssen verändert werden kann, und daß in Abhängigkeit des internen Zustands

sowie der Umwelteinflüsse nach außen hin sichtbare Outputs erzeugt werden können.

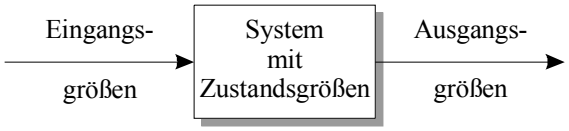

Abbildung 5.1: Grundstruktur eines dynamischen Systems mit Zustandsgrößen nach Pichler (1975)

5.1.1 Die Zustandsdarstellung dynamischer Systeme

Pichler (1975) definiert die Zustandsdarstellung eines Systems *S* formal als ein 11-Tupel

$$(A, B, X, Y, Q_0, Q, Z, F, G, T, t_0),$$

wobei gilt:

- *A*, *B* und *Q* sind beliebige Wertemengen.
- *T* ist eine Menge von Zeitwerten, deren Elemente in aufsteigender Ordnung vorliegen.
- $X: T \rightarrow A$ beschreibt die Eingangszeitfunktion des Systems *S*.
- $Y: T \rightarrow B$ beschreibt die Ausgangszeitfunktion des Systems *S*.
- $Z: T \rightarrow Q$ beschreibt die globale Zustandsüberführungsfunktion des Systems *S*.
- Q_0 beschreibt den Anfangszustand des Systems *S* zum Zeitpunkt $t_0 \in T$, d. h. es gilt: $Z (t_0) = Q_0$.
- $F: (T \times Q \times A) \rightarrow Q$ beschreibt die lokale Zustandsüberführungsfunktion des Systems *S*.
- $G: (T \times Q \times A) \rightarrow B$ beschreibt die Ausgabefunktion des Systems *S*.

Von besonderer Bedeutung für die Zustandsdarstellung eines Systems sind drei Kategorien von Größen: Eingangsgrößen, Zustandsgrößen und Ausgangsgrößen.

Als Eingangsgrößen werden Größen eines Systems bezeichnet, deren Werte das betrachtete System aus der externen Systemumgebung bezieht (Eschenbacher, 1995). Eingangsgrößen werden durch die Systemumgebung bereitgestellt und bleiben durch das betrachtete System selbst unbeeinflusst. Der zeitliche Verlauf der Werte für die Eingangsgrößen wird durch die Eingangszeitfunktion $X (t)$ beschrieben, die jedem Zeitpunkt $t \in T$ eine aktuelle Wertebelegung $a \in A$ der Eingangsgrößen zuordnet.

Zustandsgrößen beschreiben den internen Zustand eines Systems. Sie werden immer dann benötigt, wenn ein gegebenes System auf einen bestimmten Input hin nicht immer mit demselben Output antwortet. Der interne Systemzustand kann auch bei unveränderten Inputs aus der Systemumgebung zu Modifikationen des Systemverhaltens führen. Dabei resultiert der aktuelle Zustand eines Systems über eine Reihe von Zustandstransformationen aus dem Initialzustand Q_0 des Systems und dem zeitlichen Verlauf der Werte für die Eingangsgrößen.

Eine besondere Bedeutung kommt in diesem Zusammenhang der lokalen Zustandsüberführungsfunktion F zu. Die lokale Zustandsüberführungsfunktion F beschreibt, in welcher Weise ein Systemzustand zu einem gegebenen Zeitpunkt $t_n \in T$ unter Berücksichtigung eines gegebenenfalls vorliegenden Inputs $X(t_n)$ in einen Nachfolgezustand zum nächsten betrachteten Zeitpunkt $t_{n+1} \in T$ übergeht. Ausgehend vom Initialzustand Q_0 des Systems S erhält man den Verlauf der globalen Zustandsüberführungsfunktion Z durch eine sukzessive Auswertung der lokalen Zustandsüberführungsfunktion F für alle relevanten Zeitpunkte $t_i \in T$. Die lokale Zustandsüberführungsfunktion kann dabei sowohl zeitdiskrete als auch zeitkontinuierliche Zustandsübergänge beschreiben.

Für zeitdiskrete Zustandsübergänge gilt die Definitionsgleichung:

$$Z(t_{n+1}) := F(t_n, Z(t_n), X(t_n)) \qquad \text{(Gl. 5.1)}$$

Dabei sind t_n und t_{n+1} diskrete Zeitpunkte aus der Zeitmenge T.

Zeitkontinuierliche Zustandsübergänge können mit Hilfe von Differentialgleichungen der Form

$$Z'(t) := F(t, Z(t), X(t)) \qquad \text{(Gl. 5.2)}$$

beschrieben werden, wobei wiederum $t \in T$ gilt.

Als Ausgangsgrößen eines Systems werden Größen bezeichnet, die das System nach außen zur Systemumgebung hin sichtbar macht. Die Werte der Ausgangsgrößen hängen von den Werten der Eingangsgrößen und der Zustandsgrößen ab und stellen sich unmittelbar, d. h. ohne weiteren Zeitverzug, ein. Die Ausgabe $Y(t)$ eines Systems S zum Zeitpunkt $t \in T$ wird mit Hilfe der Ausgabefunktion G beschrieben, wobei gilt:

$$Y(t) := G(t, Z(t), X(t)) \qquad \text{(Gl. 5.3)}$$

Aufgrund dieser Darstellung wird ersichtlich, daß der interne Zustand $Z(t)$ eines Systems in zweierlei Hinsicht von Bedeutung ist. Zum einen bestimmt er im Zusammenspiel mit dem Input, der auf das System einwirkt, das Aussehen des Folgezustands, der sich für das System zum nächsten betrachteten Zeitpunkt ergibt. Zum anderen bedingt der interne Systemzustand darüber hinaus auch den Output, der der Umgebung durch das System zur Verfügung gestellt wird.

5.1.2 Zustandsorientierte Modellbeschreibung und die Modellierung menschlichen Verhaltens

Der im vorangegangenen Abschnitt geschilderte, systemtheoretische Ansatz führt zu einer zustandsorientierten Beschreibung dynamischer Systeme. Akzeptiert man die Grundüberzeugung, daß auch der Mensch als ein derartiges, wenn auch sehr komplexes, dynamisches System angesehen werden kann, so steht auch dem Versuch nichts mehr entgegen, Systeme, die durch menschliches Handeln, Entscheiden und Verhalten wesentlich beeinflußt werden, mit Hilfe systemtheoretischer Konzepte zu modellieren.

Der Mensch soll also im Rahmen dieser Arbeit als ein dynamisches System betrachtet werden, das Inputs aus seiner Umgebung empfängt und verarbeitet, über einen ausgeprägten internen Zustand verfügt, der physische, emotionale, kognitive und soziale Aspekte beschreibt, sowie Outputs, z. B. in Form von unterschiedlichen Verhaltensweisen und Aktionen, zu generieren in der Lage ist. Ein systemtheoretisches Modell eines Menschen mit derartigen Eigenschaften soll im Rahmen dieser Arbeit als *Agent* bezeichnet werden.

Die zustandsorientierte Modellbeschreibung erscheint in diesem Zusammenhang besonders geeignet, um die komplexen Zustände, die das menschliche Handeln, Verhalten und Entscheiden beeinflussen, sowie die zugehörigen Zustandstransformationen in adäquater und komfortabler Weise zu beschreiben.

5.1.3 Die Eingabefunktion von Agenten

Die Eingabefunktion beschreibt, welche Inputs im Laufe der Zeit auf ein System einwirken. Sie gibt den zeitlichen Verlauf der Werte für die Eingangsgrößen an, die die Systemumgebung für das betrachtete System zur Verfügung stellt.

Agenten werden von außen zum einen durch deren Umwelt und zum anderen auch durch andere Agenten beeinflußt. Zu Umgebungseinflüssen zählen in Abhängigkeit der untersuchten Situation verschiedene Arten visueller, akustischer oder auch haptischer Informationen, die in der Umgebung erzeugt und durch die Agenten wahrgenommen werden. Zusätzlich können Aktionen und Nachrichten anderer Agenten als Inputs für Agenten auftreten. Von entscheidender Bedeutung ist dabei der Sachverhalt, daß die Inputs, die auf einen Agenten einwirken, sowohl zu einer Modifikation seines internen Zustandes beitragen können, als auch die Outputs mitbestimmen, die der Agent erzeugt.

5.1.4 Der interne Zustand von Agenten

Der interne Zustand eines Agenten ergibt sich aus der aktuellen Belegung seiner internen Zustandsvariablen. Die Zustandsvariablen beschreiben dabei alle Einflüs-

se, die für die Bestimmung der Dynamik des Agenten in der betrachteten Fallstudie als relevant erachtet werden. Ein Agent wird niemals mit allen Eigenschaften ausgestattet werden, die den realen Menschen, der modelliert werden soll, insgesamt ausmachen, sondern lediglich mit einer kleinen, signifikanten Untermenge dieser Eigenschaften. Anzumerken ist an dieser Stelle, daß die Auswahl der als relevant erachteten Eigenschaften in entscheidender Weise die Qualität einer Modellierung beeinflußt und aus diesem Grund besonderer Sorgfalt bedarf.

Wie in Kapitel 4 im Überblick dargestellt wurde, sind für die Beschreibung des menschlichen Handelns, Entscheidens und Verhaltens physische, emotionale, kognitive und soziale Aspekte von Bedeutung. Um Strukturähnlichkeit zwischen der Funktionsweise eines realen Menschen und eines Agenten, der den Menschen im Modell abbilden soll, zu erreichen, müssen im Inneren des Agenten Zustandsvariablen vorgesehen werden, die diese Einflüsse zu modellieren in der Lage sind. Die Zustandsvariablen, die einem Agenten zugeordnet werden, können also entsprechend ihrer Bedeutung gedanklich in vier Kategorien eingeteilt werden: in Zustandsvariablen, die die physische Konstitution (Physis), die emotionale Befindlichkeit (Emotion), den kognitiven Zustand (Cognition), sowie die sozialen Charakteristika (Social Characteristics) des Agenten beschreiben.

Als physische Eigenschaften werden Attribute wie etwa Alter, Körpergröße, Geschlecht, Blutzuckerspiegel oder der Grad der körperlichen Ermüdung geführt, sofern sie für die gegebene Fragestellung von Bedeutung sind und den internen Zustand oder auch das Outputverhalten des Agenten beeinflussen.

Emotionale Zustandsvariablen legen fest, welche emotionalen Zustände ein Agent im Hinblick auf seine Verhaltenssteuerung berücksichtigen kann. Derartige Zustandsvariablen können emotionale Zustände wie beispielsweise Freude, Zufriedenheit, Trauer, Neid oder Wut beschreiben und diese Zustände für den Mechanismus der Verhaltensauswahl zugänglich machen.

Kognitive Zustandsvariablen speichern im wesentlichen die Informationen, die einem Agenten über seinen eigenen Zustand und den Zustand seiner Umgebung zur Verfügung stehen. Ein Agent kann also beispielsweise Informationen über seine körperliche Leistungsfähigkeit oder seinen sozialen Status innerhalb seiner Arbeitsgruppe haben, sowie auch Kenntnisse darüber besitzen, wie hoch die Temperatur in seiner Umgebung gerade ist.

Eigenschaften, die einen Agenten im Hinblick auf sein soziales Umfeld beschreiben, werden in den sozialen Charakteristika des Agenten subsumiert. In diese Kategorie fallen etwa Eigenschaften, die die Zugehörigkeit zu sozialen Gruppen beschreiben, Positionen, die der Agent innerhalb dieser Gruppen einnimmt, oder auch soziale Bedürfnisse, die sein Handeln mit beeinflussen.

5.1.5 Die lokale Zustandsüberführungsfunktion von Agenten

Interne Zustände von Menschen sind nicht statisch, sondern unterliegen ständigen Veränderungen. Um diesem Sachverhalt bei der Modellierung Rechnung zu tra-

gen, müssen auch für Agenten Zustände vorgesehen werden, die sich im Laufe der Zeit verändern können. Diese Veränderungen können dabei mit Hilfe der lokalen Zustandsüberführungsfunktion beschrieben werden.

Wie im vorhergehenden Abschnitt beschrieben, setzt sich ein interner Zustand eines Agenten aus der aktuellen Belegung von Modellgrößen zusammen, die physische, emotionale, kognitive und soziale Eigenschaften des Agenten repräsentieren. Der interne Zustand $Z_A\ (t)$ eines Agenten läßt sich damit als Vereinigungsmenge der einzelnen Zustände, die den oben angeführten Kategorien angehören, auffassen. Damit gilt

$$Z_A\ (t) := Z_P\ (t) \cup Z_E\ (t) \cup Z_C\ (t) \cup Z_S\ (t). \qquad \text{(Gl. 5.4)}$$

Dabei bezeichnet $Z_P\ (t)$ den physischen Zustand, $Z_E\ (t)$ den emotionalen Zustand, $Z_C\ (t)$ den kognitiven Zustand und $Z_S\ (t)$ die Belegung der das soziale Umfeld betreffenden Zustandsvariablen des Agenten zum Zeitpunkt $t \in T$.

Zwischen den einzelnen internen Zuständen existieren vielfältige Wechselwirkungen, die bei der Gestaltung der lokalen Zustandsüberführungsfunktion eines Agenten Berücksichtigung finden müssen. Im allgemeinen Fall können also in der Parameterliste der lokalen Zustandsüberführungsfunktion eines Agenten Zustände aus allen vier Kategorien Physis, Emotion, Cognition und Social Characteristics auftreten. Die Zustandsübergänge können dabei sowohl eigendynamisch erfolgen, als auch durch Inputs aus der Umgebung hervorgerufen werden.

Eigendynamische Zustandsübergänge bedürfen keines externen Stimulus und führen in Abhängigkeit des aktuellen internen Zustandes zu einem neuen, internen Folgezustand. Im Falle eigendynamischer Übergänge von physischen, emotionalen, kognitiven und sozialen Zuständen nehmen die zugehörigen lokalen Zustandsüberführungsfunktionen also die folgende Form an:

$$F_I: (T \times Q_P \cup Q_E \cup Q_C \cup Q_S) \rightarrow Q_I$$

Dabei gilt:

- $I \in \{P, E, C, S\}$
- Q_I bezeichnet die Wertemengen des internen physischen, emotionalen, kognitiven und sozialen Zustands des Agenten.

Beispiele für eigendynamische Zustandsänderungen im Inneren eines Agenten wären etwa das Auftreten von Hunger infolge eines Abfalls des Blutzuckerspiegels, die Aktualgenese von Emotionen im Zusammenhang mit einer Veränderung des kognitiven Zustands, oder die Veränderung der kognitiven Leistungsfähigkeit in Abhängigkeit der physischen Konstitution.

Neben den eigendynamischen Zustandsübergängen existieren jedoch auch Zustandsübergänge, die von externen Inputs, d. h. im Falle der Agenten von Sensorinformationen, abhängig sind. Inputbezogene Zustandsübergänge hängen also zusätzlich zum aktuellen internen Zustand $Z_A\ (t)$ des Agenten auch von den In-

puts $X(t)$ ab, die zum gegebenen Zeitpunkt auf den Agenten einwirken. Die lokale Zustandsüberführungsfunktion muß also für den Fall inputabhängiger Zustandstransformationen in der folgenden Weise erweitert werden:

$$F_I: (T \times Q_P \cup Q_E \cup Q_C \cup Q_S \times A) \rightarrow Q_I$$

Dabei bezeichnet A die Wertemenge der Inputs, die auf den Agenten einwirken.

Inputabhängige Zustandstransformationen des Agenten wären beispielsweise die Aktualisierung des kognitiven Zustands, nachdem neue Informationen über den Zustand der Umwelt des Agenten eingetroffen sind, oder etwa eine Veränderung der physischen Konstitution in Abhängigkeit von Umweltbedingungen wie Temperatur oder Tageszeit.

5.1.6 Die Ausgabefunktion von Agenten

Die Ausgabefunktion beschreibt die kausalen Zusammenhänge zwischen dem internen Zustand eines Systems, den Inputs, die auf das System einwirken, und den Outputs, die das System erzeugt. Die Outputs bestehen im Falle von Agenten im wesentlichen aus den Verhaltensweisen und Aktionen, die die Agenten ausführen können. Die Auswahl derartiger Verhaltensweisen und Aktionen ist dabei sowohl von dem gesamten internen Zustand des Agenten, als auch von den zum betrachteten Zeitpunkt vorhandenen Inputs abhängig.

Legt man auch hier wiederum die Annahme zugrunde, daß sich der interne Zustand eines Agenten auf physische, emotionale, kognitive und soziale Eigenschaften bezieht, so erhält die Ausgabefunktion G eines Agenten die Form

$$G: (T \times Q_P \cup Q_E \cup Q_C \cup Q_S \times A) \rightarrow B$$

Ebenso wie die lokalen Zustandsüberführungsfunktionen werden auch die Verhaltensbeschreibungen von Agenten neben der Zeit und den vorhandenen Inputs auch durch deren gesamten internen Zustand beeinflußt. Wählt man beispielsweise Wenn-Dann-Produktionen als Beschreibungsform für das Verhalten von Agenten, so können im Bedingungsteil dieser Produktionen Bedingungen auftreten, die sich auf Zustandsvariablen aus den Kategorien Physis, Emotion, Cognition und Social Characteristics beziehen. Dieser Sachverhalt soll abschließend an einem kleinen, einfachen Beispiel verdeutlicht werden:

Gegeben sei eine Fallstudie, in der das Verhalten eines Agenten untersucht wird, der in eine für ihn unbekannte Umgebung gesetzt wird und in dieser Umgebung seine Bedürfnisse zu befriedigen versucht, um zu überleben. Die grundlegenden Verhaltensstrategien, die dem Agenten dabei zugeordnet werden, bestehen darin, seine Umwelt zu erforschen (*Exploration*) und seinen Nahrungsbedarf anhand der durch die Umgebung bereitgestellten Ressourcen zu decken (*Nahrungssuche*). In der Umgebung des Agenten existieren auch Gefahrenstellen. Betritt der

Agent eine derartige Gefahrenstelle, führt dies zu einer Schädigung seiner physischen Konstitution und löst einen Angstzustand aus. Ist die Angst des Agenten hoch, so geht er bei all seinen Aktionen vorsichtiger vor, um weitere Gefährdungen zu vermeiden. Nimmt dagegen der Angstzustand des Agenten einen geringen Wert an, so handelt er wagemutiger und nimmt in Kauf, daß er in Gefahrensituationen geraten kann.

```
VALUE SET VarWerte: { gering, hoch }

Wissensstand   ( VarWerte )      # kogn. Zustandsvariable
Hunger         ( VarWerte )      # phys. Zustandsvariable
Angst          ( VarWerte )      # emot. Zustandsvariable

IF ( Wissensstand = gering
     AND Hunger = gering
     AND Angst = gering )
DO
   Handlungsstrategie := Exploration ( wagemutig );
END

IF ( Wissensstand = gering
     AND Hunger = gering
     AND Angst = hoch )
DO
   Handlungsstrategie := Exploration ( vorsichtig );
END

IF ( Wissensstand = hoch
     AND Hunger = hoch
     AND Angst = gering )
DO
   Handlungsstrategie := Nahrungssuche ( wagemutig );
END

IF ( Wissensstand = hoch
     AND Hunger = hoch
     AND Angst = hoch )
DO
   Handlungsstrategie := Nahrungssuche ( vorsichtig );
END
```

Abbildung 5.2: Produktionen für die Verhaltenssteuerung eines Agenten unter Berücksichtigung interner Zustandsvariablen

Die Abbildung 5.2 zeigt einen Ausschnitt aus den Produktionen, in denen die Verhaltensstrategie des Agenten in Abhängigkeit seines internen Zustands festgelegt wird. Beachtet werden dabei die kognitive Zustandsvariable *Wissensstand*, die physische Zustandsvariable *Hunger*, sowie die emotionale Zustandsvariable *Angst*, die jeweils die Werte *gering* und *hoch* annehmen können. Auf eine zusätzliche Beachtung sozialer Zustandsvariablen wird an dieser Stelle der Einfachheit halber verzichtet. Im allgemeinen Fall ist es jedoch auch jederzeit möglich, soziale

Zustände im Zusammenhang mit der Verhaltensbeschreibung von Agenten zu berücksichtigen.

Unter Beachtung der ersten Produktion würde der Agent exploratives Verhalten vorziehen, da sein Wissen über die Umwelt noch sehr spärlich ist und sein Hunger ihn noch nicht zur Auswahl bedürfnisbefriedigender Verhaltensweisen zwingt. Zudem wird er wagemutig vorgehen, da seine Angst gering ist. Die zweite Produktion wird aktiviert, sofern die Zustandsvariable *Angst* den Wert *hoch* annimmt, und führt aufgrund des Angstzustandes zu vorsichtigerem Explorationsverhalten.

In der dritten und vierten Produktion überwiegt der Einfluß des hohen Wertes für die Zustandsvariable *Hunger* und der Agent gibt der Nahrungssuche den Vorzug vor explorativem Verhalten, um seinen Hunger zu stillen. Auch hier entscheidet die Zustandsvariable *Angst* darüber, ob der Agent bei seiner Nahrungssuche wagemutig oder vorsichtig vorgeht.

Dieses kleine Beispiel wurde ganz bewußt sehr einfach gewählt und soll lediglich den Grundsatz vor Augen führen, daß im Zusammenhang mit der Verhaltenssteuerung im allgemeinen Fall der gesamte interne Zustand von Agenten, der sich auf physische, emotionale, kognitive und soziale Aspekte erstreckt, von Bedeutung sein kann.

5.2 Der komponentenorientierte und hierarchische Modellaufbau

Der komponentenorientierte und hierarchische Modellaufbau (Schmidt, 1994) gehört heute zu den Standardtechnologien im Bereich der Modellbildung und Simulation. Er ermöglicht eine funktionale Dekomposition komplexer Modelle in eine Menge überschaubarer, interagierender Komponenten, wobei jede einzelne Komponente einen klar definierten Teil der Gesamtfunktionalität realisiert. Die Komponententechnologie ermöglicht eine klare und übersichtliche Strukturierung komplexer Modelle und trägt dazu bei, die Strukturähnlichkeit zwischen Modell und zugrundeliegendem System zu erhöhen. Komponentenorientierte Modelle weisen ein hohes Maß an Flexibilität im Hinblick auf Modifikationen und Erweiterungen der Modellbeschreibung auf.

Modelle, in denen menschliches Handeln, Entscheiden und Verhalten untersucht werden soll, zeichnen sich insbesondere dadurch aus, daß sie eine Vielfalt unterschiedlicher Funktionalitäten, zwischen denen verschiedenartige Wechselwirkungen bestehen können, vereinen. Gerade für die Konzeption eines Referenzmodells in diesem Anwendungsbereich erweist sich der komponentenorientierte Modellaufbau als nützliches Strukturierungsprinzip und bildet die Grundlage für eine flexible Modellkonzeption. So kann auf der Grundlage des komponentenorientierten Modellaufbaus ein Referenzmodell in einer derartigen Weise strukturiert werden, daß es mit geringem Aufwand an die Anforderungen indivi-

dueller Anwendungsfälle angepaßt werden kann. Zu derartigen Anpassungen gehören beispielsweise die Ergänzung der durch das Referenzmodell bereitgestellten Komponenten durch neue Komponenten, die anwendungsspezifische Funktionalitäten einbringen, sowie das Weglassen von Komponenten, die zwar im Rahmen des Referenzmodells vorgesehen, allerdings für die gegebene Fallstudie ohne Bedeutung sind. Derartige Anpassungen können auf der Grundlage eines komponentenorientierten Modellentwurfs sehr einfach durchgeführt werden, ohne größere Modifikationen des übergeordneten Modellkonzeptes erforderlich zu machen.

5.2.1 Modellaufbau mit Komponenten

Dem komponentenorientierten Modellaufbau folgend wird ein Modell aus einzelnen, zunächst voneinander unabhängigen Teilmodellen, den sogenannten Modellkomponenten, aufgebaut (Schmidt, 2000). Jede Modellkomponente ist für die Realisierung einer bestimmten, klar definierten Funktionalität verantwortlich und läuft grundsätzlich nebenläufig zu anderen Modellkomponenten ab.

Eine Modellkomponente besteht in der Regel aus einem *Namen*, der eine eindeutige Identifikation der Komponente ermöglicht, einem *Deklarationsteil*, sowie einer *Dynamikbeschreibung* (vgl. Abbildung 5.3).

Abbildung 5.3: Grundsätzlicher Aufbau einer Modellkomponente

Im Deklarationsabschnitt werden Attribute definiert, die die Modellkomponente charakterisieren. Man kann hier entsprechend des systemtheoretischen Ansatzes zwischen *Eingangsgrößen*, *Zustandsgrößen* und *Ausgangsgrößen* unterscheiden. Die Eingangsgrößen definieren die Inputschnittstelle einer Komponente und greifen nach Art von Sensoren Informationen aus der Umgebung der Komponente ab. Die Zustandsgrößen beschreiben, in welchem Zustand sich die Komponente zu einem gegebenen Zeitpunkt befindet. Als Ausgangsgrößen werden schließlich diejenigen Modellgrößen bezeichnet, die durch die Komponente nach außen zur Umgebung hin sichtbar gemacht werden. In manchen Simulationsspra-

chen, wie beispielsweise *Simplex-MDL* (Schmidt, 2000), wird die Unterscheidung zwischen Zustandsgrößen und Ausgangsgrößen fallengelassen und das sogenannte *Glass-Box-Konzept* verfolgt. Dabei werden alle Zustandsgrößen einer Komponente nach außen hin sichtbar gemacht und können von anderen Komponenten aus lesend zugegriffen werden. Dieses Konzept trägt vor allem der Anforderung Rechnung, kausale Abhängigkeiten, die häufig zwischen Modellkomponenten existieren, in einfacher und adäquater Weise beschreiben zu können.

Neben atomaren Attributen können Modellkomponenten auch zusammengesetzte Datenobjekte, die bei Schmidt (2000) auch als *mobile Komponenten* bezeichnet werden, verwalten. Mobile Komponenten ermöglichen die Zusammenfassung inhaltlich zusammengehöriger Attribute zu gekapselten Datenobjekten. Sie werden in Warteschlangen, die auch als *Aufenthaltsorte* oder *Locations* bezeichnet werden, aufbewahrt und können bei Bedarf zwischen verschiedenen Warteschlangen transferiert werden. Die Warteschlangen werden dabei als Bestandteile von Modellkomponenten definiert. Um auch einen komponentenübergreifenden Austausch mobiler Komponenten zu ermöglichen, kann auch auf die Locations einer Komponente von anderen Komponenten aus lesend zugegriffen werden. Der Zugriff auf Locations erfolgt dabei mit Hilfe von *Eingangslocations*. Mobile Komponenten können beispielsweise zur Modellierung des Nachrichtenaustausches zwischen Agenten oder zur Codierung parametrisierter Aktionen von Agenten eingesetzt werden.

Im Rahmen der Dynamikbeschreibung wird das dynamische Verhalten einer Komponente spezifiziert. Die Dynamikbeschreibung gibt dabei die zeitliche Veränderung der Werte für die Zustands- und Ausgangsgrößen, sowie die Modifikationen, die die mobilen Komponenten betreffen, an. Gemäß des systemtheoretischen Ansatzes sind im Rahmen der Dynamikbeschreibung von Modellkomponenten sowohl zeitkontinuierliche als auch zeitdiskrete Transformationen möglich (vgl. Kapitel 5.1.1).

5.2.2 Kommunikation zwischen Modellkomponenten

Zwischen den einzelnen Komponenten eines Modells bestehen in der Regel vielfältige Wechselwirkungen. Aus diesem Grund müssen Kommunikationsmechanismen vorgesehen werden, die einen Informationsaustausch zwischen Modellkomponenten ermöglichen. Das Grundprinzip des Informationsaustausches zwischen Modellkomponenten besteht darin, Eingangsgrößen und Eingangslocations von Komponenten mit Ausgangsgrößen und Locations anderer Komponenten, von denen Informationen bezogen werden sollen, zu verbinden. Durch die Verschaltung der Ausgangselemente von Komponenten mit Eingangselementen anderer Komponenten entsteht auch die Struktur von Modellen.

In bezug auf die Kommunikationsmechanismen, die für die Interaktion von Modellkomponenten zur Verfügung stehen, können *kausale Abhängigkeiten* von *diskreten Informationsflüssen* unterschieden werden (Schmidt, 1994).

5.2.2.1 Kausale Abhängigkeiten zwischen Modellkomponenten

Zwischen zwei Modellkomponenten besteht eine kausale Abhängigkeit, wenn das dynamische Verhalten der einen Komponente vom Zustand der anderen Komponente abhängig ist. Die kausale Abhängigkeit kann dabei dadurch modelliert werden, daß Eingangsgrößen der einen Komponente mit Ausgangsgrößen der anderen Komponente verschaltet werden. Durch die Verschaltung erhält die kausal abhängige Komponente lesenden Zugriff auf die Zustandsgrößen der anderen Komponente und kann daraufhin in entsprechender Weise ihren eigenen Zustand aktualisieren. Die Informationsübertragung zwischen den Komponenten erfolgt dabei zeitlos und unverzerrt, d. h. der abgegriffene Zustand kommt bei der lesenden Komponente ohne Zeitverzug und ohne Störung an.

Kausale Abhängigkeiten zwischen Komponenten können auch graphisch veranschaulicht werden. Das folgende Beispiel zeigt, welche graphische Darstellung für kausale Abhängigkeiten zwischen Modellkomponenten im Rahmen dieser Arbeit verwendet werden soll:

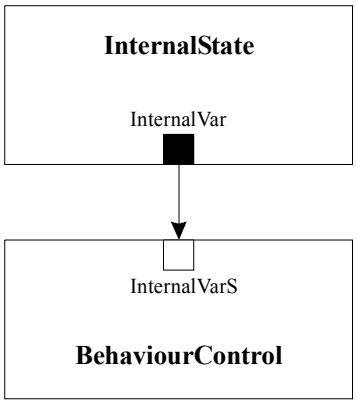

Abbildung 5.4: Graphische Repräsentation von kausalen Abhängigkeiten zwischen Modellkomponenten

Gegeben seien die beiden Modellkomponenten *InternalState* und *BehaviourControl*, die für die Speicherung des internen Zustandes, sowie für die Verhaltenssteuerung eines Agenten zuständig sind. Die Verhaltenssteuerung des Agenten sei dabei von seinem internen Zustand abhängig. Aus diesem Grund greift die Komponente *BehaviourControl* lesend auf das Attribut *InternalVar* zu, das in der Komponente *InternalState* berechnet wird. Der aktuelle Wert der Variable *InternalVar* wird innerhalb der Komponente *BehaviourControl* mit Hilfe der Eingangsvariable *InternalVarS* zugänglich gemacht.

In der graphischen Darstellung wird die Ausgangsgröße mit Hilfe eines ausgefüllten Rechtecks dargestellt, die Eingangsgröße mit Hilfe eines nicht ausgefüllten Rechtecks und die Verbindung mit Hilfe eines Wirkpfeiles, der von der Ausgangsgröße zur Eingangsgröße zeigt.

5.2.2.2 Diskrete Informationsflüsse zwischen Modellkomponenten

Diskrete Informationsflüsse zwischen Modellkomponenten können mit Hilfe von mobilen Komponenten modelliert werden. Die mobilen Komponenten beschreiben dabei Datenobjekte, die zu diskreten Zeitpunkten von einer Location innerhalb einer Ausgangskomponente auf eine Location innerhalb einer verbundenen Zielkomponente transferiert werden können. Auf diese Weise ermöglichen Informationsflußverbindungen den komponentenübergreifenden Austausch zusammengesetzter Datenobjekte.

Auch für diskrete Informationsflußverbindungen existiert eine graphische Repräsentation, die wiederum im Rahmen eines kleinen Beispiels vorgeführt werden soll:

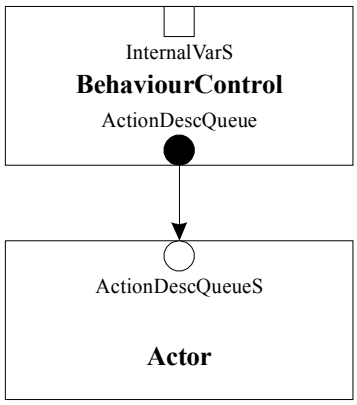

Abbildung 5.5: Graphische Repräsentation von diskreten Informationsflußverbindungen zwischen Modellkomponenten

Gegeben sei die Modellkomponente *BehaviourControl*, die bereits im vorhergehenden Beispiel eingeführt wurde, sowie eine Komponente *Actor*. Die Komponente *BehaviourControl* legt im Rahmen der Verhaltenssteuerung fest, welche Aktionen der Agent im Laufe der Zeit ausführen soll. Dazu werden mobile Komponenten des Typs *ActionDescription* generiert, die Beschreibungen der auszuführenden Aktionen enthalten, und in die Warteschlange *ActionDescQueue* einsortiert. Mit Hilfe der Eingangslocation *ActionDescQueueS*, die mit der Location *ActionDescQueue* der Komponente *BehaviourControl* verbunden wird, erhält die Komponente *Actor* lesenden Zugriff auf die Warteschlange *ActionDescQueue*. Damit kann die Komponente *Actor*, die für die Generierung von Aktionen des Agenten zuständig ist, die durch die Komponente *BehaviourControl* bereitgestellten Aktionsbeschreibungen abholen und diese im Laufe der Zeit in konkrete Aktionen des Agenten umsetzen.

In der graphischen Beschreibung sind für Locations ausgefüllte Kreise vorgesehen, für Eingangslocations nicht ausgefüllte Kreise und für die Verbindung zwischen einer Location und einer Eingangslocation Wirkpfeile, die an der Location ansetzen und in Richtung der Eingangslocation zeigen.

5.2.3 Hierarchischer Modellaufbau

Modellkomponenten können mit Hilfe von Strukturkomponenten auf der nächst höheren Abstraktionsebene zu größeren Blöcken zusammengefaßt und miteinander verschaltet werden. Diese Strukturkomponenten können ihrerseits wiederum als eigenständige Modellkomponenten aufgefaßt werden und ebenfalls als Unterkomponenten in einem zusammengesetzten Modell auftreten. Durch eine sukzessive Anwendung dieses Prinzips erhält man Modelle, die sich über mehrere Hierarchieebenen erstrecken, sowie aus einer beliebigen Anzahl an Subkomponenten aufgebaut sein können.

Um eine einfache Handhabung hierarchisch strukturierter Komponenten zu ermöglichen, die keinen Einblick in den internen Aufbau der Komponenten erfordert, müssen zusammengesetzte Komponenten mit eigenen Eingangsgrößen, Eingangslocations, Zustandsgrößen, sowie Locations ausgestattet werden. Die Eingangselemente der übergeordneten Komponente werden dabei an Eingangselemente der untergeordneten Komponenten durchgereicht. Ausgangsgrößen und Locations von untergeordneten Komponenten werden andererseits auch an die Zustandsgrößen und Locations der übergeordneten Komponente gekoppelt. Diese neuen Konzepte ermöglichen zum einen das Durchreichen von Eingangsinformationen an untergeordnete Komponenten, d. h. an diejenigen Stellen, an denen die Informationen tatsächlich benötigt und verarbeitet werden, und zum anderen die Bereitstellung interner Modellelemente auf höheren Abstraktionsebenen.

Das Konzept des hierarchischen Modellaufbaus, sowie das Durchreichen von Informationen über eine Hierarchieebene hinweg sollen abschließend an einem kleinen Beispiel verdeutlicht werden, in dem die in den letzten beiden Beispielen eingeführten Modellkomponenten zu einem kompletten, kleinen Agenten vervollständigt werden:

Gegeben seien also die in den Abbildungen 5.4 und 5.5 gezeigten Komponenten *InternalState*, *BehaviourControl* und *Actor*, sowie die dort spezifizierten Verbindungen zwischen den Komponenten. Zusätzlich soll eine Komponente *Sensor* eingeführt werden, die für die Verarbeitung der sensorischen Informationen des Agenten verantwortlich ist. Die Komponente Sensor greift mit Hilfe der Eingangslocation *EnvInfoQueue* auf Umweltinformationen zu, die für den Agenten von Bedeutung sind, und stellt diese in Form von Perzepten für die weitere Verarbeitung durch die Komponente *InternalState* zur Verfügung. Die Komponente *Actor* soll darüber hinaus um die Warteschlange *ActionQueue* erweitert werden, in der die konkreten Aktionen des Agenten abgelegt werden.

Die Aggregation der vier Subkomponenten, sowie die Spezifikation der Verbindungen zwischen den Subkomponenten erfolgt in der übergeordneten Komponente *Agent*. Die Komponente Agent verfügt zudem über eine Eingangslocation *EnvInfoQueue*, die mit dem gleichnamigen Element in der Komponente *Sensor* verbunden ist, um die Inputs des Agenten an diejenige Subkomponente weiterzuleiten, in der sie verarbeitet werden. Auf der Outputseite des Agenten wird über eine Ausgangsverbindung die *ActionQueue* der Subkomponente *Actor* mit der zu-

gehörigen Location der Komponente *Agent* verbunden, um die Aktionen des Agenten an die nächst höhere Hierarchieebene durchzureichen.

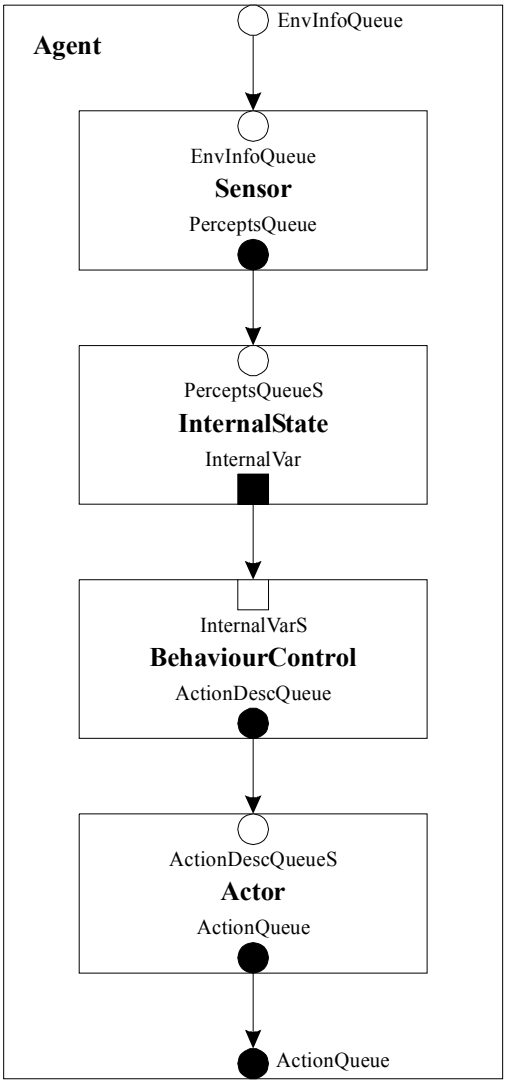

Abbildung 5.6.: Komponentenorientierte und hierarchische Strukturierung eines einfachen Agenten

Die Komponente *Agent* stellt sich nach außen hin als eine Komponente dar, die über eine Eingangslocation und eine Location verfügt. Intern ist die Komponente jedoch aus vier miteinander verschalteten Subkomponenten aufgebaut, die nach außen hin nicht sichtbar sind. Die Komponente *Agent* kann nun ihrerseits auf der nächst höheren Abstraktionsebene mit anderen Komponenten zu einem komplexeren Modell verschaltet werden, wobei keine Kenntnisse über die interne Strukturierung der Komponente *Agent* nötig sind.

Eingangsverbindungen, die Informationen von der übergeordneten an die untergeordnete Hierarchiestufe weiterleiten, werden graphisch durch die Verknüp-

fung nicht ausgefüllter Elemente repräsentiert. Ausgangsverbindungen werden durch miteinander verbundene, ausgefüllte Graphikelemente dargestellt.

6 DAS REFERENZMODELL *PECS*

Das Referenzmodell *PECS* (Physis, Emotion, Cognition, Social Characteristics) stellt ein domänenunabhängiges Konstruktionsschema für agentenbasierte Simulationsmodelle zur Verfügung, in denen menschliches Handeln, Entscheiden und Verhalten von ausschlaggebender Bedeutung sind. Es gibt einen konzeptionellen Modellierungsrahmen vor, der unabhängig von speziellen Anwendungsfeldern konzipiert ist. Die Anpassung an die Erfordernisse individueller Modelluntersuchungen erfolgt durch Ausfüllen der Leerstellen, die in der Modellarchitektur vorgesehen sind. So besteht für den Anwender des Referenzmodells grundsätzlich die Möglichkeit, die vorgegebenen Komponenten mit einer beliebigen Anzahl frei definierbarer Modellgrößen, sowie mit entsprechenden Zustandsüberführungs- und Outputfunktionen auszustatten. Dieses Vorgehen ermöglicht die Beschreibung agentenbasierter Modelle, die den unterschiedlichsten Anwendungsgebieten angehören, jedoch alle dieselbe Tiefenstruktur aufweisen.

In den folgenden Abschnitten sollen nun der Aufbau und die grundsätzliche Funktionsweise des Referenzmodells *PECS* im Überblick dargestellt werden.

6.1 Die Grundstruktur des Referenzmodells *PECS*

Die Grundstruktur des Referenzmodells *PECS* orientiert sich an den elementaren Anforderungen, die an agentenbasierte Modelle gestellt werden, wenn auf deren Grundlage menschliches Handeln, Entscheiden und Verhalten untersucht werden sollen (s. Kapitel 4 und Urban, 2000).

Um einen klaren und übersichtlichen Aufbau des Referenzmodells zu erreichen, wird als übergeordnetes Strukturierungsprinzip der komponentenorientierte und hierarchische Modellaufbau (vgl. Kapitel 5.2) gewählt. Das Referenzmodell setzt sich also aus einer Menge zunächst voneinander unabhängiger und nebenläufig arbeitender Komponenten zusammen, die auf der Grundlage von kausalen Abhängigkeiten und diskreten Informationsflüssen miteinander interagieren können. Darüber hinaus werden die einzelnen Komponenten des Referenzmodells über mehrere Ebenen hinweg zu komplexeren Teilmodellen aggregiert, so daß ein hierarchisch strukturiertes Modell entsteht.

Das Referenzmodell *PECS* umfaßt dabei insgesamt drei Hierarchieebenen. Auf der höchsten Hierarchieebene ist die Komponente *PECS_World* angesiedelt, die das Gesamtmodell beinhaltet. Auf der mittleren Hierarchieebene befinden sich gemäß der in Kapitel 4 zusammengetragenen, inhaltlichen Anforderungen drei Arten von Komponenten: die Komponente *Agent*, die Komponente *Connector*, sowie die Komponente *Environment.* Auf der untersten Ebene sind schließlich diejenigen Komponenten angesiedelt, die die Feinstruktur der Agenten beschreiben. Sie sind in der Abbildung 6.1 aus Gründen der Übersichtlichkeit zunächst noch nicht aufgeführt und werden im folgenden Abschnitt im Zusammenhang mit der Beschreibung der *PECS*-Agentenarchitektur nachgereicht.

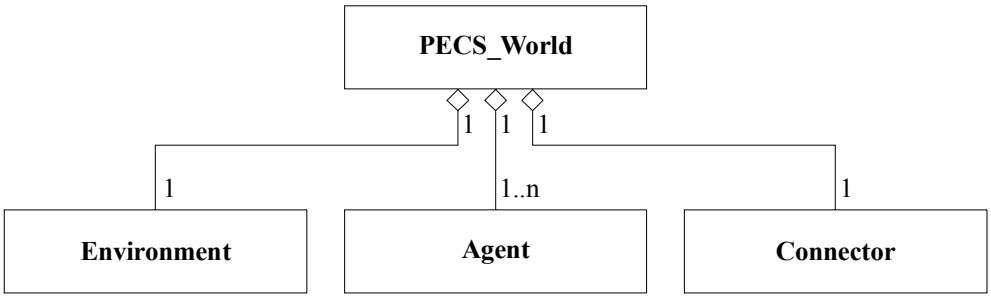

Abbildung 6.1: Die Komponentenstruktur des Referenzmodells PECS

Ein nach *PECS* strukturiertes, agentenbasiertes Modell besteht also aus einer beliebigen Anzahl an Agenten, einer Komponente *Environment* und einer Komponente *Connector*.

Die Komponente *Agent* übernimmt die Aufgabe, menschliche Akteure mit ihren für die betrachtete Fragestellung relevanten Eigenschaften und Verhaltensweisen im Modell nachzubilden. Die Komponente *Environment* dient der Modellierung der Umwelt der Akteure. Sie stellt also im Modell das externe Bezugssystem zur Verfügung, an dem das Verhalten der Agenten ausgerichtet wird. Die Komponente *Connector* realisiert eine zentral angelegte Kommunikationsinfrastruktur, die den Informationsaustausch zwischen den Agenten ermöglicht.

Die Abbildung 6.2 vermittelt einen ersten, groben Eindruck über das Zusammenspiel der einzelnen Komponenten und zeigt auf, daß die Agenten in wechselseitigen Beziehungen mit der Komponente *Environment* und der Komponente *Connector* stehen.

Die Agenten sind in der Lage, mit Hilfe von externen Aktionen den Zustand ihrer Umgebung zu verändern. Die Aktionen werden dabei auf der Grundlage von diskreten Informationspaketen (*Actions*) von den Agenten an die Komponente *Environment* übermittelt. Im Gegenzug liefert auch die Umwelt Informationen, die für die Agenten von Bedeutung sind und ihre Verhaltensweisen beeinflussen. Um die kausalen Abhängigkeiten, die zwischen dem Zustand der Umwelt und dem Verhalten der Agenten existieren, in adäquater Weise im Modell abbilden zu können, werden zwischen der Komponente *Environment* und den Agenten Verbindungen eingerichtet, die es den Agenten ermöglichen, auf entsprechende Zu-

stände innerhalb der Komponente *Environment* lesend zuzugreifen. Diese Verbindungen werden in Abbildung 6.2 zusammenfassend durch den Pfeil *EnvInfo* dargestellt.

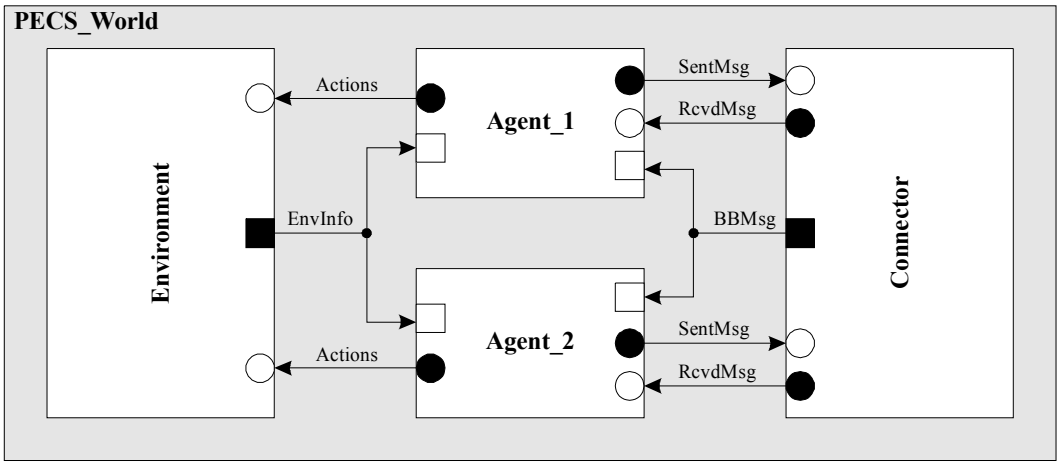

Abbildung 6.2: Grundlegende Interaktionen zwischen den Komponenten des Referenzmodells PECS

Der Informationsaustausch zwischen den Agenten erfolgt im Referenzmodell *PECS* auf der Grundlage der Komponente *Connector*. Als Kommunikationsformen stehen sowohl direkte Kommunikation zwischen den Agenten, als auch indirekte Kommunikation, die auf der Blackboard-Metapher (Erman, Hayes-Roth, Lesser & Reddy, 1980) basiert, zur Verfügung. Agenten schicken ihre Nachrichten in Gestalt von diskreten Informationspaketen über die Verbindung *SentMsg* an die Komponente *Connector*, die die Nachrichten abhängig von ihrem Typ entweder in einer für die Agenten zugreifbaren Datenstruktur zur Verfügung stellt oder direkt an diejenigen Agenten weiterleitet, die als Empfänger in der Nachricht angegeben sind. Die Verteilung der direkt adressierten Nachrichten wird dabei über die Verbindung *RcvdMsg* abgewickelt. Auf das in der Komponente *Connector* verwaltete Blackboard erhalten die Agenten mit Hilfe der Verbindung *BBMsg* lesenden Zugriff.

Die folgenden Abschnitte beschäftigen sich nun mit den Komponenten *Agent*, *Connector* und *Environment*. Dabei sollen im wesentlichen der Aufbau, die grundsätzliche Funktionsweise, sowie die Interaktion zwischen den Komponenten ins Auge gefaßt werden.

6.2 Die Architektur von *PECS*-Agenten

Wie im vorhergehenden Abschnitt bereits erwähnt, besteht die Aufgabe der Komponente *Agent* darin, menschliche Akteure mit ihren für die jeweilige Fallstudie relevanten Eigenschaften und Verhaltensweisen im Modell abzubilden. Um dies

in adäquater Weise bewerkstelligen zu können, müssen innerhalb der Komponente *Agent* eine Reihe unterschiedlicher Funktionalitäten miteinander in Einklang gebracht werden. Dazu zählen die Verarbeitung sensorischer Informationen einschließlich der Realisierung einfacher Wahrnehmungsfunktionen, die Modellierung physischer, emotionaler, kognitiver und sozialer Eigenschaften unter Berücksichtigung ihrer dynamischen Änderungen und schließlich die Nachbildung verschiedener Grundmechanismen der menschlichen Verhaltenssteuerung (s. Kapitel 4.2).

Diese Vielfalt unterschiedlicher Anforderungen, sowie die Möglichkeit, dadurch die Flexibilität der Architektur hinsichtlich struktureller Anpassungen zu erhöhen, läßt die Grundidee, auch die Komponente *Agent* modular aufzubauen, als sehr sinnvoll erscheinen. Die Komponente *Agent* setzt sich also aus den insgesamt acht Subkomponenten *Sensor*, *Perception*, *Physis*, *Emotion*, *Cognition*, *Social Characteristics*, *Behaviour* und *Actor* zusammen.

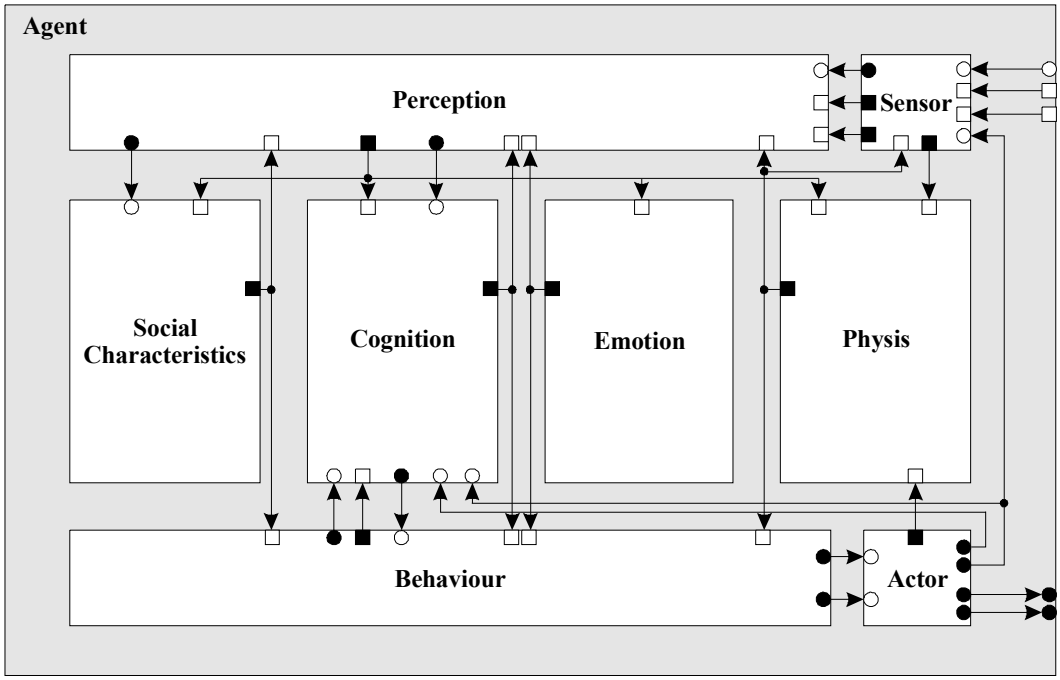

Abbildung 6.3: Die Grundstruktur von PECS-Agenten

Diese acht Komponenten können gedanklich drei unterschiedlichen Kategorien zugeordnet werden. Die erste Kategorie umfaßt die Komponenten *Sensor* und *Perception* und befaßt sich mit der Verarbeitung sensorischer Informationen, die der Agent aus seiner Umgebung aufnimmt. In die zweite Kategorie fallen die Komponenten *Physis*, *Emotion*, *Cognition* und *Social Characteristics*. Diese Komponenten beschreiben den internen Zustand des Agenten, sowie die zugehörigen Zustandsüberführungsfunktionen. Der letzten Kategorie gehören schließlich die Komponenten *Behaviour* und *Actor* an, die für die Modellierung der Verhaltenssteuerung und die Umsetzung der Aktionen des Agenten verantwortlich sind.

Auf die in Abbildung 6.3 dargestellten Verbindungen zwischen den einzelnen Komponenten der Agentenarchitektur soll an dieser Stelle noch nicht eingegangen werden. Entsprechende Ausführungen dazu finden sich in den nachfolgenden Abschnitten bei der Beschreibung der verschiedenen Subkomponenten.

6.2.1 Die Komponente *Sensor*

Wie in Kapitel 4.2.1 erörtert wurde, müssen Agenten in der Lage sein, Informationen, die in ihrer Umgebung bereitgestellt werden, aufzunehmen und zu verarbeiten, so daß ihr Verhalten in entsprechender Weise an die Gegebenheiten in der Umgebung angepaßt werden kann. Die Komponente *Sensor* realisiert in diesem Zusammenhang die Input-Schnittstelle von *PECS*-Agenten. Im Rahmen dieser Komponente werden sensorische Informationen, die für die Verhaltenssteuerung des Agenten von Bedeutung sind, registriert und für die Weiterbearbeitung in anderen Komponenten der Architektur zur Verfügung gestellt.

Komponente Sensor	**Inputs von Komponente(n)** • Agent (Environment, Connector) • Actor • Physis
Aufgabe(n) • Empfang sensorischer Informationen • Bereitstellung dieser Informationen für die interne Weiterverarbeitung	**Outputs an Komponente(n)** • Perception • Physis

Abbildung 6.4: Übersicht über die Komponente Sensor

6.2.1.1 Inputs der Komponente *Sensor*

Um die grundlegende Aufgabe, sensorische Informationen, die aus der Umgebung des Agenten stammen, zu sammeln und für weitere Verarbeitungsschritte in anderen Komponenten bereitzustellen, erfüllen zu können, muß die Komponente *Sensor* mit Verbindungen nach außen ausgestattet werden. Diese Verbindungen ermöglichen einerseits den Zugriff auf verhaltensrelevante Zustände in der Umwelt des Agenten und andererseits auch den Empfang von Nachrichten.

Die Komponente *Environment* (s. Kapitel 6.4) liefert externe Rahmen- und Randbedingungen, an denen sich das Verhalten der Agenten orientiert. Mit Hilfe der Eingangsverbindung *EnvInfoS* (s. Abbildung 6.5) erhält die Komponente *Sensor* lesenden Zugriff auf Zustände, die in der Komponente *Environment* berechnet werden.

Die Komponente *Connector* (s. Kapitel 6.3) realisiert eine zentrale Infrastruktur für den Nachrichtenaustausch zwischen Agenten. Nachrichten, die direkt an

einen einzelnen oder eine definierte Menge an Agenten adressiert sind, werden den Agenten durch die Komponente *Connector* mit Hilfe von mobilen Komponenten zugestellt. Die Nachrichten werden dabei in Warteschlangen, die individuelle Postfächer darstellen, innerhalb der Komponente *Connector* aufbewahrt und von dort durch die Agenten abgeholt. Den Zugriff auf ihre Postfächer für eingehende Nachrichten erhalten die Agenten mit Hilfe der Verbindung *RcvdMsgQS*. Ebenso müssen für die Agenten Informationen zugänglich gemacht werden, die indirekt auf der Grundlage gemeinsam nutzbarer Datenbereiche (Blackboards) kommuniziert werden. Aus diesem Grund wird für die Komponente *Sensor* eine weitere Verbindung *BBMsgS* nach außen eingerichtet, die einen lesenden Zugriff auf Blackboard-Strukturen innerhalb der Komponente *Connector* ermöglicht.

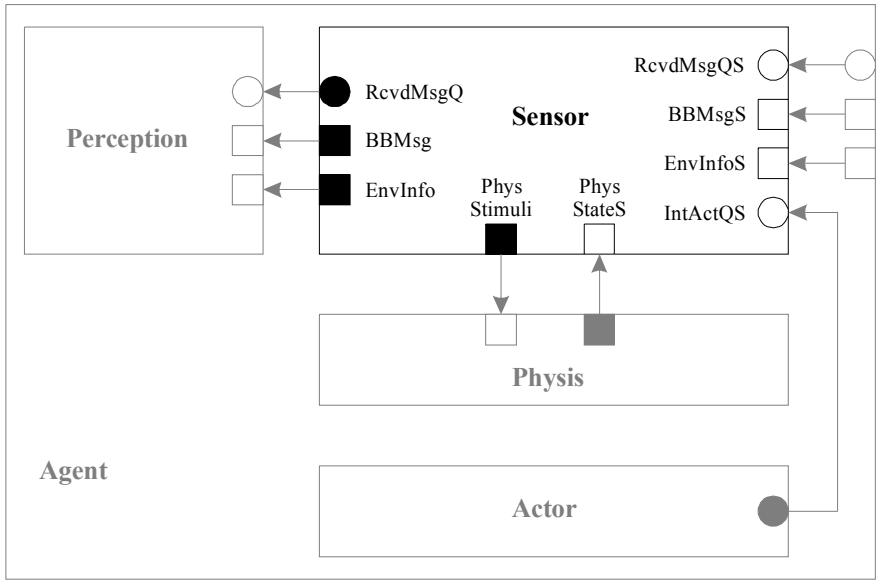

Abbildung 6.5: Grundsätzlicher Aufbau der Komponente Sensor

Anzumerken ist an dieser Stelle, daß unter Berücksichtigung des zugrunde gelegten Hierarchiekonzeptes die Komponenten *Sensor* und *Environment* bzw. *Sensor* und *Connector* nicht direkt miteinander verbunden sind. Die eigentlichen Verbindungen werden auf der nächst höheren Hierarchieebene zwischen der Komponente *Agent* und den Komponenten *Environment* bzw. *Connector* gezogen. Bei den oben aufgeführten Verbindungen *RcvdMsgQS*, *BBMsgS* und *EnvInfoS* handelt es sich also um Eingangsverbindungen, mit deren Hilfe Informationen von der Komponente *Agent* an die untergeordnete Komponente *Sensor* weitergereicht werden.

6.2.1.2 Die grundlegende Arbeitsweise der Komponente *Sensor*

Die Komponente *Sensor* empfängt sensorische Inputs aus der Umgebung des Agenten und stellt diese Inputs in Form von geeigneten internen Repräsentationen

für andere Komponenten der Agentenarchitektur zur Verfügung. Dabei bietet es sich an, Nachrichten, die als diskrete Informationspakete bei der Komponente *Sensor* eintreffen, auch als diskrete Informationspakete abzuspeichern. Blackboard-Informationen sowie Umweltzustände, die mit Hilfe von kausalen Abhängigkeiten übertragen werden, werden vorwiegend in interne Zustände der Komponente *Sensor* transformiert. In Anbetracht der weiteren Verarbeitungsschritte kann es sich jedoch auch als sinnvoll erweisen, einzelne, empfangene Zustände als Schnappschuß der aktuell vorliegenden Situation in zusammengesetzten Informationspaketen zu kapseln und in geschlossener Form weiterzuleiten. Derartig detaillierte Entwurfsentscheidungen können allerdings erst im Kontext des jeweiligen, zu realisierenden Modells getroffen werden und sollen aus diesem Grund nicht starr durch das Referenzmodell vorgegeben werden.

In Abhängigkeit von ihrer jeweiligen Bedeutung werden die eingehenden Informationen in einer derartigen Weise innerhalb der Komponente *Sensor* abgelegt, daß sie von den für ihre Weiterverarbeitung zuständigen Komponenten abgerufen werden können. Direkt adressierte Nachrichten, sowie Blackboard-Informationen werden dabei für die Komponente *Perception* zugänglich gemacht, die für die Abbildung einfacher Wahrnehmungsprozesse verantwortlich ist. In Abbildung 6.5 werden die zugehörigen Outputs der Komponente Sensor durch die Location *RcvdMsgQ* bzw. die Ausgangsgröße *BBMsg* symbolisiert, die mit entsprechenden Eingangselementen der Komponente *Perception* verbunden sind.

Umwelteinflüsse, d. h. also Größen, die innerhalb der Komponente *Environment* berechnet und bereitgestellt werden, werden nach ihrem Empfang in der Komponente *Sensor* entweder durch die Komponente *Perception* oder durch die Komponente *Physis* weiterverarbeitet. Die Verteilung erfolgt auch hier gemäß der Bedeutung der Informationen. Informationen, die visuelle Eindrücke modellieren, werden beispielsweise Wahrnehmungsprozessen innerhalb der Komponente *Perception* zugeführt. Andererseits werden Größen, die physikalische Umgebungsbedingungen wie beispielsweise Temperaturen oder Luftqualität modellieren, an die Komponente *Physis* weitergereicht, weil im Rahmen dieser Komponente die physischen Eigenschaften eines Agenten sowie die zugehörigen Zustandstransformationen berechnet werden. Die Weitergabe sensorischer Umweltinformationen an die Komponente *Perception* bzw. *Physis* ist in Abbildung 6.5 durch die Ausgangsgrößen *EnvInfo* bzw. *PhysStimuli* angedeutet.

Modifikation sensorischer Informationen

Nicht immer ist es erwünscht, daß sensorische Informationen in unveränderter Form an die verarbeitenden Komponenten weitergeleitet werden. So könnte es beispielsweise von Interesse sein, Einflüsse in einen Modellierungsansatz einzubeziehen, die zu qualitativen oder quantitativen Modifikationen der empfangenen Informationen vor deren Weitergabe führen. Um derartige Unterschiede in bezug auf die bereitgestellten, sensorischen Informationen berücksichtigen zu können,

muß der interne Zustand der Komponente *Sensor* um Einflüsse erweitert werden, die eine Modellierung derartiger Phänomene ermöglichen. Der interne Zustand der Komponente *Sensor* kann dabei mit internen Zuständen bzw. Fähigkeiten des Agenten korrelieren. Diese Abhängigkeit wird in Abbildung 6.5 vereinfachend durch die Eingangsgröße *PhysStateS* symbolisiert, die die Komponente *Sensor* mit dem internen Zustand der Komponente *Physis* verbindet. Das nachfolgende, einfache Beispiel soll diesen Sachverhalt verdeutlichen:

Ein Agent wird in eine zweidimensionale Umwelt, modelliert als 2D-Gitter, gesetzt und erhält die Möglichkeit, auf den Zustand seiner Umwelt innerhalb einer gewissen, vorgegebenen Sichtweite lesend zuzugreifen (s. Abbildung 6.6). Die Sichtweite des Agenten wird als Attribut in der Komponente *Physis* verwaltet und als Eingangsgröße in der Komponente *Sensor* bereitgestellt. Zusätzlich erhält die Komponente *Sensor* als Eingang den Zustand der gesamten Umwelt, d. h. des gesamten 2D-Gitters. Für die weitere Informationsverarbeitung innerhalb des Agenten soll nun aber nicht der Zustand des gesamten Gitters bereitgestellt werden, sondern nur derjenige Bereich, der für den Agenten in der gegebenen Situation tatsächlich sichtbar ist. Dieser sichtbare Bereich, der neben der Sichtweite des Agenten auch von seiner aktuellen räumlichen Position innerhalb der Umwelt abhängt, wird im Rahmen der Dynamik der Komponente *Sensor* aus dem ursprünglich bereitgestellten Gitter ausgeschnitten und in dieser reduzierten Form an die Komponente *Perception* zur Verarbeitung weitergereicht (s Abbildung 6.6). Die in der Abbildung gewählte Sichtweite beträgt 1 Feld in alle Himmelsrichtungen. Der grau hinterlegte Bereich im linken Gitter markiert diejenigen Zellen, über die der Agent aufgrund seiner beschränkten Sichtweite nun keine Informationen mehr besitzt. Das schwarz markierte Feld bezeichnet die aktuelle Position des Agenten.

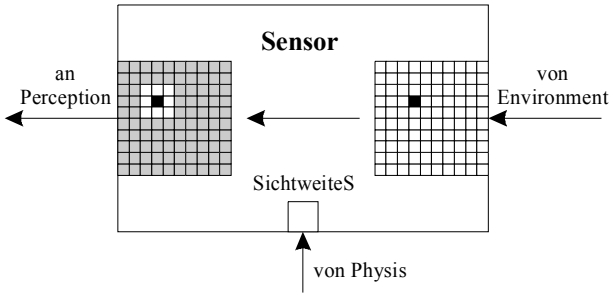

Abbildung 6.6: Einfluß interner Zustände der Komponente Sensor auf die Verarbeitung externer Informationen

Aktivierung der Komponente Sensor

Die Verarbeitung sensorischer Informationen kann innerhalb der Komponente *Sensor* auf verschiedene Weisen angestoßen werden. Eine Unterscheidung kann hier im wesentlichen nach der Art der eintreffenden Informationen erfolgen.

Nachrichten, die vorwiegend auf der Grundlage von diskreten Informationspaketen modelliert werden, werden durch die Komponente *Connector* unmittelbar

nach ihrem Eintreffen im Postfach des jeweiligen Agenten hinterlegt. Jeder Agent kann über eine Eingangslocation in der Komponente *Sensor* auf das Postfach zugreifen und ereignisgesteuert die an ihn adressierten Nachrichten abholen. Die Abbildung 6.7 zeigt einen Pseudocode-Ausschnitt, der dafür zuständig ist, die vorliegenden Nachrichten unmittelbar nach ihrem Eintreffen im Postfach des Agenten an die Komponente *Perception* weiterzuleiten.

```
WHENEVER ContainsElements ( RcvdMsgQS )
DO
   FOREACH Msg ∈ RcvdMsgQS
   DO
         Move ( Msg, RcvdMsgQ );
   END
END
```

Abbildung 6.7: Pseudocode für die Verarbeitung eingetroffener Nachrichten

Die erste Zeile des in Abbildung 6.7 angegebenen Pseudocodes spezifiziert die Ausführungsbedingung. Sobald in der Warteschlange *RcvdMsgQS* mindestens eine Nachricht vorliegt, wird der folgende Codeabschnitt aktiviert. Dadurch werden alle Nachrichten, die sich in der Warteschlange befinden, aus der Eingangslocation entfernt und in die Warteschlange *RcvdMsgQ*, die von der Komponente *Perception* aus zugreifbar ist, einsortiert.

Die Verarbeitung von Blackboard-Informationen sowie von Umwelt-Informationen kann in zweierlei Weise angestoßen werden. Zum einen können die anliegenden Inputs permanent an die konsumierenden Komponenten weitergeleitet werden. Diese Modellierung kommt einem andauernden Beobachten dieser externen Zustände gleich. Zum anderen kann die Komponente *Sensor* im Rahmen der Verhaltenssteuerung des Agenten aktiviert werden. Mit Hilfe einer internen Aktion, die durch die Komponente *Actor* an die Komponente *Sensor* geschickt wird, kann die Komponente *Sensor* beauftragt werden, bestimmte, durch die interne Aktion spezifizierte Informationen aus der Umgebung des Agenten abzurufen. Auch hier erfolgt die Aktivierung ereignisgesteuert, d. h. die Komponente *Sensor* startet entsprechende Aktivitäten, sobald eine derartige interne Aktion in der Warteschlange *IntActQS* (s. Abbildung 6.5) angekommen ist.

6.2.2 Die Komponente *Perception*

Die Komponente *Perception* dient der Modellierung von Wahrnehmungsfunktionen (vgl. Kapitel 4.2.1). Informationen über den Zustand der Umgebung sowie den internen Zustand des Agenten werden in dieser Komponente aufgenommen, in interne Repräsentationen (Perzepte) transformiert und an diejenigen Komponenten weitergeleitet, die für deren weitere Verarbeitung zuständig sind. Für die Erzeugung von Perzepten können beispielsweise komplexere Verfahren der Bild- oder Sprachverarbeitung (vgl. Dreschler-Fischer, 1995; Busemann, 1995) einge-

setzt werden, die auf der Ebene physikalischer Reize ansetzen. Im Rahmen von agentenbasierten Modellen, in denen menschliches Handeln, Entscheiden und Verhalten untersucht werden soll, spielen derartig aufwendige Verfahren allerdings eine eher untergeordnete Rolle. Wahrnehmungsprozesse werden in diesem Zusammenhang in der Regel auf einer höheren Abstraktionsebene betrachtet und vorwiegend dazu eingesetzt, um bereits symbolisch kodierte Informationen im Sinne einer Informationsselektion zu filtern und vor ihrer Weitergabe zu modifizieren.

Komponente Perception	**Inputs von Komponente(n)** • Sensor • Physis • Emotion • Cognition • Social Characteristics
Aufgabe(n) • Transformation externer und interner Informationen und Zustände in interne Repräsentationen (Perzepte) • Bereitstellung und Weiterleitung der Perzepte an die für deren Weiterverarbeitung zuständigen Komponenten	**Outputs an Komponente(n)** • Physis • Emotion • Cognition • Social Characteristics

Abbildung 6.8: Übersicht über die Komponente Perception

6.2.2.1 Inputs der Komponente *Perception*

Agenten, die menschliches Verhalten im Modell nachbilden, müssen mit der Möglichkeit ausgestattet werden, Informationen über den externen Zustand der Umgebung sowie den eigenen, internen Zustand zu sammeln, zu speichern und zu verarbeiten.

Umgebungsinformationen werden bei *PECS*-Agenten durch die Komponente *Sensor* aufgenommen und entweder als diskrete Informationspakete oder kontinuierliche Zustände an die Komponente *Perception* weitergeleitet. Zu den externen Informationen zählen, wie bereits in Kapitel 6.2.1 erläutert wurde, direkt adressierte Nachrichten, Blackboard-Informationen, die an zentraler Stelle für die Agenten zur Abholung bereitgestellt werden, und auch Zustände der Umwelt der Agenten. Um auf diese Informationen, die durch die Komponente *Sensor* gesammelt werden, in der Komponente *Perception* zugreifen zu können, werden in der Komponente *Perception* die Eingangselemente *RcvdMsgQS*, *BBMsgS* und *EnvInfoS* (vgl. Abbildung 6.9) eingerichtet und mit den korrespondierenden Elementen der Komponente *Sensor* verbunden. Die Eingangslocation *RcvdMsgQS* ermöglicht dabei den Zugriff auf direkt adressierte Nachrichten, die in einer Warteschlange in der Komponente *Sensor* abgelegt werden. Mit Hilfe der Elemente *BBMsgS* und *EnvInfoS* können Blackboard-Informationen sowie aktuelle Zustände der Umwelt von der Komponente *Sensor* abgegriffen werden. Durch das Zu-

sammenspiel der Komponenten *Sensor* und *Perception* wird also sichergestellt, daß sensorische Informationen aus der Umgebung des Agenten entsprechenden internen Wahrnehmungsprozessen zugeführt werden.

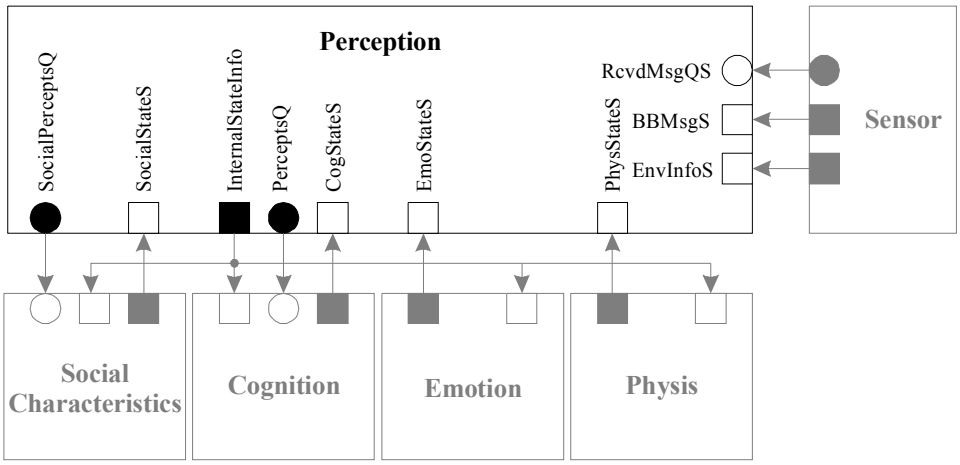

Abbildung 6.9: Grundsätzlicher Aufbau der Komponente Perception

Menschen sind, wenn auch in Grenzen, in der Lage, neben externen Gegebenheiten auch ihren eigenen, internen Zustand wahrzunehmen, auf der Grundlage dieser Wahrnehmungen eine Vorstellung von ihrem augenblicklichen Zustand zu entwickeln und daraufhin ihre Verhaltensweisen an dieser Vorstellung auszurichten. Auch Agenten, die menschliches Verhalten modellieren, können mit vergleichbaren Fähigkeiten ausgestattet werden. Dazu bedarf es einer Erweiterung der Architektur, die die Möglichkeit bietet, neben externen, sensorischen Informationen auch interne Zustände als Inputs für Wahrnehmungsprozesse zu berücksichtigen. Im Rahmen dieser Wahrnehmungsprozesse werden dann in Analogie zu externen Informationen auch für interne Informationen entsprechende Perzepte gebildet und für die Weiterverarbeitung in anderen Komponenten zur Verfügung gestellt.

Bei *PECS*-Agenten wird der interne Zustand durch die Komponenten *Physis*, *Emotion*, *Cognition* und *Social Characteristics* verwaltet. Um Wahrnehmungsprozesse zu modellieren, die auf dem internen Zustand von *PECS*-Agenten beruhen, wird zunächst einmal der interne Zustand des Agenten für die Komponente *Perception* zugänglich gemacht. Die Komponente *Perception* wird also um vier Eingangselemente erweitert, die einen lesenden Zugriff auf den internen Zustand des Agenten ermöglichen. Die Eingangsgröße *PhysStateS* stellt dabei eine Verbindung zum physischen Zustand des Agenten her, die Eingangsgröße *EmoStateS* zum emotionalen Zustand, die Eingangsgröße *CogStateS* zum kognitiven Zustand und die Eingangsgröße *SocialStateS* zum sozialen Zustand des Agenten (vgl. Abbildung 6.9). Auf der Grundlage dieser Eingänge der Komponente *Perception* können nun Wahrnehmungsprozesse modelliert werden, die sich sowohl auf externe Informationen, als auch auf interne Zustände des Agenten beziehen.

6.2.2.2 Die grundlegende Arbeitsweise der Komponente *Perception*

Die grundlegende Aufgabe der Komponente *Perception* besteht in der Bereitstellung externer Informationen wie auch interner Zustände für die Weiterverarbeitung innerhalb der Komponenten *Cognition* und *Social Characteristics*.

Die Weiterleitung eingehender Informationen

In bezug auf die zu verarbeitenden Eingangsinformationen wurde bereits bei der Beschreibung der Komponente *Sensor* (s. Kapitel 6.2.1) zwischen diskreten Informationspaketen und kontinuierlichen Signalen unterschieden. Die Weiterleitung diskreter Informationspakete erfolgt innerhalb der Komponente *Perception* ereignisorientiert, d. h. Nachrichten werden weitertransportiert, sobald sie in der Eingangswarteschlange *RcvdMsgQS* der Komponente *Perception* zur Verfügung stehen. Kontinuierliche Signale können gemäß den Anforderungen der jeweiligen Fallstudie entweder zu diskreten Informationspaketen zusammengefaßt werden und somit einen Schnappschuß der vorliegenden Situation beschreiben, oder auch einzeln als kontinuierliche Ausgangssignale bereitgestellt werden. Die kontinuierliche Informationsweiterleitung bietet sich beispielsweise für Blackboard-Informationen, Informationen über den Zustand der Umwelt, sowie für Informationen über den internen Zustand des Agenten an.

Die Verteilung der verfügbaren Informationen erfolgt in Abhängigkeit von deren Bedeutung. Informationen, die den sozialen Zustand des Agenten beeinflussen, werden mit Hilfe der Location *SocialPerceptsQ* der Komponente *Social Characteristics* zugeteilt. Informationen, die dem Aufbau bzw. der Aktualisierung des Wissens des Agenten dienen, werden auf Basis der Elemente *PerceptsQ* und *InternalStateInfo* an die Komponente *Cognition* weitergeleitet. Die Location *PerceptsQ* symbolisiert dabei eine Warteschlange, die Perzepte in Form von diskreten Informationspaketen aufnimmt. Das Element *InternalStateInfo* symbolisiert kontinuierliche Informationen über den internen physischen, emotionalen, kognitiven und sozialen Zustand des Agenten.

Die Modellierung von Wahrnehmungsprozessen

Die Hauptaufgabe der Komponente *Perception* besteht allerdings nicht nur in der Verteilung eingehender Informationen. Diese Aufgabe könnte auch bereits durch die Komponente *Sensor* übernommen werden und würde die Einführung einer eigenständigen Komponente, die alleine diesen Zweck erfüllt, nicht rechtfertigen. Vielmehr dient die Komponente *Perception* der Realisierung verschiedener Funktionalitäten im Zusammenhang mit Wahrnehmungsprozessen. Eingehende Informationen werden also in der Regel innerhalb der Komponente *Perception* nicht nur an die konsumierenden Komponenten verteilt, sondern vor deren Weiterlei-

tung modifiziert bzw. im Rahmen von Wahrnehmungsprozessen genutzt, um Perzepte zu generieren, die die Reizsituation beschreiben.

Wie in der Einleitung zu diesem Abschnitt bereits angedeutet wurde, können im Rahmen der Wahrnehmung komplexe Prozesse der Bild- oder Sprachverarbeitung eine Rolle spielen. Die Hauptaufgabe, die nachgebildeten Wahrnehmungsprozessen in diesem Zusammenhang zukommt, besteht darin, die externe Reizsituation zu analysieren und mit bekanntem Wissen in Deckung zu bringen, so daß geordnete Perzepte beispielsweise in Form von wiedererkannten Objekten oder Sachverhalten erzeugt werden können. Um auf das bestehende Wissen des Agenten zugreifen zu können, wird die Komponente *Perception* über das Eingangselement *CogStateS* mit der Komponente *Cognition* verbunden. Die neu aus den Eingangsinformationen erzeugten Perzepte werden mit Hilfe der Location *PerceptsQ* für die Komponente *Cognition* zur Verfügung gestellt und können dort in das Wissen des Agenten integriert werden.

Neben der Generierung neuer Informationen aus der verfügbaren Reizsituation, spielen für die Modellierung von Wahrnehmungsprozessen auch Mechanismen eine Rolle, die zu einer Filterung bzw. Selektion eingehender Informationen in Abhängigkeit des internen Zustands des Agenten führen. Selektionsmechanismen bewirken, daß nicht alle zur Verfügung stehende Informationen bewußt wahrgenommen werden und zu einer Erweiterung bzw. Aktualisierung des internen Zustands des Agenten führen. Verarbeitet werden vielmehr nur solche Informationen, die unter Berücksichtigung des internen Zustands des Agenten als relevant erachtet werden. Als irrelevant eingestufte Informationen werden verworfen und nicht weiter verarbeitet.

Um derartige Selektionsmechanismen realisieren zu können, benötigt die Komponente *Sensor* lesenden Zugriff auf den internen Zustand des Agenten. Die in Abschnitt 6.2.2.1 eingeführten Eingangselemente *PhysStateS*, *EmoStateS*, *CogStateS* und *SocialStateS* erhalten also eine zusätzliche Bedeutung und ermöglichen neben dem Zugriff auf interne Zustände im Rahmen der Selbstwahrnehmung auch die Modellierung kausaler Abhängigkeiten im Zusammenhang mit Prozessen der Informationsfilterung.

Ein kleines und ganz bewußt sehr einfach gehaltenes Beispiel soll das Zusammenspiel des internen Zustands des Agenten mit Selektionsmechanismen in der Komponente *Perception* verdeutlichen.

Dieses Beispiel geht von einem emotionalen Agenten aus, der in Abhängigkeit von Erlebnissen in seiner Umgebung Angst entwickeln kann. Die Höhe der Angst wird mit Hilfe einer internen Zustandsvariable des Agenten modelliert, auf die die Komponente *Perception* lesend zugreifen kann. Zudem erhält der Agent in regelmäßigen Abständen eine gleichbleibende Menge an Informationen über den Zustand seiner Umwelt. Nimmt die Zustandsvariable *Angst* nun einen hohen Wert an, so verarbeitet der Agent die verfügbaren Informationen über seine Umgebung aufmerksamer, um Gefährdungen zu vermeiden. Dies führt dazu, daß ein hoher Anteil der bereitgestellten Informationen als Wissen in die Komponente *Cognition* des Agenten durchdringt. Befindet sich der Agent in einem weniger ängstli-

chen Zustand, so bleiben aufgrund seiner geringeren Aufmerksamkeit sehr viele Informationen unbeachtet und werden in der Komponente *Perception* verworfen. Um diese Situation zu realisieren, kann in der Komponente *Perception* ein einfacher Filtermechanismus integriert werden, der um so mehr Informationen weiterleitet, je höher der Wert der Zustandsvariable *Angst* ist.

Neben quantitativen Aspekten existiert in bezug auf die Informationsfilterung auch ein qualitativer Aspekt. Die Wahrnehmung des Menschen liefert nicht immer exakte Resultate. Bei der Verarbeitung visueller Informationen treten beispielsweise in bezug auf die gelieferten Resultate qualitative Unterschiede in Abhängigkeit der Entfernung zum beobachteten Objekt auf. Die Unschärfe in den wahrgenommenen Informationen wird um so größer, je weiter ein betrachtetes Objekt vom Beobachter entfernt liegt. Auch derartige Phänomene können im Rahmen der Komponente *Perception* unter Beachtung des internen Zustands des Agenten modelliert werden. Hierzu können eingehende Informationen beispielsweise durch Störgrößen modifiziert werden, deren Einfluß in Abhängigkeit des internen Zustands des Agenten variiert. Durch Einführung der Komponente *Perception* besteht also die Möglichkeit, zur Verfügung stehende Informationen des Agenten vor deren Integration in den internen Zustand in einfacher Weise gemäß den Anforderungen der jeweiligen Fallstudie sowohl quantitativ als auch qualitativ zu modifizieren.

6.2.3 Die Komponente *Cognition*

Menschliches Verhalten setzt, um korrekt funktionieren zu können, eine gewisse Menge und Art an unterschiedlichen Informationen voraus (s. Kapitel 4.2.4). Um derartige Informationen speichern und verarbeiten zu können, verfügt der Mensch über kognitive Fähigkeiten, die sein Handeln und Verhalten funktional unterstützen.

Im Rahmen des Referenzmodells *PECS* übernimmt die Komponente *Cognition* die Aufgabe, die kognitiven Fähigkeiten eines Agenten abzubilden. Zu derartigen Fähigkeiten zählen beispielsweise die Speicherung und Verarbeitung von Informationen über die externe Umgebung des Agenten, die Speicherung und Verarbeitung von Informationen über den internen Zustand des Agenten, die Auswahl von Handlungszielen, die Vorausplanung komplexer Handlungen, oder auch die Modifikation der eigenen Vorgehensweisen durch Lernprozesse, die beispielsweise auf Erfahrungen aus der Vergangenheit aufsetzen.

6.2.3.1 Die interne Struktur der Komponente *Cognition*

Die lange Liste an Aufgaben, die für die Komponente *Cognition* in Abbildung 6.10 angeführt ist, läßt bereits darauf schließen, daß dieser Komponente eine zentrale Bedeutung innerhalb der Agentenarchitektur des Referenzmodells *PECS* zu-

kommt. Im Rahmen dieser Komponente werden eine Reihe wichtiger Informationen erzeugt und verwaltet, die im Zusammenhang mit der Verhaltenssteuerung des Agenten eine ausschlaggebende Rolle spielen. Hierzu zählen Informationen über die Umwelt des Agenten (Umweltrepräsentation), Informationen über den internen Zustand des Agenten (Selbstbild), Wissen über Handlungsziele, Handlungspläne, sowie veränderliche kognitive Zustände des Agenten.

Komponente Cognition	**Inputs von Komponente(n)** • Perception • Behaviour • Actor **Outputs an Komponente(n)** • Perception • Behaviour • Actor
Aufgabe(n) • Abbildung der kognitiven Fähigkeiten des Agenten • Verwaltung der Umweltrepräsentation des Agenten • Verwaltung des Selbstbildes des Agenten • Auswahl und Verwaltung von Handlungszielen • Ausarbeitung und Verwaltung von Handlungsplänen • Verwaltung der kognitiven Zustände des Agenten	

Abbildung 6.10: Übersicht über die Komponente Cognition

Die Umweltrepräsentation des Agenten

Im Rahmen des Umweltmodells speichert ein Agent diejenigen Informationen, die den Zustand und das Verhalten seiner Umwelt charakterisieren. In sehr einfachen Fällen beschränkt sich diese Repräsentation auf eine Menge von Zustandsvariablen, die den aktuellen Zustand einer ausgewählten, für das Verhalten des Agenten als relevant erachteten Menge an Umweltattributen wiedergeben. Diese Repräsentationsform reicht in der Regel für einfache reaktive Verhaltensweisen aus.

In komplexeren Fällen, die auch Planungs- und Prognoseprozesse im Zusammenhang mit deliberativen Verhaltensweisen mit einschließen, muß auf eine komplexere Form der Repräsentation zurückgegriffen werden, die es ermöglicht, Attribute zu komplexeren Gebilden zu aggregieren und auch das dynamische Verhalten externer Einflüsse zu beschreiben. Hier kann beispielsweise der Einsatz objekt- bzw. frameorientierter Wissensrepräsentationsmechanismen (Meyer-Fujara, Puppe & Wachsmuth, 1995; Anderson, 1996) von Nutzen sein, die sowohl eine Beschreibung der Struktur, als auch des dynamischen Verhaltens von Objekten ermöglichen. Eine grundlegende Schwierigkeit besteht in diesem Zusammenhang allerdings darin, daß für einen Agenten das dynamische Verhalten von Objekten in seiner Umgebung nicht unmittelbar ersichtlich ist. Es kann im allgemei-

nen lediglich auf der Grundlage von Beobachtungen über die Zeit hinweg erforscht werden.

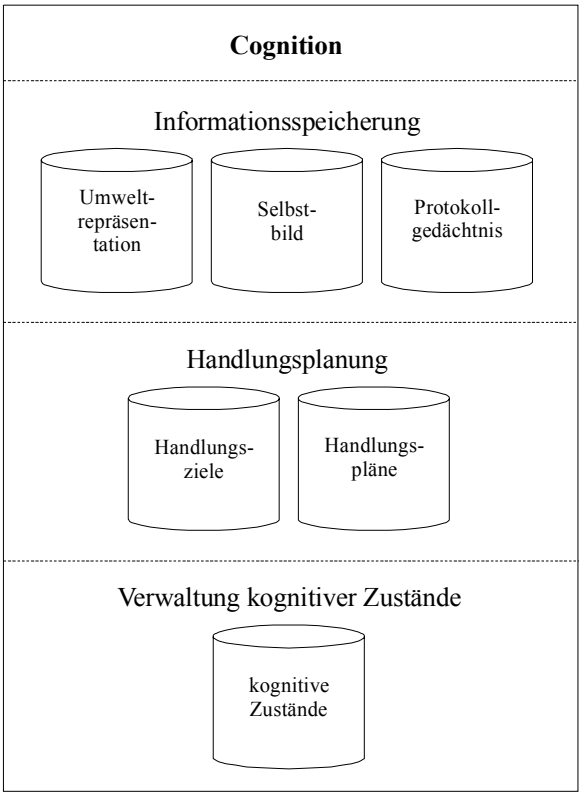

Abbildung 6.11: Die interne Struktur der Komponente Cognition

Das Selbstbild des Agenten

Ebenso wie über seine Umgebung kann ein Agent auch eine Vorstellung von seinem internen Zustand entwickeln. Diese Vorstellung bezieht sich dabei auf physische, emotionale, kognitive, wie auch soziale Eigenschaften des Agenten. Sie wird ebenso wie die Umweltrepräsentation in einem eigenen Datenbereich innerhalb der Komponente *Cognition* abgelegt und durch diejenigen internen Zustände gespeist, die der Agent im Rahmen seiner Selbstwahrnehmung aufnimmt. Neben den einzelnen Eigenschaften können auch Annahmen darüber abgespeichert werden, in welcher Weise sich die internen Zustände des Agenten im Laufe der Zeit verändern. In dieser Hinsicht besteht eine Analogie zwischen der Umweltrepräsentation des Agenten und seinem Selbstbild.

Gelingt der Aufbau eines geeigneten Selbstbildes, so kann das auf diese Weise gewonnene Wissen über den internen Zustand des Agenten im Rahmen von deliberativen Verhaltensweisen, wie z. B. bei der Planung komplexer Handlungen, ausgenutzt werden. Zudem besteht die Möglichkeit, auf der Grundlage des Selbst-

bildes eine reflektive Dynamik zu modellieren, durch die die Vorgehensweisen, Strategien und auch Zielvorstellungen des Agenten geändert werden können.

Die Handlungsziele des Agenten

Handlungsziele müssen im Rahmen der deliberativen Dynamik eines *PECS*-Agenten explizit repräsentiert werden. Sie beschreiben in Anlehnung an gängige Sichtweisen aus der Künstlichen Intelligenz (Biundo, Günter, Hertzberg, Schneeberger & Tank, 1995) Bedingungen, die an interne Zustände des Agenten bzw. externe Zustände aus der Umgebung des Agenten gestellt werden und die der Agent durch Ausführung geeigneter Verhaltensweisen zu erreichen versucht. Die Zustandsvariablen, die in die Zielbedingungen eingehen, bezieht der Agent dabei aus seiner Umweltrepräsentation und aus seinem Selbstbild.

Die Handlungsziele werden im wesentlichen eingesetzt, um den Erfolg bzw. Mißerfolg von Verhaltensweisen zu überprüfen. Nach der Ausführung von Verhaltensweisen, die den internen Zustand des Agenten oder den Zustand der Umgebung ändern, kann der neue, durch die ausgeführte Verhaltensweise entstandene Zustand mit dem vorgegeben Zielzustand verglichen werden. Entspricht der Zielzustand dem internen Zustand oder dem Zustand der Umwelt, so war die ausgeführte Verhaltensweise erfolgreich. Unterscheidet sich jedoch der angegebene Zielzustand von dem tatsächlich vorliegenden Zustand, so hat entweder die ausgeführte Verhaltensweise zu einem negativen Ergebnis geführt oder es müssen weitere Verhaltensweisen ausgelöst werden, die eine weitere Annäherung an den gewünschten Zielzustand bewirken.

Werden im Rahmen der deliberativen Dynamik eines Agenten Ziele gewählt, die nicht in einem einzelnen bzw. einer geringen Zahl an Ausführungsschritten erreicht werden können, besteht auch die Möglichkeit, Zwischenziele einzuführen, die den Lösungsweg weiter strukturieren. Eine geeignete Repräsentation von Zielen und Zwischenzielen bildet also den Ausgangspunkt für die Planung komplexer Handlungen des Agenten.

Die Handlungspläne des Agenten

Handlungspläne dienen der Strukturierung des Vorgehens bei komplexeren Handlungen und werden in einer derartigen Weise ausgelegt, daß das übergeordnete Handlungsziel am Ende der Abarbeitung des Handlungsplans erreicht werden kann. Handlungspläne speichern die Art und Reihenfolge der auszuführenden Operatoren, sowie zugehörige Vor- und Nachbedingungen. Die Vorbedingungen beschreiben Anforderungen an Situationen, die erfüllt sein müssen, damit eine Operation des Handlungsplans zur Ausführung kommen kann. Die Nachbedingungen spezifizieren Erwartungen im Hinblick auf die neue Situation, die sich durch Ausführung der jeweiligen Operation einstellen soll. Diese klassische

Sichtweise auf Handlungspläne geht im wesentlichen zurück auf den Situationskalkül von McCarthy und Hayes (1969).

Für die Erstellung von Handlungsplänen bietet die Künstliche Intelligenz eine sehr große Vielfalt unterschiedlicher Verfahren, auf die im Rahmen dieser Arbeit im einzelnen nicht eingegangen werden kann. Ein Überblick über Planungsverfahren als Teilgebiet der Künstlichen Intelligenz findet sich beispielsweise bei Russel & Norvig (1995), Allen, Hendler & Tate (1990) oder auch bei Hertzberg (1989).

Das Protokollgedächtnis des Agenten

Das Protokollgedächtnis eines Agenten dient der Aufnahme von erstellten und bereits ausgeführten Handlungsplänen. Handlungspläne können nach ihrer Ausführung im Protokollgedächtnis abgelegt werden und stehen damit für zukünftige Situationen, in denen ein Agent sein Handeln planen muß, wieder zur Verfügung. Wenn ein Agent also einen Planungsprozeß anstößt, kann er zunächst einmal in seinem Protokollgedächtnis nachsehen, ob für die vorliegende Situation bereits ein geeigneter Handlungsplan vorliegt. Ist dies der Fall, so kann der Agent diesen Handlungsplan erneut nutzen ohne weitere zeit- und arbeitsaufwendige Planungsschritte einleiten zu müssen. Im Laufe der Zeit wird sich das Protokollgedächtnis eines Agenten mit ausgeführten Handlungsplänen füllen, die der Agent zu späteren Zeitpunkten in seine deliberative Verhaltenssteuerung einfließen lassen kann.

Die kognitiven Zustände des Agenten

Neben den komplexeren Strukturen, die ein *PECS*-Agent für die Durchführung von reaktiven und deliberativen Verhaltensweisen in der Komponente *Cognition* verwaltet, können in dieser Komponente auch einfache Zustandsvariablen berechnet werden, die kognitive Zustände des Agenten beschreiben. Derartige Zustandsvariablen werden beispielsweise eingesetzt, um kognitive Bedürfnisse (s. Kapitel 4.2.3.2), die im Laufe der Zeit entstehen und wieder abklingen, oder auch kognitive Eigenschaften, wie die Intelligenz oder den aktuellen Wissensstand eines Agenten zu modellieren.

6.2.3.2 Inputs der Komponente *Cognition*

Die Komponente *Cognition* verfügt über eine Reihe von Eingangsverbindungen, die im wesentlichen dazu dienen, das Wissen des Agenten über seinen internen Zustand sowie den Zustand seiner Umgebung zu aktualisieren.

Von der Komponente *Perception* werden mit Hilfe der Eingangselemente *PerceptsQS* und *InternalStateInfoS* diskrete Informationen über eingetroffene Nachrichten und Umweltzustände, sowie aktuelle Informationen über interne physische, emotionale, kognitive und auch soziale Zustände abgerufen, die im Rah-

men des Wahrnehmungsprozesses des Agenten erarbeitet werden. Die Perzepte fließen dabei entsprechend ihrer Bedeutung entweder in die Umweltrepräsentation oder das Selbstbild des Agenten ein.

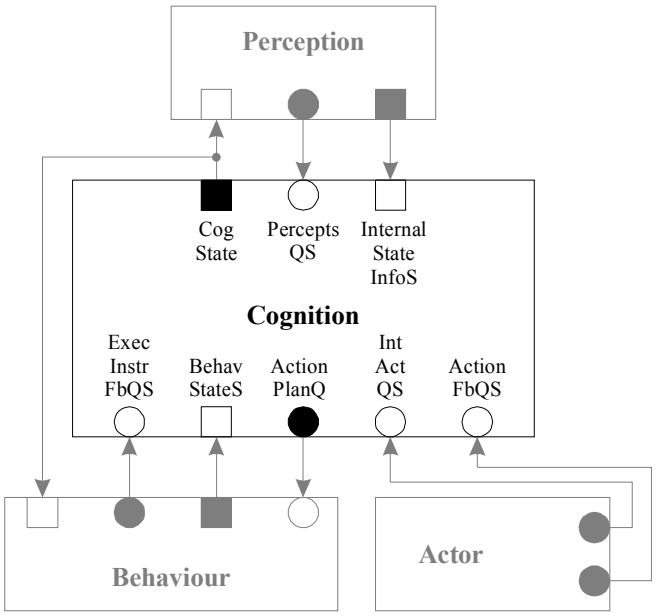

Abbildung 6.12: Ein- und Ausgangsverbindungen der Komponente Cognition

Auch zur Komponente *Behaviour* existieren Eingangsverbindungen. Die Eingangsgröße *BehavStateS* ist mit dem internen Zustand der Komponente *Behaviour* verbunden und stellt sicher, daß die Komponente *Cognition* im Rahmen der deliberativen Dynamik lesend auf Zustände innerhalb der Komponente *Behaviour* zugreifen kann. Mit Hilfe dieser Verbindung werden beispielsweise Informationen über handlungsleitende Motive ausgetauscht, die den Ausgangspunkt für kognitive Prozesse darstellen können. Zudem schickt die Komponente *Behaviour* bei Bedarf Rückmeldungen über getroffene Entscheidungen oder ausgewählte Verhaltensweisen an die Komponente *Cognition*, um das Wissen des Agenten konsistent mit ausgewählten Ausführungsanordnungen (s. Kapitel 6.2.7) zu halten. Die Komponente *Cognition* empfängt derartige Rückmeldungen in der Eingangslocation *ExecInstrFbQS*.

Vergleichbare Rückmeldungen zur Aufrechterhaltung der Konsistenz innerhalb des Wissens liefert die Komponente *Actor* an die Komponente *Cognition.* Durch die Eingangslocation *ActionFbQS* erhält die Komponente *Cognition* Zugriff auf die Rückmeldungen der Komponente *Actor*, die Informationen über ausgeführte Aktionen enthalten. Diese Rückmeldungen können beispielsweise eingesetzt werden, um innerhalb der Komponente *Cognition* anhand von neu eingetroffenen sensorischen Informationen den Erfolg oder Mißerfolg ausgeführter Aktionen festzustellen. Eine weitere Eingangslocation *IntActQS*, die eine zusätzliche Verbindung zwischen der Komponente *Cognition* und der Komponente *Actor*

herstellt, sorgt dafür, daß Aktivitäten innerhalb der Komponente *Cognition* gezielt durch interne Aktionen, die von der Komponente *Actor* abgesetzt werden, angestoßen werden können. Mit Hilfe interner Aktionen, die die Komponente *Cognition* in der Eingangslocation *IntActQS* empfängt, wird die Ausführung deliberativer Verhaltensweisen wie zum Beispiel die Auswahl von Handlungszielen und die Erstellung zugehöriger Handlungspläne in der Komponente *Cognition* ausgelöst.

6.2.3.3 Die grundlegende Arbeitsweise der Komponente *Cognition*

Das dynamische Verhalten der Komponente *Cognition* umfaßt eine Reihe unterschiedlicher Funktionalitäten, die im wesentlichen den Kategorien Aktualisierung des Wissens auf der Grundlage von externen und internen Informationen, Generierung von neuem Wissen, Abruf bestehender Wissensinhalte, Verlust von Informationen und Berechnung des dynamischen Verhaltens interner, kognitiver Zustandsvariablen zugeordnet werden können.

Aktualisierung des Wissens

Wie in Abschnitt 6.2.3.2 bereits erläutert wurde, existieren zwischen der Komponente *Cognition* und anderen Komponenten der *PECS*-Agentenarchitektur Verbindungen, die dafür sorgen, daß das Wissen des Agenten an die dynamischen Änderungen sowohl im Inneren des Agenten als auch in seiner Umgebung angepaßt werden kann. Die über die Eingangselemente eintreffenden Informationen werden gemäß ihrer Bedeutung in die verschiedenen Teilsysteme innerhalb der Komponente *Cognition* integriert.

Perzepte, die als diskrete Informationspakete modelliert werden, werden unmittelbar nach ihrem Eintreffen in der Eingangslocation *PerceptsQS* auf ihre Bedeutung hin untersucht. Betreffen die in dem jeweiligen Perzept gespeicherten Informationen die Umgebung des Agenten (`Percept.Type = ExtInfo`), werden diese Informationen in die Umweltrepräsentation des Agenten integriert. Sind in einem Perzept jedoch Informationen über den internen Zustand des Agenten codiert, führen diese Informationen zu einer Aktualisierung des Selbstbildes des Agenten. Nach Integration der Informationen in das Wissen des Agenten kann das Perzept verworfen werden. Das in Abbildung 6.13 dargestellte Code-Beispiel zeigt die grundsätzliche Funktionsweise eines Ereignisses zur Verarbeitung von Perzepten in der Komponente *Cognition*.

Signale, die durch die Komponente *Perception* fortwährend für die Komponente *Cognition* bereitgestellt werden, wie dies beispielsweise für manche interne Zustände der Fall sein kann, führen in der Regel nicht zu einer fortwährenden Aktualisierung des Wissens des Agenten, sondern werden ebenso ereignisorientiert verarbeitet, d. h. immer nur dann, wenn bestimmte zeit- oder bedingungsabhängige Ereignisse in der Komponente Cognition eintreten. Ein mögliches Beispiel für eine derartiges Ereignis wäre das Eintreffen einer internen Aktion, die die Kom-

ponente *Cognition* anweist, wahrgenommene, interne Zustände in das Selbstbild des Agenten zu integrieren.

```
WHENEVER ContainsElements ( PerceptsQS )
DO
   Percept := GetFirstElem ( PerceptsQS );
   IF ( Percept.Type = ExtInfo )
   DO
      Update_EnvRepresentation ( Percept, EnvRepresentation );
   END
   ELSE IF ( Percept.Type = IntInfo )
   DO
      Update_SelfModel ( Percept, SelfModel );
   END
   Remove ( Percept, PerceptsQS );
END
```

Abbildung 6.13: Die Verarbeitung von Perzepten innerhalb der Komponente Cognition

Rückmeldungen über ausgewählte Verhaltensweisen oder ausgeführte Aktionen werden durch die Komponenten *Behaviour* bzw. *Actor* mit Hilfe von diskreten Informationspaketen an die Komponente *Cognition* geschickt. Auch diese Information werden unmittelbar nach ihrem Eintreffen in das Planungswissen oder das Selbstbild des Agenten integriert.

Generierung von neuem Wissen

Neben der Aktualisierung des Wissens kann es im Rahmen der Verhaltenssteuerung eines Agenten auch nötig sein, neues Wissen aus bestehenden Wissensinhalten zu generieren. Dies kann etwa im Rahmen von Denk- oder Planungsprozessen erfolgen, die im Modell nachgebildet werden.

Denk- und Planungsprozesse verbrauchen in der Regel ein gewisses Maß an Zeit und können bei *PECS*-Agenten durch interne Aktionen (s. Kapitel 6.2.8) angestoßen werden. Nachdem eine entsprechende interne Aktion eingetroffen ist, werden zugehörige Mechanismen innerhalb der Komponente *Cognition* aktiviert, die für die Realisierung von Denk- und Planungsprozessen verantwortlich sind. Hierzu können beispielsweise deduktive Methoden, sowie in verstärktem Maße auch algorithmische Verfahren zum Einsatz kommen.

Denkprozesse führen zu einer Erweiterung der Umweltrepräsentation bzw. des Selbstbildes des Agenten. Planungsprozesse erzeugen neue Handlungspläne, die das Vorgehen des Agenten strukturieren. Um Planungsprozesse durchführen zu können, greift der Agent auf Informationen zu, die in der Komponente Cognition gespeichert sind. Hierzu zählen beispielsweise Informationen über den internen und externen Ausgangszustand, die gesteckten Handlungsziele, sowie Informationen über die zur Verfügung stehenden Aktionen und deren Parameter. Die erzeugten Handlungspläne werden in der Location *ActionPlanQ* abgelegt und für die Komponente *Behaviour* zur Abarbeitung zugänglich gemacht. Die Abbildung

6.14 zeigt einen Pseudocode-Ausschnitt eines Ereignisses, in dem in der Komponente *Cognition* ein Planungsprozeß infolge einer internen Aktion vom Typ *Planning* durchgeführt wird.

```
WHENEVER ContainsElements ( IntActQS )
DO
   IntAction := GetFirstElem ( IntActQS );
   IF ( IntAction.Type = Planning )
   DO
      ActionSequence := CreatePlan ( EnvRepresentation,
                                     SelfModel,
                                     Goals,
                                     Actions );
      Add ( ActionSequence, ActionPlanQ );
   END
   Remove ( IntAction, IntActQS );
END
```

Abbildung 6.14: Grundsätzlicher Ablauf eines Planungsprozesses in der Komponente Cognition

Abruf bestehender Wissensinhalte

Um die in der Komponente *Cognition* gespeicherten Informationen im Rahmen der Verhaltenssteuerung des Agenten nutzen zu können, müssen entsprechende Mechanismen vorgesehen werden, die einen Abruf vorhandener Informationen ermöglichen. Die abgerufenen Informationen werden beispielsweise mit Hilfe des Ausgangselementes *CogState* der Komponente *Behaviour* zur Verfügung gestellt. Durch die Verhaltenssteuerung des Agenten können daraufhin Verhaltensweisen ausgewählt werden, die an die gegebene Situation angepaßt sind.

Die Mechanismen für den Abruf bestehender Informationen können auch eingesetzt werden, um Erinnerungsvorgänge des Menschen nachzubilden. Denkbar wäre hier beispielsweise, daß Informationen nicht in exakt derselben Weise an verarbeitende Komponenten weitergegeben werden, wie sie im Wissen des Agenten gespeichert sind. Hierzu könnten etwa vor deren Weitergabe Transformationen auf den Informationen durchgeführt werden, die bestimmte Informationen ausblenden und damit zu einer Unvollständigkeit der gelieferten Informationen führen, oder auch einzelne Teilinformationen modifiziert werden, so daß Unschärfen in den Informationen zustande kommen können.

Anzumerken ist an dieser Stelle, daß im Rahmen der Denk- und Planungsprozesse, sowie beim Abruf bestehender Wissensinhalte, die in der Komponente *Cognition* stattfinden, der interne Zustand des Agenten von ausschlaggebender Bedeutung sein kann. Beispielsweise kann das Ergebnis von Planungsprozessen in entscheidender Weise vom emotionalen Zustand des Agenten abhängen. Befindet sich ein Agent aufgrund einer externen Bedrohung in einem sehr ängstlichen Zustand, wird etwa eine Planung sehr schnell und ungenau erfolgen mit dem Ziel möglichst schnell in irgendeiner möglichen Weise der Bedrohung zu entkommen. In einer anderen Situation, in der sich der Agent in keiner bedrohlichen Situation

befindet, mag die Planung zu einem besseren Resultat führen und vielleicht sogar auch noch ein gestecktes Nebenziel erfüllen.

Verlust von Informationen

Im menschlichen Gedächtnis können Informationen nicht nur angefügt werden, sondern durch Vergessensprozesse auch wieder verlorengehen. Um Vergessensprozesse bei *PECS*-Agenten nachbilden zu können, besteht innerhalb der Komponente *Cognition* auch die Möglichkeit, Informationen teilweise aus den einzelnen Wissensspeichern zu entfernen. Hier können beispielsweise Zufallsmechanismen eingesetzt werden, die von Zeit zu Zeit angestoßen werden und in zufälliger Weise Informationen aus dem Wissen des Agenten entfernen. Aber auch deterministische Verfahren sind hier denkbar, die in Anlehnung an psychologische Theorien lange nicht mehr benötigte oder unwichtige Informationen aus dem Wissen des Agenten entfernen.

Berechnung des dynamischen Verhaltens kognitiver Zustandsvariablen

Zusätzlich zu den Informationsspeichern können in der Komponente *Cognition* auch Zustandsvariablen verwaltet und mit Hilfe des Ausgangselementes *CogState* nach außen zur Verfügung gestellt werden, um Wechselwirkungen zu modellieren, die zwischen dem kognitiven Zustand und den anderen internen Zuständen des Agenten sowie mit dessen Verhaltenssteuerung bestehen. Für die Änderungen dieser Zustandsvariablen stehen gemäß des systemtheoretischen Ansatzes sowohl zeitdiskrete als auch zeitkontinuierliche Mechanismen zur Verfügung.

6.2.4 Die Komponente *Emotion*

Emotionen liefern essentielle Impulse für das menschliche Verhalten. Sie stehen in bezug auf ihre Entstehung und auch ihre Auswirkungen in enger Wechselwirkung mit anderen psychischen und sozialen Funktion des Menschen (s. Kapitel 4.2.5).

Aus diesem Grund wird innerhalb der *PECS*-Agentenarchitektur eine eigenständige interne Komponente *Emotion* eingeführt, die für die Realisierung der emotionalen Dynamik eines Agenten verantwortlich ist und dazu mit anderen internen Komponenten des Agenten zusammenarbeitet.

6.2.4.1 Inputs der Komponente *Emotion*

Wie in Kapitel 4.2.5 bereits diskutiert wurde, existieren in der Literatur vielfältige und auch zum Teil konträre Erklärungsansätze für die Entstehung, die Existenz,

sowie die Auswirkungen von Emotionen. Den meisten Ansätzen ist jedoch die Überzeugung gemein, daß Emotionen nicht isoliert vom restlichen psychischen, physischen und sozialen Geschehen des Menschen arbeiten, sondern vielmehr in enger Wechselwirkung mit diesen Einflußbereichen stehen.

Komponente Emotion	**Inputs von Komponente(n)** • Perception
Aufgabe(n) • Modellierung der emotionalen Dynamik des Agenten	**Outputs an Komponente(n)** • Perception • Behaviour

Abbildung 6.15: Übersicht über die Komponente Emotion

Die Komponente *Emotion* nimmt also, ebenso wie die Komponente *Cognition* eine zentrale Rolle innerhalb der *PECS*-Agentenarchitektur ein. Sie erhält über das Eingangselement *InternalStateInfoS* (s. Abbildung 6.16), das an die korrespondierenden Zustände in der Komponente *Perception* gekoppelt ist, die Möglichkeit, auf relevante interne Zustände des Agenten lesend zuzugreifen und damit diese Zustände bei der Bestimmung des emotionalen Zustands sowie der emotionalen Dynamik des Agenten zu berücksichtigen. Auf diese Weise wird eine grundsätzliche Möglichkeit geschaffen, emotionale Zustände in Beziehung zu den internen physischen, kognitiven und sozialen Zuständen des Agenten zu setzen. Dabei werden allerdings durch die Architektur keinerlei Kausalbeziehungen fest vorgegeben, so daß die Architektur grundsätzlich in der Lage ist, die Anforderungen unterschiedlicher Emotionstheorien in bezug auf das konkrete Zusammenspiel interner Prozesse und Zustände umzusetzen.

6.2.4.2 Die grundlegende Arbeitsweise der Komponente *Emotion*

Die Komponente *Emotion* verwaltet eine Menge von Zustandsvariablen, die den emotionalen Zustand des Agenten beschreiben. Die Zustandsvariablen können dabei Werte annehmen, die sich im Laufe der Zeit verändern können. Zustandsübergänge, die derartige Wertänderungen hervorrufen, erfolgen dabei sowohl zeitdiskret als auch zeitkontinuierlich und können durch eigendynamische Prozesse innerhalb der Komponente *Emotion*, sowie durch externe Ereignisse in anderen Komponenten der Agentenarchitektur hervorgerufen werden, die eine Anpassung des emotionalen Zustandes des Agenten erforderlich machen.

Die Beschreibung zeitdiskreter Zustandsübergänge kann auf der Grundlage von Ereignissen oder Regeln erfolgen, die angeben, unter welchen Bedingungen welche Zustandsübergänge erfolgen sollen. Für die Beschreibung zeitkontinuierlicher Wertänderungen von Zustandsvariablen können beispielsweise Differentialgleichungen oder herkömmliche algebraische Gleichungen herangezogen werden.

Wertänderungen emotionaler Zustandsvariablen können von außen induziert werden. Um das Zusammenspiel von Emotionen mit anderen internen Zuständen

im Hinblick auf deren Aktualgenese modellieren zu können, greift die Komponente *Emotion* mit Hilfe der Eingangsverbindung *InternalStateInfoS* aktuelle interne Zustände ab. Die importierten Zustandsvariablen können dann sowohl im Rahmen von Regeln bzw. Ereignissen, als auch innerhalb von Gleichungen in der Komponente *Emotion* auftreten und dadurch die Zustandsübergänge emotionaler Zustandsvariablen beeinflussen. Das in Abbildung 6.17 angegebene Beispiel schildert eine Situation, in der eine emotionale Zustandsvariable infolge einer Änderung einer kognitiven Zustandsvariable auch ihren Wert ändert.

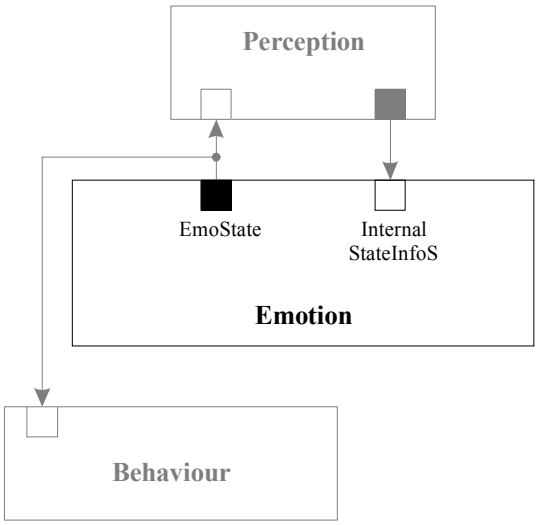

Abbildung 6.16: Grundsätzlicher Aufbau der Komponente Emotion

Den Ausgangspunkt bildet eine Situation, in der ein Agent innerhalb seiner Umgebung in eine bedrohliche Situation gerät. Er stellt die Bedrohung fest, indem er die durch die Umgebung bereitgestellten Informationen über die vorliegende Situation innerhalb seiner Komponente *Cognition* auswertet. Das Ergebnis seiner Auswertung speichert er in der kognitiven Zustandsvariable *InDanger*, die im Falle einer Bedrohung den Wert *True* erhält. Auf diese Zustandsvariable kann von der Komponente *Emotion* aus mit Hilfe der Eingangsgröße *InDangerS* lesend zugegriffen werden. Sobald der Agent die Bedrohung realisiert, wird er ängstlich. Dies kann durch einen plötzlichen, diskreten Anstieg einer Zustandsvariable *Fear* innerhalb der Komponente *Emotion* modelliert werden. Der Anstieg wird dabei als unmittelbare Konsequenz der Wertänderung der Variable *InDanger* realisiert. Der folgende Pseudocode-Ausschnitt zeigt das Ereignis der Komponente *Emotion*, in dem der Wert für die Zustandsvariable *Fear* nach Feststellung der Bedrohung erhöht wird. Der Einfachheit halber wird für dieses Beispiel angenommen, daß die Erhöhung des Wertes für die Zustandsvariable *Fear* auf den Maximalwert *MaxFear* erfolgt. Im allgemeinen Fall könnten an dieser Stelle auch beliebige andere funktionale Zusammenhänge Verwendung finden, die in detaillierter Weise an die jeweils vorliegende Situation angepaßt sind.

Das eben geschilderte Beispiel stellt eine einfache Umsetzung einer sogenannten Appraisal-Theorie dar, wie sie beispielsweise von Plutchik (1994) vorgestellt wird. An dieser Stelle sei allerdings noch einmal deutlich darauf hingewiesen, daß die Agentenarchitektur des Referenzmodells *PECS* nicht nur Emotionstheorien unterstützt, die kognitive Einschätzungen als Ursache für die Entstehung von Emotionen voraussetzen. Auch andere Ansätze, die auf abweichenden Theorien beruhen, können in flexibler Weise auf der Grundlage der für *PECS* gewählten Integration emotionaler Zustände in die Agentenarchitektur umgesetzt werden.

```
ON InDangerS = True
DO
   Fear := MaxFear;
END
```

Abbildung 6.17: Ereignis für die Modifikation einer emotionalen Zustandsvariable in Abhängigkeit der Wertänderung einer kognitiven Zustandsvariable

Neben der von außen induzierten Dynamik kann in der Komponente *Emotion* auch eine gewisse Eigendynamik stattfinden. So können sich beispielsweise emotionale Zustände im Laufe der Zeit ohne Einwirkung von außen sprunghaft oder kontinuierlich verändern. Kontinuierliche Wertänderungen können beispielsweise verwendet werden, um ein stetiges Abklingen emotionaler Zustände zu modellieren. Zudem ist auch eine gegenseitige Beeinflussung unterschiedlicher emotionaler Zustände möglich. Velásquez (1997) spricht hier beispielsweise von einer gegenseitigen Verstärkung oder auch Hemmung verschiedener Emotionen (Trauer hemmt beispielsweise die Entstehung von Freude etc.). Auch für die Modellierung derartiger Phänomene, die innerhalb der Komponente *Emotion* Berücksichtigung finden können, sind zeitdiskrete, aber auch zeitkontinuierliche Wertänderungsmechanismen von Nutzen.

Um eine konkrete Situation zu veranschaulichen, in der sich aufgrund eines eigendynamischen Prozesses der Wert einer emotionalen Zustandsvariable ändert, soll nun das oben eingeführte Beispiel fortgesetzt werden. Unser Agent soll dazu mit einer zusätzlichen emotionalen Dynamik ausgestattet werden, die dafür sorgt, daß die Angst nach ihrer Entstehung im Laufe der Zeit auch wieder abklingt. Dieser Sachverhalt soll dabei mit Hilfe einer Differentialgleichung beschrieben werden, die einen negativ exponentiellen Verlauf für die Zustandsvariable *Fear* bewirkt:

$$Fear' := -FearDecFactor * Fear \qquad \text{(Gl. 6.1)}$$

Die Variable *FearDecFactor* beschreibt dabei einen konstanten Abnahmeparameter der Angst. Insgesamt ergibt sich also für das obige Beispiel ein Kurvenverlauf für die Zustandsvariable *Fear*, wie er in Abbildung 6.18 dargestellt ist.

Interne emotionale Zustände können ihrerseits andere interne Zustände des Agenten wie auch seine Verhaltenssteuerung beeinflussen. Um anderen Komponenten der Agentenarchitektur den Zugriff auf emotionale Zustände zu ermögli-

chen, wird die Komponente *Emotion* mit der Komponente *Perception* verbunden. Die Komponente *Perception* erhält dadurch die Möglichkeit interne emotionale Zustände wahrzunehmen und an andere interne Komponenten weiterzureichen. Zudem wird der emotionale Zustand an die Komponente *Behaviour* bekanntgegeben, damit auch emotionale Einflüsse für die Bestimmung der Verhaltensweisen des Agenten berücksichtigt werden können.

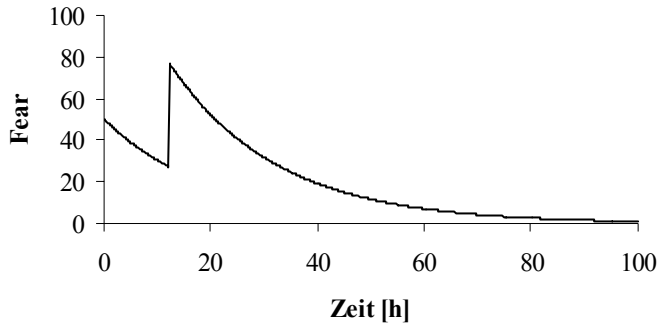

Abbildung 6.18: Kurvenlauf für die Zustandsvariable Fear

6.2.5 Die Komponente *Physis*

Als ein weiterer übergeordneter Einflußbereich auf das menschliche Handeln, Entscheiden und Verhalten wurden in Kapitel 4.2.6 physische Eigenschaften und Prozesse angeführt. Physische Eigenschaften des Menschen unterliegen stetigen Veränderungen und stehen auch in enger Beziehung zu anderen internen wie externen Vorgängen. Um derartige physische Einflüsse in adäquater Weise auch im Rahmen von *PECS* berücksichtigen zu können, wird die Agentenarchitektur um eine zusätzliche interne Komponente *Physis* erweitert. Die Komponente *Physis* übernimmt dabei die Aufgabe, physische Prozesse abzubilden, die im Rahmen der Verhaltenssteuerung von Agenten von Bedeutung sind.

6.2.5.1 Inputs der Komponente *Physis*

Das dynamische Verhalten der Komponente *Physis* kann im allgemeinen durch die Komponenten *Sensor*, *Perception* und *Actor* beeinflußt werden.

Die Komponente *Sensor* sammelt, wie in Abschnitt 6.2.1 erläutert, externe Reize auf und leitet diese an die für deren Verarbeitung zuständigen Komponenten weiter. Unter anderem können hier auch Reize auftreten, die unmittelbar auf den physischen Zustand des Agenten einwirken. Hierzu zählen beispielsweise Einflüsse wie Umgebungstemperatur, Lichtstärke, Luftqualität oder auch andere Reize, die in dem betrachteten Szenario für das Verhalten von Agenten von Bedeutung sein können. Diese Einflüsse werden mit Hilfe von Zustandsvariablen in

der Komponente *Sensor* zur Verfügung gestellt und können von der Komponente *Physis* aus mit Hilfe des Eingangselementes *PhysStimuliS* abgegriffen werden.

Komponente Physis	**Inputs von Komponente(n)** • Sensor • Perception • Actor
Aufgabe(n) • Modellierung der physischen Dynamik des Agenten	**Outputs an Komponente(n)** • Sensor • Perception • Behaviour

Abbildung 6.19: Übersicht über die Komponente Physis

Neben externen Einflüssen kann der physische Zustand eines Agenten auch mit anderen internen Zuständen korrelieren. Um Informationen über interne Zustände zu beziehen und entsprechend auf diese Zustände reagieren zu können, wird für die Komponente *Physis* eine weitere Verbindung zur Komponente *Perception* eingerichtet, die den Zugriff auf interne Zustände ermöglicht. Das Eingangselement *InternalStateInfoS* ist für die Bereitstellung dieser internen Zustandsinformationen zuständig.

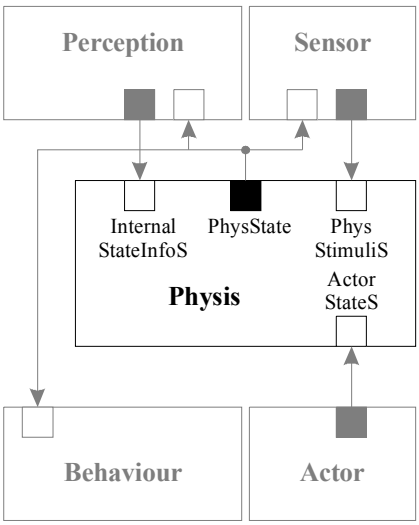

Abbildung 6.20: Grundsätzlicher Aufbau der Komponente Physis

Nicht zuletzt kann der physische Zustand eines Agenten auch davon abhängen, welche Aktionen er über die Zeit hinweg ausführt. Derartige Wechselwirkungen können dadurch abgebildet werden, daß der Komponente *Physis* lesender Zugriff auf interne Zustände innerhalb der Komponente *Actor* eingeräumt wird. Mit Hilfe des Eingangselementes *ActorStateS* wird also innerhalb der Komponente *Physis* bekanntgemacht, welche Aktionen jeweils durch den Agenten ausge-

führt werden, damit physische Zustandsübergänge, die mit den ausgeführten Aktionen einhergehen, in entsprechender Weise angestoßen werden können.

6.2.5.2 Die grundlegende Arbeitsweise der Komponente *Physis*

Die Komponente *Physis* verwaltet eine Menge von Zustandsvariablen, die die physische Konstitution eines Agenten beschreiben. In Übereinstimmung mit der internen Komponente *Emotion* sind auch für die Komponente *Physis* zeitdiskrete und zeitkontinuierliche Zustandsänderungen möglich, die sowohl im Rahmen der Eigendynamik, als auch im Rahmen der von außen induzierten Dynamik der Komponente *Physis* zur Anwendung kommen können. Als Beschreibungsmechanismen für die Dynamik der Komponente finden aus diesem Grund wiederum Ereignisse bzw. Regeln für zeitdiskrete Zustandsübergänge und Differentialgleichungen bzw. algebraische Gleichungen für zeitkontinuierliche Zustandsübergänge Anwendung.

Im Rahmen der von außen induzierten Dynamik werden in der Komponente *Physis* Zustandsübergänge berechnet, deren eigentliche Ursachen mit den aktuellen Zuständen anderer Komponenten zusammenhängen. Ein Beispiel hierfür wäre die Änderung des physischen Zustandes nach der Aktualgenese einer Emotion. Emotionen gehen oftmals mit physischen Symptomen einher. Mit dem Auftreten von Emotionen können sich physische Phänomene wie etwa Errötung, Veränderungen der Herzrate, Veränderungen der Atmung oder auch Schweißbildung einstellen (Meyer, Schützwohl & Reisenzein, 1993). Eine Modellierung dieser Situation kann dabei beispielsweise auf der Grundlage eines Ereignisses innerhalb der Komponente *Physis* erfolgen, in dessen Auslösebedingung der emotionale Zustand des Agenten abgefragt wird. Die benötigten Informationen über den emotionalen Zustand des Agenten bezieht die Komponente *Physis* dabei über das Eingangselement *InternalStateInfoS*. Derselbe Mechanismus greift auch für die Modellierung von Zustandsübergängen, die von bestimmten Gegebenheiten innerhalb der Komponenten *Sensor* und *Actor* abhängen.

Zusätzlich ist die Komponente *Physis* in der Lage, ihren Zustand im Rahmen eigendynamischer Prozesse, d. h. ohne Anregung von außen, zu ändern. Dies kann dann erforderlich sein, wenn beispielsweise homöostatische Triebe (s. Kapitel 4.2.6), Alterungsvorgänge oder tagesrhythmisch bedingte Ermüdungsprozesse nachgebildet werden sollen. Zustandsänderungen beziehen sich in diesem Zusammenhang nur auf interne Zustandsvariablen der Komponente *Physis* und können situationsabhängig sowohl zeitdiskreter als auch zeitkontinuierlicher Natur sein.

In Analogie zu emotionalen Zuständen des Agenten können auch physische Zustände ihrerseits Zustandsübergänge in anderen Komponenten bewirken, wie auch Einfluß auf die Verhaltenssteuerung des Agenten nehmen. Aus diesem Grund wird der interne Zustand, der in Abbildung 6.20 durch das Element

PhysState repräsentiert wird, für die Komponenten *Sensor*, *Perception* und auch *Behaviour* zugänglich gemacht.

6.2.6 Die Komponente *Social Characteristics*

Menschliches Handeln, Entscheiden und Verhalten hängt in entscheidender Weise mit dem sozialen Umfeld zusammen. In Kapitel 4.2.7 wurde angeführt, daß beispielsweise soziale Rollen, soziale Positionen oder auch soziale Bedürfnisse den Menschen in seinen Verhaltensweisen beeinflussen. Im Rahmen der *PECS*-Agentenarchitektur kommt der Komponente *Social Characteristics* die Aufgabe zu, Eigenschaften sowie deren dynamische Veränderungen abzubilden, die den Agenten im Hinblick auf sein soziales Umfeld charakterisieren.

Komponente Social Characteristics	**Inputs von Komponente(n)** • Perception
Aufgabe(n) • Modellierung der sozialen Eigenschaften des Agenten	**Outputs an Komponente(n)** • Perception • Behaviour

Abbildung 6.21: Übersicht über die Komponente Social Characteristics

6.2.6.1 Inputs der Komponente *Social Characteristics*

Soziale Eigenschaften des Menschen hängen mit Vorgängen in seiner Umgebung und auch mit internen Prozessen zusammen. So kann sich beispielsweise die soziale Rolle einer Person mit einem Wechsel des sozialen Umfelds verändern. Ein und dieselbe Person kann etwa im Rahmen seiner Familie die Vaterrolle übernehmen und in einer anderen Situation aber auch Geschäftsführer eines Unternehmens sein. Ebenso können sich im Laufe der Zeit soziale Bedürfnisse wie etwa ein Bedürfnis nach Geselligkeit einstellen, wenn seit dem letzten Kontakt zu anderen Menschen bereits ein langer Zeitraum verstrichen ist. Soziale Eigenschaften hängen also, wie eingangs erwähnt, mit äußeren wie inneren Gegebenheiten zusammen und beeinflussen auch das Verhalten eines Agenten.

Um diese Wechselwirkungen in der Agentenarchitektur von *PECS* in geeigneter Weise zu berücksichtigen, werden für die Komponente *Social Characteristics* Verbindungen zu anderen Komponenten angelegt. Inputs bezieht die Komponente *Social Characteristics* im wesentlichen von der Komponente *Perception*. Mit Hilfe der Eingangslocation *SocialPerceptsQS* erhält die Komponente *Social Characteristics* die Möglichkeit, Informationen über die in der Umgebung des Agenten vorliegende Situation abzurufen. Zudem greift die Komponente *Social Characteristics* über das Eingangselement *InternalStateInfoS* auf Informationen zu, die den internen physischen, emotionalen und auch kognitiven Zustand des Agenten betreffen.

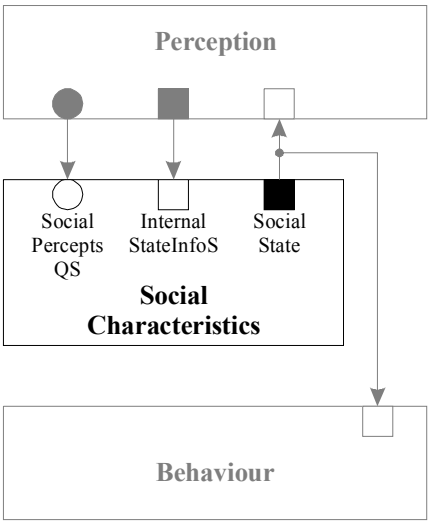

Abbildung 6.22: Grundsätzlicher Aufbau der Komponente Social Characteristics

6.2.6.2 Die grundlegende Arbeitsweise der Komponente *Social Characteristics*

Wie in den anderen internen Komponenten der Agentenarchitektur können auch in der Komponente *Social Characteristics* Zustandsübergänge zeitdiskret und zeitkontinuierlich erfolgen. Ebenso können diese Zustandsübergänge von außen angeregt sein, sowie im Rahmen der Eigendynamik der Komponente ablaufen.

Von außen angeregte Zustandsübergänge dienen dazu, den internen, sozialen Zustand an die Gegebenheiten im sozialen Umfeld des Agenten anzupassen. Stellt sich eine neue Situation in der Umgebung des Agenten ein, die eine Neuberechnung des internen, sozialen Zustands des Agenten erforderlich macht, so wird diese externe Veränderung zunächst durch die Komponente *Perception* registriert und beispielsweise in Form von diskreten Informationspaketen für die Komponente *Social Characteristics* zugänglich gemacht. Sobald ein derartiges Informationspaket eintrifft, wird es mit Hilfe der Eingangslocation *SocialPerceptsQS* durch die Komponente *Social Characteristics* in Empfang genommen. In Abhängigkeit der in dem Informationspaket gespeicherten Informationen über die externe, soziale Situation wird daraufhin eine Aktualisierung des internen, sozialen Zustands des Agenten innerhalb der Komponente *Social Characteristics* vorgenommen.

Das folgende Beispiel dient zur Verdeutlichung eines derartigen, von außen angeregten Zustandsüberganges in der Komponente *Social Characteristics*. Betrachtet wird dabei eine Zustandsvariable *ActRole*, die in der Komponente *Social Characteristics* verwaltet wird und die soziale Rolle repräsentiert, die ein Agent in Abhängigkeit von seinem aktuellen sozialen Umfeld einnimmt. Für dieses Beispiel werden der Einfachheit halber nur die zwei sozialen Rollen *Father* und *Managing_Director* angenommen, die der Agent ausfüllen kann. Die Rolle *Father*

nimmt der Agent ein, sobald er zu Hause bei seiner Familie angekommen ist. Die Rolle *Managing_Director* wird aktiviert, wenn der Agent in seinem Unternehmen eintrifft. Die Information über den Aufenthaltsort des Agenten wird durch die Komponente *Perception* bereitgestellt und in Form eines Informationspaketes *SocialPercept* gespeichert, das in der Variable *Loc* den aktuellen Aufenthaltsort des Agenten ablegt. Der folgende Pseudocode-Ausschnitt zeigt die Zuteilung der Rollen in Abhängigkeit der vorliegenden, externen Situation.

```
WHENEVER ContainsElements ( SocialPerceptsQS )
DO
   SocialPercept := GetFirstElem ( SocialPerceptsQS );
   IF ( SocialPercept.Loc = AtHome )
   DO
      Role := Father;
   END
   ELSE IF ( SocialPercept.Loc = AtCompany )
   DO
      Role := Managing_Director;
   END
   Remove ( SocialPercept, SocialPerceptsQS );
END
```

Abbildung 6.23: Ereignis für die Modifikation einer sozialen Zustandsvariable in Abhängigkeit von externen Gegebenheiten

Zustandsübergänge können in der Komponente *Social Characteristics* auch eigendynamisch erfolgen. Derartige Transformationen dienen beispielsweise der Modellierung sozialer Bedürfnisse, die im Laufe der Zeit entstehen und wieder abklingen, und können von anderen internen Zuständen des Agenten abhängen, die mit Hilfe des Eingangselementes *InternalStateInfoS* abgelesen werden.

Um die internen, sozialen Eigenschaften des Agenten für die restlichen internen Komponenten und auch für die Verhaltenssteuerung des Agenten berücksichtigen zu können, wird der interne soziale Zustand (*SocialState*) sowohl an die Komponente *Perception* als auch an die Komponente *Behaviour* verteilt.

6.2.7 Die Komponente *Behaviour*

Die Komponente *Behaviour* nimmt eine zentrale Stellung innerhalb der *PECS*-Agentenarchitektur ein. Sie ist verantwortlich für die Beschreibung der Verhaltenssteuerung des Agenten. Das Referenzmodell geht von einer regelbasierten Verhaltensbeschreibung aus. In der Komponente *Behaviour* wird daher eine Menge von Regeln verwaltet, die angeben, welche Verhaltensweisen der Agent in Abhängigkeit von seinem internen physischen, emotionalen, kognitiven und sozialen Zustand zur Ausführung auswählt.

6.2.7.1 Inputs der Komponente *Behaviour*

Für die Festlegung des Verhaltens ist der gesamte interne Zustand des Agenten von ausschlaggebender Bedeutung (vgl. Kapitel 5.1.6). Damit im Rahmen der Verhaltenssteuerung des Agenten auf diesen internen Zustand zugegriffen werden kann, werden für die Komponente *Behaviour* Verbindungen zu den internen Komponenten *Physis*, *Emotion*, *Cognition* und *Social Characteristics* eingerichtet. Mit Hilfe der Eingangselemente *PhysStateS*, *EmoStateS*, *CogStateS* und *SocialStateS* greift die Komponente *Behaviour* auf den physischen, emotionalen, kognitiven und sozialen Zustand des Agenten zu und kann damit seine Verhaltenssteuerung an dem aktuellen internen Zustand ausrichten.

Komponente Behaviour	**Inputs von Komponente(n)** • Physis • Emotion • Cognition • Social Characteristics **Outputs an Komponente(n)** • Actor • Cognition
Aufgabe(n) • Modellierung der Verhaltenssteuerung des Agenten • Regelbasierte Generierung von Ausführungsanordnungen	

Abbildung 6.24: Übersicht über die Komponente Behaviour

Im Rahmen deliberativer Verhaltensweisen arbeitet die Komponente *Behaviour* eng mit der Komponente *Cognition* zusammen. Die Komponente *Behaviour* steuert dabei die Ausführung von Handlungsplänen, die in der Komponente *Cognition* generiert wurden. Mit Hilfe der Eingangslocation *ActionPlanQS* erhält die Komponente *Behaviour* die Möglichkeit, Handlungspläne, die in der Komponente *Cognition* gespeichert werden, in Empfang zu nehmen.

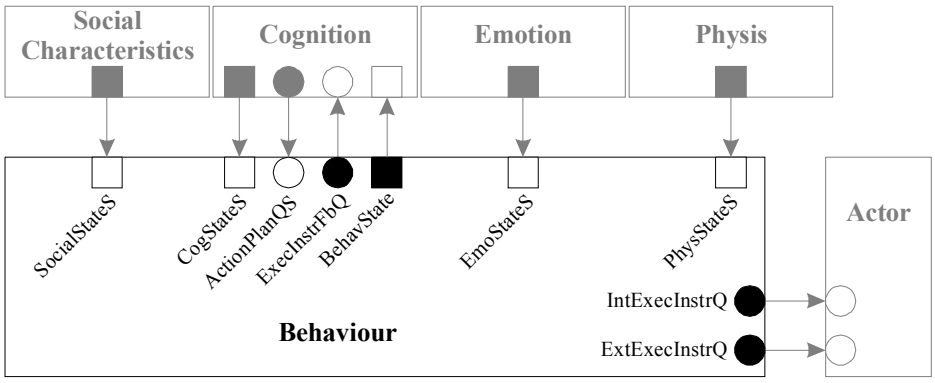

Abbildung 6.25: Grundsätzlicher Aufbau der Komponente Behaviour

6.2.7.2 Die grundlegende Arbeitsweise der Komponente *Behaviour*

Wie eingangs dieses Abschnittes erwähnt, steuert die Komponente *Behaviour* das Verhalten des Agenten. Zu diesem Zweck müssen im Rahmen dieser Komponente Regeln angegeben werden, die beschreiben, wie sich der Agent in verschiedenen Situationen, mit denen er konfrontiert wird, verhalten wird. Die Beschreibung dieser Regeln kann dabei auf der Grundlage des Referenzmodells *SSA* für Strategien (Klinger, 1999) erfolgen.

Die Verhaltensbeschreibung von Agenten und das Referenzmodell SSA

Das Referenzmodell *SSA* gibt ein allgemeingültiges Schema vor, das für die Strukturierung von Strategien genutzt werden kann. Gemäß *SSA* gliedert sich eine Strategie in die Bestandteile *Strategieaufruf*, *Strategie* und *Ausführung* (s. Abbildung 6.26).

Strategieaufruf	
Strategieaufrufbedingung	
↓	
Strategie	
Berechnungsabschnitt	
Regel 1	Regelbedingung
	Berechnungsabschnitt
	Ausführungsanordnung 1
Regel 2	Regelbedingung
	Berechnungsabschnitt
	Ausführungsanordnung 2
...	...
Regel n	Regelbedingung
	Berechnungsabschnitt
	Ausführungsanordnung n
↓	
Ausführung	
Handlungsalternative 1	Ausführungsanordnung 1
	Anweisungen 1
Handlungsalternative 2	Ausführungsanordnung 2
	Anweisungen 2
...	...
Handlungsalternative n	Ausführungsanordnung n
	Anweisungen n

Abbildung 6.26: Das Referenzmodell SSA für Strategien

Der Strategieaufruf umfaßt die Strategieaufrufbedingung, die in Form eines prädikatenlogischen Ausdrucks eine Beschreibung von Situationen liefert, in der die Strategie aufgerufen werden soll.

Strategieaufruf	
Hunger ≥ Aktivierungsschwelle	
↓	
Strategie	
Regel 1	Bedingung: *Nahrungsmittelreserven sind erschöpft*
	Ausführungsanordnung: *im Restaurant essen*
Regel 2	Bedingung: *Nahrungsmittel sind noch vorhanden*
	Ausführungsanordnung: *zu Hause essen*
↓	
Ausführung	
Handlungsalternative 1	Ausführungsanordnung: *im Restaurant essen*
	Aktionen: *zum Restaurant fahren*, *eine Speise bestellen*, *die servierte Speise verzehren*
Handlungsalternative 2	Ausführungsanordnung: *zu Hause essen*
	Aktionen: *Mahlzeit zubereiten*, *die selbst gekochte Speise verzehren*

Abbildung 6.27: Beispiel für eine nach SSA strukturierte Strategie zur Verhaltenssteuerung eines Agenten

Innerhalb der Strategie wird eine Entscheidung getroffen, welche Handlungsalternative in der gegebenen Situation ausgewählt werden soll. Im Strategieabschnitt befindet sich ein Satz an Regeln, die sich jeweils aus einer Regelbedingung sowie einer Ausführungsanordnung zusammensetzen. Regelbedingungen werden dabei ebenso wie Strategieaufrufbedingungen mit Hilfe von prädikatenlogischen Ausdrücken angegeben und dienen zur Unterscheidung verschiedener Situationen. Ausführungsanordnungen beschreiben Handlungsalternativen und umfassen im allgemeinen Parameter, die die jeweilige Handlungsalternative näher spezifizieren. In Berechnungsabschnitten können lokale Berechnungen durchgeführt werden, die entweder die Auswahl der anzuwendenden Regeln beeinflussen oder regelintern Daten für die Entscheidungsfindung bereitstellen.

Im Ausführungsabschnitt werden die ausgewählten Ausführungsanordnungen in konkrete Aktionen umgesetzt. Dabei besteht die Möglichkeit, daß für die Realisierung einer Ausführungsanordnung eine einzige Aktion nicht ausreicht, sondern eine Liste von Aktionen bzw. eine Aktionsfolge generiert werden muß, wobei die einzelnen Aktionen der Aktionsfolge sequentiell ausgeführt werden.

Das Verhalten von *PECS*-Agenten wird durch den gesamten internen Zustand des Agenten, der sich auf physische, emotionale, kognitive und soziale Aspekte bezieht, beeinflußt. Diesem Sachverhalt kann bei Einsatz des Referenzmodells *SSA* zur Beschreibung der Verhaltensstrategien von *PECS*-Agenten dadurch Rechnung getragen werden, daß in den Bedingungen sowohl des Strategieaufrufes als auch der einzelnen Regeln, Zustandsvariablen aus allen vier oben genannten Kategorien auftreten können.

Das in Abbildung 6.27 dargestellte, wiederum ganz bewußt sehr einfach gehaltene Beispiel soll aufzeigen, wie das Referenzmodell *SSA* für die Beschreibung von Strategien eingesetzt werden kann, die das Verhalten von Agenten festlegen. Wir gehen dazu von einem Agenten aus, der von Zeit zu Zeit Hunger entwickeln kann und zwei Möglichkeiten hat, um seinen Hunger zu stillen: er kann entweder im Restaurant essen, wenn die Nahrungsmittelreserven zu Hause erschöpft sind, oder eben zu Hause essen, wenn noch Nahrungsmittel vorrätig sind. Das Restaurant liegt allerdings etwas entfernt vom Aufenthaltsort des Agenten, so daß zunächst eine kurze Anfahrt nötig ist. Auch zu Hause kann der Agent nicht sofort mit dem Essen beginnen, da zuvor die Mahlzeit erst noch zubereitet werden muß. Die Abbildung 6.27 zeigt eine nach SSA strukturierte Strategie, die zur Beschreibung entsprechender Verhaltensregeln des Agenten eingesetzt werden könnte.

Ausführungsanordnungen und Aktionen

Ausführungsanordnungen beinhalten abstrakte Beschreibungen ausgewählter Handlungsalternativen. Sie umfassen sowohl den Typ der auszuführenden Handlung, als auch Parameter, die die Handlung genauer bestimmen.

In Abhängigkeit von aktivierten Ausführungsanordnungen und dem aktuell vorliegenden internen Zustand des Agenten werden innerhalb der Komponente *Behaviour* Aktionen erzeugt, die für die Realisierung der beabsichtigten Handlung benötigt werden, und anschließend für die Ausführung durch die Komponente *Actor* bereitgestellt. Aktionen beschreiben dabei atomare Bestandteile des Handelns, die nach ihrer Ausführung zu konkreten Zustandsänderungen führen. In einfachen Fällen genügt zur Ausführung einer Handlung eine einzige Aktion. Komplexere Handlungen können es jedoch erforderlich machen, daß mehrere Aktionen hintereinander ausgeführt werden müssen, damit das gewünschte Ergebnis der Handlung erreicht werden kann (vgl. Abbildung 6.27).

Im Rahmen des Referenzmodells *PECS* wird zwischen *internen Aktionen* und *externen Aktionen* unterschieden. Die Unterscheidung basiert dabei auf dem Typ

der Komponente, in der durch Ausführung einer Aktion Zustandsübergänge angestoßen werden.

Interne Aktionen dienen der Aktivierung einer internen Komponente der Agentenarchitektur. Mit Hilfe einer internen Aktion ist es beispielsweise möglich, Denk- bzw. Planungsprozesse in der Komponente *Cognition* anzustoßen oder die Komponente *Sensor* zu beauftragen, benötigte Informationen aus der Umgebung abzurufen. Für die Speicherung interner Aktionen wird innerhalb der Komponente *Behaviour* eine eigene Warteschlange *IntExecInstrQ* angelegt.

Externe Aktionen stehen einem *PECS*-Agenten zur Verfügung, um seine Umgebung zu modifizieren oder auch Nachrichten an andere Agenten zu versenden. Durch die Ausführung externer Aktionen werden also Zustandsübergänge in den Komponenten *Environment* und *Connector* ausgelöst. Externe Aktionen werden innerhalb der Komponente *Behaviour* durch die Warteschlange *ExtExecInstrQ* (s. Abbildung 6.25) aufgenommen.

Dieser Aufteilung liegt die Annahme zugrunde, daß ein Agent interne und externe Aktionen nebenläufig ausführen kann. Ein *PECS*-Agent wäre also beispielsweise in der Lage, Planungsprozesse in der Komponente *Cognition* auszuführen und sich gleichzeitig in seiner Umwelt zu bewegen. Reicht dieses Maß an Nebenläufigkeit in bezug auf die Aktionen des Agenten nicht aus, so können in beliebiger Weise weitere Warteschlangen eingeführt werden, die für jeden Aktuator des Agenten Aktionsbefehle, die zur Abarbeitung anstehen, aufnehmen können. Jedem Aktuator kann dann im allgemeinen Fall ein eigener Kontrollprozeß innerhalb der Komponente *Behaviour* zugeordnet werden, der dafür sorgt, daß entsprechende Aktionsbefehle generiert und für die Ausführung freigegeben werden.

Das Abarbeitungsprinzip der Verhaltensregeln

Bei jeder Aktivierung der Komponente *Behaviour* werden die Strategieaufrufbedingungen aller Strategien bzw. Verhaltensregeln überprüft. Für alle diejenigen Verhaltensregeln, deren Aufrufbedingung erfüllt ist, werden die zugehörigen Strategien und Ausführungsabschnitte zur Ausführung gebracht. Dadurch kann es also durchaus vorkommen, daß zu einem gegebenen Zeitpunkt in der Komponente *Behaviour* mehrere Verhaltensregeln schalten. Ebenso können die durch eine ausgeführte Strategie verursachten Zustandsübergänge dazu führen, daß im nächsten Schritt Aufrufbedingungen weiterer Strategien erfüllt sind. Die in der Komponente *Behaviour* spezifizierte Regelmenge kann also in Anlehnung an das Prinzip der Vorwärtsverkettung (Puppe, 1996) grundsätzlich für einen Zeitpunkt mehrmals iterativ durchlaufen werden. Die Iteration ist dann beendet, wenn keine Regel mehr existiert, deren Aufrufbedingung erfüllt ist oder ein anderes Terminierungskriterium, greift.

Bei der Konzeption der Verhaltensregeln für einen Agenten ist ganz besonders darauf zu achten, daß die angegebene Regelmenge eindeutig und vollständig

ist. Die Forderung nach Eindeutigkeit gewährleistet, daß ein Agent in einer gegebenen Entscheidungssituation genau eine Verhaltensweise bestimmen kann, die er zur Ausführung bringen wird. Die Vollständigkeit der Regelmenge stellt sicher, daß keine Situation existiert, für die der Agent keine passende Verhaltensregel zur Verfügung hat, und damit der Agent jederzeit handlungsfähig ist.

Reaktives Verhalten

Die Komponente *Behaviour* ist in der Lage, das Verhalten des Agenten sowohl selbständig, als auch in Zusammenarbeit mit der Komponente *Cognition* zu steuern. Im Falle reaktiven Verhaltens werden die Regeln bzw. Strategien, an denen sich das Verhalten des Agenten orientiert, ausschließlich in der Komponente *Behaviour* spezifiziert. Die Aufrufbedingungen der Verhaltensregeln beziehen sich dabei auf Zustandsvariablen aus allen vier internen Komponenten *Physis*, *Emotion*, *Cognition* und *Social Characteristics*. Sobald sich der interne Zustand des Agenten in einer derartigen Weise verändert, daß Aufrufbedingungen von Verhaltensregeln in der Komponente *Behaviour* erfüllt sind, werden dort die zugehörigen Strategien aufgerufen und entsprechende reaktive Ausführungsanordnungen erzeugt. Die reaktive Verhaltenssteuerung wird also in Abhängigkeit von internen Zustandsänderungen des Agenten angestoßen und erzeugt einzelne Aktionen oder auch automatisch ablaufende Aktionssequenzen, die der Komponente *Actor* zur Ausführung übergeben werden.

Deliberatives Verhalten

Im Rahmen deliberativer Verhaltensweisen arbeitet die Komponente *Behaviour* sehr eng mit der Komponente *Cognition* zusammen. In der Komponente *Cognition* werden zunächst einmal Handlungspläne erzeugt (vgl. Kapitel 6.2.3) und mit Hilfe der Eingangslocation *ActionPlanQS* für die Komponente *Behaviour* zugänglich gemacht.

Wurde für den Handlungsplan eine hohe Abstraktionsebene gewählt, so daß darin nur Teilziele für das geplante Vorgehen gespeichert sind, erzeugt die Komponente *Behaviour* für jedes der im Handlungsplan abgelegten Teilziele eine Aktionsfolge, deren spätere Ausführung zur Erreichung des jeweils zugrunde liegenden Teilziels führen soll. Bei der Generierung von Aktionsfolgen zur Umsetzung deliberativ geplanter Handlungen kommen in der Komponente *Behaviour* ebenso Strategien zur Anwendung, die in deren Bedingungen neben dem jeweils gültigen Teilziel auch den gesamten internen Zustand des Agenten in Betracht ziehen. Werden durch den Handlungsplan, der durch die Komponente *Cognition* bereitgestellt wird, bereits konkrete Aktionen spezifiziert, so besteht die Aufgabe der Komponente *Behaviour* im wesentlichen darin, diese Aktionen für die Komponente *Actor* zur Ausführung freizugeben.

Die Freigabe von Aktionen zur Ausführung

Aktionen, deren Ausführung im Rahmen der Verhaltenssteuerung des Agenten festgelegt wurde, werden in der Regel nicht sofort an die Komponente *Actor* weitergeleitet, sondern zunächst einmal in der Komponente *Behaviour* zwischengespeichert. Für die tatsächliche Freigabe dieser Aktionen werden in der Komponente *Behaviour* eigene Regeln angegeben, die in ihren Bedingungen den internen Zustand des Agenten, den Ausführungszustand der Komponente *Actor*, sowie unter Umständen explizit angegebene Vorbedingungen der jeweiligen Aktionen berücksichtigen können. Zudem können Situationen auftreten, in denen sowohl reaktive als auch deliberativ geplante Aktionen gleichzeitig zur Ausführung anstehen. In derartigen Fällen bedarf es eines Zusatzmechanismus, der dafür Sorge trägt, daß die vorliegenden Aktionen geordnet, d. h. beispielsweise auf der Grundlage einer Prioritätenordnung oder auch verschränkt, zur Ausführung freigegeben werden.

Das Verwerfen von Handlungsplänen und Aktionsfolgen

Manche Situationen können eine Rücknahme von bereits zur Ausführung vorgesehenen Handlungsplänen und Aktionen erforderlich machen. Dies kann beispielsweise dann der Fall sein, wenn sich Zustände, die ausschlaggebend für die Auswahl von Handlungsplänen oder Aktionen waren, bis zur tatsächlichen Ausführung der Handlungspläne oder Aktionen so grundlegend verändert haben, daß eine Fortführung der angestoßenen Handlungen nicht mehr sinnvoll erscheint. Derartige Situation können etwa bei einer Veränderung der handlungsrelevanten Motive eines Agenten eintreten oder auch durch eine sich sehr schnell verändernde Umgebung des Agenten hervorgerufen werden.

Zudem erscheint es sinnvoll, eine Möglichkeit vorzusehen, um Handlungspläne oder Aktionen zu verwerfen, sofern diese nicht erfolgreich zu Ende geführt werden können. Ein derartiger Abbruchmechanismus würde verhindern, daß Agenten über längere Zeiträume hinweg unablässig versuchen, Verhaltensweisen auszuführen, die sich unter Berücksichtigung der gegebenen Umstände als nicht realisierbar erweisen. Auch für die Rücknahme von Handlungsplänen und Aktionsfolgen müssen in der Komponente *Behaviour* entsprechende Regeln vorgegeben werden. In den Aufrufbedingungen derartiger Regeln werden die Umstände spezifiziert, die dazu führen sollen, daß bereits ausgewählte Verhaltensweisen obsolet werden. Im Strategie- bzw. Ausführungsabschnitt werden Ausführungsanordnungen angegeben, die beschreiben, welche Zustandsübergänge für die Rücknahme der in der vorliegenden Situation aktiven Verhaltensweisen erforderlich sind.

Rückmeldungen an die Komponente Cognition

Die Komponente *Behaviour* kann mit Hilfe von Rückmeldungen die Komponente *Cognition* über Entscheidungen in Kenntnis setzen, die im Rahmen der Verhaltenssteuerung des Agenten getroffen wurden. Diese Rückmeldungen dienen dazu, eine gewisse Konsistenz zwischen dem Wissen des Agenten und seinen Verhaltensweisen herzustellen, sowie sensorische Informationen, die der Agent aus seiner Umgebung empfängt, mit seinem Verhalten in Beziehung zu setzen. Ein derartiger Mechanismus wird beispielsweise benötigt, um den Erfolg oder Mißerfolg ausgeführter Verhaltensweisen festzustellen. In der Komponente *Cognition* würde dazu ein Bewertungsmechanismus eingeführt, der sowohl auf den ausgeführten Verhaltensweisen, als auch auf den empfangenen sensorischen Informationen über die tatsächlich eingetretenen Konsequenzen der Verhaltensweisen beruht. Die Rückmeldungen, die von der Komponente *Behaviour* an die Komponente *Cognition* gesendet werden, werden dabei in der Location *ExecInstrFbQ* (s. Abbildung 6.25) innerhalb der Komponente *Behaviour* abgelegt.

Der interne Zustand der Komponente Behaviour

Auch die Komponente *Behaviour* besitzt einen eigenen, internen Zustand. Dieser Zustand macht Informationen über den aktuellen Stand der Verhaltenssteuerung des Agenten verfügbar. So werden hier etwa Resultate von Entscheidungen, wie beispielsweise das aktuell gültige, handlungsleitende Motiv, gespeichert, die die Verhaltenssteuerung des Agenten über längere Zeiträume hinweg beeinflussen. Da dieser interne Zustand (*BehavState*) der Komponente *Behaviour* auch für die deliberativen Prozesse in der Komponente *Cognition* von Bedeutung ist, wird zwischen der Komponente *Behaviour* und der Komponente *Cognition* eine weitere Verbindung eingerichtet, die es der Komponente *Cognition* ermöglicht, auf den internen Zustand der Komponente *Behaviour* lesend zuzugreifen.

6.2.8 Die Komponente *Actor*

Die Komponente *Actor* ist für die Ausführung der Aktionen des Agenten zuständig. Im Rahmen dieser Komponente werden interne wie auch externe Aktionen, die durch die Verhaltenssteuerung in der Komponente *Behaviour* ausgewählt werden, in konkrete Aktionsbefehle umgesetzt.

6.2.8.1 Inputs der Komponente *Actor*

Die Komponente *Actor* ist in bezug auf die Verhaltenssteuerung des Agenten der Komponente *Behaviour* untergeordnet. Sie realisiert diejenigen Aktionen, die durch die Komponente *Behaviour* zur Ausführung freigegeben werden.

Komponente Actor	**Inputs von Komponente(n)** • Behaviour
Aufgabe(n) • Ausführung der im Rahmen der Verhaltenssteuerung festgelegten internen und externen Aktionen des Agenten	**Outputs an Komponente(n)** • Agent (Environment, Connector) • Physis • Cognition • Sensor

Abbildung 6.28: Übersicht über die Komponente Actor

Wie im Abschnitt 6.2.7.2 erläutert wurde, wird im Rahmen von *PECS* zwischen internen und externen Aktion unterschieden. Interne und externe Aktionen können nebenläufig ausgeführt werden und werden zu diesem Zweck in der Komponente *Behaviour* in unterschiedlichen Warteschlangen gesammelt. Den Zugriff auf die Warteschlangen für interne und externe Aktionen erhält die Komponente *Actor* mit Hilfe der beiden Eingangslocations *IntExecInstrQS* und *ExtExecInstrQS*. Diese beiden Warteschlangen dienen damit als Eingangsbereich der Komponente *Actor*, in dem zur Ausführung anstehende Aktionen bereitgestellt werden können.

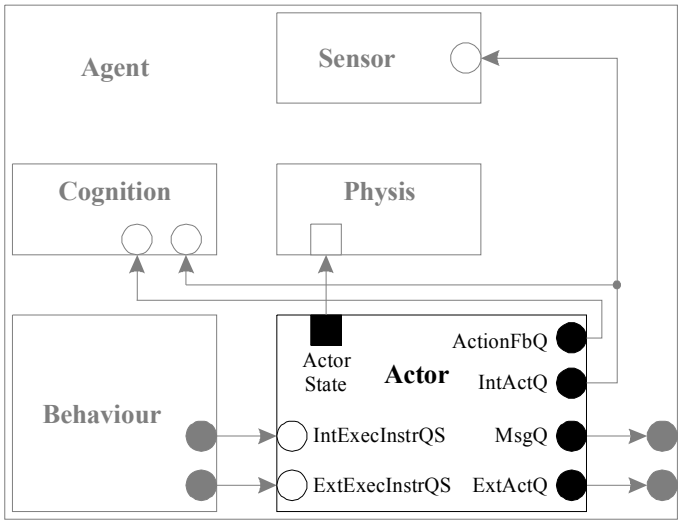

Abbildung 6.29: Grundsätzlicher Aufbau der Komponente Actor

6.2.8.2 Die grundlegende Arbeitsweise der Komponente *Actor*

Die Komponente *Actor* arbeitet ereignisorientiert nach dem Prinzip einer Bedieneinheit. Die Abarbeitung einer anstehenden Aktion wird dabei als zeitverbrauchender Vorgang aufgefaßt. Sobald eine Aktion zur Bearbeitung ansteht, wird ein zeitverbrauchender Vorgang eingeleitet, der die Ausführungsdauer der Aktion nachbildet. Nachdem die Bearbeitungszeit einer Aktion verstrichen ist, wird die

Berechnung der Konsequenzen der ausgeführten Aktion angestoßen. Dazu wird eine Benachrichtigung an diejenige Komponente verschickt, deren Zustand durch die Ausführung der Aktion beeinflußt wird.

Das Verarbeitungsprinzip von Aktionen in der Komponente Actor

Innerhalb der Komponente *Actor* wird für jeden einzelnen Aktuator des Agenten eine eigene Bedieneinheit eingerichtet. Jede Bedieneinheit hat dabei die Kapazität eins, d. h. sie ist in der Lage, zu einem gegebenen Zeitpunkt genau eine Aktion auszuführen. Für die Beschreibung des Referenzmodells *PECS* gehen wir der Einfachheit halber zunächst von zwei Aktuatoren aus, die zum einen für die Bearbeitung interner und zum andern für die Bearbeitung externer Aktionen zuständig sind. Die Aktuatoren für interne und externe Aktionen können dabei nebenläufig agieren. Es sei an dieser Stelle allerdings ausdrücklich darauf hingewiesen, daß die Beschränkung auf zwei Aktuatoren keine grundsätzliche Einschränkung des Referenzmodells *PECS* darstellt. Es ist jederzeit möglich, gemäß des nachfolgend angegebenen Verarbeitungsprinzips und in Abhängigkeit der Anforderungen der jeweiligen Fallstudie innerhalb der Komponente *Actor* eine beliebige Anzahl an Aktuatoren vorzusehen, die durch die Komponente *Behaviour* angesteuert werden.

Strategieaufrufbedingung
Wenn • eine durch die Komponente *Behaviour* freigegebene Aktion zur Ausführung vorliegt und • der Aktuator nicht bereits mit der Abarbeitung einer anderen Aktion beschäftigt ist
Strategie
Start der Aktion
Ausführung
• lege den Endezeitpunkt der Aktion fest • leite den zeitverbrauchenden Vorgang ein • markiere den Aktuator als ‚arbeitend'

Abbildung 6.30: Ereignis zum Start einer Aktion in der Komponente Actor

Wie eingangs bereits erwähnt, arbeitet die Komponente *Actor* ereignisorientiert. Im wesentlichen müssen für eine korrekte Funktionsweise der Komponente *Actor* zwei Ereignisse spezifiziert werden: ein Ereignis, das dafür sorgt, daß die Ausführung einer Aktion eingeleitet wird, sowie ein zweites Ereignis, das nach Ablauf der Dauer der Aktion die zugehörigen Zustandsübergänge in die Wege leitet. Beide Ereignisse können unter Zuhilfenahme zweier einfacher *SSA*-Strategien näher spezifiziert werden.

Die Strategie für den Start einer Aktion hat dabei die in Abbildung 6.30 dargestellte Form. Die Verfügbarkeit auszuführender Aktionen wird durch einen Zugriff auf die Eingangslocations der Komponente *Actor* festgestellt. Liegt eine auszuführende Aktion vor und arbeitet der Aktuator nicht gerade eine andere Aktion ab, so kann mit der Ausführung der Aktion begonnen werden. Dazu wird zunächst einmal die Dauer der auszuführenden Aktion festgestellt, die in der Regel als Parameter der Aktion angegeben wird. Daraufhin wird für die Dauer der Aktionsausführung ein zeitverbrauchender Vorgang angestoßen und schließlich der beanspruchte Aktuator als ‚arbeitend' markiert, um sicherzustellen, daß einem Aktuator zu einem Zeitpunkt nicht mehrere Aktionen zur Ausführung übergeben werden.

Strategieaufrufbedingung
Wenn • der Endezeitpunkt der ausgeführten Aktion erreicht ist und • der Aktuator als ‚arbeitend' markiert ist
Strategie
Veranlassung der durch die Aktion verursachten Zustandsänderungen
Ausführung
• erzeuge ein diskretes Informationspaket, das die durch die Aktion beabsichtigten Zustandsänderungen beschreibt • schicke dieses Informationspaket an diejenige Komponente, deren Zustand aufgrund der ausgeführten Aktion aktualisiert werden muß • entferne die ausgeführte Aktion aus der Warteschlange der noch zu bearbeitenden Aktionen • markiere den Aktuator wieder als ‚frei'

Abbildung 6.31: Ereignis zur Veranlassung der aus einer ausgeführten Aktion resultierenden Zustandsübergänge

Nach Ablauf der Bearbeitungszeit für die ausgeführte Aktion muß die durch die Aktion betroffene Komponente darüber benachrichtigt werden, daß eine für sie relevante Aktion stattfand und deshalb in entsprechender Weise die mit der Aktion verknüpften Zustandsübergänge berechnet werden müssen. Dazu wird zunächst einmal ein diskretes Informationspaket erzeugt, in dem die benötigten Informationen über die auszuführenden Zustandsübergänge hinterlegt werden. Im nächsten Schritt wird dieses Informationspaket dann an diejenige Komponente verschickt, die von den Auswirkungen der ausgeführten Aktion betroffen ist. Die Information, welche Komponente durch eine Aktion betroffen ist, wird wiederum entweder als Parameter einer Aktion angegeben oder liegt bereits durch den Typ der Aktion fest. Abschließend muß die gerade ausgeführte Aktion noch aus der Warteschlange für auszuführende Aktionen entfernt werden, um eine erneute Ausführung zu vermeiden, und der Aktuator wieder als ‚frei' markiert werden, damit

weitere Aktionen abgearbeitet werden können. Das Ereignis, das nach Ablauf der Bearbeitungszeit einer Aktion für die Ausführung der zugehörigen Zustandsübergänge sorgt, ist in der Übersicht in Abbildung 6.31 dargestellt.

Die Verarbeitung interner Aktionen

Interne Aktionen werden der Komponente *Actor* mit Hilfe der Eingangslocation *IntExecInstrQS* übergeben. Sie dienen im wesentlichen der Aktivierung interner Komponenten des Agenten. So können interne Aktionen beispielsweise eingesetzt werden, um Denk- bzw. Planungsprozesse in der Komponente *Cognition* anzustoßen oder die Komponente *Sensor* zu beauftragen, gezielt Informationen aus der Umgebung abzurufen. Die Zeit, die durch die Ausführung interner Aktionen verbraucht wird, wird der Komponente *Actor* zugerechnet. Für die Berechnung der mit den internen Aktionen verbundenen Zustandsübergänge werden hingegen die dafür zuständigen internen Komponenten des Agenten aufgerufen. Der Aufruf erfolgt dabei, wie im vorhergehenden Abschnitt beschrieben, mit Hilfe von diskreten Informationspaketen, die alle für die Beschreibung von internen Aktionen benötigten Attribute umfassen. Die Informationspakete für interne Aktionen werden in der Location *IntActQ* der Komponente *Actor* abgelegt und können von dort aus von den zuständigen internen Komponenten abgerufen werden.

Die Verarbeitung externer Aktionen

Externe Aktionen werden eingesetzt, um die Umgebung des Agenten zu beeinflussen oder auch Nachrichten an andere Agenten bzw. zur Bereitstellung auf einem Blackboard abzusenden. Auch hier wird wiederum die für die Aktion benötigte Zeitdauer innerhalb der Komponente *Actor* modelliert und für die Ausführung der zur Aktion gehörigen Zustandsübergänge die dafür zuständige Komponente aufgerufen.

Bei der Benachrichtigung der für die Zustandsübergänge verantwortlichen Komponenten muß zwischen Nachrichten und Aktionen, die die Umgebung betreffen, unterschieden werden. Diese Unterscheidung kann in der Regel anhand des Typs einer Aktion erfolgen. So werden Aktionen, die Nachrichten an andere Agenten oder Blackboard-Informationen modellieren, mit Hilfe der Location *MsgQ* an die Komponente *Connector* zur Bearbeitung weitergeleitet. Externe Aktionen, die die Umgebung des Agenten betreffen, werden in der Location *ExtActQ* abgelegt und dadurch zur Verarbeitung an die Komponente *Environment* weitergereicht. Neben der Realisierung des zeitverbrauchenden Vorgangs nimmt die Komponente *Actor* in diesem Fall also auch eine Verteilung der aus den Aktionen resultierenden Informationspakete an die betroffenen Komponenten vor.

Der Abbruch laufender Aktionen

In manchen Situationen kann es erforderlich sein, laufende Aktionen abzubrechen. Dies ist beispielsweise dann der Fall, wenn aufgrund von zwingenden Umständen Aktionen ausgeführt werden müssen, die keinen Aufschub dulden, oder noch während der Ausführung einer Aktion festgestellt wird, daß sie nicht zu dem gewünschten Ergebnis führt und es damit keinen Sinn mehr macht, sie zu Ende zu bringen.

Der Abbruch von Aktionen kann mit Hilfe von speziellen Abbruchaktionen ausgelöst werden, die durch die Komponente *Behaviour* erzeugt und in der herkömmlichen Weise an die Komponente *Actor* zur Ausführung weitergegeben werden. Bei der Generierung der Abbruchaktionen ist lediglich dafür Sorge zu tragen, daß sie an die erste Position der Aktionswarteschlange gestellt werden, damit sie sofort durch die Komponente *Actor* abgearbeitet werden können und nicht durch andere Aktionen, die sich ebenfalls noch in der Warteschlange befinden können, blockiert werden.

Für die Abarbeitung von Abbruchaktionen kann in der Komponente *Actor* ein eigenes Ereignis eingeführt werden, das ausgeführt wird, sobald eine derartige Abbruchaktion eintrifft. Als Konsequenz einer Abbruchaktion muß in der Komponente *Actor* lediglich der aktuelle Zustand des betroffenen Aktuators auf ‚frei' zurückgesetzt werden und die abgebrochene Aktion aus der Aktionswarteschlange entfernt werden. Weitere Maßnahmen wie beispielsweise die Annullierung von Zustandsübergängen, sind in diesem Fall nicht erforderlich, da die mit der Aktion verbundenen Zustandsübergänge ohnehin erst nach Ablauf der vollständigen Dauer der Aktion in Kraft getreten wären.

Der interne Zustand der Komponente Actor

Auch die Komponente *Actor* kann über einen eigenen, internen Zustand verfügen. Dieser Zustand gibt die Ausführungszustände der Aktuatoren an, d. h. ob die einzelnen Aktuatoren gerade arbeiten oder frei sind. Für eine detailliertere Betrachtung kann jedoch auch eine feinere Unterscheidung von Ausführungszuständen vorgenommen werden, die unter anderem Rückschlüsse darauf zulassen würde, welcher Typ von Aktion gerade in dem betrachteten Aktuator ausgeführt wird.

Dieser interne Zustand, der in Abbildung 6.29 durch das Modellelement *ActorState* repräsentiert wird, kann auch von außen, beispielsweise durch die Komponente *Physis*, abgelesen werden. Auf der Grundlage einer derartigen kausalen Abhängigkeit können dann Korrelationen zwischen ausgeführten Aktionen und dem physischen Zustand des Agenten abgebildet werden. Ein Beispiel hierfür wäre etwa körperliche Ermüdung infolge von schweren körperlichen Tätigkeiten.

Weiterhin können mit Hilfe von internen Zuständen der Komponente *Actor* auch Einflüsse berücksichtigt werden, die sich unmittelbar auf die Funktionsfähigkeit der Aktuatoren auswirken. Hierdurch ließen sich beispielsweise temporäre oder permanente Defekte, wie auch Fehlfunktionen von Aktuatoren modellieren.

Rückmeldungen an die Komponente Cognition

Um das Wissen des Agenten konsistent mit den ausgeführten Aktionen zu halten, besteht die Möglichkeit, daß die Komponente *Actor* Informationspakete an die Komponente *Cognition* versendet. Mit dem Ablauf der Bearbeitungszeit für eine bestimmte Aktion wird also nicht nur eine Benachrichtigung an die durch die Aktion betroffene Komponente versandt, sondern gegebenenfalls auch eine Rückmeldung an die Komponente *Cognition*, in der der Typ der ausgeführten Aktion sowie unter Umständen benötigte Zusatzinformationen festgehalten werden. Die Rückmeldungen an die Komponente *Cognition* werden in der Location *ActionFbQ* der Komponente *Actor* abgelegt.

6.3 Die Komponente *Connector*

Kommunikation spielt eine wichtige Rolle für den Informationsaustausch zwischen Menschen. Agentenbasierte Modelle, in denen menschliches Verhalten sowie die Interaktion zwischen Menschen abgebildet wird, müssen daher in entsprechender Weise Kommunikationsmechanismen vorsehen, die es ermöglichen, Informationen zwischen Agenten, die wirkliche Menschen im Modell repräsentieren, auszutauschen.

Im Rahmen des Referenzmodells *PECS* wird zur Abwicklung des Informationsaustausches zwischen Agenten die zentrale Komponente *Connector* eingeführt. Diese Komponente realisiert einen Verteilungsmechanismus für Nachrichten und stellt verschiedene Kommunikationsformen zur freien Nutzung durch die Agenten zur Verfügung. Die auf der Grundlage der Komponente *Connector* realisierbaren Kommunikationsformen umfassen direkte Kommunikation, Broadcasting, das als Sonderform der direkten Kommunikation angesehen werden kann, sowie Blackboard-Kommunikation.

Komponente Connector	**Inputs von Komponente(n)** • Agent (Actor)
Aufgabe(n) • Informationsverteilung und Informationsbereitstellung • Unterstützung von direkter Kommunikation • Unterstützung von Broadcasting • Unterstützung von Blackboard-Kommunikation	**Outputs an Komponente(n)** • Agent (Sensor)

Abbildung 6.32: Übersicht über die Komponente Connector

6.3.1 Die Grundstruktur der Komponente *Connector*

Nachrichten werden von Agenten als eine mögliche Form von externen Aktionen erzeugt und als diskrete Informationspakete in der Location *MsgQ* der Komponente *Actor* abgelegt. Die Komponente *Connector* stellt für jeden Agenten einen eigenen Ausgangsbereich in Form einer Eingangslocation *SentMsgQS* bereit. Besteht also ein Modell beispielsweise aus *n* Agenten, so werden in der Komponente *Connector n* solcher Eingangslocations eingerichtet. Jede dieser Eingangslocations ist dabei mit dem Nachrichtenausgang *MsgQ* eines bestimmten Agenten verbunden.

Weiterhin verfügt die Komponente *Connector* über einen Verteilmechanismus, der dafür sorgt, daß die von den Agenten abgeschickten Nachrichten bei den angegebenen Empfängern eintreffen oder in einer durch die Agenten von außen zugreifbaren Datenstruktur, dem sogenannten Blackboard, bereitgestellt werden.

Um eine korrekte Zustellung der Nachrichten an ihre Empfänger zu ermöglichen, wird in der Komponente *Connector* für jeden Agenten ein eigener Bereich für eingehende Nachrichten (*RcvdMsgQ*) eingerichtet. Diese Zustellbereiche werden mit den Eingangslocations *RcvdMsgQS* in der Komponente *Sensor* der jeweils zugeordneten Agenten verbunden. Mit Hilfe dieser Verbindung erhalten die Agenten die Möglichkeit, die an sie gerichteten Nachrichten aus ihrem Ablagefach in der Komponente *Connector* abzuholen.

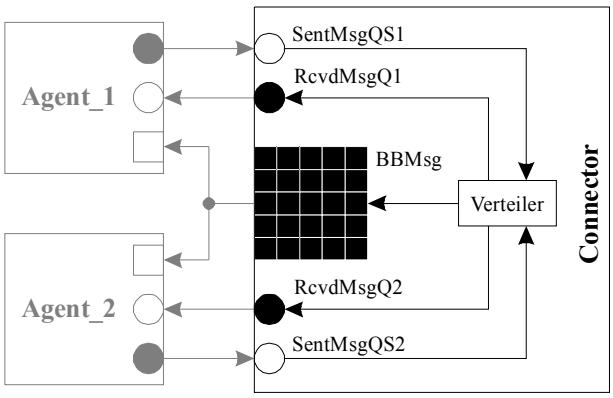

Abbildung 6.33: Der grundsätzliche Aufbau der Komponente Connector

6.3.2 Die grundlegende Arbeitsweise der Komponente *Connector*

Die Komponente *Connector* realisiert einen zentralen Zustelldienst für Nachrichten, die zwischen Agenten ausgetauscht werden. Darüber hinaus verwaltet sie zentral angelegte Datenbereiche im Sinne einer Blackboard-Kommunikation. Die Aktivierung der Komponente *Connector* erfolgt ausschließlich ereignisorientiert. Aktivitäten im Zusammenhang mit der Verarbeitung von Nachrichten werden in

dieser Komponente angestoßen, sobald neue Nachrichten in den Ausgangsbereichen eines oder mehrerer Agenten vorliegen. In Abhängigkeit der in einer eingetroffenen Nachricht angegebenen Kommunikationsform, stellt die Komponente *Connector* diese Nachricht entweder dem angegebenen Empfänger zu oder leitet entsprechende Operationen zur Verwaltung des zuständigen Blackboards ein.

6.3.2.1 Direkte Kommunikation

Im Falle der direkten Kommunikation (Message Passing) wird eine Nachricht von einem Sender zu einem direkt adressierten Empfänger übermittelt. Die Rolle, die der Komponente *Connector* im Zusammenhang mit direkter Kommunikation zukommt, besteht im wesentlichen darin, eine im Ausgangsbereich *SentMsgQS* eines Agenten bereitgestellte Nachricht in den Eingangsbereich *RcvdMsgQ* des als Empfänger spezifizierten Agenten zu transferieren. Die Information, welcher Agent als Empfänger einer Nachricht vorgesehen ist, wird dabei in der Regel als Attribut der abgeschickten Nachricht angegeben.

6.3.2.2 Broadcasting

Broadcasting kann als eine Sonderform von direkter Kommunikation angesehen werden. Dabei wird eine Nachricht nicht an einen einzelnen Agenten versendet, sondern an alle Agenten, die sich zum Zeitpunkt des Nachrichtenversands in einem Bereich aufhalten, in dem die Nachricht vernehmbar ist. Besteht keine Einschränkung in bezug auf die Reichweite einer Nachricht, so erfolgt die Zustellung einer per Broadcast übermittelten Nachricht an alle in einem Modell existierenden Agenten.

Um die Kommunikationsform Broadcast unterstützen zu können, muß die Funktionalität der Komponente *Connector* im Vergleich zur direkten Kommunikation ein wenig erweitert werden. Sollte eine Einschränkung in bezug auf die Reichweite der Nachricht existieren, muß zunächst einmal anhand von bereitgestellten Zusatzinformationen die Menge derjenigen Agenten ermittelt werden, die die Nachricht empfangen können. Daraufhin muß für die vorliegende Nachricht eine entsprechende Anzahl an Kopien angefertigt werden, bevor im letzten Schritt schließlich in Analogie zur direkten Kommunikation die Nachrichten zu den Eingangsbereichen der als Empfänger identifizierten Agenten transferiert werden.

6.3.2.3 Blackboard-Kommunikation

Als dritte Kommunikationsform ist im Rahmen des Referenzmodells *PECS* Blackboard-Kommunikation vorgesehen. Durch die Agenten bereitgestellte Informationen werden dabei in einem oder mehreren öffentlich zugänglichen Da-

tenbereichen, die durch die Komponente *Connector* verwaltet werden, zugänglich gemacht.

Trifft eine neue Blackboard-Nachricht bei der Komponente *Connector* ein, so muß im Rahmen des Verteilmechanismus zunächst einmal geprüft werden, welche Operation im Zusammenhang mit dieser Nachricht ausgeführt werden muß. Die Information über die gewünschte Operation kann in der Regel mit Hilfe eines dafür vorgesehenen Attributes direkt in der Nachricht codiert werden.

Die ausführbaren Operationen umfassen dabei sowohl das Hinzufügen von Nachrichten zu einem Blackboard oder das Entfernen von Nachrichten von einem Blackboard.

Soll auf einem Blackboard eine Nachricht hinzugefügt werden, muß zunächst einmal das zuständige Blackboard, sowie die zugehörige Position auf dem Blackboard identifiziert werden. Beide Informationen werden in der Regel als Parameter einer Nachricht angegeben. Stehen diese beiden Informationen fest, so kann die in der Nachricht angegebene Nutzinformation auf dem zuständigen Blackboard veröffentlicht werden.

Das Entfernen von Informationen von einem Blackboard kann in einer ähnlichen Weise erfolgen. Dazu ist zunächst wiederum festzustellen, welches Blackboard durch die Operation betroffen ist. Zudem muß angegeben werden, welche Information auf dem Blackboard entfernt werden soll. Sofern diese Daten in Erfahrung gebracht wurden, kann die entsprechende Nachricht aus dem angegebenen Datenbereich entfernt werden.

Die Agenten haben dabei zu jeder Zeit die Möglichkeit, die auf einem Blackboard veröffentlichten Informationen nach Belieben zu nutzen und weiterzuverarbeiten.

Der in diesem Abschnitt geschilderte Grundmechanismus kann gemäß den speziellen Anforderungen realer Anwendungsszenarien in vielfältiger Weise erweitert werden. Beispielsweise können Zugriffsrechte für Blackboardinformationen eingeführt werden, die eine restriktivere Handhabung der Lese- und Schreibrechte auf den Blackboards ermöglichen. Schreibzugriffe können beispielsweise dadurch eingeschränkt werden, daß der Verteilmechanismus vor der Publikation von Informationen auf einem Blackboard überprüft, ob der Agent, der als Absender in einer Nachricht verzeichnet ist, über die erforderlichen Rechte verfügt, um die gewünschte Schreiboperation anzustoßen. Ebenso können in bezug auf Lesezugriffe Einschränkungen vorgenommen werden. Dies kann beispielsweise dadurch erfolgen, daß gesamte Blackboards oder auch Teile davon für Lesezugriffe gesperrt werden.

6.3.3 Der grundsätzliche Aufbau von Nachrichten

Nachrichten werden in *PECS* mit Hilfe von diskreten Informationspaketen abgebildet, die zwischen den verschiedenen Warteschlangen der Agenten und der

Komponente *Connector* transferiert werden. Sie bestehen in der Regel aus einem Nachrichtenkopf und einem Nachrichtenrumpf.

Der Nachrichtenrumpf umfaßt die Nutzinformationen, die zwischen den Agenten ausgetauscht werden sollen, und muß an die Anforderungen der jeweils betrachteten Anwendungsszenarien angepaßt werden. Grundsätzlich können im Rumpf einer Nachricht sowohl einzelne, atomare Attribute übertragen werden, als auch komplexere Informationen, die auf der Grundlage einer eigenen, anwendungsspezifischen Ontologie codiert werden.

Der Nachrichtenkopf enthält Meta-Informationen, die für die Verarbeitung der Nachrichten benötigt werden. Im wesentlichen wurden diese Informationen bereits in den Abschnitten 6.3.2.1 bis 6.3.2.3 bei der Beschreibung der unterschiedlichen Kommunikationsformen angesprochen. Die folgende Liste stellt in Form einer Gesamtschau noch einmal diejenigen Attribute von Nachrichten zusammen, die für eine korrekte Funktionsweise der Kommunikationsformen *direkte Kommunikation*, *Broadcast* und *Blackboard-Kommunikation* im Rahmen des Referenzmodells *PECS* benötigt werden:

Nachrichtenkopf	
Attribut	Bedeutung
Communication_Type ∈ { Direct_Communication, Broadcast, Blackboard }	Bezeichnung der verwendeten Kommunikationsform; es kann zwischen direkter Kommunikation, Broadcast und Blackboard-Kommunikation ausgewählt werden
Sender	ID des Absenders der Nachricht
Receiver	ID des Empfängers der Nachricht
Blackboard_ID	ID des zuständigen Blackboards
Blackboard_Operation ∈ { Add, Delete }	Bezeichnung der auszuführenden Blackboard-Operation; gibt an, ob eine Nachricht auf einem Blackboard veröffentlicht oder von einem Blackboard entfernt werden soll
Blackboard_MsgID	ID für eine Nachricht auf dem Blackboard; wird für Delete-Operation benötigt
Nachrichtenrumpf	
Attribut	Bedeutung
Msg_Content	Nutzinformation der Nachricht

Abbildung 6.34: Grundlegende Attribute von Nachrichten

Damit die Kommunikation zwischen Agenten in einem realen Anwendungsbeispiel funktionieren kann, müssen die Agenten über den grundlegenden Aufbau und die Bedeutung der verwendeten Nachrichten informiert sein. Der Aufbau und insbesondere die Nutzinformationen, die durch die Nachrichten transportiert werden, können in beliebiger Weise an die Anforderungen unterschiedlicher Fallstudien angepaßt werden.

Der in Abbildung 6.35 dargestellte Pseudocode-Ausschnitt stellt die Verteilung, sowie die Verwaltung von Nachrichten auf einem Blackboard durch die Komponente *Connector* noch einmal im Überblick dar.

```
WHENEVER ContainsElements ( SentMsgQS[i] )
DO
   Msg := GetFirstElem ( SentMsgQS[i] );
   IF ( Msg.Communication_Type = Direct_Communication )
   DO
      Add ( Msg, RcvdMsgQ[Msg.Receiver] );
   END
   ELSE IF ( Msg.Communication_Type = Broadcast )
   DO
      FOR j FROM 1 TO NrOfAgents
      DO
         IF ( IsReceiver ( Agent[j], Msg.Sender ) AND
              j <> Msg.Sender )
         DO
            Add ( Msg, RcvdMsgQ[j] );
         END
      END
   END
   ELSE IF ( Msg.Communication_Type = Blackboard )
   DO
      IF ( Msg.Blackboard_Operation = Add )
      DO
         AddToBB ( Msg, Blackboard[Msg.Blackboard_ID],
                   Msg.Sender );
      END
      ELSE IF ( Msg.Blackboard_Operation = Delete )
      DO
         DeleteFromBB ( Msg.Blackboard_MsgID,
                        Blackboard[Msg.Blackboard_ID],
                        Msg.Sender );
      END
   END
   Remove ( Msg, SentMsgQS[i] );
END
```

Abbildung 6.35: Die grundlegende Funktionsweise der Komponente Connector

Dem vorliegenden Codeausschnitt liegt die Annahme zugrunde, daß die Agenten der Einfachheit halber mit aufsteigend sortierten, ganzzahligen Ordnungsnummern als Identifikatoren versehen sind. Diese Ordnungsnummern werden in Abbildung 6.35 verwendet, um den Agenten in der Komponente *Connector* eindeutige Ausgangs- und Eingangsbereiche für Nachrichten zuzuordnen.

Sobald im Ausgangsbereich *SentMsgQS[i]* eines Agenten *i* eine neue Nachricht eingetroffen ist, wird diese Nachricht durch die Komponente *Connector* verarbeitet. Eine erste Unterscheidung wird anhand der in der Nachricht angegebenen Kommunikationsform vorgenommen. Handelt es sich dabei um direkte Kommunikation, so wird die eingetroffene Nachricht direkt in den Eingangsbereich des

als Empfänger angegebenen Agenten verschoben. Soll eine Nachricht per Broadcast gleich an eine ganze Menge an Agenten verschickt werden, so muß für alle Agenten des Modells im allgemeinen Fall zunächst einmal überprüft werden, ob sie als Empfänger der Nachricht in Frage kommen. Stellt sich ein Agent *j* als Empfänger der Nachricht *Msg* heraus, so wird ihm eine eigene Kopie der Nachricht in seinem Ausgangsbereich *RcvdMsgQ[j]* zugestellt.

Im Falle der Blackboard-Kommunikation können sowohl Einfüge- als auch Lösch-Operationen durchgeführt werden. Beim Einfügen einer neuen Nachricht auf einem Blackboard muß, sofern mehrere Blackboards existieren, angegeben werden, auf welchem Blackboard und an welcher Stelle auf dem jeweiligen Blackboard die Nachricht platziert werden soll. Ähnliches gilt auch für Lösch-Operationen. Hier wird neben dem Identifikator des zuständigen Blackboards auch ein Identifikator für die zu löschende Nachricht benötigt. Zusätzlich kann für die Durchführung von Blackboard-Operationen auch der Absender der Nachricht eine Rolle spielen. Diese Information kann beispielsweise verwendet werden, um die Berechtigung für die Durchführung einer angeforderten Operation auf einem Blackboard zu überprüfen.

Abschließend wird die verarbeitete Nachricht aus dem Ausgangsbereich des Agenten *i* entfernt und die Komponente *Connector* geht in einen Wartezustand über, bis die nächste Nachricht zur Verarbeitung eintrifft.

6.4 Die Komponente *Environment*

Wie in Kapitel 4.3 erläutert wurde, hängt das Verhalten von Menschen in entscheidender Weise von der Umgebung ab, in die sie eingebettet sind. Die Umgebung setzt dabei Rahmenbedingungen, an denen Menschen ihr Handeln, Entscheiden und Verhalten ausrichten.

Im Rahmen des Referenzmodells *PECS* übernimmt die Komponente *Environment* die Aufgabe, die externe Umgebung der Agenten zu modellieren. Im Rahmen dieser Komponente werden allerdings nur solche Einflüsse abgebildet, die für das Verhalten von Agenten bestimmend sind. Andere Einflüsse, die in dieser Hinsicht ohne Bedeutung sind, fallen dem filternden Abstraktionsprozeß zum Opfer und werden im allgemeinen nicht weiter berücksichtigt.

Komponente Environment	**Inputs von Komponente(n)** • Agent (Actor)
Aufgabe(n) • Modellierung von Umgebungseinflüssen, die für das Verhalten der Agenten relevant sind	**Outputs an Komponente(n)** • Agent (Sensor)

Abbildung 6.36: Übersicht über die Komponente Environment

6.4.1 Die Grundstruktur der Komponente *Environment*

Zwischen der Komponente *Environment* und den Agenten besteht eine enge Wechselwirkung. Zum einen orientieren sich die Agenten bei der Bestimmung ihres Verhaltens an den aktuellen Umständen in ihrer Umgebung. Zum anderen beeinflussen sie jedoch auch den Zustand ihrer Umgebung, indem sie extern wirksame Aktionen ausführen.

Um den Agenten die Möglichkeit zu geben, im Rahmen ihrer Verhaltenssteuerung externe Umstände zu berücksichtigen, wird im Referenzmodell *PECS* eine Verbindung zwischen der Komponente *Environment* und der Komponente *Sensor* der Agenten hergestellt. Mit Hilfe dieser Verbindung wird den Agenten ein lesender Zugriff auf den aktuellen Zustand ihrer Umgebung, der in der Abbildung 6.37 durch das Modellelement *EnvInfo* repräsentiert wird, eingeräumt.

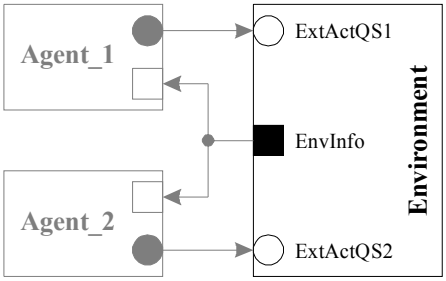

Abbildung 6.37: Die Grundstruktur der Komponente Environment

Agenten sind, wie eingangs erwähnt, auch in der Lage, Aktionen auszuführen, die den Zustand ihrer Umgebung beeinflussen. Aktionen werden in *PECS* auf der Grundlage von diskreten Informationspaketen modelliert und als Resultat der Verhaltenssteuerung in der Komponente *Actor* der Agenten erzeugt. Um die durch die Agenten bereitgestellten Aktionen an die Komponente *Environment* übermitteln zu können, müssen zwischen den Agenten und der Komponente *Environment* weitere Verbindungen eingerichtet werden. Die Komponente *Environment* wird dazu mit Eingangslocations ausgestattet, die den Zugriff auf die durch die Agenten abgesetzten Aktionen ermöglichen. Für jeden Agenten *i* wird dabei eine eigene Eingangslocation *ExtActQSi* eingerichtet, die mit dem zugehörigen Ausgangselement *ExtActQ* in der Komponente *Actor* des Agenten *i* in Verbindung steht.

6.4.2 Die grundlegende Arbeitsweise der Komponente *Environment*

Die Komponente *Environment* berechnet Zustände, die Randbedingungen für das Verhalten der Agenten vorgeben, sowie deren dynamische Veränderungen. In bezug auf das dynamische Verhalten der Komponente *Environment* kann ganz allgemein zwischen einer Eigendynamik der Komponente und einer durch die Akti-

onen der Agenten bedingten Dynamik der Komponente unterschieden werden. In beiden Fällen sind sowohl zeitdiskrete als auch zeitkontinuierliche Zustandsübergänge möglich.

Als Beschreibungsmechanismen für zeitkontinuierliche Zustandsübergänge kommen in Analogie zu den internen Komponenten der Agenten Ereignisse bzw. Transformationsregeln zum Einsatz, die angeben, unter welchen Umständen welche Zustandsvariablen ihre Werte ändern. Zeitkontinuierliche Zustandsübergänge können auf der Grundlage von Differentialgleichungen und algebraischen Gleichungen beschrieben werden.

Eigendynamisches Verhalten ist für die Komponente *Environment* von Bedeutung, wenn Zustandsübergänge modelliert werden sollen, die keinen externen Auslöser voraussetzen. In diese Kategorie fallen beispielsweise Zustandsübergänge in Verbindung mit Wachstums- oder Regenerationsprozessen, die zu kontinuierlichen Wertänderungen von Zustandsvariablen führen. Im Rahmen der Eigendynamik der Komponente *Environment* können jedoch auch diskrete Ereignisse oder Zufallsprozesse eine Rolle spielen. Ein Beispiel für den Einsatz von Zufallsprozessen im Rahmen der Eigendynamik der Komponente *Environment* wäre etwa die Modellierung zufällig eintretender Katastrophen, die zu einer diskreten Änderung der Verfügbarkeit von Ressourcen oder der Gestalt der Umgebung der Agenten führen.

Neben den eigendynamischen Änderungen hängt das Verhalten der Komponente *Environment* zusätzlich noch von Aktionen ab, die die Agenten im Laufe der Zeit ausführen. Mit Hilfe von externen Aktionen (vgl. Abschnitt 6.2.8) besitzen Agenten die Möglichkeit, gezielt auf ihre Umwelt einzuwirken und deren Zustand zu verändern. Zu diesem Zweck generieren die Agenten externe Aktionen in Form von diskreten Informationspaketen, die eine genaue Beschreibung der gewünschten Zustandsänderungen in der Umwelt umfassen. Diese Aktionen werden an die Komponente *Environment* übermittelt, die daraufhin die an sie gerichteten Aktionen interpretiert und entsprechende Zustandsübergänge in die Wege leitet. Bei den mit externen Aktionen verbundenen Zustandsübergängen kann es sich einerseits um zeitdiskrete Änderungen von Zustandsvariablen in der Komponente *Environment* handeln. Zusätzlich können jedoch durch externe Aktionen auch länger andauernde, kontinuierliche Prozesse in der Komponente *Environment* in Gang gesetzt werden. Wird die Komponente *Environment* beispielsweise zur Modellierung eines Ökosystems genutzt und bewirkt die Ausführung einer externen Aktion eine drastische Reduzierung des Bestands eines vorhandenen Rohstoffes, so kann ein durch die externe Aktion ausgelöster, kontinuierlicher Prozess eingesetzt werden, um langfristige Konsequenzen dieses externen Eingriffs, sowie die damit verbundenen Konsequenzen für die Lebensbedingungen der Agenten abgebildet werden.

Je nach Komplexität und den Gegebenheiten der zu realisierenden Fallstudie kann im Sinne des hierarchischen Modellaufbaus (s. Kapitel 5.2) auch eine weitere Untergliederung der Komponente *Environment* in eine Menge von Subkomponenten mit individuellen Zuständigkeiten erforderlich sein. Eine derartige Verfei-

nerung würde es ermöglichen, die interne Struktur der Komponente *Environment* nach unten weiter aufzulösen, ohne dadurch die Schnittstelle zu den Agenten hin zu beeinflussen.

6.5 Zusammenfassung

In diesem Kapitel wurde die grundlegende Strukturierung des Referenzmodells *PECS* beschrieben. *PECS* stellt ein domänenunabhängiges Konstruktionsschema für agentenbasierte Simulationsmodelle zur Verfügung, in denen menschliches Handeln, Entscheiden und Verhalten von ausschlaggebender Bedeutung sind. Es gibt einen konzeptionellen Modellierungsrahmen vor, der unabhängig von speziellen Anwendungsfeldern konzipiert ist. Die Anpassung des Referenzmodells an die Gegebenheiten spezieller Anwendungsszenarien erfolgt durch ein Ausfüllen der Leerstellen, die in der Modellarchitektur vorgesehen sind. Für den Anwender bietet sich in dieser Hinsicht die Möglichkeit, die durch das Referenzmodell vorgesehenen Komponenten mit entsprechenden Zustandsvariablen sowie Zustandsüberführungs- und Outputfunktionen zu füllen. Auf diese Weise wird erreicht, daß agentenbasierte Modelle entwickelt werden können, die unterschiedlichen Anwendungsgebieten entstammen, jedoch alle dieselbe Tiefenstruktur aufweisen.

Die Struktur des Referenzmodells *PECS* orientiert sich an den elementaren Anforderungen, die an agentenbasierte Modelle gestellt werden, wenn auf deren Grundlage menschliches Handeln, Entscheiden und Verhalten untersucht werden sollen (s. Kapitel 4). Es stellt drei unterschiedliche Klassen von Komponenten zur Verfügung. Die Komponente *Agent* wird eingesetzt, um Lebewesen und insbesondere Menschen mit ihren relevanten Eigenschaften und Verhaltensweisen im Modell zu repräsentieren. Die Komponente *Connector* stellt eine Infrastruktur zur Verfügung, die durch die Agenten zum Informationsaustausch genutzt werden kann. Die Komponente *Environment* dient schließlich dazu, Umgebungseinflüsse abzubilden, die sich auf das Handeln, Entscheiden und Verhalten der Agenten auswirken. Aufgrund der hohen Komplexität, die entsteht, wenn menschliches Verhalten mit Hilfe von Agenten abgebildet werden soll, ist für die Komponente *Agent* eine modular-hierarchische Struktur vorgesehen. Diese Komponente zerfällt also in eine Menge von Subkomponenten, die für die Modellierung unterschiedlicher Funktionalitäten im Zusammenhang mit der menschlichen Verhaltenssteuerung benötigt werden. Es handelt sich dabei um die Komponenten *Sensor*, *Perception*, *Physis*, *Emotion*, *Cognition*, *Social Characteristics*, *Behaviour* und *Actor*. Der grundlegende Aufbau sowie die Funktionsprinzipien dieser Komponenten wurden in den jeweiligen Unterabschnitten dieses Kapitels ausführlich dargestellt.

7 FALLSTUDIE I: EVOLUTION VON SOLIDARNETZWERKEN

Die Sozialwissenschaften bedienen sich seit langer Zeit spieltheoretischer Modellierungsansätze, um die Interaktion sozialer Entitäten aus verschiedensten Blickwinkeln heraus zu untersuchen. Als Grundlage für diese Untersuchungen dienen dabei sehr häufig Simulationsmodelle, die auf zellularen Automaten aufsetzen oder sich, wenn auch in der Regel nur in sehr einfacher Form, der Agententechnologie bedienen. Untersucht werden in derartigen Simulationsstudien nicht selten Strukturbildungsprozesse oder andere emergente Phänomene, die aus der Interaktion der Entitäten bzw. Agenten des Modells heraus entstehen.

Die Fallstudie *Evolution von Solidarnetzwerken* beschäftigt sich mit der Entstehung sozialer Interaktionsstrukturen in einer Population rationaler Egoisten. Die Agenten sind dabei als rationale Entscheider konzipiert, die grundsätzlich eigennützig handeln. Kooperationsbeziehungen zwischen jeweils zwei Agenten können unter dieser Voraussetzung also nur dann entstehen, wenn beide an einer Kooperationsbeziehung beteiligte Agenten einen Nutzen daraus ziehen.

Der in den folgenden Abschnitten beschriebene Modellierungsansatz geht zurück auf ein ab 1997 durch Rainer Hegselmann und Andreas Flache (Hegselmann & Flache, 1997; Hegselmann & Flache, 1998) veröffentlichtes, spieltheoretisches Modell, das auf der Grundlage von zellularen Automaten realisiert wurde. Die von Hegselmann und Flache getroffenen Annahmen sollen nun dazu verwendet werden, um ein agentenbasiertes Simulationsmodell auf der Grundlage des Referenzmodells *PECS* aufzubauen.

7.1 Modellbeschreibung

Das Modell *Evolution von Solidarnetzwerken*, das im folgenden kurz *Solidarnetzwerk* genannt werden soll, beschäftigt sich mit der grundlegenden Fragstellung, unter welchen Bedingungen in einer Gesellschaft, die ausschließlich aus rational handelnden Egoisten besteht, ein Netzwerk bilateraler Solidarbeziehungen evolvieren kann. Alle Agenten im Modell *Solidarnetzwerk* befolgen grundsätzlich

dieselben Verhaltensregeln. Sie unterscheiden sich allerdings in ihrer sog. Bedürftigkeit. Die Bedürftigkeit gibt an, welches Maß an Unterstützung ein Agent im Rahmen einer Solidarbeziehung einem anderen Agenten geben kann bzw. wie viel Unterstützung der Agent andererseits aber auch selbst im Rahmen dieser Beziehung von seinem Kooperationspartner erwartet und dann auch in Anspruch nimmt. Die Agenten entscheiden eigenständig über die Wahl ihrer Kooperationspartner, gehen dabei allerdings in opportunistischer Weise vor. Als sozialer Raum dient ein zweidimensionales Spielfeld, wobei jede Position des Spielfelds maximal durch einen Agenten besetzt werden kann.

Der zentrale Ansatz des Modells besteht darin, bilaterale Solidarbeziehungen zwischen jeweils zwei Agenten auf der Grundlage eines Zwei-Personen-Spiels, des sog. *Support Games*, zu modellieren. Solidarbeziehungen können dabei grundsätzlich nur zwischen solchen Agenten entstehen, die sich auf dem Spielfeld in unmittelbarer Nachbarschaft zueinander befinden. Die Nachbarschaftsbeziehungen bleiben dabei in der Regel nicht stabil, sondern verändern sich im Laufe der Zeit dadurch, daß Agenten sog. Migrationsoptionen, d. h. Chancen, ihre aktuelle Position auf dem Spielfeld zu verändern, wahrnehmen. Im Rahmen ihrer Umzüge verfolgen die Agenten das Ziel, in eine neue Nachbarschaft zu gelangen, in der ein möglichst hohes Maß an Unterstützung durch andere Agenten zu erwarten ist. Die eben geschilderten Grundannahmen des Modells führen zu einer regen Bewegung der Agenten auf dem Spielfeld. Es stellt sich die Frage, ob und in welcher Weise sich im Laufe der Zeit erkennbare Strukturen auf dem Spielfeld ausbilden.

7.1.1 Das räumliche Bezugssystem der Agenten

Wie eingangs geschildert, ist für die Ausbildung einer Solidarbeziehung zwischen je zwei Agenten deren räumliche Beziehung von entscheidender Bedeutung. Jeder Agent nimmt daher eine ganz bestimmte Position in seiner Umwelt ein, die er im Laufe der Zeit im Rahmen von Umzügen verändern kann. Die Umwelt im Modell *Solidarnetzwerk* besteht aus einem Torus, d. h. einem nach allen Seiten hin offenen Gitter, mit insgesamt 21 × 21 Spielfeldern. Jedes Spielfeld kann durch maximal einen Agenten belegt werden. Es existieren jedoch weniger Agenten als Spielfelder, so daß nicht alle Spielfelder immer von Agenten besetzt sind. Zu Beginn werden die Agenten zufällig über das Spielfeld verteilt.

Im Rahmen der Umwelt werden ausgehend von der aktuellen Position eines Agenten zwei Regionen definiert, die seinen Aktionsradius vorgeben:

- Das Interaktionsfenster

  Das Interaktionsfenster legt fest, welche Felder in der unmittelbaren Nachbarschaft eines Agenten grundsätzlich für den Aufbau einer Kooperationsbeziehung in Betracht gezogen werden können. Zum Interaktionsfenster eines Agenten im Modell *Solidarnetzwerk* gehören ausgehend von seiner aktuellen

Position nicht alle acht angrenzenden Felder, sondern im Sinne einer Von Neumann-Nachbarschaft nur die Felder im Norden, Süden, Osten und Westen. Die Abbildung 7.1 zeigt das Interaktionsfenster eines Agenten als dunkelgrau hinterlegte Felder.

- Das Migrationsfenster

  Das Migrationsfenster beschränkt die Mobilität der Agenten. Nimmt ein Agent eine Migrationsoption wahr, die ihm im Rahmen eines Zufallsprozesses zugeteilt wird, so darf er nur auf solche Felder in seiner Umwelt umziehen, die zum betrachteten Zeitpunkt durch keinen anderen Agenten besetzt sind und innerhalb seines Migrationsfensters liegen. Das Migrationsfenster erstreckt sich in der Grundvariante des Modells in alle Richtungen auf fünf Felder ausgehend von der aktuellen Position eines Agenten. Es ergibt sich also ein quadratischer Bereich mit Kantenlänge 11. Das Migrationsfenster eines Agenten wird in Abbildung 7.1 durch den hellgrau hinterlegten Bereich dargestellt.

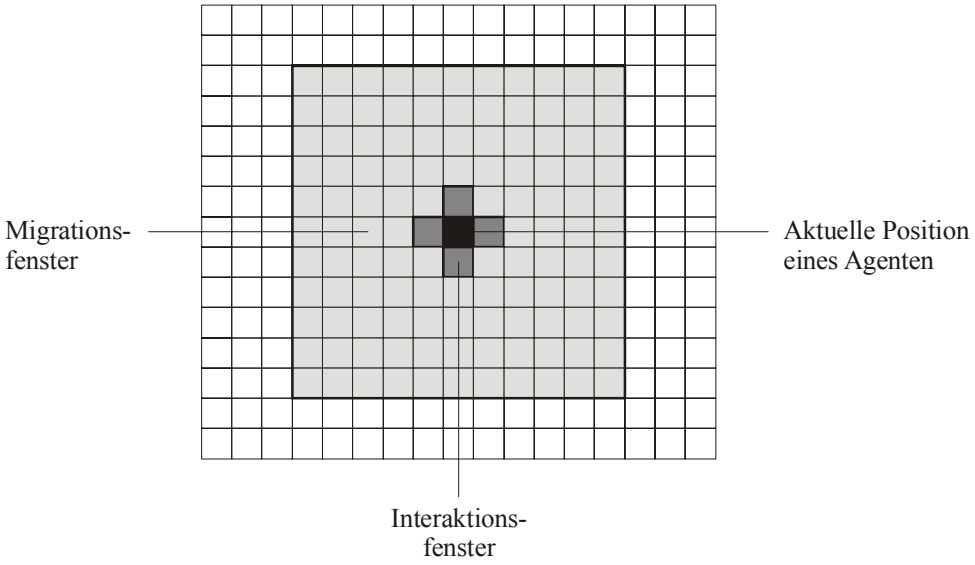

Abbildung 7.1: Die Regionen der Umwelt im Modell Solidarnetzwerk

7.1.2 Eigenschaften und Verhaltensweisen der Agenten

7.1.2.1 Die Bedürftigkeit der Agenten

Die Agenten im Modell *Solidarnetzwerk* zeichnen sich durch ihre Zugehörigkeit zu einer ganz bestimmten Bedürftigkeitsklasse aus. Die Bedürftigkeitsklasse beeinflusst dabei grundlegend, welches Maß an Unterstützung ein Agent in Rahmen einer Solidarbeziehung einem Kooperationspartner geben kann. Ebenso wird durch die Bedürftigkeitsklasse festgelegt, wie hoch der eigene Bedarf eines Agen-

ten an Unterstützung durch andere Agenten ist. Insgesamt sieht das Modell *Solidarnetzwerk* neun Bedürftigkeitsklassen $N_1, ..., N_9$ vor, denen die folgenden Werte zugeordnet sind:

$$N_1 = 0.1, N_2 = 0.2, ..., N_9 = 0.9$$

Je geringer die Bedürftigkeit $n_i \in \{ N_1, ..., N_9 \}$ eines Agenten i ist, d. h. je kleiner die ihm zugeordnete Bedürftigkeitsklasse ist, um so mehr Unterstützung kann er einem anderen Agenten im Rahmen einer Kooperationsbeziehung geben und um so weniger Unterstützung wird er von sich aus in einer derartigen Beziehung einfordern.

Die Bedürftigkeitsklasse wird jedem Agenten zu Beginn eines Simulationslaufes fest zugeordnet und in der Folge als konstant angenommen. Kooperationsbeziehungen, die Agenten mit anderen Agenten eingehen, haben also keinerlei Auswirkungen auf deren Bedürftigkeit.

7.1.2.2 Perfekte Information

Im Rahmen spieltheoretischer Modelle wird häufig angenommen, daß die Akteure bei ihrer Entscheidungsfindung perfekt informiert sind. Diese Annahme bedeutet insbesondere, daß Akteure in einer Entscheidungssituation vollumfänglichen Zugriff auf alle entscheidungsrelevanten Parameter haben und auch die Konsequenzen ihrer Entscheidungen zweifelsfrei abschätzen können.

Die Annahme perfekter Information gilt auch für die Agenten des Modells *Solidarnetzwerk*, die im wesentlichen in zwei Entscheidungssituationen stehen. Zum einen muß im Rahmen der aktuellen Nachbarschaft entschieden werden, ob es sich lohnt, mit einem benachbarten Agenten eine Kooperationsbeziehung einzugehen. Zum anderen ist im Falle der Annahme einer Migrationsoption die Entscheidung zu treffen, welche Nachbarschaft innerhalb des Migrationsfensters des Agenten zu präferieren ist. Beide Entscheidungen basieren auf demselben Bewertungsschema, das durch das Support Game sowie die Verhaltensregeln der Agenten eindeutig vorgegeben ist. Wesentlicher Entscheidungsparameter ist dabei die Bedürftigkeit des betrachteten Kooperationspartners. Die Konsequenz einer Entscheidung entspricht unmittelbar dem individuellen Nutzen, den ein Agent aus einer Interaktion mit einem anderen Agenten im Rahmen des Support Games zieht. Die perfekte Information der Agenten im Modell *Solidarnetzwerk* begründet sich dabei auf der Annahme, daß alle Agenten grundsätzlich die Bedürftigkeitsklassen aller anderen Agenten kennen und für alle Agenten identische Verhaltensregeln zugrunde gelegt werden, so daß jeder Agent im voraus seinen Individualnutzen aus einer Kooperationsbeziehung ohne Zweifel bestimmen kann.

Die Annahme der perfekten Information, wenn auch häufig grundsätzlich im Zusammenhang mit spieltheoretischen Modellen kritisiert, stellt hinsichtlich der Implementierung des Modells eine wesentliche Vereinfachung dar, da für die Abbildung einer Kooperationsbeziehung auf eine direkte Kommunikation zwischen

den betroffenen Agenten verzichtet werden kann. Jeder Agent trifft auf der Grundlage seiner vollständigen Informationen und seiner Verhaltensregeln eine Kooperationsentscheidung ohne explizit mit seinem Kooperationspartner zu interagieren. Beide Kooperationspartner kommen dabei aufgrund derselben Informationen und Verhaltensregeln zu identischen Entscheidungen. Aus diesem Grund kann von einer direkten und expliziten Interaktion beider betroffener Agenten im Rahmen einer Kooperationsbeziehung abstrahiert werden.

7.1.2.3 Das Support Game

Das Support Game beschreibt die Solidarbeziehung, die je zwei Agenten nach eigener Entscheidung im Modell *Solidarnetzwerk* miteinander eingehen. Der Ausgang bzw. individuelle Nutzen einer Solidarbeziehung wird dabei maßgeblich durch die Bedürftigkeitsklassen bestimmt, die den betreffenden Agenten zugeordnet sind.

Die folgenden Betrachtungen gehen von zwei Agenten i und j aus, die den beiden Bedürftigkeitsklassen n_i und n_j angehören. Für beide Agenten ist eine Kooperationsbeziehung sowohl mit einem Zugewinn als auch mit einem gewissen Aufwand bzw. Verlust verbunden.

Der Gewinn G_{ij} des Agenten i aus einer Kooperationsbeziehung mit Agent j ist in der folgenden Weise definiert:

$$G_{ij} := n_i * (1 - n_j) * B \qquad \text{(Gl. 7.1)}$$

B bezeichnet dabei den maximal möglichen Gewinn, der genau dann einträte, wenn ein maximal bedürftiger Agent i (mit $n_i = 1$) mit einem vollständig unbedürftigen Agenten j (mit $n_j = 0$) eine Kooperationsbeziehung einginge.

Ebenso kann der Aufwand bzw. Verlust L_{ij} eines Agenten i aus einer Kooperationsbeziehung mit Agent j quantifiziert werden:

$$L_{ij} := (1 - n_i) * n_j * E \qquad \text{(Gl. 7.2)}$$

Auch der Faktor E stellt eine Konstante dar und bezeichnet die maximal mögliche Hilfeleistung, die ein gänzlich unbedürftiger Agent i (mit $n_i = 0$) für einen maximal bedürftigen Agenten j (mit $n_j = 1$) in einer Kooperationsbeziehung erbringen könnte.

Der Gesamtnutzen des Agenten i aus einer einmaligen Durchführung des Support Games mit Agent j hängt in entscheidender Weise vom Verhalten beider Agenten ab. Beide Agenten haben die Möglichkeit entweder zu kooperieren (C) oder zu defektieren (D). Kooperiert Agent i, so entsteht für ihn zumindest der in Gleichung Gl. 7.2 dargestellte Aufwand. Ein Gewinn ergibt sich für Agent i jedoch nur dann, wenn Agent j sich ebenfalls für die kooperative Strategie entscheidet. In diesem Fall tritt für Agent i ein Gewinn in Höhe von G_{ij} ein (s. Gleichung Gl. 7.1). Falls beide Agenten kooperieren, entsteht für Agent i also ein Gesamt-

nutzen in Höhe von G_{ij} - L_{ij} (analog ergibt sich für Agent j ein Gesamtnutzen in Höhe von G_{ji} - L_{ji}). Wählt ein Agent hingegen die defektive Strategie, entsteht für ihn auf der einen Seite kein Aufwand, auf der anderen Seite für den potentiellen Kooperationspartner aber auch kein Nutzen.

Die folgende Tabelle stellt die Nutzenbilanz zweier Agenten A_i und A_j im Support Game des Modells *Solidarnetzwerk* unter Berücksichtigung der beiden alternativen Entscheidungen *C* und *D* in der Übersicht dar.

	A_j wählt *C*		**A_j wählt *D***	
	Nutzen für A_i	Nutzen für A_j	Nutzen für A_i	Nutzen für A_j
A_i wählt *C*	$R_{ij} = G_{ij} - L_{ij}$	$R_{ji} = G_{ji} - L_{ji}$	$S_{ij} = - L_{ij}$	$T_{ji} = G_{ji}$
A_i wählt *D*	$T_{ij} = G_{ij}$	$S_{ji} = - L_{ji}$	$P_{ij} = 0$	$P_{ji} = 0$

Abbildung 7.2: Nutzen-Matrix im Support Game des Modells Solidarnetzwerk

Der Nutzen, der sich in den Fällen *(CC)*, *(DC)*, *(CD)* und *(DD)* ergibt, wird dabei in Anlehnung an die in der Spieltheorie gebräuchliche Terminologie als **R**eward, **T**emptation, **S**ucker's payoff und **P**unishment bezeichnet.

Aus der für das Support Game zugrunde gelegten Nutzenmatrix lässt sich sofort ableiten, daß Defektion *(D)* gegenüber der Kooperation *(C)* im vorliegenden Fall grundsätzlich die dominante Strategie ist, denn es gilt: $T_{ij} > R_{ij}$ und $P_{ij} > S_{ij}$.

Damit es unter den gegebenen Bedingungen im Falle einer beidseitigen Kooperation *(CC)* zu einem beidseitigen, positiven Nutzen kommen kann, muß folgende Bedingung für beide Agenten erfüllt sein:

$$G_{ij} > L_{ij} \quad \Leftrightarrow \quad n_i * (1 - n_j) * B > (1 - n_i) * n_j * E \qquad \text{(Bed. 7.1)}$$

Die Bedingung Bed. 7.1 ist daher als Minimalkriterium anzusehen, das mindestens erfüllt sein muß, um zwei rational egoistisch handelnde Agenten dazu zu bewegen, eine Solidarbeziehung miteinander eingehen. Dabei gilt die Annahme, daß rationale Egoisten nur dann eine Kooperationsbeziehung miteinander eingehen, wenn für beide Seiten ein positiver Nutzen aus dieser Beziehung entsteht.

Setzt man nun die Bedingung Bed. 7.1 für das Support Game als gegeben voraus, so erhält man ein Gefangenendilemma (Prisoners' Dilemma), weil damit nun $R_{ij} > P_{ij}$ gilt. Die Bedingung Bed. 7.1 soll aus diesem Grund als *PD-Bedingung* bezeichnet werden und deren Erfüllung als notwendige Voraussetzung für die Entstehung einer bilateralen Solidarbeziehung zwischen je zwei rationalen Egoisten gelten. Ist die Bedingung Bed. 7.1 für beide Agenten im Rahmen eines Support Games nicht erfüllt, so entscheiden sich beide Agenten für die Alternative Defektion *(D)*.

Die PD-Bedingung reicht allerdings noch nicht aus, um rationale Egoisten nachhaltig zur Kooperation mit anderen Agenten zu bewegen. Kurzfristig betrachtet bleibt Defektion die dominante Strategie. In einem wiederholten Spiel mit einer entsprechenden Anzahl an Iterationen könnte sich allerdings ein rationaler Entscheider davon abbringen lassen, seinen Partner kurzfristig auszubeuten, da er

sich sicherlich darüber im Klaren sein wird, daß der Partner auf längere Sicht nur dann kooperieren wird, wenn er auch selbst im Sinne einer Gegenleistung Hilfe erhalten wird. Trotzdem wird sich eine bilaterale Kooperationsbeziehung im Vergleich zur Ausbeutung eines Partners erst dann lohnen, wenn der Zeithorizont für eine Kooperationsbeziehung ausreichend lang gewählt wird. Wie lang muß also eine Interaktion im Modell *Solidarnetzwerk* grundsätzlich angesetzt werden, damit sich ein rationaler und egoistischer Agent dazu bewegen lässt, sich von der kurzfristig effektiveren Strategie der Defektion abzuwenden und Kooperation zu wählen?

Für die Beantwortung dieser Frage wird die Fortsetzungswahrscheinlichkeit α eingeführt, die angibt, wie hoch die Wahrscheinlichkeit ist, daß zwei Agenten ihre Interaktionsbeziehung in der nächsten Spielrunde weiter fortsetzen. Je höher die Fortsetzungswahrscheinlichkeit α dabei angesetzt wird, um so länger wird eine Interaktionsbeziehung zweier Agenten erwartungsgemäß andauern. Umgekehrt darf die Fortsetzungswahrscheinlichkeit α einen gewissen Schwellwert nicht unterschreiten, um bei rationalen und egoistischen Agenten im Sinne einer längerfristigen Beziehung eine Kooperation zu erreichen. Friedman (1986) und Taylor (1987) definieren in diesem Zusammenhang einen Schwellwert für die Fortsetzungswahrscheinlichkeit α aus einer Gleichgewichtsbetrachtung heraus in der folgenden Weise:

$$\alpha \geq L_{ij} / G_{ij} \qquad \text{(Bed. 7.2)}$$

Natürlich gilt auch hier, daß es für einen Agenten i unattraktiv ist, einen Agenten j zu unterstützen, wenn die Verluste L_{ij} aus der Kooperationsbeziehung mit Agent j größer sind als die daraus entstehenden Gewinne G_{ij}. Wenn aber die Gewinne aus einer derartigen Beziehung größer sind als die Verluste, so wird auch eine Wahrscheinlichkeit $\alpha \leq 1$ existieren, mit der Agent i bereit ist, mit Agent j zu kooperieren. Hierbei ist zu beachten, daß das Verhältnis L_{ij} / G_{ij} maßgeblich sowohl durch die Bedürftigkeitsklassen der beteiligten Agenten, als auch durch die beiden Konstanten B und E beeinflusst wird.

Die Fortsetzungswahrscheinlichkeit α wird mit Hilfe der Wahrscheinlichkeit q abgeschätzt, daß ein Agent eine Migrationsoption erhält. Eine pessimistische Schätzung für die Fortsetzungswahrscheinlichkeit ergibt sich dadurch mit Hilfe der folgenden Beziehung:

$$\alpha := (1 - q) * (1 - q) \qquad \text{(Gl. 7.3)}$$

$1 - q$ gibt dabei die Wahrscheinlichkeit dafür an, daß ein Agent keine Migrationsoption erhält und sich damit in der nächsten Spielrunde immer noch an derselben Position befindet. Stabilität tritt also dann ein, wenn beide an einer Kooperationsbeziehung beteiligten Agenten keine Migrationsoption erhalten. Als pessimistisch kann diese Abschätzung deswegen eingestuft werden, weil sie davon ausgeht, daß ein Agent jede sich bietende Migrationsoption auch ausnutzen und seine aktuelle Position verändern wird. Dies wird allerdings nicht der Fall sein, da

für die Annahme einer Migrationsoption noch weitere Bedingungen erfüllt sein müssen (s. Abschnitt 7.1.2.4).

Jeder Agent überprüft im Rahmen seiner Nachbarschaft, ob beide Bedingungen Bed. 7.1 und Bed. 7.2 erfüllt sind. Eine Solidarbeziehung zwischen zwei Nachbarn kommt genau dann zustande, wenn für beide Nachbarn sowohl die PD-Bedingung Bed. 7.1 als auch die sog. COOP-Bedingung Bed. 7.2 erfüllt sind.

7.1.2.4 Die Qualität der aktuellen Position eines Agenten

Sowohl die PD- als auch die COOP-Bedingung schränken die Kooperationsbereitschaft der Agenten in ganz entscheidender Weise ein. Agenten einer bestimmten Bedürftigkeitsklasse werden nur mit einer Auswahl an Agenten anderer Bedürftigkeitsklassen zu kooperieren bereit sein.

Jeder Agent nimmt in jeder Spielrunde eine Bewertung seiner aktuellen Nachbarschaft bzw. sozialen Position vor. Zu diesem Zweck führt er das Support Game simultan mit allen Agenten seiner vier Nachbarfelder durch. Die Qualität Q_i der aktuellen Position eines Agenten i ergibt sich dabei als Summe der Ausschüttungen U_{ij} mit $j \in \{ 1, ..., 4 \}$, die aus den Support Games mit den auf den Nachbarfeldern befindlichen Agenten resultieren. Ist zum betrachteten Zeitpunkt ein Nachbarfeld nicht durch einen Agenten besetzt, so fließt dieses Feld mit dem Wert 0 in die Summe ein. Es gilt:

$$Q_i := \sum_{j=1}^{4} U_{ij} \qquad \text{(Gl. 7.4)}$$

Die bestmögliche soziale Position eines Agenten i liegt innerhalb einer Nachbarschaft, in der sich vier Agenten der besten Bedürftigkeitsklasse befinden, deren Mitglieder gerade noch bereit sind, eine Kooperationsbeziehung mit dem Agenten i einzugehen. In einer derartigen Nachbarschaft erzielt ein Agent i die unter Berücksichtigung der PD- und COOP-Bedingung maximal mögliche Gesamtausschüttung Q_i^{max} aus vier simultan gespielten Support Games.

Die schlechtest mögliche soziale Position ergibt sich dann, wenn ein Agent i ausschließlich von leeren Feldern umringt ist oder alle relevanten Felder durch Mitglieder derjenigen Bedürftigkeitsklassen besetzt sind, mit denen aufgrund der PD- und COOP-Bedingung keine Kooperationsbeziehungen zustande kommen. Man erkennt leicht, daß im vorliegenden Fall die schlechtest mögliche Ausschüttung Q_i^{min} immer den Wert 0 annehmen wird.

7.1.2.5 Die Zufriedenheit der Agenten

Unmittelbar mit der Qualität der aktuellen sozialen Position korreliert die Zufriedenheit eines Agenten. Im Rahmen des Modells *Solidarnetzwerk* besteht die An-

nahme, daß ein Agent entweder zufrieden oder unzufrieden mit seiner sozialen Position sein kann. Feinere Abstufungen werden hier zunächst nicht berücksichtigt. Zufrieden wird ein Agent im vorliegenden Modellierungsansatz genau dann sein, wenn die Qualität seiner aktuellen sozialen Position einen bestimmten Schwellwert überschreitet. Dieser Schwellwert liegt dabei zwischen der Qualität der bestmöglichen und der schlechtest möglichen sozialen Position und wird in eindeutiger Weise durch das sog. *Minimalniveau* definiert. Ein Agent *i* gilt als zufrieden mit seiner aktuellen sozialen Position, falls die folgende Bedingung erfüllt ist:

$$Q_i \geq Minimalniveau * Q_i^{max} \qquad \text{(Bed. 7.3)}$$

Für die Modellgröße *Minimalniveau* wird der Wert 0.5 angenommen. Damit zeigt sich ein Agent bereits als zufrieden, wenn er an seiner aktuellen Position die Hälfte seiner Maximalausschüttung erreicht. Ist die Bedingung 7.3 nach Auswertung der Nachbarschaft für einen Agenten nicht erfüllt, so ist er unzufrieden. Die Zufriedenheit bzw. Unzufriedenheit eines Agenten wirkt sich maßgeblich auf dessen Migrationsverhalten aus.

7.1.2.6 Migration

Im Rahmen einer zufällig zugeteilten Migrationsoption erhält ein Agent von Zeit zu Zeit die Möglichkeit, seine Position in der Umwelt zu verändern. Dabei wird er allerdings eine Migrationsoption nicht immer nutzen, um seine aktuelle Position tatsächlich zu verlassen, sondern zunächst einige Überlegungen dazu anstellen, ob und wohin ein Umzug erfolgen kann. Konkret geht ein Agent nach Erhalt einer Migrationsoption in der folgenden Weise vor:

Zunächst einmal werden alle unbesetzten Felder innerhalb des Migrationsfensters auf ihre Qualität hin untersucht. Zu diesem Zweck spielt der Agent, analog zur Bewertung seiner eigenen Position, das Support Game mit allen angrenzenden Nachbarn. Dasjenige Feld, das sich nach der vollständigen Analyse des Migrationsfensters hinsichtlich der Qualität der Nachbarschaft als das beste herausgestellt hat, wird dann als potentielles Ziel eines Umzugs in Betracht gezogen. Ob der betrachtete Agent dann tatsächlich auf dieses Feld umzieht, hängt im wesentlichen von der Zufriedenheit des Agenten, der Qualität seiner aktuellen sozialen Position sowie der Qualität des Umzugsziels ab.

Ein zufriedener Agent wird nur dann seine aktuelle Position verlassen und sein potentielles Umzugsziel auch tatsächlich beziehen, wenn die Qualität des Zielfeldes mindestens der Qualität des aktuellen Feldes entspricht.

Anders stellt sich die Situation bei einem unzufriedenen Agenten dar. Ein unzufriedener Agent zeigt wesentlich mehr Risikobereitschaft als ein zufriedener Agent und bezieht sein potentielles Sprungziel immer und selbst dann, wenn es im Vergleich zur aktuellen Position eine geringere Qualität aufweist. Für einen unzu-

friedenen Agenten stellt sich ein derartiges Zielfeld in gewisser Weise als Sprungbrett dar, das ihm die Möglichkeit gibt, von dort aus im Rahmen der nächsten Migrationsschritte in eine akzeptable Nachbarschaft zu gelangen. Für den Erhalt einer derartigen Veränderungsmöglichkeit nimmt ein unzufriedener Agent sogar eine kurzfristige Verschlechterung seiner Situation in Kauf.

7.1.3 Der grundsätzliche Spielablauf

Das Spiel beginnt mit einer Initialisierungsphase. Im Rahmen dieser Initialisierungsphase werden zunächst einmal das Spielfeld angelegt für jede Bedürftigkeitsklasse jeweils 35 Agenten konfiguriert. Die Agenten werden zu Beginn willkürlich und ungeachtet ihrer Bedürftigkeitsklassen auf dem Spielfeld verteilt. Auf dem Spielfeld befinden sich nach der Initialisierungsphase insgesamt 315 Agenten. Zudem bleiben 126 Felder auf dem Spielfeld frei, die anschließend durch die Agenten für Umzüge genutzt werden können.

In der Folge läuft das Spiel rundenbasiert ab. Zu Beginn einer neuen Spielrunde wird zunächst einmal festgelegt, in welcher Reihenfolge die Agenten ihre Spielzüge durchführen dürfen. Die Reihenfolge wird dabei von Runde zu Runde mit Hilfe eines Zufallsprozesses neu bestimmt, um alle Agenten gleichberechtigt zu behandeln. Eine Sequentialisierung der Aktivitäten der einzelnen Agenten ist aus dem Grund erforderlich, weil simultan ausgeführte Migrationen mehrerer Agenten zwangsläufig zu Konflikten auf dem Spielfeld führen würden.

Nachdem die Reihenfolge der Agenten für eine Runde bestimmt wurde, wird mit Hilfe eines Zufallsprozesses ausgewürfelt, ob der betreffende Agent eine Migrationsoption erhält. Wird ihm aufgrund des Ergebnisses des Zufallsexperiments eine Migrationsoption verwehrt, verbleibt der Agent ohne weitere Aktivitäten auszuführen auf seiner aktuellen Position. Erhält der Agent hingegen eine Migrationsoption, wird er dazu aufgefordert, seine entsprechenden Spielaktivitäten auszuführen. Hierzu geht der Agent in der folgenden Weise vor:

Zunächst einmal spielt er mit allen aktuell angrenzenden Nachbarn das Support Game, um die Qualität seiner Position zu bestimmen. Unmittelbar mit der Qualität der Position korreliert die Zufriedenheit des Agenten, die im darauf folgenden Schritt neu berechnet wird. Damit sind alle Aktivitäten im Zusammenhang mit der Bewertung der aktuellen Situation bzw. Position abgeschlossen und alle Informationen vorhanden, die für die Auswertung der Migrationsoption erforderlich sind. Im Zuge der Auswertung seiner Migrationsoption führt der Agent als nächstes eine Analyse aller freien Felder innerhalb seines Migrationsfensters durch, um festzustellen, welches der Felder das präferierte Migrationsziel ist. Liegt dieses Ziel fest, so entscheidet er schließlich unter Berücksichtigung der Qualität seiner aktuellen Position, seiner aktuellen Zufriedenheit, sowie der Qualität seines präferierten Sprungziels, ob eine Migration tatsächlich in Frage kommt. Entschließt sich der Agent in diesem Zusammenhang für einen Umzug, so wechselt er tatsächlich seine aktuelle Position auf dem Spielfeld. Lehnt der Agent nach

Berücksichtigung aller Entscheidungsparameter einen Umzug trotz zugeteilter Migrationsoption ab, so verbleibt er an seiner ursprünglichen Position und beendet seine Aktivitäten.

Sind im Rahmen einer Spielrunde alle Agenten aufgefordert worden, ihre Spielaktivitäten durchzuführen, werden zum Ende der Spielrunde hin noch einmal alle Agenten dazu aufgerufen, die Qualität ihrer aktuellen Position neu zu berechnen, um statistische Informationen, wie etwa den Gesamtnutzen des Netzwerks, zu erheben. Zudem wird die Anzahl an Agenten festgehalten, die in der vorliegenden Spielrunde tatsächlich ihren Aufenthaltsort gewechselt haben, sowie die Anzahl an zufriedenen und unzufriedenen Spielern protokolliert.

Damit sind alle Aktivitäten, die im Rahmen einer Spielrunde durchgeführt werden müssen, vollständig erledigt und es kann eine neue Spielrunde beginnen. Die Anzahl der Spielrunden kann dabei zu Beginn frei gewählt werden.

Die Abbildung 7.2 stellt die wesentlichen Parameter des Modells *Solidarnetzwerk* noch einmal in der Übersicht dar:

Bezeichner	**Initialisierung**	**Erläuterung**
Spielfeld	21 × 21 Felder	Räumliches Bezugssystem der Agenten (Torus)
Interaktionsfenster	von Neumann-Nachbarschaft	Positionen in der Umwelt, zu denen eine Nachbarschaftsbeziehung existiert
Migrationsfenster	11 × 11 Felder	Positionen in der Umwelt, die im Rahmen einer Migration grundsätzlich als Zielfelder in Frage kommen
n_i	$n_i \in \{ 0.1, ..., 0.9\}$	Bedürftigkeit eines Agenten i 35 Agenten je Bedürftigkeitsklasse
B	*4*	theoretischer Maximal-Nutzen einer Kooperationsbeziehung
E	*1*	theoretischer Maximal-Aufwand einer Kooperationsbeziehung
Minimalniveau	*0.5*	Stellgröße für die Zufriedenheit der Agenten
q	*0.05*	Wahrscheinlichkeit für den Erhalt einer Migrationsoption
α	$(1 - q) * (1 - q)$	Fortsetzungswahrscheinlichkeit einer Nachbarschaftsbeziehung

Abbildung 7.2: Die wesentlichen Parameter des Modells Solidarnetzwerk

7.2 Entwurf und Implementierung auf der Grundlage des Referenzmodells *PECS*

Der folgende Abschnitt beschäftigt sich mit dem Implementierungskonzept für das Modell *Solidarnetzwerk*. Ausgehend von der Grundstruktur des Modells werden dazu alle wesentlichen Modellkomponenten hinsichtlich ihres Aufbaus und ihres dynamischen Verhaltens betrachtet. Darüber hinaus wird anhand eines typischen Szenarios das dynamische Zusammenspiel der *PECS*-Komponenten, aus denen die Agenten bestehen, beschrieben.

Für die Realisierung des Modells wurde das Referenzmodell *PECS* (s. Kapitel 6) sowie das Simulationssystem *Simplex3* mit der zugehörigen Modellbeschreibungssprache *Simplex-MDL* (Schmidt, 2000) verwendet. *Simplex-MDL* liegt ein systemtheoretischer Ansatz zur Modellbeschreibung zugrunde und unterstützt einen komponentenorientierten und hierarchischen Modellaufbau. Im Zusammenspiel mit dem Referenzmodell *PECS* wird dadurch ein komfortabler Aufbau agentenbasierter Simulationsmodelle möglich.

7.2.1 Die Grundstruktur des Modells *Solidarnetzwerk*

Das Modell *Solidarnetzwerk* besteht aus einer Menge von unabhängig und nebenläufig arbeitenden Modellkomponenten, die sich über insgesamt drei Hierarchieebenen erstrecken.

Auf der höchsten Hierarchieebene ist das Gesamtmodell *Solidarnetzwerk* angesiedelt. Das Modell *Solidarnetzwerk* umfasst insgesamt 315 Agenten, sowie eine Komponente *Umwelt*, die das Spielfeld zur Verfügung stellt und die grundlegende Steuerung des Spielablaufes übernimmt. Die Agenten des Modells sind nach den Vorgaben des Referenzmodells *PECS* konzipiert und bestehen auf der nächst tieferen Hierarchieebene aus den Subkomponenten *Sensor*, *Wahrnehmung*, *Wissen*, *Emotion*, *Status*, *Verhalten* und *Aktor*.

Als zentrale Steuerungsinstanz versorgt die Komponente *Umwelt* die Agenten mit wesentlichen Informationen zur aktuellen Spielsituation und steuert die Reihenfolge, in der die Agenten ihre Spielaktivitäten durchführen. Als wesentliche Eingangsinformationen stellt die Komponente *Umwelt* den Agenten ihre Initialposition auf dem Spielfeld (*InitRow*, *InitCol*), die aktuelle Situation auf dem Spielfeld (*Grid*), sowie alle erforderlichen Parameter des Support Games zur Verfügung. Hierzu zählen insbesondere die Bedürftigkeitsklasse (*InitClass*), die den Agenten zu Beginn zugeteilt wird, die Ausschüttungen, die ein Agent auf der Grundlage seiner Bedürftigkeitsklasse durch eine Kooperationsbeziehung mit anderen Agenten erzielen kann, sowie der absolute Schwellwert hinsichtlich der Qualität der aktuellen Position, ab dem ein Agent mit seiner Position in der Umwelt zufrieden ist. Die Spielinformationen werden der Einfachheit halber in der Komponente *Umwelt* vorgegeben, da sie während des Spielablaufes als statisch

betrachtet und für alle Agenten einer ganz bestimmten Bedürftigkeitsklasse in identischer Weise angesetzt werden. Eine individuelle Berechnung der Spielparameter in den einzelnen Agenten ist aus diesen Gründen nicht erforderlich.

Darüber hinaus fordert die Komponente *Umwelt* die Agenten mit Hilfe von Steuerungssignalen auf, bestimmte Aktivitäten durchzuführen. Hierzu zählen etwa die Signale *MIGRAT* oder *PLAYGAME*, die die Agenten dazu aufrufen, eine erhaltene Migrationsoption auszuwerten bzw. gegen Ende einer Runde das Support Game in der aktuellen Nachbarschaft zu spielen.

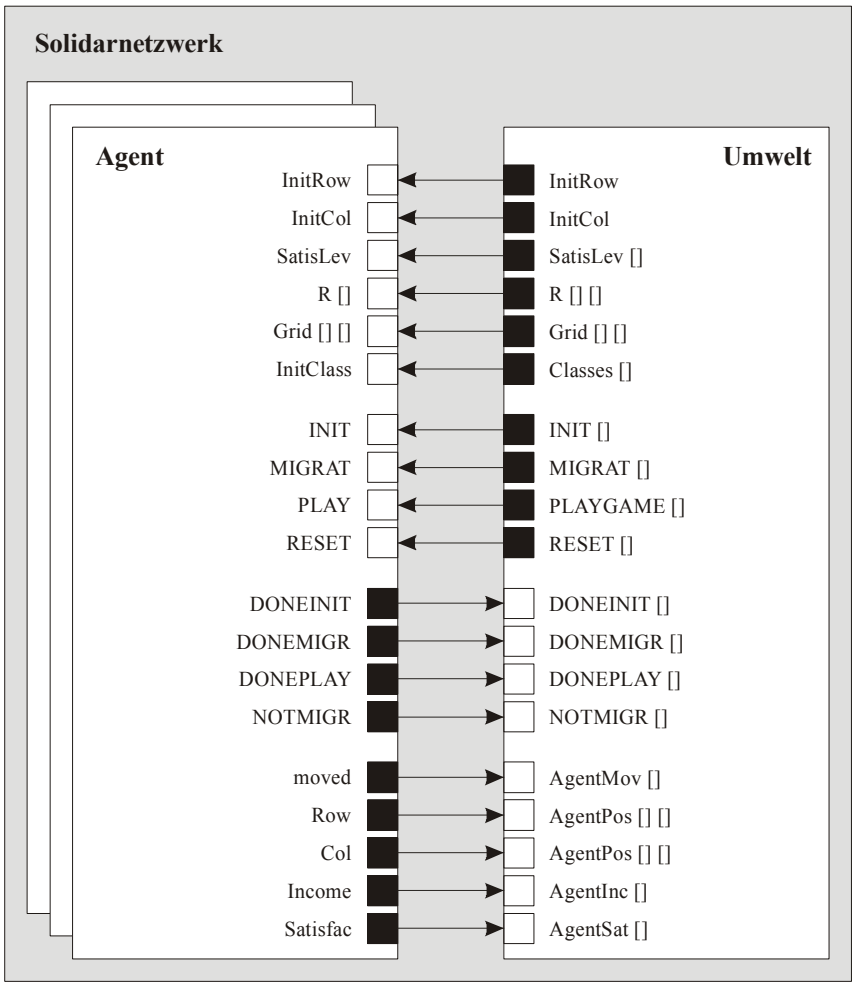

Abbildung 7.3: Die grundlegende Struktur des Modells Solidarnetzwerk

Die Agenten geben im Gegenzug ihre neue Position (*Row*, *Col*), die Qualität ihrer aktuellen Nachbarschaft (*Income*), ihre Zufriedenheit, sowie die Information, ob eine Bewegung in der aktuellen Spielrunde stattgefunden hat (*moved*), zur weiteren Bearbeitung an die Komponente *Umwelt* weiter. Hinzu kommen Steuerungsrückmeldungen (z. B. *DONEMIGR*, *DONEPLAY*), die die Umwelt über getroffene Entscheidungen bzw. den Abschluss der Aktivitäten der Agenten informieren.

Im Vergleich zur grundlegenden Komponentenstruktur des Referenzmodells *PECS* fehlt im Modell *Solidarnetzwerk* die Komponente *Connector*. Diese Komponente kann im vorliegenden Modell entfallen, da aufgrund der Modellspezifikation nach Hegselmann und Flache alle Spieler perfekt informiert und mit identischen Verhaltensregeln ausgestattet sind, so daß für eine tatsächliche Durchführung des Support Games keine direkte Interaktion im Sinne einer Kommunikation zwischen den einzelnen Agenten erforderlich ist.

Die Abbildung 7.3 zeigt den grundlegenden Aufbau der Strukturkomponente *Solidarnetzwerk*.

7.2.2 Die Komponente *Umwelt*

Die Komponente *Umwelt* stellt das räumliche Bezugssystem für die Agenten des Modells *Solidarnetzwerk* zur Verfügung. Sie verwaltet grundlegende Spielinformationen, steuert den Ablauf des Spiels und berechnet statistische Informationen über das Netzwerk.

7.2.2.1 Grundlegende Modellelemente der Komponente *Umwelt*

Innerhalb der Komponente *Umwelt* werden eine Reihe von Modellelementen verwaltet, die zum Teil als wesentliche Eingangsinformation an die Agenten des Modells *Solidarnetzwerk* weitergeleitet werden.

Von grundlegender Bedeutung ist dabei das Spielfeld, auf dem sich die Agenten befinden und bewegen. Das Spielfeld wird als zweidimensionales Array gespeichert. Die einzelnen Elemente des Arrays repräsentieren die entsprechenden Felder auf dem Spielfeld und enthalten als wesentliche Information für die Durchführung des Support Games die Bedürftigkeitsklassen der jeweils auf den Feldern befindlichen Agenten.

Darüber hinaus stellt die Komponente *Umwelt* die Nutzenmatrix für das Support Game zur Verfügung. Die Nutzenmatrix enthält dabei in jedem Element *(i, j)* die Auszahlung G_{ij}, die sich aus einer Kooperationsbeziehung eines Agenten der Bedürftigkeitsklasse *i* mit einem Agenten der Bedürftigkeitsklasse *j* ergibt. Diese Nutzenmatrix gilt als Eingangsinformation in identischer Weise für alle Agenten.

Als weitere Eingangsinformation wird in der Komponente *Umwelt* je Bedürftigkeitsklasse der zugehörige Schwellwert für die Zufriedenheitsberechnung der Agenten bereitgestellt. Hierzu werden in einem Array die absoluten Werte für die Qualität der Umgebung hinterlegt, die die Agenten der einzelnen Bedürftigkeitsklassen mindestens erreichen müssen, damit sie mit ihrer aktuellen Nachbarschaft zufrieden sind.

Die Komponente *Umwelt* dient darüber hinaus der Generierung statistischer Informationen über das Netzwerk. Am Ende jeder Spielrunde wird innerhalb der Komponente *Umwelt* protokolliert, wie hoch der Gesamtnutzen des Netzwerks ist,

wie viele Agenten insgesamt zufrieden mit ihrer aktuellen Position sind und wie viele Agenten sich in der letzten Spielrunde bewegt haben.

7.2.2.2 Die grundlegende Arbeitsweise der Komponente *Umwelt*

Der Ablauf innerhalb der Komponente *Umwelt* zerfällt in zwei aufeinander folgende Phasen. In der ersten Phase erfolgt die Initialisierung des Spiels, während in der zweiten Phase das eigentliche Spiel rundenbasiert durchgeführt wird.

Die Initialisierungsphase umfasst die folgenden Aktivitäten:

- Berechnung der Nutzenmatrix für das Support Game

  Um die Nutzenmatrix aufzustellen, wird für alle möglichen Kombinationen von Bedürftigkeitsklassen das Support Game gespielt und im Falle einer Kooperation der resultierende Nutzen in der Nutzenmatrix eingetragen. Der in Abbildung 7.4 dargestellte Codeabschnitt zeigt die verwendete *MDL*-Prozedur zur Berechnung der Nutzenmatrix.

```
PROCEDURE CalcPay(INT:b, INT:e, REAL:alpha
                  --> ARRAY [a][a] REAL)
DECLARE
  ARRAY [a][a] p (REAL),
  Gij(REAL),
  Gji(REAL),
  Lij(REAL),
  Lji(REAL)

BEGIN
  FOR i FROM 1 TO 9 REPEAT
    FOR j FROM 1 TO 9 REPEAT
      Gij := i/10 * (1 - j/10) * b;
      Gji := j/10 * (1 - i/10) * b;
      Lij := (1 - i/10) * j/10 * e;
      Lji := (1 - j/10) * i/10 * e;

      # Ueberpruefen der PD- und COOP-Bedingung
      # --------------------------------------
      IF ((Gij-Lij > 0) AND (Gji-Lji > 0)) AND
          (alpha >= (Lij / Gij)) AND (alpha >= (Lji/Gji))
      DO
        p[i][j] := Gij - Lij;
      END
      ELSE DO
        p[i][j] :=0;
      END
    END_LOOP
  END_LOOP
RETURN (ARRAY p);
END_PROC
```

Abbildung 7.4: MDL-Prozedur zur Berechnung der Nutzenmatrix für das Support Game

- Berechnung der Mindestqualität für die Bestimmung der Zufriedenheit

  Für jede Bedürftigkeitsklasse wird in der Komponente *Umwelt* eine Mindestqualität hinterlegt, die ein Agent dieser Bedürftigkeitsklasse hinsichtlich seiner aktuellen Position erreichen muss, um zufrieden zu sein. Die Berechnung der Mindestqualität setzt dabei auf der Nutzenmatrix auf. Für jede in der Nutzenmatrix vertretene Bedürftigkeitsklasse wird zunächst einmal der Maximalwert für den möglichen Nutzen eines Agenten dieser Klasse bestimmt. Anhang des Maximalnutzens wird dann auf der Grundlage der Gleichung Gl. 7.4 und des rechten Teils der Bedingung Bed. 7.3 die Mindestqualität je Bedürftigkeitsklasse festgelegt.

- Verteilung der Agenten über das Spielfeld

  Die Komponente *Umwelt* weist den Agenten im Rahmen der Initialisierung auf der Grundlage eines Zufallsprozesses individuelle Startpositionen zu.

- Anstoß der Initialisierung der Agenten

  Am Ende der Initialisierungsphase werden die Agenten dazu aufgefordert, ihre Startwerte aus der Komponente *Umwelt* zu übernehmen.

Im Anschluss an die Initialisierungsphase beginnt die eigentliche Spielphase, die in aufeinander folgende Runden mit jeweils identischem Ablauf aufgeteilt ist. Zu Beginn einer neuen Spielphase wird mit Hilfe eines Zufallsexperimentes die Reihenfolge festgelegt, nach der die Agenten ihre Aktivitäten in dieser Spielrunde ausführen dürfen. Entsprechend dieser Reihenfolge wird daraufhin für jeden Agenten ausgewürfelt, ob er eine Migrationsoption erhält. Ist dies der Fall, so wird der entsprechende Agent mit Hilfe einer Nachricht aktiviert, um seine Migrationsoption zu bearbeiten. Weiterhin werden die ursprünglichen Koordination des Agenten in der Komponente *Umwelt* gespeichert, um sie bei einer späteren Modifikation zur Verfügung zu haben. Der Agent meldet sich nach vollständiger Ausführung seiner Spielaktivitäten bei der Komponente *Umwelt* zurück und gibt dabei gegebenenfalls die neue Position bekannt, für die er sich im Rahmen seiner Migrationsoption entschieden hat. Die Komponente *Umwelt* vollzieht daraufhin den Positionswechsel durch eine entsprechende Modifikation der Koordinaten des Agenten, schließt dann die Aktivitäten für den aktiven Agenten ab und fährt mit dem nächsten Agenten fort, bis alle Agenten in der laufenden Spielrunde abgearbeitet wurden.

Zum Abschluss einer Spielrunde werden alle Agenten mit Hilfe des Signals PLAYGAME noch einmal dazu aufgefordert, das Support Game in ihrer aktuellen Nachbarschaft zu spielen und dadurch die Qualität ihrer Positionen zu aktualisieren. Die Qualität der aktuellen Position, die Information, ob sich die Position eines Agenten in der letzten Spielrunde verändert hat, sowie die Zufriedenheit der Agenten werden zum Ende der Spielrunde hin durch die Komponente *Umwelt* von allen Agenten abgelesen und dort zur Erstellung einer Rundenstatistik verwendet. Die Komponente Umwelt verfügt dabei der Einfachheit halber über Lesezugriff auf die entsprechenden Modellgrößen der einzelnen Agenten.

7.2.3 Die Architektur der Agenten im Modell *Solidarnetzwerk*

Die folgenden Abschnitte befassen sich mit dem Aufbau und der Funktionsweise der Agenten im Modell *Solidarnetzwerk*. Wie die Abbildung 7.5 zeigt, besteht ein Agent im vorliegenden Modell aus den an *PECS* angelehnten Komponenten *Sensor*, *Wahrnehmung*, *Wissen*, *Emotion*, *Status*, *Verhalten* und *Aktor*. Die Komponente *Physis* entfällt im Vergleich zur Agentenarchitektur von *PECS*, da im Modell *Solidarnetzwerk* physische Eigenschaften von Agenten ohne Bedeutung sind.

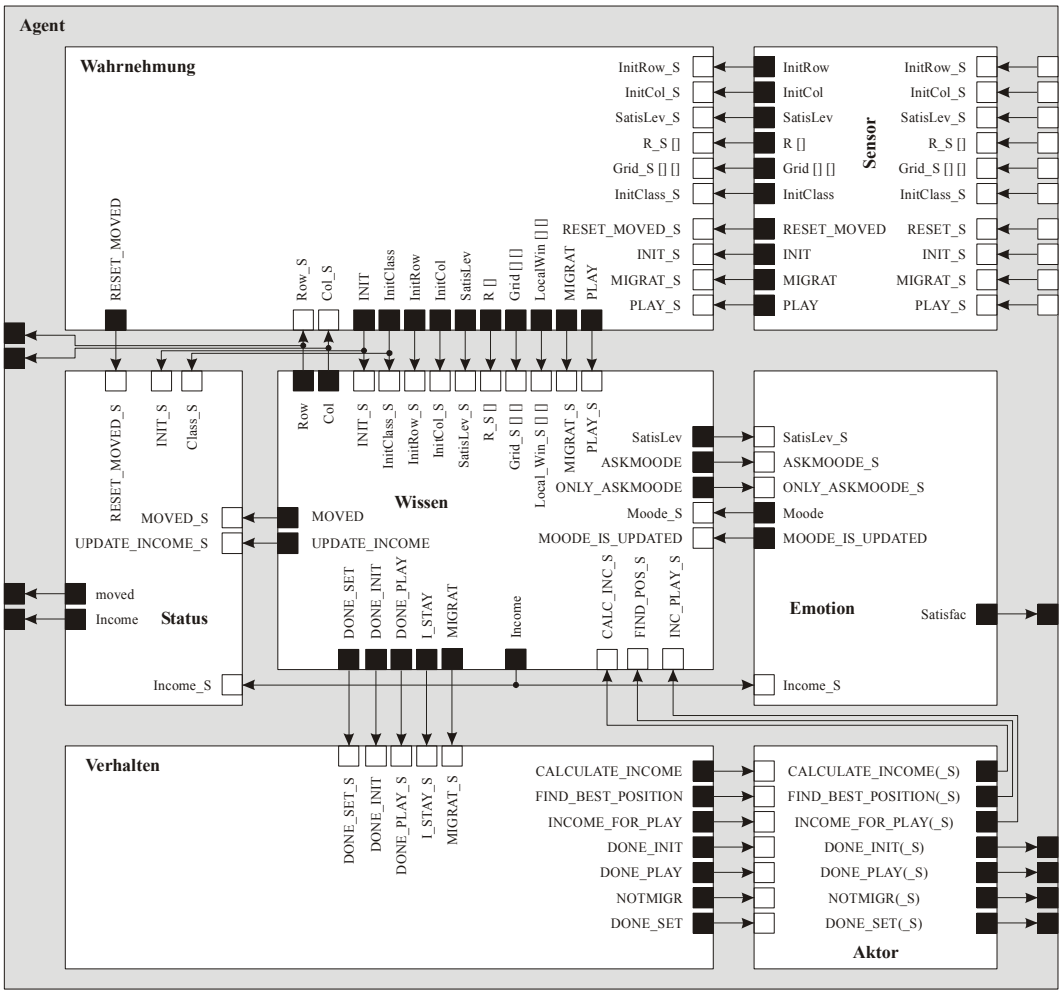

Abbildung 7.5 Der grundlegende Aufbau der Agenten im Modell Solidarnetzwerk

7.2.3.1 Die Komponente *Sensor*

Die Komponente *Sensor* stellt die Eingangsschnittstelle eines Agenten für Informationen aus der Umgebung dar und steht daher in enger Verbindung mit der Komponente *Umwelt*. Mit Hilfe der Komponente *Sensor* nimmt ein Agent im

Modell *Solidarnetzwerk* Initialisierungsdaten, Informationen über die aktuelle Spielsituation, sowie Steuerungsinformationen entgegen. Zu den Initialisierungsdaten gehören die initiale Position des Agenten (*InitRow_S*, *InitCol_S*), die Bedürftigkeitsklasse (*InitClass_S*), die Qualitätsschranke für die Bestimmung der Zufriedenheit (*SatisLev_S*), sowie der Nutzenvektor *R_S []*, der die Auszahlungen aus den Support Games mit Agenten aus allen Bedürftigkeitsklassen beinhaltet. Als Steuerungsinformationen erhält die Komponente *Sensor* von der Komponente *Umwelt* die Signale *INIT_S*, *MIGRAT_S*, *PLAY_S* und *RESET_S*. *INIT_S* bewirkt dabei, dass ein Agent seine Initialisierungsinformationen aus der Umwelt in den internen Zustand übernimmt. Mit Hilfe des Signals *MIGRAT_S* informiert die Komponente Umwelt einen Agenten darüber, dass er eine Migrationsoption erhalten hat. Das Signal *PLAY_S* fordert einen Agenten dazu auf, auf der Grundlage des Support Games seine aktuelle Nachbarschaft auszuwerten. Eine Initialisierung der Bewegungsinformationen eines Agenten zu Beginn einer neuen Runde bewirkt das Signal *RESET_S*.

Alle Informationen, die die Komponente *Sensor* von außen erhält, werden ohne Ausnahme und auch ohne Veränderung an die Komponente *Wahrnehmung* zur weiteren Verarbeitung weitergeleitet.

7.2.3.2 Die Komponente *Wahrnehmung*

Die Komponente *Wahrnehmung* nimmt diejenigen Informationen entgegen, die durch die Komponente *Sensor* bereitgestellt werden, und leitet diese je nach Spielsituation an die internen Komponenten *Wissen* und *Status* weiter.

Liegt ein Initialisierungssignal aus der Umwelt vor, so werden an die Komponente *Wissen* die Anfangsposition (*InitRow*, *InitCol*), die zugewiesene Bedürftigkeitsklasse (*InitClass*), der Nutzenvektor, sowie der Qualitätsstandard zur Zufriedenheitsberechnung weitergeleitet.

Signalisiert die Komponente *Sensor* der Komponente *Wahrnehmung*, dass eine Migrationsoption vorliegt, so werden das aktuelle Spielfeld und das Migrationsfenster durch die Komponente *Wahrnehmung* an die Komponente *Wissen* weitergeleitet. Das Migrationsfenster wird in diesem Zusammenhang erst innerhalb der Komponente *Wahrnehmung* erstellt, wobei die aktuelle Position des Agenten, die in der Komponente *Wissen* gespeichert ist, von ausschlaggebender Bedeutung ist. Zur Berechnung des Migrationsfensters findet also in der Komponente *Wahrnehmung* ausgehend von der aktuellen Position des Agenten ein Selektionsprozess statt. Dabei wird die aktuelle Information über das Spielfeld zunächst einmal dupliziert. Anschließend werden auf dem duplizierten Spielfeld alle Felder ausgeblendet, die außerhalb des Migrationsfensters des Agenten liegen.

Im Falle eines *PLAY*-Signals aus der Umwelt, reicht die Komponente *Wahrnehmung* nur das aktuelle Spielfeld an die Komponente *Wissen* weiter. Eine Berechnung des Migrationsfensters ist in diesem Fall nicht erforderlich.

Ein *RESET*-Signal wird an die Komponente *Status* weitergeleitet, um dort die Bewegungsinformationen aus der letzten Runde für die anstehende, neue Runde zu initialisieren.

7.2.3.3 Die Komponente *Wissen*

Die Komponente *Wissen* stellt den zentralen Informationsspeicher eines Agenten dar. Sie nimmt die Informationen aus der Umwelt entgegen und aktualisiert damit das Umweltmodell des Agenten. Zudem verwaltet die Komponente *Wissen* Informationen über den Zustand des Agenten, wie etwa seine aktuelle Position, und führt, aufgefordert durch die Komponente *Aktor*, interne Aktionen aus.

Die Komponente *Wissen* steht in enger Verbindung mit der Komponente *Wahrnehmung*. Aus der Komponente *Wahrnehmung* empfängt die Komponente *Wissen* je nach Spielphase Informationen über die aktuelle Situation auf dem Spielfeld, im Falle einer Migrationsoption das Migrationsfenster, sowie zu Beginn die grundlegenden Parameter des Support Games einschließlich der Startposition des Agenten. Darüber hinaus erhält die Komponente *Wissen* die Steuerungsinformationen aus der Umwelt, die über die Komponenten *Sensor* und *Wahrnehmung* aufgenommen werden, und legt diese ebenfalls in ihrem Umweltmodell ab. Diese Steuerungsinformationen werden im Rahmen der Verhaltenssteuerung des Agenten verwendet, um die anstehenden Spielaktivitäten auszulösen.

Im Rahmen einer Migrationsoption berechnet ein Agent zunächst die Qualität seiner aktuellen Umgebung und analysiert daraufhin sein Migrationsfenster hinsichtlich eines potentiellen Migrationsziels. Beide Aktivitäten werden in Folge der internen Aktionen *UPDATE_INCOME* und *FIND_BEST_POSITION*, die durch die Komponente *Aktor* beauftragt werden, innerhalb der Komponente *Wissen* ausgeführt. Zu diesem Zweck werden in der Komponente *Wissen* zwei Prozeduren vorgehalten. Die Prozedur *CalcIncome* berechnet auf der Grundlage der aktuellen Nachbarschaftsbeziehungen des Agenten und des Nutzenvektors die Qualität der aktuellen Position. Die Prozedur *FindBestPosition* sucht ausgehend von der aktuellen Position des Agenten nach dem besten Sprungziel innerhalb des Migrationsfensters. Nachdem die Qualität der aktuellen Position festgestellt wurde, wird die Komponente *Status* dazu aufgefordert, die neue Qualität in ihren aktuellen internen Zustand aufzunehmen. Ebenso wird innerhalb der Komponente *Emotion* die Neubestimmung der Zufriedenheit des Agenten angestoßen. Ausgehend von der neuen Zufriedenheit des Agenten und dem Nutzen, den der Agent aus einem möglichen Umzug ziehen würde, fällt innerhalb der Komponente *Wissen* dann die Entscheidung, ob ein Umzug tatsächlich stattfinden wird. Stellt sich im Rahmen seiner Abwägungen heraus, daß die Migrationsoption wahrgenommen werden soll, speichert der Agent in seiner Wissensbasis die neue Position und stößt über die Komponenten *Verhalten* und *Aktor* die Aktualisierung seiner Position auch in der Komponente *Umwelt* an. Entscheidet sich der Agent im Rahmen seines Deliberationsprozesses dafür, an seiner aktuellen Position zu bleiben, sind keine wei-

teren Zustandsübergänge in der Komponente *Wissen* erforderlich. Es ergeht lediglich eine Rückmeldung über diese Entscheidung an die Komponente *Umwelt*.

Die Auswertung der aktuellen Nachbarschaft zum Ende einer Spielrunde hin erfolgt in der Komponente *Wissen* ebenso auf Grundlage der Prozedur *CalcIncome*. Auch hier schließt sich die Übernahme der Qualität in den internen Zustand der Komponente *Status* sowie die Neubestimmung der Zufriedenheit des Agenten unmittelbar an die Berechnung der Qualität der aktuellen Position an.

7.2.3.4 Die Komponente *Emotion*

Innerhalb der Komponente *Emotion* wird die Zufriedenheit des Agenten verwaltet und in der Modellgröße *Moode* gespeichert. Eine Neuberechnung der Zufriedenheit erfolgt dabei immer dann, wenn sich innerhalb der Komponente *Kognition* die Qualität der aktuellen Umgebung des Agenten verändert hat. Der Berechnung der Zufriedenheit liegt eine einfache Schwellwertbetrachtung zugrunde, die in Bedingung Bed. 7.3 dargelegt ist.

7.2.3.5 Die Komponente *Status*

Die Komponente *Status* enthält wesentliche Status-Informationen, die einen Agenten sowohl grundsätzlich als auch zum aktuellen Zeitpunkt hinsichtlich des Support Games auszeichnen. Hierzu gehören die Bedürftigkeitsklasse, die in der Modellgröße *Class* abgelegt wird und zu Beginn eines Simulationslaufes einmal fest für jeden Agenten vorgegebenen wird, sowie die Qualität der aktuellen Position des Agenten. Ebenso wird in der Komponente *Status* zu statistischen Zwecken die Information erfasst, ob der betreffende Agent in der aktuellen Spielrunde eine Migrationsoption wahrgenommen hat. Die Komponente *Status* kapselt einen großen Teil der Modellgrößen eines Agenten, die auch außerhalb des Agenten entweder zu statistischen Zwecken oder zur Datenbereitstellung für die Animation des Modells benötigt werden.

7.2.3.6 Die Komponente *Verhalten*

Die Verhaltenssteuerung der Agenten im Modell Solidarnetzwerk basiert auf rein reaktiven Mechanismen. Die Agenten reagieren im wesentlichen auf Steuerungsinformationen, die aus der Umwelt eintreffen und in der Komponente *Wissen* gespeichert werden. Aufgrund der Anlage des Spiels in vordefinierten Phasen haben die Agenten keine Möglichkeit, in die grundsätzliche Steuerung des Spiels einzugreifen. Sie verfügen lediglich über die Möglichkeit, über ihre Spielzüge zu entscheiden, nachdem sie von der Komponenten *Umwelt* dazu aufgerufen wurden, tun dies aber auch streng nach den Vorgaben eines festen Regelwerks, das in Abschnitt 7.1.2 dargestellt ist.

Die Komponente *Verhalten* übernimmt im Modell *Solidarnetzwerk* im wesentlichen zwei Aufgaben. Zum einen überprüft sie permanent die in der Komponente *Wissen* eingegangenen Steuerungsinformationen zum Spiel und generiert entsprechend dieser Informationen Ausführungsanordnungen, die in der Komponente *Aktor* in interne Aktionen umgesetzt werden. Zum anderen stößt sie Rückmeldungen über ausgeführte Spielaktivitäten an, die in der Komponente *Aktor* als externe Aktionen an die Komponente *Umwelt* weitergeleitet werden. Die Ausführungsanordnungen beschränken sich dabei rein auf die Vorgabe der auszuführenden Aktion und enthalten keine Parameter. Aus diesem Grund basiert die Interaktion der Komponenten *Verhalten* und *Aktor* im vorliegenden Modell der Einfachheit halber auf Steuerungssignalen und nicht, wie in der Agentenarchitektur des Referenzmodells *PECS* vorgeschlagen, auf mobilen Komponenten.

Nach Erhalt einer Migrationsoption weist die Komponente *Verhalten* die Komponente *Aktor* an, die Aktionssequenz *CALCULATE_INCOME* und *FIND_BEST_POSITION* an die Komponente *Wissen* zur Ausführung weiterzuleiten. Im Falle einer *PLAY*-Anweisung ist nur eine Neuberechnung der Qualität der aktuellen Position erforderlich. Dies geschieht mit Hilfe der Ausführungsanordnung *CALC_INCOME_FOR_PLAY*, die die Komponente *Aktor* als interne Aktion an die Komponente *Verhalten* weiterleitet.

Die Interaktion der Agenten mit der Komponente *Umwelt* ist im Modell *Solidarnetzwerk* so konzipiert, daß die Agenten auf alle Steuerungsanweisungen der Komponente *Umwelt* mit einer definierten Rückmeldung antworten. In dieser Hinsicht ist die Komponente *Verhalten* auch dafür verantwortlich, dass nach Abschluss aller Aktivitäten eines Agenten infolge einer Steuerungsanweisung eine entsprechende Rückmeldung an die Komponente *Umwelt* generiert wird. Die Komponente *Verhalten* gibt dabei mit Hilfe einer Ausführungsanordnung nur den Anstoß. Die tatsächliche Rückmeldung erfolgt dann auf der Grundlage einer externen Aktion durch die Komponente *Aktor*.

7.2.3.7 Die Komponente *Aktor*

Die Komponente *Aktor* dient der Generierung der internen und externen Aktionen und steht zu diesem Zweck in Verbindung mit der Komponente *Wissen* des Agenten, sowie der Komponente *Umwelt*. Die Komponente *Wissen* ist dabei zuständig für die Ausführung der internen Aktionen. Die Komponente *Umwelt* erhält die externen Aktionen.

Da im Rahmen der Modells Solidarnetzwerk mit der Ausführung von Aktionen kein zeitverbrauchender Vorgang angestoßen wird, beschränken sich die Aktivitäten der Komponente *Aktor* auf die Umsetzung der Ausführungsanordnungen in Aktionen. Auch hier werden der Einfachheit halber für die Abbildung von Aktionen keine mobilen Komponenten verwendet sondern direkte Steuerungssignale, da die Aktionen der Agenten im Modell *Solidarnetzwerk* keine Kommunikation weiterer Parameter erfordern.

Die Abbildung 7.6 zeigt abschließend die Interaktion aller Modellkomponenten des Modell *Solidarnetzwerk*, nachdem in der Umwelt für einen betreffenden Agenten eine Migrationsoption ausgelost wurde.

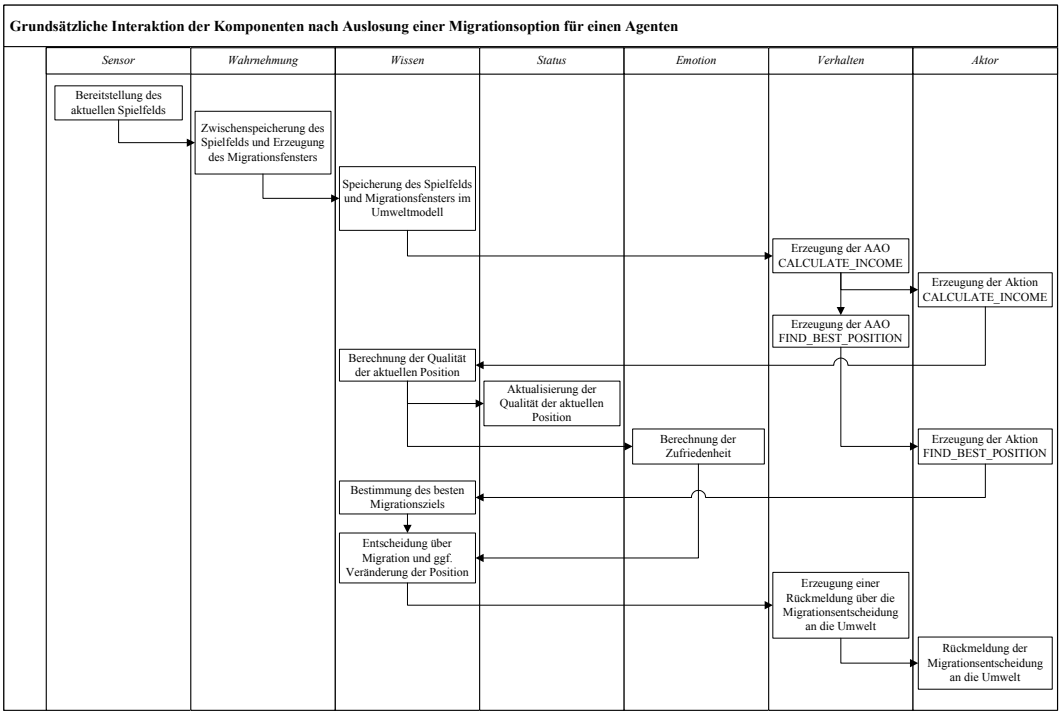

Abbildung 7.6: Das dynamische Zusammenspiel der Komponenten des Modells Solidarnetzwerk am Beispiel der Auswertung einer Migrationsoption

7.3 Simulationsexperimente

Im Rahmen von Simulationsexperimenten kann das dynamische Verhalten des Modells *Solidarnetzwerk* im Detail untersucht werden. Die Durchführung eines Simulationsexperimentes zerfällt im wesentlichen in drei Phasen. In der ersten Phase werden wesentliche Eingangsparameter des Modells festgelegt. In der zweiten Phase wird die eigentliche Simulation durchgeführt und es werden in diesem Zusammenhang die Zustandsübergänge des Modells, die im Laufe der Zeit stattfinden, automatisch berechnet. Am Ende schließt sich in der Regel eine Auswertung des Modells und eine Bewertung bzw. Interpretation der Ergebnisse an.

Um diesen Experimentierprozess im Zusammenhang mit dem Modell *Solidarnetzwerk* zu vereinfachen und eine adäquate Darstellung der Simulationsergebnisse sicherzustellen, wurde das Modell *Solidarnetzwerk* mit einer graphischen Benutzungsoberfläche ausgestattet, die im folgenden Abschnitt kurz dargestellt wird. Der darauf folgende Abschnitt beinhaltet ein Simulationsexperiment, das das charakteristische Verhalten des Modells *Solidarnetzwerk* demonstriert.

7.3.1 Die graphische Benutzungsoberfläche

Die graphische Benutzungsoberfläche des Modells *Solidarnetzwerk* soll ein einfaches Experimentieren mit dem Modell ermöglichen. Die Grundfunktionalität der Benutzungsoberfläche besteht darin, Initialwerte für ausgewählte Modellgrößen zu Beginn eines Simulationslaufes festzulegen, Simulationsläufe durchzuführen und anschließend das dynamische Verhalten des Modells im Rahmen einer Animation darzustellen.

Die graphische Benutzungsoberfläche ermöglicht eine Variation der folgenden Parameter:

- Ausdehnung des Migrationsfensters

  Die Ausdehnung des Migrationsfensters gibt an, welcher Aktionsradius den Agenten im Rahmen einer Migrationsoption zur Verfügung steht. Je größer das Migrationsfenster dabei gewählt wird, um so größere Distanzen können die Agenten bei einem Umzug auf dem Spielfeld zurücklegen und um so schneller wird sich die gegenwärtige Situation auf dem Spielfeld ändern.

- Wahrscheinlichkeit für die Zuteilung einer Migrationsoption

  Die Zuteilung einer Migrationsoption an einen Agenten erfolgt im Modell *Solidarnetzwerk* mit Hilfe eines Zufallsprozesses. Je höher dabei die Wahrscheinlichkeit gewählt wird, mit der ein Agent eine Migrationsoption erhält, um so öfter werden Migrationsoptionen auch tatsächlich ausgewürfelt werden und um so größer wird die Dynamik auf dem Spielfeld sein.

- Minimalniveau

  Das Minimalniveau (s. Abschnitt 7.1.2.5) nimmt Einfluss auf die Zufriedenheitsberechnung der Agenten. Je höher dieser Parameter gewählt wird, um so höher wird auch die Schranke liegen, ab der Agenten mit ihrer aktuellen Position zufrieden sind. Bei einem hohen Wert für das Minimalniveau wird im Modell eine relativ hohe Anzahl an unzufriedenen Agenten existieren, die auch verhältnismäßig mutig und dynamisch ihre Positionen auf dem Spielfeld verändern werden.

Im Anschluss an die Vorgabe der Modellparameter kann die Simulation gestartet werden, wobei angegeben werden muss, wie viele Runde des Support Games gespielt werden sollen. Nach Beendigung des Simulationslaufes stehen die Ergebnisse der Simulation für die Darstellung in der Animation zur Verfügung.

Die graphische Benutzungsoberfläche der Animation für das Modell *Solidarnetzwerk* kann in drei Bereiche untergliedert werden (s. Abbildung 7.7). Im oberen Bereich befinden sich die Menüleiste sowie die grundlegenden Bedienelemente zur Steuerung der Animation. Im mittleren Bereich ist das Spielfeld mit der aktuellen Verteilung der Agenten dargestellt. Wie die Legende, die im unteren Drittel der Anzeige angesiedelt ist, zeigt, werden die Agenten entsprechend ihrer Zugehörigkeit zu einer spezifischen Bedürftigkeitsklasse sowie entsprechend ihrer

aktuellen Zufriedenheit dargestellt. Ein Kreis symbolisiert dabei einen unzufriedenen Agenten, während ein Quadrat einen zufriedenen Agenten repräsentiert. Je dunkler ein Agent eingefärbt ist, um so geringer ist seine Bedürftigkeit.

Unmittelbar neben dem Spielfeld befinden sich pulsierende Balken, die die jeweils aktuelle Anzahl an zufriedenen Agenten, die Anzahl an Agenten, die in der letzten Spielrunde ihre Position verändert haben, sowie die Gesamtauszahlung, die sich aus allen aktuellen Solidarbeziehungen auf dem Spielfeld ergibt, darstellen.

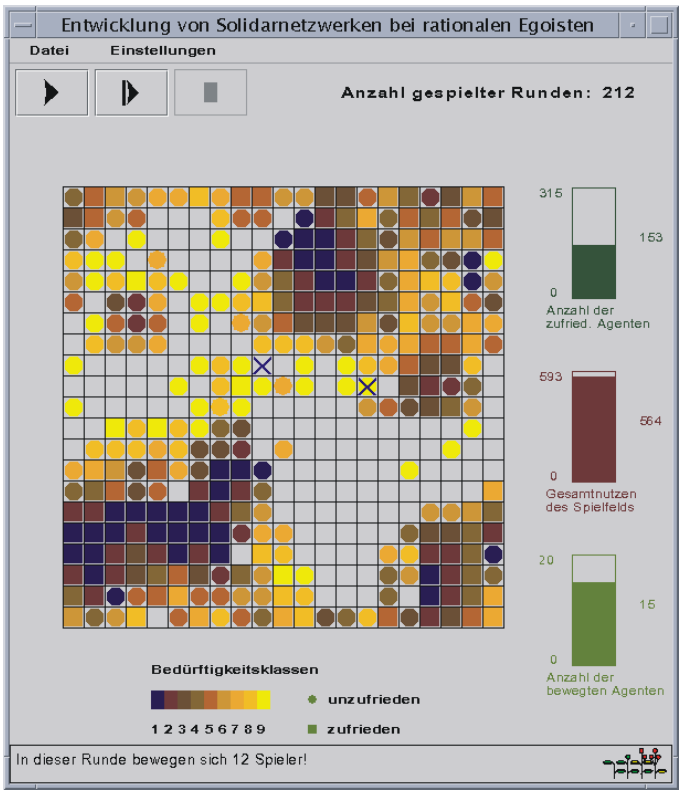

Abbildung 7.7: Die graphische Benutzungsoberfläche der Animation für das Modell Solidarnetzwerk (Süß, 1999)

7.3.2 Das Standardexperiment des Modells *Solidarnetzwerk*

Zur Darstellung des dynamischen Verhaltens des Modells *Solidarnetzwerk* soll nun das Standardexperiment herangezogen werden, das auf den Grundeinstellungen des Modells beruht. Die wesentlichen Parameterwerte, die dem hier vorgestellten Simulationslauf zugrunde liegen, sind in Abbildung 7.2 zusammengefasst.

Zu Beginn des Simulationslaufes werden alle Agenten zufällig auf dem Spielfeld verteilt. Anhand der in Abbildung 7.8 dargestellten Spielzustände lässt sich allerdings deutlich erkennen, dass auf dem Spielfeld bereits nach wenigen Runden

ein Strukturbildungsprozess einsetzt, der darauf beruht, dass alle Agenten in eine Nachbarschaft drängen, die für sie einen möglichst hohen Nutzen bietet. Es entstehen Cluster bzw. Segregationsmuster. Im Kern der Cluster befinden sich Agenten aus den niedrigen Bedürftigkeitsklassen, die aufgrund der hohen Hilfeleistung, die sie anderen Agenten im Rahmen des Support Games in Aussicht stellen können, eine entsprechend hohe Anziehungskraft auf andere Agenten ausüben. Je weiter man in die Peripherie dieser Cluster blickt, um so mehr tauchen dort auch Agenten der höheren Bedürftigkeitsklassen auf. Derartige Agenten finden in den Zentren der dunklen Cluster keine zufrieden stellende Nachbarschaft, da die dort befindlichen Agenten der niedrigen Bedürftigkeitsklassen aufgrund des fehlenden Nutzens nicht bereit sind, mit diesen Agenten Kooperationsbeziehungen einzugehen. Trotzdem bleiben aber auch unattraktive Agenten der hohen Bedürftigkeitsklassen nicht alleine, sondern schließen sich mit Agenten ähnlich hoher Bedürftigkeitsklassen ebenfalls zu Kooperationsbeziehungen zusammen. Es entsteht also ein ausgedehntes Netzwerk von Kooperationsbeziehungen, obwohl die Agenten als rationale Egoisten konzipiert sind, die einander nur dann unterstützen, wenn die Unterstützung auch zum eigenen Vorteil gereicht.

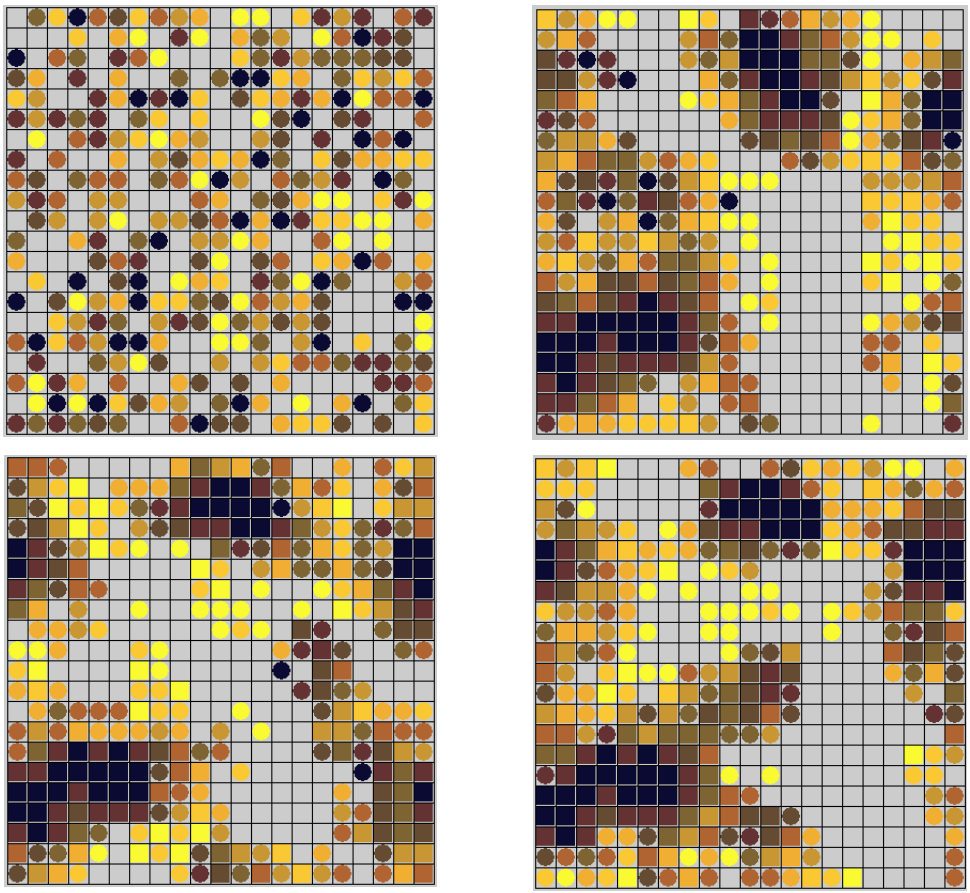

Abbildung 7.8: Die zeitliche Entwicklung des Spielfelds im Modell Solidarnetzwerk

Die Abbildung 7.8 zeigt von links nach rechts die Spielsituationen in den Runden 0, 400, 700 und 1000. Wie die einzelnen Abbildungen belegen, entwickelt sich sehr schnell, d. h. in deutlich weniger als 400 Spielrunden, eine Struktur von Kooperationsbeziehungen, die ein hohes Maß an Stabilität aufweist. Dies ist daran erkennbar, dass sich an der grundlegenden Aufteilung des Spielfeldes von Runde 400 bis Runde 1000 nicht sehr viel verändert. Die beiden Cluster, die am linken unteren Spielfeldrand und oben in der Mitte platziert sind, bleiben über die gesamte Rundenanzahl hinweg nahezu unverändert. Dieselbe Stabilität zeigt der etwas kleinere Cluster am rechten oberen Rand.

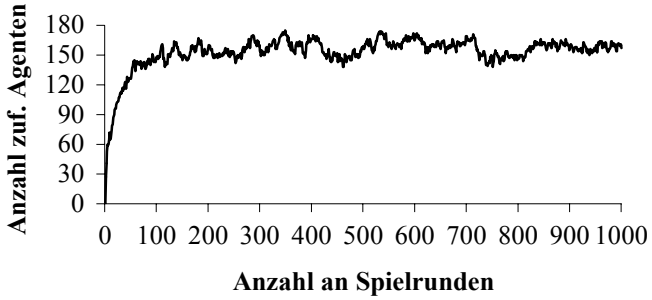

Abbildung 7.9: Die zeitliche Entwicklung der Anzahl an zufriedenen Spielern

Wie die folgenden Zeitverläufe einiger statistischer Größen des Netzwerks zeigen, tritt allerdings zu keiner Zeit völlige Stabilität auf dem Spielfeld ein. Es existiert beispielsweise immer eine gewisse Anzahl an Agenten, die sich nicht mit ihrer aktuellen Position zufrieden geben und damit für eine gewisse Bewegung auf dem Spielfeld sorgen (s. Abbildung 7.9). Im Mittel verbleiben im vorliegenden Simulationsmodell nach der Einschwingphase bis zu einer Spieldauer von 1000 Runden etwa 159 Agenten, also ungefähr die Hälfte der Agenten, unzufrieden.

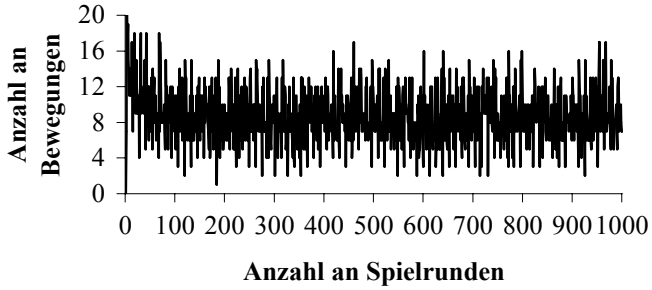

Abbildung 7.10: Die zeitliche Entwicklung der Anzahl an Bewegungen auf dem Spielfeld

Da allerdings nicht alle Agenten in allen Spielrunden Migrationsoptionen erhalten, liegt nach der Einschwingphase die Anzahl an Bewegungen pro Spielrun-

de wesentlich tiefer als die Anzahl an unzufriedenen Agenten. Im Mittel ergeben sich im Standardlauf je Spielrunde etwa 8 bis 9 Positionsveränderungen auf dem Spielfeld.

Der Gesamtnutzen, der aus den bilateralen Kooperationsbeziehungen auf dem Spielfeld entsteht, steigt zu Beginn des Simulationslaufes deutlich an, da sich während dieser Phase sehr schnell erste Solidarbeziehungen ausbilden. Ab der Spielrunde 33 tritt allerdings schon eine gewisse Stabilität ein, wobei sich der Gesamtnutzen des Spielfeldes ab diesem Zeitpunkt etwa bei dem Wert 567 einpendelt (s. Abbildung 7.11). Der theoretisch mögliche Maximalwert für den Gesamtnutzen des Spielfeldes, der genau dann erreicht würde, wenn jeder Agent in der für ihn optimalen Nachbarschaft angesiedelt wäre, läge im Vergleich dazu etwa bei einem Wert von 1268.

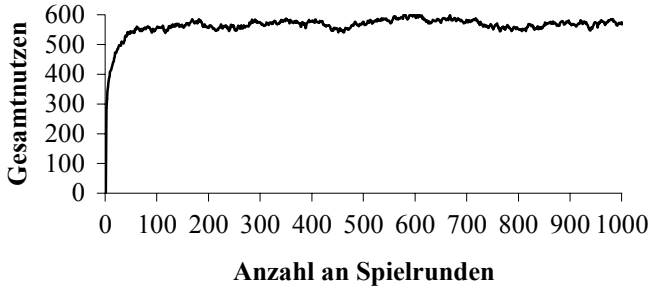

Abbildung 7.11: Die zeitliche Entwicklung des Gesamtnutzens auf dem Spielfeld

7.4 Zusammenfassung und Fazit

Das Modell *Solidarnetzwerk* beschäftigt sich auf der Grundlage eines spieltheoretischen Ansatzes mit der Frage, unter welchen Bedingungen Solidarbeziehungen zwischen rational und egoistisch handelnden Agenten entstehen können. Es demonstriert, dass rationale Egoisten durchaus bereit sind, unter ganz bestimmten Bedingungen Kooperationsbeziehungen mit anderen Agenten einzugehen. Zu diesen Bedingungen gehören im wesentlichen, dass aus einer Solidarbeziehung mit einem anderen Agenten ein signifikanter, eigener Nutzen entspringen muss, sowie eine entsprechende Stabilität und Kontinuität der Beziehung.

Das Modell *Solidarnetzwerk* zeigt das emergente Phänomen der Strukturbildung. Durch die Interaktionsbeziehungen der Agenten auf der Mikroebene des Modells entstehen auf der Makroebene sichtbare Strukturen, sog. Segregationsmuster und Cluster. Den Kern der Cluster bilden dabei Agenten mit hoher Attraktivität. In der Peripherie der Cluster befinden sich mit zunehmendem Abstand vom Zentrum Agenten aus immer höheren Bedürftigkeitsklassen, die nur noch wenige Kooperationspartner finden.

Der Implementierungsabschnitt des vorliegenden Kapitels zeigt, daß das Referenzmodell *PECS* sehr gut geeignet ist, die Konstruktion spieltheoretischer Simulationsmodelle gewinnbringend zu unterstützen. Durch den Einsatz des Referenzmodells *PECS* entsteht eine klare und übersichtliche Modellstruktur, die zu einer sehr leichten Erweiterbarkeit und Modifizierbarkeit derartiger Modelle führt. Auch der systemtheoretische Ansatz, auf dem das Referenzmodell *PECS* basiert, führt zu einer sehr einfachen Abbildung der diskreten Dynamik des Modells *Solidarnetzwerk*.

Aufgrund der sehr einfachen Struktur und Dynamik des von Hegselmann und Flache beschriebenen Modells entsteht auf der anderen Seite jedoch auch ein gewisser Overhead durch die Verwendung des Referenzmodells *PECS*. Eine derartige Strukturierung agentenbasierter Simulationsmodelle, wie sie das Referenzmodell *PECS* aus dem Blickwinkel komplexer Modelle heraus vorschlägt, erscheint für das vorliegende, spieltheoretische Modell in der Nachbetrachtung beinahe als zu feingranular.

Als Fazit lässt sich also an dieser Stelle die Aussage treffen, dass das Referenzmodell *PECS* zwar ohne Zweifel gewinnbringend für den Aufbau spieltheoretischer Simulationsmodelle, die Agenten als elementares Modellierungsparadigma heranziehen, eingesetzt werden kann. Die eigentliche Stärke des Referenzmodells wird sich allerdings erst bei Modellen zeigen, die über eine wesentlich höhere Komplexität verfügen. Beispiele für derartige Modelle zeigen die folgenden Fallstudien.

8 FALLSTUDIE II: DAS MARKTMODELL

Klassische ökonomische Theorien gehen für die Modellierung des Verhaltens von Marktteilnehmern in der Regel von logischen und rationalen Entscheidungsmechanismen aus. Es wird sehr häufig angenommen, dass die Entscheidungsträger in Kenntnis aller Entscheidungsalternativen sind, für alle Entscheidungsalternativen die möglichen Konsequenzen korrekt und zweifelsfrei abschätzen können, den Entscheidungsalternativen eindeutige Präferenzen zuordnen können und über eine ausreichende technische Unterstützung verfügen, um all diese Berechnung in Zuge einer Entscheidungsfindung durchführen zu können.

In der Realität stellt sich die Situation allerdings so dar, dass ein Großteil ökonomischer Entscheidungen auf der Grundlage unvollständiger Informationen gefällt wird, so dass weder alle Handlungsalternativen bekannt sind, noch deren mögliche Konsequenzen. Darüber hinaus müssen Entscheidungen in der Regel sehr schnell getroffen werden. Damit stehen sehr häufig weder ausreichend Zeit noch genügend Ressourcen zur Verfügung, um alle möglichen Alternativen inklusive der daraus resultierenden Konsequenzen extensiv zu untersuchen. Vielmehr werden Entscheidungen in der Praxis auf Grundlage von Faustregeln und Intuition getroffen. Entscheidungstheoretische Ansätze stehen damit im Bereich der Ökonomie in der Kritik.

Abhilfe sollen hier nun Ansätze aus dem Bereich der experimentellen Ökonomie schaffen. In diesem Umfeld dient die Untersuchung des Verhaltens realer Akteure als Grundlage für die ökonomische Theoriebildung. Am Beginn eines derartigen Forschungsansatzes steht in der Regel ein Experiment mit realen Akteuren in einer kontrollierten Laborumgebung. Aus diesem Experiment werden Informationen über das Verhalten der Akteure gewonnen. Im nächsten Schritt werden die gewonnen Informationen analysiert und Theorien über die möglichen Verhaltensweisen und Entscheidungen der Testpersonen entwickelt. Dieser Vorgehensweise liegt die Annahme zugrunde, dass sich anhand der im Experiment gewonnen Daten Regularitäten im Verhalten der Testpersonen erkennen lassen. Liegt eine entsprechende Theorie vor, besteht im nächsten Schritt die Notwendigkeit, diese Theorie zu validieren, d. h. deren Übereinstimmung mit den im realen Experiment aufgetretenen Phänomenen zu überprüfen. Dieser Schritt findet häufig

unter Verwendung entsprechend konstruierter Simulationsmodelle statt, da statische und analytische Betrachtungen in diesem Zusammenhang oftmals nicht verwendbar sind. Insbesondere bietet sich hier der Einsatz agentenbasierter Simulationsmodelle an, da in derartigen Modellierungsansätzen eine adäquate Abbildung menschlicher Verhaltensweisen einschließlich individueller Informationen, Präferenzen, Kommunikations- und Lernfähigkeiten von ausschlaggebender Bedeutung ist.

Die Fallstudie *Marktmodell* basiert auf einer Veröffentlichung von José Castro Caldas und Helder Coelho aus dem Jahre 1992 (Caldas & Coelho, 1992) und beschreibt ein agentenbasiertes Simulationsmodell, das auf der Grundlage von Theorien, die aus einem Laborexperiment gewonnen wurden, mit dem Ziel der Theorievalidierung entstanden ist. Das nachstehend beschriebene Modell dient der Demonstration der Einsatzfähigkeit des Referenzmodells *PECS* im Rahmen der agentenbasierten Modellbildung in der Ökonomie.

8.1 Modellbeschreibung

Dieser Abschnitt widmet sich dem Aufbau und der Dynamik des Marktmodells. Zum besseren Verständnis der Annahmen, die dem Marktmodell nach Caldas und Coelho zugrunde liegen, soll zunächst kurz das dem Modell zugrunde liegende Laborexperiment geschildert werden.

8.1.1 Das Laborexperiment

Im Laborexperiment wird eine Duopol-artige Marktsituation nachgestellt, in der zwei Produzenten eines normalen Gutes einer gewissen Anzahl an Nachfragern dieses Gutes gegenüberstehen. Untersuchungsgegenstand sind dabei die Verhaltensweisen der Produzenten.

Die Nachfrageseite wird auf der Grundlage eines Computerprogramms realisiert. Für die Rolle der Produzenten auf dem Markt werden Testpersonen verwendet, deren Verhalten analysiert werden soll.

Das Experiment geht von einer Reihe von Marktperioden aus, wobei die Anzahl an Marktperioden zu Beginn des Experimentes nicht bekannt gegeben wird. Die Produzenten verfügen über keine Möglichkeit, direkt mit ihrem Konkurrenten zu kommunizieren und kennen sich gegenseitig auch nicht. Jeder der beiden Produzenten befindet sich in einem isolierten Bereich. Zudem existiert ein Experimentleiter, der mit den beiden Produzenten jeweils in telefonischer Verbindung steht.

8.1.1.1 Eingangsinformationen der Produzenten

Zu Beginn wird den Produzenten der grundlegende Ablauf des Experimentes vorgestellt, wobei die folgenden Informationen bereitgestellt werden:

- Preisfestlegung

  Vor jeder neuen Marktperiode muss durch jeden Produzenten ein Preis für sein Gut festgelegt werden. Dabei darf der Preis eine gewisse Obergrenze nicht übersteigen, damit eine Nachfrage zustande kommt. Der Preis, den ein Produzent zu Beginn einer Marktperiode festlegt, hat über die gesamte Marktperiode Gültigkeit und kann zwischenzeitlich nicht geändert werden.

- Unendlich schnelle Anpassung des Angebotes an die Nachfrage

  Es existieren ausreichend Vorräte bzw. keine Beschaffungsengpässe, d. h. die Nachfrage kann in jedem Fall durch das Angebot befriedigt werden. Diese Annahme entbindet die Produzenten davon, eine Produktionsplanung aufzusetzen.

- Undifferenzierte Produkte

  Die Produkte der beiden Produzenten sind absolut vergleichbar und es besteht keinerlei Möglichkeit, beispielsweise durch technische Veränderungen die Produkte voneinander abzuheben und damit die Präferenzen der Konsumenten zu beeinflussen. Der Absatz der Produzenten wird damit ausschließlich durch den Preis bestimmt, der für das angebotene Gut festgelegt wird.

- Preis-Absatz-Relation

  Da es sich bei den auf dem Markt angebotenen Gütern um normale Güter handelt, geht das Experiment von der Annahme aus, dass mit einer Erhöhung des Preises für das Gut die Nachfrage zurückgeht und umgekehrt, wobei in der vorgegebenen Duopol-Situation auch die Preisfestlegung des Konkurrenten von ausschlaggebender Bedeutung ist.

- Durchschnittliche Bezugskosten

  Die durchschnittlichen Bezugskosten für das angebotene Produkt werden in der Zeit als unveränderlich betrachtet und sind ebenso unabhängig von der Produktionsmenge.

- Statistische Informationen

  Nach jeder Marktperiode erhält jeder Produzent statistische Informationen über die Ergebnisse der letzten Marktperiode. Hierzu zählen die individuelle Absatzmenge, der Gesamtabsatz auf dem Markt, der Gewinn aus der letzten Periode, sowie der Preis, zu dem der Konkurrent das Produkt in der letzten Periode auf dem Markt angeboten hat.

- Keine Initialinformationen über das Nachfrageverhalten

  Zu Beginn des Experimentes verfügen beide Produzenten über keinerlei Informationen zum Verhalten der Nachfrager. Sie wissen also nicht, welcher

Absatz sich konkret bei bestimmten Preisen ergeben wird. Da die Nachfrage allerdings deterministisch modelliert ist, lernen die Produzenten im Laufe der Zeit das Nachfrageverhalten auf der Grundlage ihrer Erfahrungen kennen.

8.1.1.2 Das Verhalten der Konsumenten

Für die Darstellung der Nachfrageseite wird ein einfaches, agentenbasiertes Programm verwendet, wobei die folgenden Annahmen zugrunde liegen:

- Nachfrageverhalten

  Jeder Agent folgt hinsichtlich der Festlegung seiner Nachfragemenge einer vorgegebenen Preis-Nachfrage-Kurve. Die Kurve ist dabei so ausgelegt, dass bei einem geringeren Preis für das nachgefragte Gut eine höhere, aber dennoch nach oben hin begrenzte Nachfrage entsteht. Die Preis-Nachfrage-Kurven werden für jeden Konsumenten individuell vorgegeben.

- Präferenzen

  Jeder Konsument verfügt über eine gewisse Präferenz für das Produkt von Produzent 1 oder Produzent 2. Ein Konsument wird entsprechend seiner Präferenz seine Nachfrage erst dann auf den weniger bevorzugten Produzenten verlagern, wenn der präferierte Produzent sein Produkt zu einem deutlich höheren Preis auf dem Markt anbietet als sein Konkurrent. Die Präferenz bewirkt also eine gewisse Stabilität im Nachfrageverhalten und bringt die Konsumenten dazu, trotz geringfügiger Preisnachteile bei ihrem bevorzugten Produzenten zu bleiben.

- Symmetrische Nachfrage

  Das Nachfrageverhalten ist symmetrisch, d. h. es existieren ebenso viele Konsumenten, die das Produkt von Produzent 1 bevorzugen, wie Konsumenten, die das Produkt von Produzent 2 bevorzugen.

- Gleichgewichtspunkt

  Die Nachfragekurven und Präferenzen sind so eingerichtet, dass sich für die Nachfrage, die aus einer identischen Preissetzung beider Produzenten resultiert, eine Kurve mit genau einem Maximum ergibt.

- Ablauf der Nachfrage

  In jeder Marktperiode bestimmen die Konsumenten sequentiell und in statischer Reihenfolge ihre Nachfrage gemäß der vorgegebenen Preis-Nachfrage-Kurve und ihrer Präferenzen. Haben alle Konsumenten in einer Marktperiode ihre Nachfrage festgelegt, so schließt die aktuelle Marktperiode und eine neue Marktperiode beginnt.

Auf der Grundlage dieses Nachfrageverhaltens teilt sich die Gesamtnachfrage unter der Voraussetzung identischer Preise beider Produzenten in zwei Hälften

auf. Beide Produzenten verfügen also in einer derartigen Situation über gleich große Marktanteile. Darüber hinaus existiert genau eine Kombination aus Preisen, die zu einem Maximalgewinn für beide Produzenten führt.

Der Markt, der diesem Experiment zugrunde liegt, geht von sehr einfachen Annahmen aus. Trotzdem bietet dieses Szenario einen sehr interessanten Ausgangspunkt für die Untersuchung des Verhaltens je zweier Produzenten, die eine Reihe von Entscheidungen treffen müssen, im Rahmen ihrer eigenen Interessen agieren, auf die Preisfestlegungsstrategie ihres Kontrahenten reagieren müssen und dabei nicht in der Lage sind, in eine direkte Verhandlung über das gemeinsame Verhalten einzutreten.

Das Marktmodell geht von der grundlegenden Annahme aus, dass beide Produzenten versuchen werden, ihre Gewinne zu maximieren. Daraus ergeben sich im wesentlichen die beiden folgenden Fragen:

- Werden die beiden Produzenten es schaffen, gemeinsam das Paar von Preisen zu erreichen, bei denen der Gewinn für beide Produzenten maximal sein wird, oder endet das Szenario in einem Ungleichgewicht, das aus einem Preiskampf resultiert?
- In welcher Weise werden die beiden Produzenten in der Lage sein, aus den Verhaltensweisen des jeweiligen Konkurrenten zu lernen und ihre eigenen Entscheidungen daraufhin anzupassen?

8.1.1.3 Grundlegende Ergebnisse des realen Experiments

Auf der Grundlage des im letzten Abschnitt beschriebenen Setups wurden drei Experimente mit insgesamt sechs Testpersonen durchgeführt. Drei der Testpersonen waren Ökonomen, die anderen drei Testpersonen Sozialwissenschaftler. Die Abbildung 8.1 zeigt exemplarisch die Sequenz der Preisentscheidungen in einem der drei realen Experimente, in dem die beiden Produzenten nach 36 Marktperioden einen Gleichgewichtspunkt bei einem gemeinsamen Preis von 55 finden.

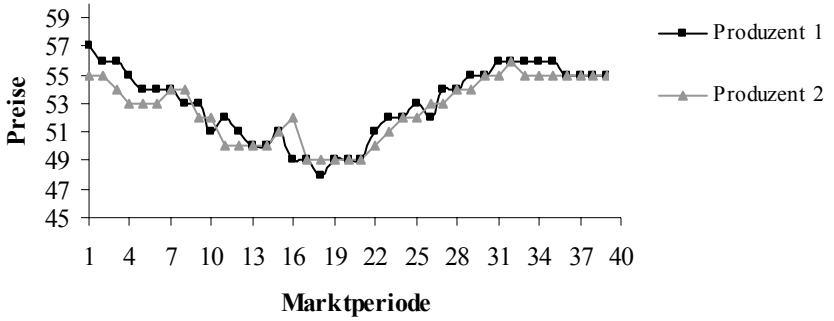

Abbildung 8.1: Sequenz der Preisentscheidungen in einem der realen Experimente

Aus der Auswertung der Experimente, die darauf abzielte, die wesentlichen Verhaltensweisen der Testpersonen bei ihren Preisentscheidungen zu erschließen, gingen drei wesentliche Erkenntnisse hervor:

- Alle drei Experimente lassen sich in zwei Phasen aufteilen. In der ersten Phase senken beide Produzenten ihre Preise ab. In der zweiten Phase tendieren beide Preise nach oben.
- Die Sequenz der Preisentscheidungen unterscheidet sich in allen drei Experimenten.
- Zwei Experimente enden in einem Gleichgewichtszustand, d. h. bei einem Preis von 55 für beide Produzenten (s. Abbildung 8.1). Ein Experiment führte selbst nach 53 Marktperioden zu keinem Gleichgewicht.

Die Entstehung zweier Phasen lässt sich in der folgenden Weise erklären: zu Beginn verfügen beide Produzenten nur über ein unvollständiges Wissen über die tatsächliche Situation. Sie wissen, dass eine Reduktion des Preises zu einer Erhöhung der Nachfrage führen wird. Die Vermutung liegt nahe, dass ein höherer Marktanteil dadurch zu erreichen ist, dass der Preis für das Produkt reduziert wird. Dabei wird allerdings die Reaktion des jeweiligen Konkurrenten unterschätzt. Bei jeder Preisreduktion zieht der jeweilige Konkurrent seinen Preis nach, so dass dieses Verhalten nach einer gewissen Zeit für beide Produzenten zu sehr schlechten Gewinnen führt.

Die Produzenten erkennen also, dass die reine Wettbewerbsstrategie nicht zu den gewünschten Ergebnissen führt und aus diesem Grund durch eine kooperative Strategie ersetzt werden muss. Das Problem dabei besteht darin, dass die beiden Produzenten keine Möglichkeit haben, direkt miteinander zu kommunizieren und in dieser Hinsicht auch keine direkten Preisabsprachen stattfinden können. Als einziges Mittel, um dem Konkurrenten zu signalisieren, dass der Preiskampf beendet werden soll, bleibt einem Produzenten die Anhebung seines Preises. Der Produzent, der als erster den Preiskampf verlässt und seinen Preis anhebt, geht dabei von der Hoffnung aus, dass sein Konkurrent dieses Signal richtig interpretiert und in der Folge eine gemeinsame Suche nach dem für beide Produzenten optimalen Marktpreis beginnen kann. Dieser Marktpreis liegt genau auf dem Niveau, auf dem der gemeinsame Gewinn beider Produzenten maximiert wird.

Falls der Konkurrent ein derartiges Preissignal nicht versteht oder sich nicht auf eine gemeinsame Preissuche einlässt, beginnt ein Interaktions- bzw. Verhandlungsprozess zwischen beiden Produzenten. Verbleibt der Konkurrent trotz eines Preissignals auf seinem niedrigen Preisniveau, so führt der Produzent, der die Preissuche anführt, einen Vergeltungsschlag aus, indem er seinen Preis auf das Niveau seines Konkurrenten reduziert und ihm damit Marktanteile abnimmt. Mit Hilfe dieses Vorgehens hofft der Initiator einer gemeinsamen Preissuche seinen Konkurrenten davon überzeugen zu können, dass ihm keine andere Möglichkeit bleibt als an einer gemeinsamen Preissuche teilzunehmen. Die beiden Phasen, die in allen Laborexperimenten entstanden sind, deuten darauf hin, dass alle Testpersonen im Laufe ihrer Experimente einen Lernprozess durchlaufen haben.

Die grundlegenden Strategien der Testpersonen

Caldas und Coelho leiten aus den Ergebnissen der Experimente die folgenden, grundlegenden Strategien für das Preisbildungsverhalten der Produzenten ab:

In der ersten Phase der Experimente verfolgen alle Testpersonen eine relativ aggressive *Wettbewerbsstrategie*. Diese Wettbewerbsstrategie beruht auf der Annahme, dass durch Preisreduktionen auf Kosten des Konkurrenten eine dominante Marktposition erreicht werden kann, in der dann eine Maximierung der Gewinne möglich ist. Testpersonen, die dieser Strategie folgen, werden also in der Regel versuchen, einen geringeren Preis festzulegen als ihre Konkurrenten auf dem Markt.

In der zweiten Phase der Experimente kristallisiert sich eine *Kooperationsstrategie* heraus, die auf der Einsicht beruht, dass ein beidseitiger, maximaler Gewinn nur bei einer gleichmäßigen Aufteilung des Marktes eintreten wird. Aus diesem Grund basiert diese Strategie auf einer gegenseitigen Kompromissbereitschaft, die sich allerdings erst nach einer gewissen Zeit einstellt. Testpersonen, die der Kooperationsstrategie folgen, werden versuchen, gemeinsam mit ihrem Konkurrenten einen Gleichgewichtspreis zu finden, bei dem die Gewinne beider Produzenten maximiert werden.

Wesentliche Eigenschaften der Testpersonen

Die unterschiedlichen Preisverläufe in den drei Experimenten resultieren im wesentlichen aus den Anfangsbedingungen, mit denen beide Testpersonen zu Beginn der Experimente konfrontiert werden, sowie aus den Eigenschaften und Verhaltensweisen der Testpersonen. Die Testpersonen unterscheiden sich dabei hinsichtlich ihrer Lern- und Anpassungsfähigkeiten, ihrer Fähigkeit, eine gemeinsame Preissuche zu initiieren, sowie in ihrer individuellen Preisfestlegung. Hierbei ist sowohl ein ruhiges, besonnenes Preisbildungsverhalten zu erkennen, das zu kleinen, vorsichtigen Preisveränderungen führt, als auch ein nervöses Verhalten in Verbindung mit relativ großen Preissprüngen. Ebenso lässt sich feststellen, dass sich manche Testpersonen eher wie Anführer, andere eher wie Nachahmer bzw. Gefolgsleute verhalten.

Die Abbildung 8.1 zeigt die aufeinander folgenden Preisentscheidungen zweier moderat agierender Testpersonen. Beide Testpersonen verändern ihre Preise in der Regel um eine bzw. maximal zwei Einheiten. In den beiden anderen Experimenten machen die Testpersonen aber auch wesentlich größere Preissprünge, die als deutliche Aufforderungen an den Konkurrenten zu interpretieren sind, die Preise mit dem Ziel einer gemeinsamen Preissuche anzuheben.

Allen Experimenten ist gemein, dass beide Produzenten immer versuchen, ihre Gewinne zu maximieren.

Die Aktionen der Testpersonen

Das Aktionsrepertoire der Testpersonen beschränkt sich auf Aktionen zur Festlegung ihres Preises für die nächste Marktperiode. Die Aktionen hängen dabei sowohl von internen als auch externen Bedingungen ab. Zu diesen Bedingungen zählen

- die Einschätzung der Testperson hinsichtlich der Qualität der verfolgten Strategie,
- die aktuell durch den Konkurrenten verfolgte Strategie,
- die Richtung, in die sich die Gewinne bewegen,
- die Höhe des aktuellen Marktanteils,
- die Preise der letzten Marktperiode,
- die Persönlichkeit der Testperson hinsichtlich ihrer Preissprünge und ihrer Fähigkeit bzw. Bereitschaft, eine Kooperation zu initiieren.

Diese Bedingungen verändern sich also sowohl mit den aktuellen Gegebenheiten auf dem Markt sowie mit den Annahmen der Testperson über sich und ihren Konkurrenten.

Der Ablauf der Entscheidungsfindung

Um über den Preis für die nächste Marktperiode zu entscheiden, durchläuft eine Testperson drei Phasen.

In der ersten Phase werden die Ergebnisse der letzten Marktperiode analysiert. Hierbei versucht die Testperson die Resultate ihrer grundlegenden Strategie zu bewerten und entscheidet daraufhin, ob die Strategie beibehalten oder verändert wird. Zudem versucht die Testperson, Rückschlüsse auf die Strategie des Konkurrenten zu ziehen, und baut sich dafür eine entsprechende Annahme auf, die nach jeder Marktperiode entweder verifiziert oder revidiert wird. Schließlich entscheidet eine Testperson auf der Grundlage der gesammelten Nachfragedaten in welche Richtung eine Preisänderung stattfinden muss, um den Gewinn zu erhöhen.

In der zweiten Phase generiert die Testperson alle Informationen, die für die Festlegung einer Aktion zur Preissetzung erforderlich sind.

In der letzten Phase wird schließlich auf der Grundlage der erarbeiteten Informationen die Entscheidung getroffen, mit welchem Preis die Testperson in die nächste Marktperiode geht.

8.1.2 Aufbau und Dynamik des Marktmodells

Die folgenden Abschnitte beschäftigen sich mit dem Aufbau des Marktmodells sowie den grundlegenden Eigenschaften und Verhaltensweisen der Agenten, die im Modell die Angebots- und Nachfrageseite repräsentieren.

Das Marktmodell besteht aus insgesamt 18 Agenten, wobei zwei Agenten die Produzenten des Marktes abbilden und die restlichen 16 Agenten die Nachfrage festlegen.

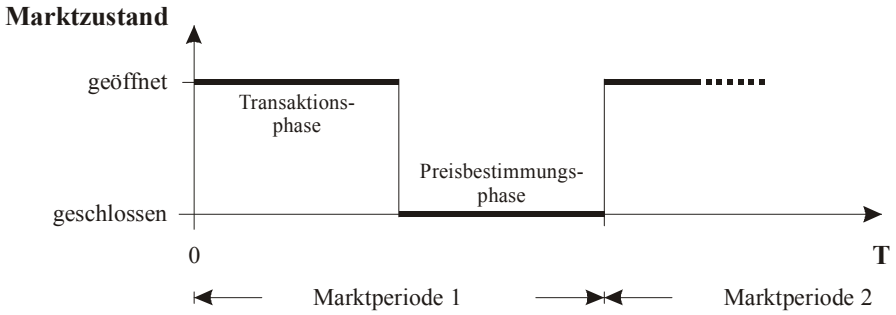

Abbildung 8.2: Der grundlegende Ablauf des Marktmodells in Phasen (Weppner, 1998)

8.1.2.1 Die grundlegende Ablauf des Modells

Der zeitliche Ablauf des Marktmodells gliedert sich in aufeinander folgende Marktperioden. Jede Marktperiode besteht aus einer sog. Transaktionsphase und einer Preisbildungsphase (s. Abbildung 8.2).

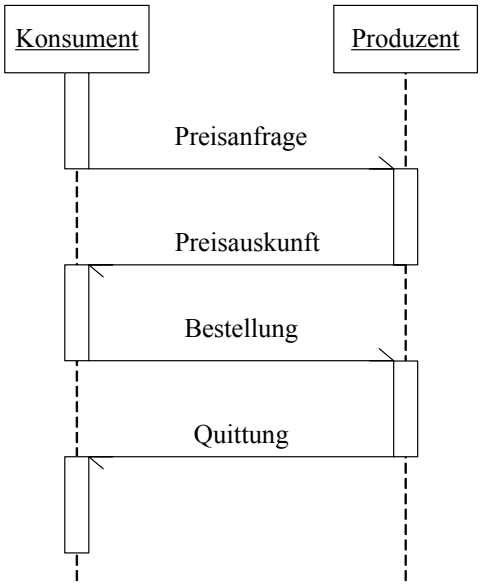

Abbildung 8.3: Der grundlegende Ablauf der Interaktionen in der Transaktionsphase

Während der Transaktionsphase ist der Markt geöffnet. Es findet eine Kommunikation zwischen den Produzenten und Konsumenten statt mit dem Ziel, auf der Grundlage der vorhandenen Preise die Nachfrage auf dem Markt festzulegen. Um seine Nachfrage zu bestimmen, holt jeder Konsument zunächst einmal bei den ihm bekannten Produzenten eine Preisauskunft ein. Die Produzenten teilen

daraufhin den Konsumenten ihren aktuellen Preis mit. Auf der Grundlage der vorhandenen Preise wählen die Konsumenten den Produzenten aus, bei dem ein Kauf getätigt werden soll. Weiterhin legen die Konsumenten unter Berücksichtigung ihrer individuellen Preis-Nachfrage-Kurve (s. Abschnitt 8.1.2.2) ihre Nachfragemenge fest und teilen diese wiederum dem entsprechenden Produzenten mit. Der Produzent bestätigt schließlich die Transaktion mit einer abschließenden Kaufbestätigung an den Konsumenten. Den Gesamtablauf während der Transaktionsphase zeigt die Abbildung 8.3.

Nachdem alle Konsumenten ihre Nachfrage festgelegt haben, endet die Transaktionsphase und der Markt wird für eine gewisse Zeit für Transaktionen geschlossen. Es folgt die sog. Preisbildungsphase, in der die Produzenten die Ergebnisse der letzten Transaktionsphase erheben und auswerten, um auf deren Grundlage einen neuen Preis für die Transaktionsphase der nächsten Marktperiode festzulegen. Die Preisbildung wird von einer Reihe weiterer Parameter beeinflusst, die ausführlich in Abschnitt 8.1.2.3 dargestellt werden.

8.1.2.2 Die Modellierung der Konsumenten

Die Nachfrageseite des Marktmodells wird mit Hilfe von insgesamt 16 Agenten des Typs *Consumer* abgebildet. Die *Consumer*-Agenten sind nur während der Transaktionsphasen aktiv, um ihre Nachfrage zu bestimmen. Im Laufe einer Transaktionsphase treffen die *Consumer*-Agenten zwei Entscheidungen. Zunächst muss festgelegt werden, bei welchem Produzenten die Nachfrage befriedigt werden soll. Steht dieser Produzent fest, wird daraufhin festgelegt, welche Menge des angebotenen Produktes bestellt wird.

Die Auswahl des Produzenten

Zu Beginn einer Transaktionsphase stellt jeder *Consumer*-Agent Preisanfragen an alle ihm bekannten Produzenten. Im Marktmodell wird vereinfachend angenommen, dass jeder Konsument beide Produzenten kennt. Ist die Preisinformation beider Produzenten eingetroffen, legt der *Consumer*-Agent zunächst einmal fest, welcher der beiden Produzenten den Vorzug erhält. Diese Entscheidung hängt im wesentlichen von den Preisen beider Produzenten sowie den Präferenzen des Konsumenten für den jeweiligen Produzenten ab. Die Präferenz für einen Produzenten ist eine unveränderliche und individuelle Eigenschaft, die zu Beginn für jeden Konsumenten festgelegt wird.

Sind $Price_i$, $i \in \{ 1, 2 \}$, die Preise beider Produzenten und $Pref_i$ die Präferenzen des Konsumenten für beide Produzenten, so wählt ein Konsument zur Befriedigung seines Bedarfes denjenigen Produzenten aus, für den das Verhältnis ω_i aus Preis und Präferenz seinen Minimalwert annimmt. Bevorzugt wird also tendenziell der Produzent mit geringerem Preis und höherer Präferenz. Es gilt

$$\omega_i := Price_i / Pref_i \qquad \text{(Gl. 8.1)}$$

mit $Pref_i \in [0, 1]$. Ergibt sich bei beiden Produzenten ein identischer Wert für ω_i, so wird einer der beiden Produzenten zufällig ausgewählt.

Die Festlegung der Nachfragemenge

Die Bestimmung der Nachfragemenge beruht auf einer individuell für jeden Agenten festgelegten Preis-Nachfrage-Kurve. Für die Festlegung der Preis-Nachfrage-Kurve gilt die Annahme eines normalen Gutes. Daraus folgt, dass ein höherer Preis zu einer geringeren Nachfrage führen wird und umgekehrt. Es existiert ein sog. Prohibitivpreis, ab dem die Konsumenten nicht mehr bereit sein werden, das Produkt zu kaufen. Ebenso existiert ein Sättigungspunkt, ab dem die Konsumenten nicht mehr bereit sein werden, noch mehr Einheiten des angebotenen Produktes zu kaufen, selbst wenn dessen Preis noch weiter fallen würde.

Die Abbildung 8.4 zeigt eine charakteristische Nachfragekurve für einen Konsumenten im Marktmodell. Im vorliegenden Beispiel liegt der Sättigungspunkt bei einem Preis von 51 und einer Nachfrage von 6 Mengeneinheiten. Der Prohibitivpreis liegt bei 62.

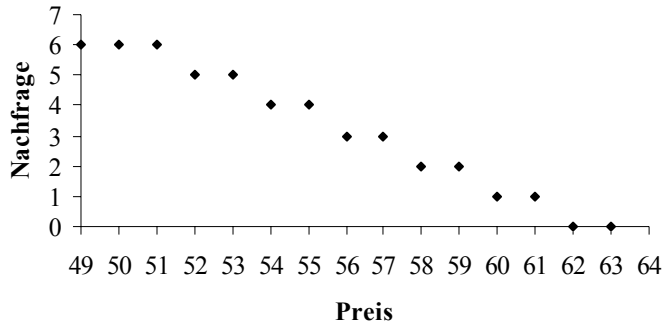

Abbildung 8.4: Nachfragekurve für einen Konsumenten im Marktmodell

Die aus der Preis-Nachfrage-Kurve ermittelte Nachfragemenge teilt der *Consumer*-Agent dem ausgewählten Produzenten mit, wodurch ein verbindlicher Kauf zustande kommt. Mit Entgegennahme der Kaufbestätigung des Produzenten beendet ein Konsument schließlich seine Aktivitäten innerhalb der aktuellen Transaktionsphase und ruht bis zum Beginn der nächsten Transaktionsphase.

Zu Beginn eines Simulationslaufes werden sowohl die Präferenzen als auch die Preis-Nachfrage-Kurven für alle *Consumer*-Agenten individuell aber dennoch ausgewogen eingerichtet, so dass insgesamt gesehen eine symmetrische Nachfrage entsteht. Eine symmetrische Nachfrage ist die Grundvoraussetzung dafür, dass unter den gegebenen Bedingungen bei gleichen Preisen beider Produzenten auch ein Gleichgewicht auf dem Markt entstehen kann.

8.1.2.3 Die Modellierung der Produzenten

Das Marktmodell umfasst insgesamt zwei Agenten des Typs *Producer* auf der Angebotsseite des Marktes. Die Aufgabenstellung der *Producer*-Agenten variiert in Abhängigkeit der aktuellen Marktphase.

In der Transaktionsphase besteht die Aufgabe der *Producer*-Agenten darin, die Preisanfragen der Konsumenten zu beantworten sowie eingegangene Bestellungen zu registrieren und eine entsprechende Bestellbestätigung an die Konsumenten zu verschicken.

Die wesentliche Aufgabe erfüllt ein *Producer*-Agent jedoch in der Preisbestimmungsphase einer Marktperiode. In dieser Phase legt jeder *Producer*-Agent den Marktpreis für sein Produkt fest, der ausschlaggebend für seinen wirtschaftlichen Erfolg in der nächsten Marktperiode sein wird. Bei dieser Entscheidung spielen ein Reihe von Parametern sowie Strategien eine Rolle, die in den folgenden Abschnitten detailliert dargestellt werden.

Das Aktionsrepertoire der Producer-Agenten

Im Rahmen der Transaktionsphase führt jeder *Producer*-Agent auf Anforderung der *Consumer*-Agenten die folgenden beiden Aktion aus:

- *Preis mitteilen*

  Zu Beginn einer Transaktionsphase teilen die *Producer*-Agenten den *Consumer*-Agenten nach Aufforderung ihre aktuellen Marktpreise mit.

- *Bestellung bestätigen*

  Nachdem bei einem *Producer*-Agenten eine Bestellung eingegangen ist, bestätigt er dem entsprechenden *Consumer*-Agenten die Bestellung. Die Bestätigung enthält dabei entsprechend den Vorgaben des Marktmodells immer die gewünschte Menge sowie den Preis, der für die aktuelle Marktperiode gültig ist.

Während der Preisbestimmungsphase legt jeder *Producer*-Agent i seinen Marktpreis $P_{i,T}$ für die nächste Transaktionsphase T fest. Der Marktpreis wird dabei immer als ganzzahlig angenommen. Die Aktionsauswahl wird durch eine Reihe von Parametern, wie etwa den Preis $P_{j,T}$ des Konkurrenten in der letzten Transaktionsphase, beeinflusst und an späterer Stelle genauer dargestellt. Für die Bestimmung des neuen Preises $P_{i,T}$ stehen die folgenden, internen Aktionen zur Verfügung:

- *raise1*: $P_{i,T} := P_{i,T-1} + 1$

  Der Preis wird im Vergleich zur vorhergehenden Marktperiode um eine Einheit erhöht.

- *raise2*: $P_{i,T} := max\ (P_{i,T-1}, P_{j,T-1} - 3)$

  Der Preis wird gegebenenfalls bis auf drei Einheiten unter den Preis des Mitbewerbers *j* angehoben. Diese Aktion wird in der Regel nur dann ausgeführt, wenn der Preis des Mitbewerbers *j* deutlich über dem Preis des betrachteten Agenten *i* liegt.

- *keep*: $P_{i,T} := P_{i,T-1}$

  Durch Ausführung der Aktion *keep* wird der in der letzten Marktperiode gültige Preis auch in die nächste Periode übernommen.

- *lower1*: $P_{i,T} := P_{i,T-1} - 1$

  Der Preis wird im Vergleich zur vorhergehenden Marktperiode um eine Einheit verringert.

- *lower2*: $P_{i,T} := min\ (P_{i,T-1}, P_{j,T-1} + 3)$

  Der Preis wird gegebenenfalls bis auf drei Einheiten über den Preis des Mitbewerbers *j* reduziert. Diese Aktion wird in der Regel nur dann ausgeführt, wenn der Preis des Mitbewerbers *j* deutlich unter dem Preis des betrachteten Agenten *i* liegt.

- *signal*: $P_{i,T} := rnd\ (P_{i,T-1} * (1 + T / a))$

  Durch die Aktion *signal* wird ein Preissprung nach oben durchgeführt. Durch den Faktor T / a wird der Preissprung um so größer ausfallen, je mehr Marktperioden bereits vergangen sind. Der Parameter *a* ist mit dem Wert *75* vorbelegt, so dass sich nach *75* Marktperioden und Ausführung der Aktion *signal* der Preis des betrachteten Agenten verdoppeln würde. Die Funktion *rnd* bezeichnet dabei die kaufmännische Rundungsfunktion.

- *retaliate*: $P_{i,T} := P_{j,T-1}$

  Der Preis des Produzenten *i* wird auf das Preisniveau des Mitbewerbers *j* aus der vorhergehenden Marktperiode gebracht.

Die grundlegenden Preisbestimmungsstrategien und deren Auswahl

Wie in Abschnitt 8.1.1 bereits erwähnt, verfolgen die Produzenten im Rahmen ihrer Preisbestimmung entweder eine Wettbewerbsstrategie oder eine Kooperationsstrategie.

Die Wettbewerbsstrategie wird nach Caldas und Coelho (1992) auch als *Cut-throat-Strategie* bezeichnet, weil sie darauf abzielt, in gieriger Weise durch fortwährende Preisreduktionen Marktanteile zu gewinnen und den Konkurrenten aus dem Markt zu drängen. Die Produzenten sehen allerdings nach einer gewissen Zeit ein, dass die Wettbewerbsstrategie unter den gegebenen Bedingungen des Marktmodells zu keinen zufrieden stellenden Gewinnen führen kann. Sie geben also im Laufe der Zeit den Wettbewerb auf und treten in eine Kooperationsphase ein.

Für die Kooperationsstrategie gibt es zwei Ausprägungen, die sich durch die Rolle, die ein Produzent im Rahmen einer gemeinsamen Preissuche einnimmt, voneinander unterscheiden. Führt ein Produzent eine kooperative Preissuche an, indem er proaktiv neue Preise vorgibt, so verfolgt dieser die sog. *Cooperation-leader-Strategie*. Nimmt ein Produzent bei der gemeinsamen Preissuche eine eher passive Rolle eine und übernimmt die Preisvorgaben seines Kooperationspartners, so verfolgt dieser die sog. *Cooperation-follower-Strategie*.

Jedem der beiden *Producer*-Agenten stehen diese drei Strategien zur Preisbestimmung zur Verfügung. Ein vorgegebenes Regelwerk legt dabei fest, unter welchen Bedingungen ein *Producer*-Agent seine Preisbestimmungsstrategie wechselt. Zentrale Steuerungsgröße ist hierbei das Vertrauen, das ein *Producer*-Agent noch in seine aktuell verfolgte Strategie legt. Ist ein Produzent mit seiner verfolgten Strategie erfolgreich, so steigt sein Vertrauen in diese Strategie an. Führt die Strategie hingegen zu Misserfolgen, so nimmt das Vertrauen in die Strategie ab. Die aktuell verfolgte Strategie wird in der Modellgröße *Strategy*, das zugehörige Vertrauen in diese Strategie in der Modellgröße *Credibility* festgehalten.

Bezeichner	**Wertebereich**	**Erläuterung**
Strategy	{ *cut, cooplead, coopfoll* }	Mögliche Preisbestimmungsstrategien der *Producer*-Agenten
Credibility	*Real*	Vertrauen in die verfolgte Preisbestimmungsstrategie

Abbildung 8.5: Die Strategien der Producer-Agenten zur Preisbestimmung und die zugehörige Steuerungsgröße

Die Auswahl der Preisbestimmungsstrategie. Zu Beginn jeder neuen Preisbestimmungsphase findet eine Bewertung der bisher verfolgten Preisbestimmungsstrategie statt. Aus dieser Bewertung heraus wird eine Entscheidung getroffen, ob die bisher verfolgte Strategie beibehalten werden soll oder ein Strategiewechsel stattfinden muss. Im Rahmen des Marktmodells wird angenommen, dass jeder Produzent zu Beginn die *Cut-throat-Strategie* verfolgt. Das Vertrauen in diese Strategie wird mit einem individuell für jeden Produzenten vorgegebenen Wert *InitialCredit* initialisiert und verändert sich im Laufe der Zeit. Ist das Vertrauen in die *Cut-throat*-Strategie aufgebraucht, so wechselt der entsprechende *Producer*-Agent zur *Cooperation-leader*- oder *Cooperation-follower-Strategie*. Ein erneuter Rückwechsel in die *Cut-throat-Strategie* ist nicht mehr möglich, nachdem sich ein *Producer*-Agent für eine der beiden *Cooperation*-Strategien entschieden hat.

Für die Beurteilung der *Cut-throat-Strategie* ist die Veränderung des Gewinns im Vergleich zur Vorperiode ausschlaggebend. Ist eine Steigerung des Gewinns eingetreten, so wird die Strategie als erfolgreich erachtet und das Vertrauen in die Strategie erhöht. Der Zuwachs, den die Modellgröße *Credibility* dabei erfährt,

entspricht 30% der Gewinnzunahme. Ist hingegen der Gewinn im Vergleich zur Vorperiode gesunken, so geht auch das Vertrauen in die *Cut-throat*-Strategie zurück. Der Wert für die Modellgröße *Credibility* wird dabei um die gesamte Gewinneinbuße verringert. Solange die Modellgröße *Credibility* einen positiven Wert annimmt bzw. noch ein gewisses Restvertrauen in die verfolgte Strategie vorliegt, behält der entsprechende *Producer*-Agent die *Cut-throat-Strategie* bei. Sinkt der Wert der Modellgröße *Credibility* unter den Schwellwert Null, so wechselt der *Producer*-Agent zur *Cooperation*-Strategie. Ob der *Producer* in diesem Fall die *Leader*- oder *Follower*-Rolle einnimmt, hängt davon ab, ob seiner Einschätzung nach bereits ein *Cooperation-leader* existiert. Ist dies der Fall, so wählt er die *Cooperation-follower-Strategie*, andernfalls entscheidet er sich für die *Cooperation-leader-Strategie*, um selbst eine gemeinsame Preissuche anzuführen. Bei einem Wechsel der Strategie erhält deren Vertrauen wieder den ursprünglichen Wert *InitialCredit*.

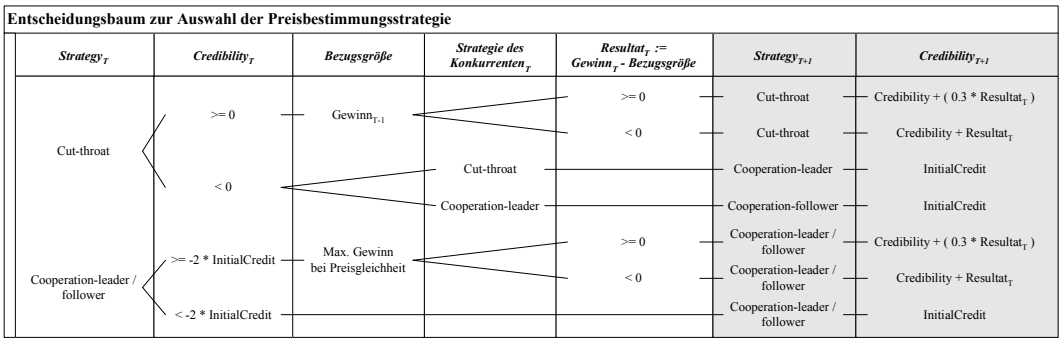

Abbildung 8.6: Entscheidungsbaum zur Auswahl der Preisbestimmungsstrategie

Die Beurteilung der *Cooperation*-Strategien folgt einem ähnlichen Schema. Maßstab für die Beurteilung des Erfolgs ist allerdings nicht die Gewinnentwicklung im Vergleich zur Vorperiode, sondern die Differenz zwischen dem aktuellen Gewinn und dem höchsten Gewinn, der bisher bei Preisgleichheit beider Produzenten erzielt wurde. Der höchste Gewinn bei Preisgleichheit wird als Bezugsgröße gewählt, weil das grundlegende Ziel der *Cooperation*-Strategien darin besteht, eine gleichmäßige Aufteilung des Marktes bei beidseitig maximalen Gewinnen zu erreichen. Die Differenz zwischen aktuellem Gewinn und maximalem Gewinn bei Preisgleichheit bestimmt die Veränderung des Vertrauens in die *Cooperation*-Strategien. Fällt die Differenz positiv aus, so wird die *Cooperation*-Strategie als erfolgreich betrachtet. In diesem Fall steigt der Wert für die Modellgröße *Credibility* an. Andernfalls nimmt as Vertrauen in die *Cooperation*-Strategie ab. Für die Modellgröße *Credibility* wird im Falle der *Cooperation*-Strategie eine untere Schranke von *-2 * InitialCredit* angesetzt. Wird dieser Schwellwert unterschritten, waren die bisherigen Kooperationsbemühungen des betreffenden Agenten noch nicht erfolgreich. Der Agent bleibt in diesem Fall aber trotzdem bei der Kooperationsstrategie und unternimmt einen weiteren Versuch, seinen Mitbewerber zur

Kooperation zu bewegen. Die Modellgröße *Credibility* erhält vor diesem neuen Anlauf wieder den Ausgangswert *InitialCredit*.

Die Abbildung 8.6 zeigt in der Übersicht alle Verhaltensregeln, die in den *Producer*-Agenten zur Auswahl der Preisbestimmungsstrategie Verwendung finden. Die grau hinterlegten Bereiche des Entscheidungsbaumes markieren dabei den Aktionsteil der Regeln.

Die Steuerungsgrößen im Rahmen der Preisbestimmung

Mit jeder Preisbestimmungsstrategie ist eine besondere Art und Weise verbunden, den Marktpreis für die nächste Transaktionsphase festzulegen. Für jede dieser drei Strategien existiert also ein eigener Satz an Verhaltensregeln, der vorgibt, welcher neue Marktpreis in den jeweiligen Situationen zustande kommt. Dabei werden eine Reihe von Steuerungsgrößen berücksichtigt, die nun in den folgenden Abschnitten näher erläutert werden.

Die Persönlichkeit der *Producer*-Agenten. Caldas und Coelho (1992) unterscheiden in ihrem Marktmodell zwischen ruhigen und nervösen Produzenten. Ruhige Produzenten zeichnet aus, dass sie von einer Marktperiode zur nächsten nur geringe Preisveränderungen vornehmen und sehr stabil in ihren Preisreaktionen sind. Nervöse Produzenten hingegen tendieren zu starken Preissprüngen und verändern damit sehr schnell und grundlegend die Situation auf dem Markt.

Die Persönlichkeit der Producer-Agenten findet in der Modellgröße *Personality* Berücksichtigung. Sie wird zu Beginn für jeden Producer-Agenten mit einem der Werte *calm* oder *nervous* belegt und im weiteren Verlauf als unveränderlich betrachtet.

Der Preis der aktuellen Marktperiode. Der Preis für die Transaktionsphase der nachfolgenden Marktperiode wird in der Regel auf der Grundlage des aktuellen Marktpreises bestimmt. Seinen aktuellen Marktpreis legt jeder *Producer*-Agent in seiner Modellgröße *Price* ab.

Der aktuelle Marktanteil. Der Marktanteil eines Producer-Agenten wird über den Anteil seines eigenen Absatzes am Gesamtabsatz auf dem Markt gemessen. Diese statistische Größe wird allen *Producer*-Agenten am Ende jeder Transaktionsaktionsphase zur Auswertung zur Verfügung gestellt und in deren Modellgröße *MShare* gespeichert.

Der Marktanteil wird in die fünf Kategorien *very high*, *high*, *fair*, *low* und *very low* unterteilt, wobei die in Abbildung 8.7 dargestellten Intervalle zur Anwendung kommen.

Kategorie	Bereich
very high	$80\% < MShare \leq 100\%$
high	$50\% < MShare \leq 80\%$
fair	$MShare = 50\%$
low	$40\% < MShare \leq 50\%$
very low	$0\% < MShare \leq 40\%$

Abbildung 8.7: Klassifikation der Marktanteile

Die Preisbestimmungsstrategie des Mitbewerbers. Für die Strategieauswahl sowie die Preisbestimmung im Rahmen der *Cooperation-leader-Strategie* ist es für einen *Producer*-Agenten von ausschlaggebender Bedeutung, welche Strategie der Mitbewerber auf dem Markt verfolgt. Jeder *Producer*-Agent versucht also aus den vorhandenen Marktinformationen und dem Verhalten des Mitbewerbers Rückschlüsse auf dessen Preisbestimmungsstrategie zu ziehen. Dabei gilt die Annahmen, dass alle *Producer*-Agenten über dasselbe Repertoire an Preisbestimmungsstrategien verfügen und mit der *Cut-throat-Strategie* beginnen. Damit reicht es für die Ermittlung der Strategie des Mitbewerbers zunächst einmal aus, den Zeitpunkt festzustellen, an dem dieser von der *Cut-throat-Strategie* zur *Cooperation-Strategie* wechselt.

Zu diesem Zweck untersucht jeder *Producer*-Agent, ob die ausgeführten Preisänderungen denen eines *Cut-throat*-Produzenten entsprechen. Als Eingangsinformationen dienen dabei der Preis sowie der Marktanteil des Mitbewerbers in der letzten Marktperiode. Decken sich die Beobachtungen mit den Vorgaben der *Cut-throat-Strategie* zieht der *Producer*-Agent den Schluss, dass der Mitbewerber noch die *Cut-throat-Strategie* verfolgt. Andernfalls geht der *Producer*-Agent von der Annahme aus, dass der Mitbewerber in der Zwischenzeit zu einer der beiden *Cooperation-Strategien* gewechselt ist. Die Zuordnung zur *Cooperation-leader*- oder *Cooperation-follower-Strategie* basiert auf der Einschätzung, ob zum betrachteten Zeitpunkt bereits ein *Cooperation-leader* existiert. Ist dies der Fall, wird der Mitbewerber als *Cooperation-follower* eingestuft. Existiert nach Einschätzung des *Producer*-Agenten hingegen noch kein Agent, der die *Cooperation-Strategie* verfolgt, so wird dem betrachteten Mitbewerber die *Cooperation-leader-Strategie* zugeordnet.

Innerhalb der Producer-Agenten wird die Strategie des Mitbewerbers in der Modellgröße *CompStrat* gespeichert. Mögliche Werte für die Strategie entsprechend der Annahmen des Marktmodells sind *cut* für *Cut-throat*, *cooplead* für *Cooperation-leader* und *coopfoll* für *Cooperation-follower*.

Die Gewinnentwicklung. Die Entwicklung der Gewinne ist für die Preisbestimmung im Rahmen der *Cooperation-leader-Strategie* von Bedeutung, die darauf abzielt, maximale Gewinne bei einer fairen Aufteilung des Marktes zu erreichen.

Bei *n* Produzenten ergibt sich nach den Annahmen des Marktmodells eine faire Marktaufteilung, wenn alle Produzenten einen Marktanteil von *(100 / n)%* aufweisen. Diese Marktaufteilung tritt nur dann ein, wenn alle Produzenten zu demselben Marktpreis anbieten.

Bei einer fairen Marktaufteilung gilt es nun, das Preisniveau zu ermitteln, auf dem die gemeinsamen Gewinne maximiert werden. Im Rahmen der Preisbestimmung ist dabei festzulegen, in welche Richtung der aktuelle Preis für die nachfolgende Marktperiode verändert werden muss.

Zur Bestimmung der Gewinnentwicklung vergleicht der *Producer*-Agent den Gewinn G auf dem aktuellen Preisniveau mit den Gewinnen G^+ und G^- auf dem nächst höheren bzw. nächst niedrigeren Preisniveau. Allerdings kann es dabei vorkommen, dass nicht alle erforderlichen Gewinninformationen vollständig vorliegen. Dies ist insbesondere dann der Fall, wenn sich die Produzenten in der Vergangenheit noch nicht gemeinsam auf dem betrachteten Preisniveau befanden.

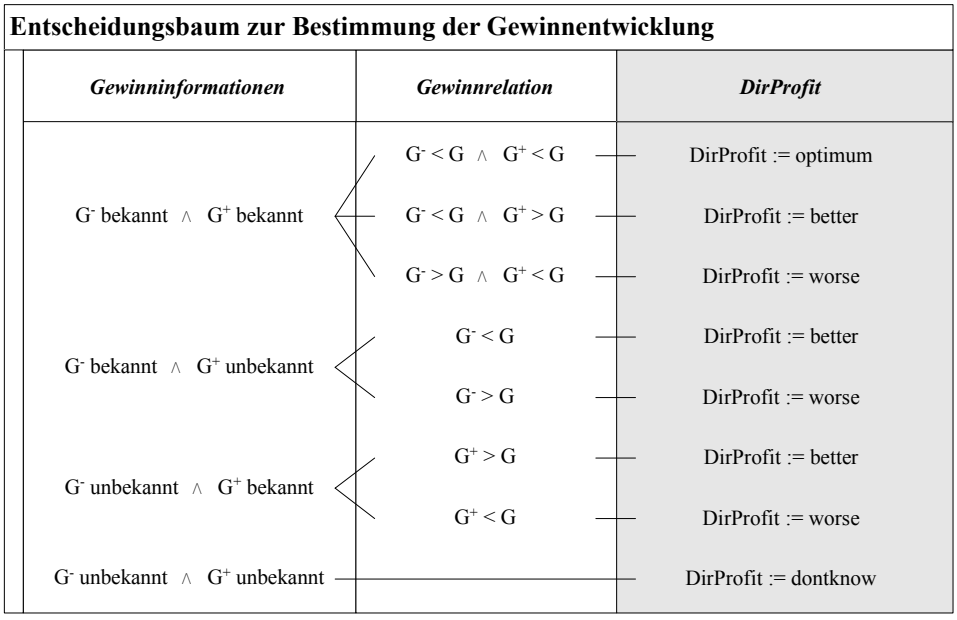

Abbildung 8.8: Der Entscheidungsbaum zur Bestimmung der Gewinnentwicklung

Sind alle drei Gewinne bekannt, so existieren für die Relationen zwischen G, G^- und G^+ drei mögliche Fälle:

- $G > G^-$ und $G > G^+$

  Aufgrund der Annahme des Marktmodells, dass nur eine Kombination von Preisen existiert, bei der der gemeinsame Gewinn maximiert wird, liegt in diesem Fall bereits das Optimum der Gewinnkurve vor. Weder eine Preiserhöhung, noch eine Preisverringerung können in diesem Fall zu einer weiteren

Erhöhung der Gewinne führen. Der Preis bleibt also in diesem Fall für alle nachfolgenden Marktperioden unverändert.

- $G < G^+$

  In diesem Fall wären auf dem nächst höheren Preisniveau höhere Gewinne als auf dem aktuellen Preisniveau zu erreichen. Der Preis ist also für die nachfolgende Marktperiode zu erhöhen.

- $G < G^-$

  Dieser Fall zeigt an, dass der aktuelle Marktpreis bereits zu hoch gewählt wurde, da auf dem nächst niedrigeren Preisniveau in der Vergangenheit bereits höhere Gewinne erzielt wurden.

Für die Gewinnentwicklung lässt sich in eindeutiger Weise ein Wert bestimmen, solange einer der beiden Gewinne G^- oder G^+ bekannt ist. Sind beide Gewinne unbekannt, kann auf die Gewinnentwicklung im Rahmen der Preisbestimmung der *Cooperation-leader-Strategie* nicht zurückgegriffen werden.

Die Gewinnentwicklung wird innerhalb der *Producer*-Agenten in der Modellgröße *DirProfit* abgelegt, die die Werte *dontknow*, *better*, *worse* oder *optimum* annehmen kann. Der in Abbildung 8.8 dargestellte Entscheidungsbaum zeigt alle möglichen Fälle für die Bestimmung der Gewinnentwicklung *DirProfit*.

Die Abbildung 8.9 zeigt in der Übersicht alle Steuerungsgrößen, die bei der Preisbestimmung im Marktmodell von Bedeutung sind.

Bezeichner	**Wertebereich**	**Erläuterung**
Personality	*{ calm, nervous }*	Persönlichkeit des *Producer*-Agenten
Price	*Integer*	Aktueller Marktpreis
MShare	*Real, 0 - 100%*	Aktueller Marktanteil
CompStrat	*{ cut, cooplead, coopfoll }*	Strategie des Mitbewerbers
DirProfit	*{ better, worse, optimum, dontknow }*	Gewinnentwicklung
Credibility	*Real*	Vertrauen in die verfolgte Preisbestimmungsstrategie

Abbildung 8.9: Die Steuerungsgrößen für die Preisbestimmung im Marktmodell

Nachdem die Strategieauswahl und die Steuerungsgrößen für die Preisbestimmung geklärt sind, muss abschließend noch erläutert werden, welche Regeln für die Preisbestimmung im Rahmen der einzelnen Strategien zum Einsatz kommen.

Die Preisbestimmung in der Cut-throat-Strategie

Das Ziel der *Cut-throat-Strategie* besteht darin, einen möglichst hohen Gewinn auf der Grundlage eines hohen Marktanteils zu erzielen. Ein *Cut-throat*-Produzent richtet seine Preisbestimmung daher im wesentlichen am aktuellen Marktanteil aus. Hält ein *Cut-throat*-Produzent zum betrachteten Zeitpunkt nur einen geringen Marktanteil, so wird er versuchen, durch eine Verringerung des Preises seinen Marktanteil zu vergrößern. Dabei wird er seinen Preis allerdings nie unterhalb seiner unteren Schranke *lowlimit* ansetzen. Zur Steuerung der *Cut-throat-Strategie* dienen die Modellgrößen *Price*, *MShare* und *Personality*.

Preisbestimmung *Cut-throat*		R1	R2	R3	R4	R5	R6	R7	R8
B1	*Price*	= *low-limit*	= *low-limit*	> *low-limit*	> *low-limit*	> *low-limit*	> *low-limit*	> *low-limit*	> *low-limit*
B2	*MShare*	*very high*	-	*very high*	*very high*	*high*	*fair*	*low*	*very low*
B3	*Personality*	*nervous*	-	*nervous*	*calm*	-	-	-	-
A1	*raise2*	X		X					
A2	*raise1*		X		X				
A3	*keep*					X			
A4	*lower1*						X	X	
A5	*lower2*								X

Abbildung 8.10: Die Verhaltensregeln zur Preisbestimmung im Rahmen der Cut-throat-Strategie

Ein *Cut-throat*-Produzent gibt sich zufrieden, sobald er einen hohen Marktanteil erreicht hat. In diesem Fall sieht er keine Notwendigkeit für eine Preisänderung und belässt seinen Preis für die folgende Marktperiode auf dem aktuellen Niveau. Er führt also in einer derartigen Situation die Aktion *keep* aus.

Fällt der Marktanteil eines *Cut-throat*-Produzenten fair oder niedrig aus, so versucht er, durch Verringerung des Preises um eine Einheit (*lower1*) mehr Konsumenten für sich zu gewinnen und damit seinen Marktanteil zu erhöhen. Eine deutlichere Preisreduktion um mehr als eine Einheit (*lower2*) findet nur dann statt, wenn ein *Cut-throat*-Produzent einen sehr niedrigen Marktanteil hat.

Verfügt ein *Cut-throat*-Produzent über einen sehr hohen Marktanteil von über 80%, so wird er versuchen, seinen Gewinn zu steigern, indem er seinen Preis erhöht. Dieses Vorgehen wird allerdings nicht zu einem stabilen Marktanteil führen, da mit steigenden Preisen immer mehr Konsumenten zum Mitbewerber abwandern werden, um ihre Nachfrage dort zu befriedigen.

Die Persönlichkeit des *Cut-throat*-Produzenten wirkt sich nur bei sehr hohen Marktanteilen und den damit verbundenen Preiserhöhungen aus. Ein *nervöser Cut-throat*-Produzent wird seinen Preis in einer derartigen Situation knapp unter dem Preis seines Mitbewerbers ansiedeln (*raise2*) und sofort eine maximale Gewinnsteigerung anstreben. Ein ruhiger Produzent wird eine gemächliche Preissteigerung durchführen und seinen Preis je Periode um nur eine Einheit erhöhen (*raise1*).

Ist im Verlauf der Wettbewerbsstrategie der Preis an der Untergrenze *lowlimit* angekommen, wird ein *Cut-throat*-Produzent ebenso durch Erhöhung seines Preises versuchen, eine Verbesserung seiner Situation herbeizuführen. Ein *nervöser* Produzent mit einem sehr hohen Marktanteil wird dabei einen Preissprung durch Ausführung der Aktion *raise2* herbeiführen, während in allen anderen Fällen der Preis nur um eine Einheit nach oben korrigiert wird, indem die Aktion *raise1* zur Ausführung kommt.

Die Entscheidungstabelle in Abbildung 8.10 zeigt im Überblick alle Verhaltensregeln, die die Preisbestimmung in der *Cut-throat-Strategie* beeinflussen. Die Regeln *R1* und *R2* sind dabei bewusst nicht überschneidungsfrei spezifiziert. *R2* ist vielmehr als Default-Regel zu *R1* zu verstehen.

Die Preisbestimmung in der Cooperation-follower-Strategie

Das Ziel der *Cooperation-follower-Strategie* besteht darin, einen fairen Marktanteil und damit eine Aufteilung des Marktes in zwei gleiche Teile zu erreichen. Als Steuerungsgrößen dienen bei dieser Strategie der Marktanteil *MShare* und die Persönlichkeit *Personality* des *Producer*-Agenten.

Verfügt ein *Producer*-Agent zum betrachteten Zeitpunkt über einen geringen bzw. sehr geringen Marktanteil, so geht er von der Annahme aus, dass sein gegenwärtiger Preis zu hoch ist. Er wird in einer derartigen Situation versuchen, sich einem fairen Marktanteil dadurch zu nähern, dass er seinen Preis entsprechend verringert, wobei seine Preisänderung um so größer ausfallen wird, je weiter er von einer fairen Marktteilung entfernt ist. Ist der aktuelle Marktanteil sehr niedrig, wird die Aktion *lower2* zur Ausführung ausgewählt. Bei einem niedrigen Marktanteil führt der *Cooperation-follower* die Aktion *lower1* mit einer daraus resultierenden Preiskorrektur um eine Einheit aus.

Verfügt ein Produzent über einen hohen bzw. sehr hohen Marktanteil, so wird er im Rahmen der *Cooperation-follower-Strategie* Marktanteile an seinen Mitbewerber abtreten müssen, indem er seinen Marktpreis erhöht. Liegt ein sehr hoher Marktanteil vor und handelt es sich bei dem betrachteten Produzenten um einen *nervösen* Agenten, so wird dieser in der gegebenen Situation die Aktion *raise2* ausführen, mit der er seinen Preis beinahe auf das Niveau des Mitbewerbers anhebt. Ein *ruhiger* Produzent oder ein Produzent mit einem hohen Marktanteil wird in dieser Situation seinen Preis mit Hilfe der Aktion *raise1* nur um eine Einheit nach oben korrigieren.

Liegt bereits eine faire Marktteilung vor, so wird im Rahmen der *Cooperation-follower-Strategie* auf eine Preisänderung verzichtet und die Aktion *keep* ausgeführt.

Die Abbildung 8.11 stellt die Verhaltensregeln im Rahmen der *Cooperation-follower-Strategie* im Überblick dar.

Preisbestimmung ***Cooperation-follower***		R1	R2	R3	R4	R5	R6
B1	*MShare*	*very high*	*very high*	*high*	*fair*	*low*	*very low*
B2	*Personality*	*nervous*	*calm*	-	-	-	-
A1	*raise2*	X					
A2	*raise1*		X	X			
A3	*keep*				X		
A4	*lower1*					X	
A5	*lower2*						X

Abbildung 8.11: Die Verhaltensregeln zur Preisbestimmung im Rahmen der Cooperation-follower-Strategie

Die Preisbestimmung in der Cooperation-leader-Strategie

Die *Cooperation-leader-Strategie* zielt darauf ab, einen gemeinsamen Preis zu finden, bei dem beide Produzenten einen identischen Marktanteil besitzen und die gemeinsamen Gewinne maximiert werden. Die Schwierigkeit dieser Strategie besteht darin, den Mitbewerber ebenfalls zur Kooperation zu bewegen, da eine gemeinsame Preissuche mit einem *Cut-throat*-Produzenten unmöglich ist.

Solange der Mitbewerber noch die *Cut-throat-Strategie* verfolgt, wird ihn der *Cooperation-leader* zu überzeugen versuchen, auch zur Kooperationsstrategie zu wechseln. Zu diesem Zweck hebt der *Cooperation-leader* in Abhängigkeit seiner Persönlichkeit seinen Preis an (*signal* / *raise1*), um zu signalisieren, dass er die Phase des Preiskampfes bereits beendet hat. Verharrt der Mitbewerber trotz allem auf der *Cut-throat-Strategie*, so übt der *Cooperation-leader* nach einer gewissen Zeit einen Vergeltungsschlag aus, indem er mit Hilfe der Aktion *retaliate* seinen Preis auf das Niveau seines Konkurrenten verringert und dadurch dessen Marktanteil reduziert. Anschließend hebt der *Cooperation-leader*, wie schon zu Beginn, seinen Preis wieder an, um erneut seine Kompromissbereitschaft zu signalisieren. Diese Vorgehensweise wiederholt der *Cooperation-leader* so lange, bis sein Mitbewerber bemerkt bzw. sich schließlich davon überzeugen lässt, dass Kooperation auf längere Sicht für beide Produzenten die bessere Strategie ist. Die Regeln *R1*

bis *R4* in Abbildung 8.12 zeigen die für dieses Vorgehen verantwortlichen Verhaltensregeln.

Preisbestimmung ***Cooperation-leader I***		R1	R2	R3	R4	R5	R6	R7
B1	*Price*	= *low-limit*	= *low-limit*	> *low-limit*	> *low-limit*	> *low-limit*	> *low-limit*	> *low-limit*
B2	*CompStrat*	*cut*	-	*cut*	*cut*	-	-	-
B3	*Credibility*	= *Initial-credit*	-	= *Initial-credit*	< *-0.3 * Initial-credit*	-	-	-
B4	*MShare*	-	-	-	-	*very high*	*very high*	*high*
B5	*Personality*	*nervous*	-	*nervous*	-	*nervous*	*calm*	-
B6	*DirProfit*	-	-	-	-	-	-	-
A1	*signal*	X		X				
A2	*raise1*		X				X	
A3	*raise2*					X		
A4	*keep*							X
A5	*retaliate*				X			

Abbildung 8.12: Die Verhaltensregeln zur Preisbestimmung im Rahmen der Cooperation-leader-Strategie (Teil I)

Hat der Mitbewerber die Wettbewerbsstrategie verlassen, initiiert der *Cooperation-leader* eine gemeinsame Suche nach dem für beide Produzenten optimalen Preisniveau. Die mit der Preissuche verbundenen Preisanpassungen orientieren sich dabei im wesentlichen am aktuellen Marktanteil (*MShare*) sowie der Richtung, in die sich die Gewinne entwickeln (*DirProfit*).

Im Falle eines sehr hohen Marktanteils wird der *Cooperation-leader* zunächst einmal seinen Preis anheben müssen, um sich einer gerechten Aufteilung des Marktes anzunähern. Ein nervöser Produzent wird in dieser Situation durch Ausführung der Aktion *raise2* einen größeren Preissprung vollziehen als ein ruhiger Produzent, der in diesem Fall die Aktion *raise1* bevorzugen wird (s. Regeln *R5* und *R6* in Abbildung 8.12).

Verfügt der *Cooperation-leader* über einen hohen Marktanteil, geht er eher konservativ vor und wartet darauf, dass sein Mitbewerber durch eine Preissen-

kung seinen Marktanteil erhöht. Er stabilisiert in dieser Situation seinen Preis und wählt die Aktion *keep* (s. Regel R7 in Abbildung 8.12).

Preisbestimmung *Cooperation-leader II*		R8	R9	R10	R11	R12	R13
B1	*Price*	> *low-limit*	> *low-limit*	> *low-limit*	> *low-limit*	> *low-limit*	> *low-limit*
B2	*CompStrat*	-	-	-	-	-	-
B3	*Credibility*	-	-	-	-	-	-
B4	*MShare*	*fair*	*fair*	*fair*	*fair*	*low*	*very low*
B5	*Personality*	-	-	-	-	-	-
B6	*DirProfit*	*dontknow*	*better*	*worse*	*optimum*	-	-
A1	*raise1*	X	X				
A2	*lower1*			X			X
A3	*keep*				X	X	

Abbildung 8.13: Die Verhaltensregeln zur Preisbestimmung im Rahmen der Cooperation-leader-Strategie (Teil II)

Der interessanteste Fall ergibt sich bei einer gerechten Aufteilung des Marktes auf beide Produzenten. In einer derartigen Situation muss überprüft werden, ob das für beide Produzenten optimale Preisniveau bereits erreicht ist oder noch weitere Preisanpassungen erforderlich sind. An dieser Stelle kommt nun die Gewinnentwicklung ins Spiel. Existiert auf dem betrachteten Preisniveau bislang keine Aussage über die Gewinnentwicklung, so wählt der *Cooperation-leader* die Aktion *raise1*, um zu prüfen, wie sich in der Folge die Gewinne auf dem nächst höheren Preisniveau verhalten (s. Regel *R8* in Abbildung 8.13). Ebenso wird der *Cooperation-leader* seinen Preis um eine Einheit anheben, wenn die Gewinnentwicklung in eine positive Richtung zeigt (s. Regel *R9* in Abbildung 8.13). Verspricht das nächst höhere Preisniveau keine Verbesserung der Gewinne und ist das Optimum noch nicht erreicht, wird der Marktpreis mit Hilfe der Aktion *lower1* für die Folgeperiode um eine Einheit verringert (s. Regel *R10* in Abbildung 8.13). Hat der *Cooperation-leader* schließlich zusammen mit seinem Mitbewerber das optimale Preisniveau erreicht, besteht keine Notwendigkeit für weitere Preisveränderungen mehr. Er wählt in diesem Fall die Aktion *keep* (s. Regel *R11* in Abbildung 8.13).

Verfügt der *Cooperation-leader* nur über einen geringen Marktanteil, so verhält er sich ebenso wie im Falle eines hohen Marktanteils zunächst einmal konservativ und wartet auf entsprechende Aktivitäten seines Mitbewerbers. Er führt in einer derartigen Situation die Aktion keep aus (s. Regel *R12* in Abbildung 8.13).

Erst dann, wenn der Marktanteil eines *Cooperation-leader* sehr gering wird, wirkt er wieder proaktiv auf seine Situation ein und versucht, mit Hilfe der Aktion *lower1* und der damit verbundenen Preissenkung in der Folgeperiode wieder Marktanteile für sich zu erobern (s. Regel *R13* in Abbildung 8.13).

8.2 Entwurf und Implementierung auf der Grundlage des Referenzmodells *PECS*

Der folgende Abschnitt erläutert das Implementierungskonzept für das Marktmodell. Ausgehend von der Grundstruktur des Modells werden alle wesentlichen Modellkomponenten hinsichtlich ihres Aufbaus und dynamischen Verhaltens betrachtet.

Als Grundlage für die Strukturierung des Marktmodells dient das Referenzmodell *PECS*. Die Implementierung des Modells erfolgt auf der Grundlage des Simulationssystems *Simplex3* und der darin enthaltenen Modellbeschreibungssprache *Simplex-MDL* (Schmidt, 2000).

8.2.1 Die Grundstruktur des Marktmodells

Das Simulationsmodell setzt sich aus einer Menge von unabhängigen und nebenläufig arbeitenden Modellkomponenten zusammen, die über insgesamt drei Hierarchieebenen hinweg zum Gesamtmodell *Market* aggregiert werden. Eine Interaktion der Modellkomponenten findet dabei auf der Grundlage von diskreten Informationsflüssen und kausalen Abhängigkeiten statt.

Auf der höchsten Hierarchieebene befindet sich das Gesamtmodell *Market*. Das Modell *Market* umfasst zwei Agenten der Klasse *Producer*, 16 Agenten der Klasse *Consumer*, sowie die Komponenten *Connector*, *Control* und *Statistic*. Die Komponente *Control* steuert den Gesamtablauf des Modells und gibt sowohl die Marktperioden als auch deren Unterteilung in die Transaktions- und Preisbestimmungsphase vor. In der Komponente *Statistic* werden statistische Daten (Preise und Absatzmengen) über die Marktaktivitäten der Transaktionsphase gesammelt, ausgewertet und als Eingangsinformationen für die Preisbestimmungsaktivitäten an die Producer-Agenten weitergeleitet. Die Komponente *Connector* stellt das Bindeglied zwischen den Produzenten und Konsumenten dar und ermöglicht den Informationsaustausch zwischen den Marktteilnehmern in der Transaktionsphase. Sie stellt für die Agenten beider Klassen entsprechende Postfächer für ein- und

ausgehende Nahrichten bereit. Die *Consumer*-Agenten modellieren schließlich die Nachfrageseite des Marktes, während die *Producer*-Agenten die Angebotsseite des Marktes repräsentieren.

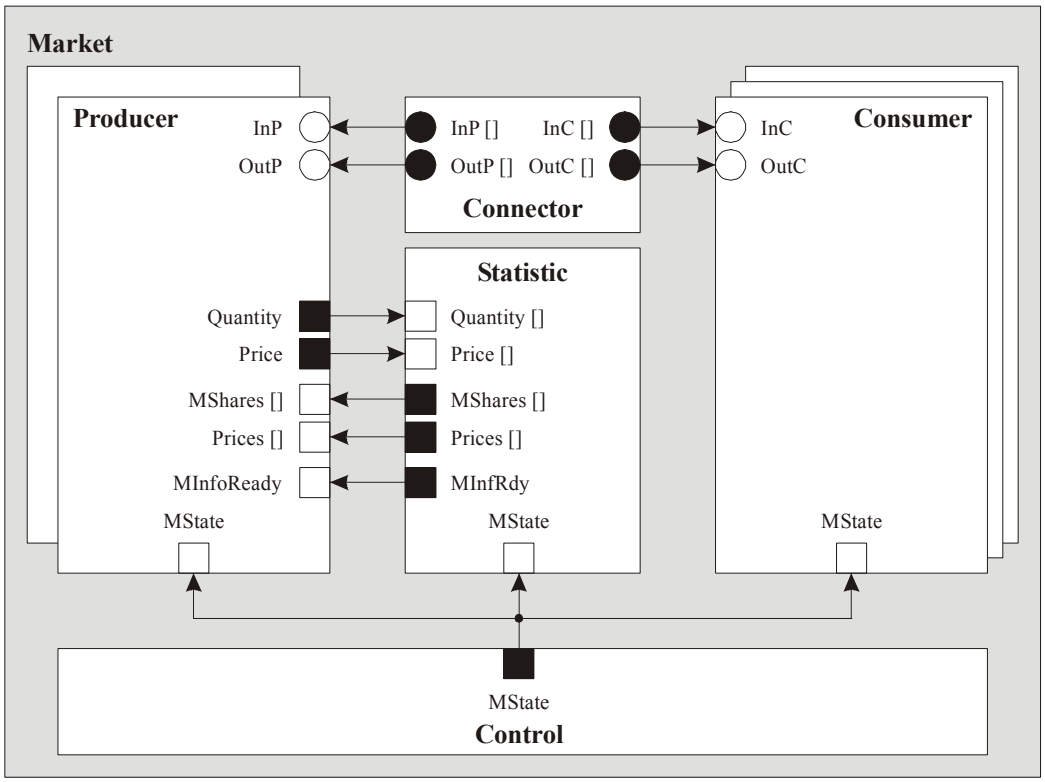

Abbildung 8.14: Die Grundstruktur des Marktmodells

Im Vergleich zur grundlegenden Komponentenstruktur des Referenzmodells *PECS* fehlt im Marktmodell die Komponente *Environment*. Diese Komponente wurde dem filternden Abstraktionsprozess unterzogen, da weder im Laborexperiment noch im daraus resultierenden Modellierungsansatz von Caldas und Coelho auf Umweltbedingungen, die sich auf Marktentwicklungen auswirken könnten, Rücksicht genommen wurde.

8.2.2 Die Komponente *Control*

Die Komponente *Control* steuert die Marktphasen entsprechend den Vorgaben aus Abschnitt 8.1.2. Mit Hilfe einer Zeitsteuerung wird der Markt abwechselnd für Transaktionen geöffnet und geschlossen. Die Information über die aktuelle Marktphase wird innerhalb der Komponente *Control* in der Modellgröße *MState* abgelegt und den *Consumer*-Agenten, den *Producer*-Agenten sowie der Komponente *Statistic* mit Hilfe von kausalen Abhängigkeiten zur Verfügung gestellt.

8.2.3 Die Komponente *Connector*

Die Interaktion der Konsumenten und Produzenten während der Transaktionsphasen beruht ausschließlich auf direkter Kommunikation zwischen den an einer Transaktion beteiligten Parteien. Die Komponente *Connector* unterstützt dabei den Informationsaustausch zwischen den Mitgliedern der beiden Agentenklassen, indem sie einen entsprechenden Mechanismus zur direkten Kommunikation zur Verfügung stellt.

Für jeden Agenten werden innerhalb der Komponente *Connector* zwei Bereiche eingerichtet. In den Ausgangsbereichen *OutP* bzw. *OutC* werden Nachrichten abgelegt, die für den betreffenden Agenten als Eingangsinformation bestimmt sind. In den Eingangsbereichen *InP* bzw. *InC* der Komponente *Connector* legen die Agenten Informationspakete ab, die jeweils an ein bestimmtes Mitglied der anderen Agentenklasse gerichtet sind. Die Komponente *Connector* spielt dabei lediglich eine Vermittlerrolle und stellt die Nachrichten in unveränderter Form den jeweiligen Empfängern zu.

```
MOBILE COMPONENT Message

LOCAL DEFINITIONS

  # Definition der Nachrichtentypen
  # -------------------------------
  VALUE SET MType  :  ('CAsk', 'COrder', 'POffer', 'PAck',
                       'MUnknown')

  # Definition der Agentenklassen (Absender)
  # ----------------------------------------
  VALUE SET MFrom  :  ('Cons', 'Prod', 'Unknown')

DECLARATION OF ELEMENTS

  STATE VARIABLES
  DISCRETE

  # Header-Informationen
  # --------------------
  Agent    (MFrom)   := 'Unknown',  # Typ des Absenders
  Sender   (INTEGER) := -1,         # ID des Senders
  Receiver (INTEGER) := -1,         # ID des Empfaengers
  Type     (MType)   := 'MUnknown', # Nachrichtentyp
  TStamp   (REAL)    := -1,         # Versandzeitpunkt

  # Nachrichteninhalt
  # -----------------
  Price    (REAL)    := -1,         # Preis
  Quantity (REAL)    := -1          # Menge

END OF Message
```

Abbildung 8.15: Der grundlegende Aufbau von Nachrichten im Marktmodell

Als Trägerobjekte für die zwischen den Agenten ausgetauschten Nachrichten dienen mobile Komponenten der Klasse *Message*. Jede Nachricht trägt dabei sowohl Metainformationen, die etwa den korrekten Ablauf der Nachrichtenübermittlung sicherstellen oder dem Nachrichtenempfänger eine korrekte Auswertung der Nachrichteninhalte ermöglichen, sowie die eigentlichen Nutzinformationen. Der Kopf einer Nachricht im Marktmodell enthält als Metainformationen die Agentenklasse des Absenders, die Identifikation des Absenders, den Empfänger, den Typ sowie den Absendezeitpunkt der Nachricht. Im Rumpf einer Nachricht werden in Abhängigkeit des Nachrichtentyps Preisinformationen und Kaufmengen übertragen. Für die Abbildung der Kommunikation zwischen den Agenten im Marktmodell sind die folgenden Nachrichtentypen erforderlich:

- *CAsk*

  Mit Hilfe einer Nachricht des Typs *CAsk* fordert ein Konsument einen Produzenten dazu auf, seinen aktuellen Marktpreis bekannt zu geben.

- *POffer*

  Die Antwort des Produzenten auf eine derartige Preisanfrage wird mit Hilfe einer Nachricht des Typs *POffer* codiert. Als Nutzinformation enthält eine derartige Nachricht den Preis des Produzenten.

- *COrder*

  Nachdem ein Konsument festgelegt hat, wie hoch seine Nachfrage in der aktuellen Transaktionsphase ausfallen wird, löst er unter Verwendung einer Nachricht des Typs *COrder* und Angabe einer entsprechenden Menge eine Bestellung beim betreffenden Produzenten aus.

- *PAck*

  Der Produzent bestätigt dem Konsumenten schließlich zum Abschluss einer Transaktion die Liefermenge und den zugrunde gelegten Preis auf der Grundlage einer Nachricht des Typs *PAck*.

Die Abbildung 8.15 zeigt den grundsätzlichen Aufbau einer Nachricht in der Modellbeschreibungssprache *Simplex-MDL*.

8.2.4 Die Komponente *Statistic*

Die Komponente *Statistic* protokolliert während der Transaktionsphasen die Preise sowie die Absatzmengen der beiden Produzenten. Diese Informationen werden innerhalb der Komponente *Statistic* dazu verwendet, um statistische Daten zu generieren, die anschließend an die beiden *Producer*-Agenten zur weiteren Verarbeitung übergeben werden. Zu den erzeugten Marktdaten gehören dabei die Preise, zu denen beide Produzenten in der letzten Periode ihre Produkte auf dem Markt angeboten haben, sowie die aus den jeweiligen Absatzmengen und dem Gesamtabsatz auf dem Markt resultierenden Marktanteile beider Produzenten.

8.2.5 Die *Consumer*-Agenten

Die *Consumer*-Agenten realisieren im Marktmodell die Nachfrageseite des Marktes. Sie sind nach dem Vorbild von *PECS*-Agenten konzipiert und stellen somit hierarchisch strukturierte Modellkomponenten dar. Die Modellkomponente *Consumer* instanziiert und aggregiert als Strukturkomponente auf der übergeordneten Hierarchieebene die Subkomponenten *CSensor*, *CPerception*, *CCognition*, *CStatus*, *CBehaviour* und *CActor*. Die beiden *PECS*-Komponenten *Emotion* und *Physis* sind für die Modellierung der Konsumenten des Marktmodells ohne Bedeutung und fehlen daher in der Architektur der *Consumer*-Agenten.

Wesentlich für die Funktionalität der *Consumer*-Agenten sind die Komponenten *CCognition* und *CBehaviour*. Innerhalb der Komponente *CCognition* werden alle Informationen verwaltet und aktualisiert, auf denen die Verhaltenssteuerung der *Consumer*-Agenten aufsetzt. Die Verhaltenssteuerung wird dabei innerhalb der Komponente *CBehaviour* realisiert und umfasst die Aktivitäten Einholen von Preisangeboten, Auswahl des bevorzugten Produzenten und Absenden einer Bestellung.

Die Komponenten *CSensor* und *CPerception* dienen der Verarbeitung von Informationen aus der Umgebung der *Consumer*-Agenten. Zu diesen Informationen zählen der aktuelle Marktzustand, der durch die Komponente *Control* berechnet und bereitgestellt wird, sowie die Nachrichten, die im Verlauf der Transaktionsphasen von den Produzenten abgeschickt werden. Die Komponente *CStatus* protokolliert die Historie der Käufe, die ein Konsument im Verlauf der Marktperioden getätigt hat. Die Komponente *CActor* ist für die Ausführung der Aktionen des *Consumer*-Agenten zuständig. Die *Consumer*-Agenten verfügen dabei lediglich über die beiden Aktionen *Preisanfrage abschicken* und *Bestellung auslösen*, die jeweils an *Producer*-Agenten gerichtet sind. Die Codierung der Aktionen erfolgt mit Hilfe von mobilen Komponenten der Klasse *CAction*.

Informationen, die zwischen den Subkomponenten der *Consumer*-Agenten ausgetauscht werden, werden durch eine eigene Klasse *CThought* von mobilen Komponenten gekapselt. Auf diese Weise entstehen innerhalb der Agentenarchitektur sehr schlanke Schnittstellen, die eine sehr einfache Erweiterung der Wechselwirkungen zwischen den einzelnen Subkomponenten zulassen.

Aufgrund der großen Ähnlichkeit, die die Funktionalitäten der einander entsprechenden Komponenten in den *Consumer*- und den *Producer*-Agenten aufweisen, soll an dieser Stelle auf eine detaillierte Darstellung der Subkomponenten *CSensor*, *CPerception*, *CStatus* und *CActor* verzichtet werden. Entsprechende Ausführungen, die zum großen Teil auf die *Consumer*-Agenten übertragen werden können, finden sich bei der Implementierungsbeschreibung der Producer-Agenten (s. Abschnitt 8.2.6). Die nun folgenden Abschnitte beschäftigen sich in ausführlicherer Weise nur mit den Subkomponenten *CCognition* und *CBehaviour* der *Consumer*-Agenten.

8.2.5.1 Die Komponente *CCognition*

Innerhalb der Komponente *CCognition* verwalten und aktualisieren *Consumer*-Agenten ihre Wissensbasis. Zu den in dieser Komponente vorgehaltenen Informationen gehören im wesentlichen die Liste der bekannten Produzenten einschließlich der zugeordneten Präferenzen, der aktuelle Marktzustand, Abarbeitungsinformationen, die im Laufe der Transaktionsphasen entstehen, sowie die individuelle Preis-Nachfrage-Kurve des Konsumenten. Die Liste der bekannten Produzenten, die Präferenzen und auch die Preis-Nachfrage-Kurve werden im Marktmodell als konstant betrachtet und nur zu Beginn eines Simulationslaufes einmalig belegt. Der aktuelle Marktzustand und die Abarbeitungsinformationen der Transaktionsphasen, die das Arbeitsgedächtnis eines *Consumer*-Agenten bilden, werden nach Verfügbarkeit entsprechender Informationen fortlaufend aktualisiert. Den Anstoß dafür geben neu eingetroffene, mobile Komponenten der Klasse *CThought*, die sowohl durch die Komponente *CPerception* als auch durch die Komponente *CBehaviour* an die Komponente *CCognition* herangetragen werden.

Die Komponente *CPerception* versorgt die Komponente *CCognition* dabei mit neuen Informationen aus der Umgebung und verwendet dafür *PThought*-Nachrichten der folgenden Typen:

- *CTMOpen*

  Mit Hilfe einer Nachricht des Typs *CTMOpen* wird die Komponente *CCognition* darüber informiert, dass eine neue Transaktionsphase begonnen hat. Sie speichert diese Information daraufhin in der Modellgröße *MState* und aktiviert dadurch die Komponente *CBehaviour*.

- *CTMClose*

  Nachrichten des Typs *CTMClose* werden verwendet, um der Komponente *CCognition* das Ende einer Transaktionsphase anzuzeigen und alle im Rahmen einer Transaktionsphase verwendeten Modellgrößen für die nächste Periode in ihren Initialzustand zu versetzen.

- *CTOffer*

  Eine *CThought*-Nachricht des Typs *CTOffer* signalisiert die Verfügbarkeit eines neuen Preisangebotes. Der mitgeteilte Preis des Produzenten wird innerhalb der Komponente *CCognition* gespeichert. Liegen für alle Preisanfragen der aktuellen Transaktionsphase entsprechende Angebote vor, so wird die Komponente *CBehaviour* aufgefordert, eine Kaufentscheidung zu treffen und anschließend eine entsprechende Bestellung auszulösen.

- *CTAck*

  Mit Hilfe einer Nachricht des Typs *CTAck* wird bekannt gegeben, dass der Produzent, bei dem eine Bestellung ausgelöst wurde, diese Bestellung akzeptiert hat. Es kommt ein Kauf zustande, der anschließend innerhalb der Komponente *CStatus* protokolliert wird

Die Komponente *CBehaviour* informiert die Komponente *CCognition* unter Verwendung der folgenden Typen von *CThought*-Nachrichten über Entscheidungen, die im Rahmen der Verhaltenssteuerung getroffen wurden:

- *CTAskUpd*

  Innerhalb der Komponente *CCognition* wird vermerkt, wie viele Preisanfragen im Rahmen der Transaktionsphase an welche Produzenten verschickt wurden, um die später eintreffenden Preisangebote mit den Anfragen abzugleichen.

- *CTSelPId*

  Mit Hilfe einer *CThought*-Nachricht des Typs *CTSelPId* gibt die Komponente *CBehaviour* bekannt, welcher der beiden Produzenten in der betrachteten Marktperiode den Vorzug erhält. Zugleich stellt diese Nachricht für die Komponente *CCognition* eine Aufforderung dar, anhand der Preis-Nachfrage-Kurve und des Preises des ausgewählten Produzenten die Bestellmenge für die vorliegende Periode festzulegen.

- *CTOrdUpd*

  Zum Abschluss einer Transaktion protokolliert die Komponente *CCognition* die Bestellung der aktuellen Marktperiode, um später die Bestellbestätigung des Produzenten überprüfen zu können.

8.2.5.2 Die Komponente *CBehaviour*

Die Komponente *CBehaviour* ist verantwortlich für die Verhaltenssteuerung der *Consumer*-Agenten. Sie löst Preisanfragen an die *Producer*-Agenten aus und entscheidet im Zusammenspiel mit der Komponente *CCognition* über das Kaufverhalten des *Consumer*-Agenten.

Unmittelbar nach Beginn einer neuen Transaktionsphase veranlasst die Komponente *CBehaviour* die Komponente *CActor* dazu, Preisanfragen an alle bekannten Produzenten abzuschicken. Zu diesem Zweck iteriert die Komponente *CBehaviour* über die Liste der bekannten Produzenten, die in der Komponente *CCognition* verwaltet wird, und erzeugt dabei entsprechende mobile Komponenten der Klasse *CAction*, die als Ausführungsanordnungen an die Komponente *CActor* weitergeleitet werden. Parallel zur Ausführungsanordnung an die Komponente *CActor* wird auch jeweils eine Update-Information (*CThought*) an die Komponente *CCognition* geschickt, um dort die Liste der Preisanfragen im Arbeitsgedächtnis zu erweitern.

Die eingetroffenen Preisangebote der Produzenten werden in der Komponente *CCognition* gespeichert. Sind alle Preisangebote vorhanden, setzt die Komponente *CCognition* die Verhaltenssteuerung davon in Kenntnis. In der Komponente *CBehaviour* wird daraufhin eine Sequenz von Ereignissen ausgelöst, die das Ziel verfolgen, das Nachfrageverhalten des Konsumenten zu bestimmen. Im ersten Schritt

werden dazu die Präferenzen für die Produzenten nach dem in Abschnitt 8.1.2.2 dargestellten Schema ermittelt und der in der aktuellen Marktperiode bevorzugte Produzent ausgewählt. Im nächsten Schritt wird die Komponente *CCognition* aufgefordert, anhand des aktuellen Preises des ausgewählten Produzenten und der Preis-Nachfrage-Kurve die Nachfragemenge für die aktuelle Periode festzulegen. Steht die entsprechende Mengeninformation zur Verfügung, übernimmt die Komponente *CBehaviour* erneut die Kontrolle und erzeugt eine Ausführungsanordnung, mit der die Komponente *CActor* dazu aufgefordert wird, eine Bestellung über die entsprechende Menge an den ausgewählten Produzenten zu schicken. Analog zur Preisanfrage wird auch die Bestellinformation an die Komponente *CCognition* zur Ablage im Arbeitsgedächtnis weitergeleitet, um die anschließend eintreffende Bestellbestätigung des Produzenten mit den ursprünglichen Bestelldaten abgleichen zu können.

8.2.6 Die *Producer*-Agenten

Die *Producer*-Agenten stellen die komplexesten Komponenten im Marktmodell dar. Sie haben die Aufgabe, die Angebotsseite des Marktes zu repräsentieren. In dieser Hinsicht stehen sie im Rahmen von Transaktionen mit den *Consumer*-Agenten in Verbindung. Ebenso interagieren die *Producer*-Agenten auch indirekt mit ihrem jeweiligen Konkurrenten auf dem Markt, indem sie bei der Preisbildung nicht nur ihre eigenen Interessen verfolgen, sondern ihre Aktivitäten strategisch auch auf das Preisbildungsverhalten ihres Konkurrenten ausrichten.

Die *Producer* sind als *PECS*-Agenten konzipiert und bestehen aus den Komponenten *PSensor*, *PPerception*, *PCognition*, *PStatus*, *PBehaviour* und *PActor*. Auch bei den Produzenten spielen in Analogie zu den Konsumenten weder physische noch emotionale Befindlichkeiten ein Rolle. Aus diesem Grund finden auch bei dieser Agentenklasse die Komponenten *Physis* und *Emotion*, die grundsätzlich Bestandteil der *PECS*-Architektur sind, keine Verwendung.

Die Komponentenstruktur der *Producer*-Agenten zeigt die Abbildung 8.16. Wie aus dieser Abbildung sehr gut erkennbar ist, verfügen auch die *Producer*-Agenten intern über sehr schlanke Schnittstellen. Die Ursache dafür liegt darin, dass für den Informationstransport zwischen den Komponenten der Agentenarchitektur weitestgehend eine eigene Klasse *PThought* mobiler Komponenten verwendet wird, die Informationen verschiedener Typen aufnehmen und transportieren kann. Auf diese Weise entsteht eine sehr überschaubare Verbindungsstruktur, auf deren Grundlage Erweiterungen an der Agentenarchitektur sehr einfach möglich werden. Der einzige Ort, an dem innerhalb der Agentenarchitektur in größerem Umfang von der Informationsübermittlung mit Hilfe von diskreten Informationspaketen abgewichen wird, befindet sich an der Schnittstelle der Komponenten *CCognition* und *CBehaviour*. An dieser Stelle werden zur Informationsübergabe der Einfachheit halber Lesezugriffe auf die erforderlichen Modellgrößen verwendet.

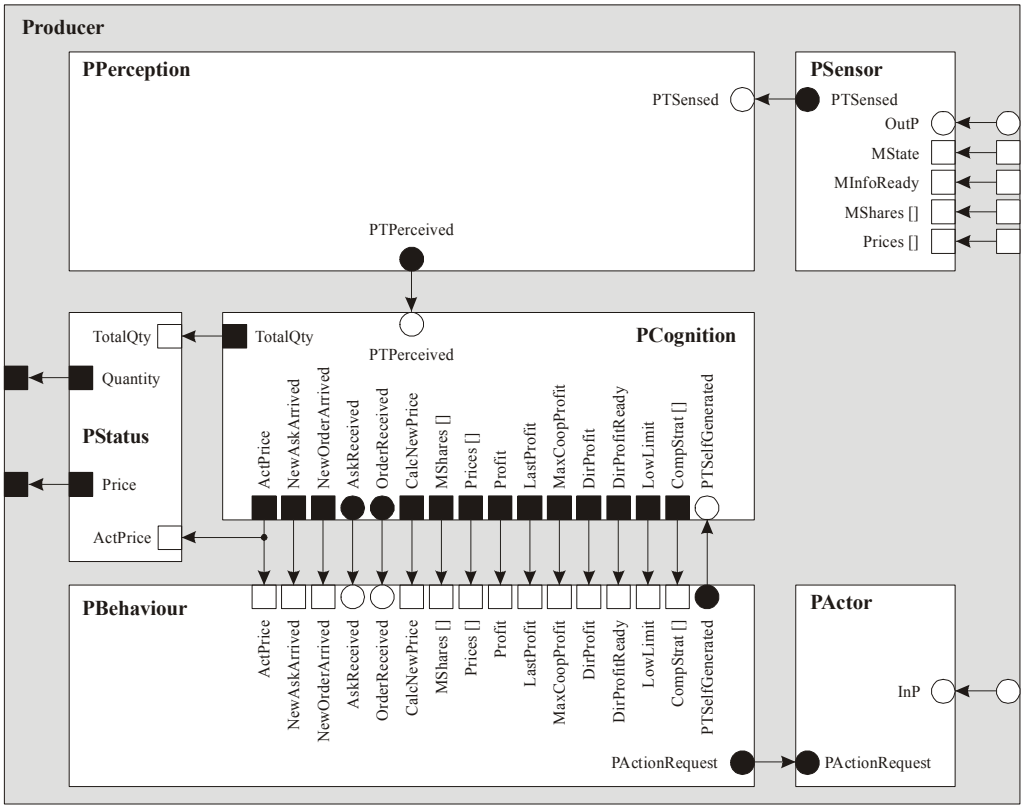

Abbildung 8.16: Die Architektur der Producer-Agenten

8.2.6.1 Die Komponente *PSensor*

Die Komponente *PSensor* bildet die Eingangsschnittstelle der *Producer*-Agenten. Sie steht über die Eingangslocation *OutP* mit der Komponente *Connector* in Kontakt, um die von den Konsumenten abgeschickten Nachrichten in Empfang zu nehmen. Ebenso übernimmt die Komponente *PSensor* in der Modellgröße *MState* den aktuellen Marktzustand, der in der Komponente *Control* verwaltet wird, sowie die statistischen Marktdaten zu den Preisen (*Prices*) und Marktanteilen (*MShares*) der Produzenten aus der Komponente *Statistic*.

Im Zuge der Verarbeitung der Eingangsinformationen generiert die Komponente *PSensor* mobile Komponenten der Klasse *PThought* und weist diesen je nach Inhalt unterschiedliche Typen zu. Diese Informationspakete werden anschließend in der Warteschlange *PTSensed* abgelegt und von dort aus zur weiteren Verarbeitung durch die Komponente *PPerception* abgezogen.

8.2.6.2 Die Komponente *PPerception*

Innerhalb der Komponente *PPerception* werden die mobilen Komponenten der Klasse *PThought* unmittelbar nach ihrem Eintreffen ohne weitere Modifikation an

die Komponente *PCognition* weitergeleitet. Die Komponente *PPerception* erfüllt im hier dargestellten Status des Marktmodells also noch keine grundlegende Aufgabe. Sie wurde allerdings trotzdem als Platzhalter mit in die Agentenarchitektur aufgenommen, um spätere Erweiterungen des Modells zu vereinfachen.

8.2.6.3 Die Komponente *PCognition*

Die Komponente *PCognition* verwaltet grundlegende Informationen eines Producer-Agenten über die aktuelle Marktsituation, das Verhalten des Mitbewerbers, sowie sein Arbeitsgedächtnis. Die Komponente *PCognition* steht mit den Komponenten *PPerception*, *PBehaviour* und *PStatus* in Verbindung.

Die Verbindungsstruktur der Komponente PCognition

Die Komponente *PPerception* trägt in Form von mobilen Komponenten der Klasse *PThought* Informationen aus der Umgebung des Agenten an die Komponente *PCognition* heran. Als Eingangsinformationen erhält die Komponente *PCognition* dabei Preisanfragen und Bestellungen der Konsumenten, den aktuellen Marktzustand, sowie die Marktanteile und Preise der letzten Transaktionsphase.

Zur Komponente *PBehaviour* besteht eine wechselseitige Verbindung. Die Komponente *PCognition* liefert Informationen aus dem Wissensspeicher des Agenten an die Komponente *PBehaviour*, die im Rahmen der Verhaltenssteuerung von Bedeutung sind. Hierzu zählen, um nur einige zu nennen, die Marktanteile, Preise, Gewinne oder auch die Gewinnentwicklung, die im Rahmen der *Cooperation-leader-Strategie* Verwendung findet. Die Komponente *PBehaviour* gibt ihrerseits Informationen über zur Ausführung angeordnete Aktion an die Komponente *PCognition* zurück, die dort zu einer Aktualisierung des Arbeitsgedächtnisses führen. Übergeben werden hierbei beispielsweise Entscheidungen im Rahmen der Preisbestimmung oder auch Informationen über bestätigte Bestellungen, die zu einer Erhöhung der Absatzmenge in der laufenden Periode führen.

Der aktuelle Preis des Produzenten sowie der Gesamtabsatz je Periode werden von der Komponente *PCognition* an die Komponente *PStatus* zur Aufzeichnung übergeben.

Die Funktionsweise der Komponente PCognition

Die Komponente *PCognition* verfügt im Marktmodell über keine Eigendynamik. Sie nimmt nur dann ihre Arbeit auf, wenn sie durch einen externen Stimulus dazu aufgefordert wird. Derartige Stimuli werden, wie eingangs bereits erwähnt, durch die beiden Komponenten *PPerception* und *PBehaviour* bereitgestellt.

Die Verarbeitung von Eingangsinformationen aus der Komponente *PPerception*. Im Zusammenspiel mit der Komponente *PPerception* überprüft die Komponente *PCognition* die Eingangslocation *PTPerceived* auf neu eingetroffene *PThought*-Komponenten. Steht in *PTPerceived* ein neues Informationspaket bereit, wird es in die Komponente *PCognition* übernommen und ausgewertet. Die Auswertung wird durch den Typ der *PThought*-Nachricht gesteuert, der angibt, um welche Art von Information es sich handelt. Hierbei sind die folgenden Typen für *PThought*-Nachrichten vorgesehen:

- *PTAsk*

  Mit Hilfe einer Nachricht des Typs *PTAsk* wird bekannt gegeben, dass eine neue Preisanfrage eines Konsumenten eingetroffen ist. Im Rahmen der Verarbeitung dieser Nachricht werden die Preisanfrage in einer eigenen Liste, sowie der Absender der Preisanfrage als spätere Referenz gespeichert. Die weitere Verarbeitung findet anschließend in der Komponente *PBehaviour* statt.

- *PTOrder*

  Der Typ *PTOrder* kennzeichnet eine Bestellung eines Konsumenten. Auch in diesem Fall werden sowohl die Bestellung als auch deren Absender für die nachgelagerten Verarbeitungsschritte in der Komponente *PCognition* gespeichert. Zudem erhöht sich die Gesamtbestellmenge für die laufende Periode nach Eingang einer neuen Bestellung. Eine entsprechende Reaktion des Produzenten auf eine Bestellung wird in der Komponente *PBehaviour* generiert.

- *PTMOpen / PTMClose*

  Die beiden Typen *PTMOpen* und *PTMClose* kennzeichnen den Beginn bzw. das Ende einer Transaktionsphase und führen zu einer Aktualisierung der innerhalb der Komponente *PCognition* gespeicherten Modellgröße *MState*, die den aktuellen Marktzustand aufnimmt.

- *PTInfRdy*

  Das Eintreffen einer mobilen Komponente des Typs *PTInfRdy* signalisiert, dass die statistischen Daten der letzten Transaktionsphase bereitstehen und damit eine neue Preisbestimmungsphase beginnen kann. Im ersten Schritt werden zu diesem Zweck die Steuerungsgrößen der Preisbestimmungsstrategien ermittelt. Dazu gehört, dass die aktuellen Preise und Marktanteile aller Produzenten in der Komponente *PCognition* gespeichert werden, ebenso wie der aktuelle Gewinn und der Gewinn der Vorperiode. Haben in der betrachteten Periode beide Produzenten ihre Produkte zu demselben Preis angeboten, wird der daraus entstandene Gesamtgewinn und dessen Maximalwert hinterlegt. Diese Informationen dienen einem *Cooperation-leader* als wesentliche Eingangsinformation für seine Preisbestimmung. Abschließend versucht der Produzent auf Grundlage der Preisbildung und der Marktanteile seines Mitbewerbers Rückschlüsse auf dessen Preisbestimmungsstrategie zu ziehen und speichert diese Annahme ebenso für die spätere Verwendung im Rahmen sei-

ner eigenen Strategiefestlegung. Im Anschluss daran wird die Komponente *PBehaviour* dazu aufgefordert, die eigentliche Preisbestimmung vorzunehmen.

Die Verarbeitung von Eingangsinformationen aus der Komponente *PBehaviour*. Die Komponente *PBehaviour* legt im Rahmen ihrer Aktivitäten in der Eingangslocation *PTUpdate* Rückmeldungen für die Komponente *PCognition* ab. Die Komponente *PCognition* nimmt diese Nachrichten in Empfang und wertet sie entsprechend ihrer Typen aus. Hierbei treten die folgenden Nachrichtentypen auf:

- *PTOffUpd*

  Eine Nachricht des Typs *PTOffUpd* informiert die Komponente *PCognition* über ein an einen Konsumenten abgeschicktes Preisangebot. Der angebotene Preis wird im Zuge der Verarbeitung dieser Nachricht beim betreffenden Konsumenten gespeichert.

- *PTAckUpd*

  Mit Hilfe einer Nachricht des Typs *PTAckUpd* wird der Komponente *PCognition* bekannt gegeben, dass infolge einer Bestellung eine Bestellbestätigung an den entsprechenden Konsumenten geschickt wurde und die damit verbundene Transaktion als abgeschlossen zu betrachten ist.

- *PTPrcUpd*

  Die Komponente *PBehaviour* legt im Rahmen der Preisbestimmungsstrategie des Agenten den Marktpreis für die jeweils folgende Transaktionsphase fest. Diesen neuen Marktpreis teilt die Komponente *PBehaviour* der Komponente *PCognition* mit Hilfe einer *PThought*-Komponente des Typs *PTPrcUpd* zur Aktualisierung des Arbeitsgedächtnisses mit.

- *PTFindDir*

  Im Rahmen der *Cooperation-leader-Strategie* wird die Gewinnentwicklung als elementarer Parameter benötigt. Die Komponente *PCognition* generiert den aktuellen Wert für die Gewinnentwicklung unmittelbar nach Anfrage durch die Komponente *PBehaviour*. Diese Anfrage wird mit Hilfe einer *PThought*-Nachricht des Typs *PTFindDir* gestellt. Zur Ermittlung der Gewinnentwicklung geht die Komponente *PCognition* nach dem in Abbildung 8.8. dargestellten Schema vor und übergibt anschließend das Resultat zur weiteren Verarbeitung an die Komponente *PBehaviour*.

8.2.6.4 Die Komponente *PBehaviour*

Die Komponente *PBehaviour* steht in enger Verbindung mit der Komponente *PCognition* und steuert auf der Grundlage der im Arbeitsgedächtnis des Agenten

vorhanden Informationen dessen Verhalten. Die Verhaltenssteuerung der *Producer*-Agenten basiert auf reaktiven Mechanismen, die über interne oder externe Trigger aktiviert werden. Im Rahmen der Verhaltenssteuerung werden sowohl intern als auch extern wirksame Aktionen ausgelöst. Zu den externen Aktionen gehören die Beantwortung von Preisanfragen sowie die Bestätigung von Bestellungen. Nach innen hin wirksam sind die Aktionen zur Preisbestimmung.

Die Beantwortung einer Preisanfrage wird durch die Komponente *PCognition* angestoßen. Die Komponente *PCognition* signalisiert zu diesem Zweck der Komponente *PBehaviour*, dass eine Preisanfrage eines Konsumenten vorliegt, die beantwortet werden muss. Die Komponente *PBehaviour* generiert als Reaktion darauf eine Ausführungsanordnung für die Komponente *PActor*, die im wesentlichen den Preis sowie die Identifikation des betreffenden Konsumenten als Adressinformation enthält.

In ähnlicher Weise wird verfahren, wenn eine Bestellung vorliegt, die dem betreffenden Konsumenten bestätigt werden muss. Auch in diesem Fall gibt die Komponente *PCognition* den Anstoß, während die Komponente *PBehaviour* eine entsprechende Ausführungsanordnung erzeugt, in der die bestellte Menge sowie der gültige Preis hinterlegt sind.

Nachdem in der Komponente *PCognition* die aktuellen Marktdaten in das Arbeitsgedächtnis integriert wurden, beginnt die Komponente *PBehaviour* mit der Bestimmung des neuen Marktpreises. Im ersten Schritt werden hierzu zunächst einmal das Vertrauen in die verfolgte Strategie bestimmt und die eigentliche Preisbildungsstrategie festgelegt. Liegen diese grundsätzlichen Informationen vor, beginnt die eigentliche Preisbestimmung, wie sie in Abschnitt 8.1.2.3 im Rahmen der Strategien *Cut-throat*, *Cooperation-leader* und *Cooperation-follower* dargestellt wurde. Als Resultat der Preisbestimmung wird eine der möglichen Aktionen zur Preisfestlegung ausgewählt. Dadurch wird eine entsprechende Rückmeldung an die Komponente *PCognition* erzeugt, mit deren Hilfe schließlich eine Anpassung des für die nächste Periode gültigen Preises vorgenommen wird.

8.2.6.5 Die Komponente *PStatus*

Die Komponente *PStatus* übernimmt den Marktpreis sowie die gesamte Absatzmenge einer Periode aus der Komponente *PCognition* und stellt diese Informationen nach außen hin der Komponente *Statistik* zur Auswertung bereit.

8.2.6.6 Die Komponente *PActor*

Die Komponente *PActor* bildet die Aktionsschnittstelle der *Producer*-Agenten. Sie nimmt die Ausführungsanordnungen, die in der Komponente *PBehaviour* erzeugt werden, entgegen und setzt diese in entsprechende Aktionen um. Die Aktionen werden dabei mit Hilfe von mobilen Komponenten der Klasse *Message* abgebildet, da es sich bei allen externen Aktionen um Nachrichten handelt, die über

die Komponente *Connector* an die *Consumer*-Agenten verschickt werden. Die Verbindung zur Komponente *Connector* wird dabei über die Eingangslocation *InP* hergestellt, in der die für die Übertragung bestimmten Nachrichten unmittelbar nach ihrer Generierung abgelegt werden.

8.3 Simulationsexperimente

Der folgende Abschnitt beschäftigt sich mit den Experimentiermöglichkeiten, die das Marktmodell bietet, und geht zu diesem Zweck exemplarisch von einem Simulationsexperiment aus, in dem der Versuch unternommen wird, die in Abschnitt 8.1.1.3 dargestellten Ergebnisse des Laborexperiments im Modell zu reproduzieren. Das im folgenden dargestellte Experiment lässt also in erster Näherung Rückschlüsse auf die Validität des Simulationsmodells und insbesondere auf die Annahmen hinsichtlich der Verhaltenssteuerung der Produzenten zu, die Hauptgegenstand der Untersuchung von Caldas und Coelho waren.

Das Experiment geht von insgesamt 18 Agenten aus, wobei die Nachfrageseite durch 16 *Consumer*-Agenten und die Angebotsseite, wie im Laborexperiment, durch zwei *Producer*-Agenten abgebildet wird.

Die *Consumer*-Agenten werden zu Beginn des Experimentes mit individuellen Präferenzen und Preis-Nachfrage-Kurven ausgestattet. Die Parametrisierung wird dabei allerdings in der Weise vorgenommen, dass gemäß den Grundannahmen des Laborexperimentes eine symmetrische Nachfrage entsteht. (Die gesamte Parametrisierung des vorliegenden Experimentes ist in Anhang B dargestellt.)

Die Parametrisierung der Producer-Agenten zeigt die Abbildung 8.17.

Modellgröße	**Startwert**	
	Producer 1	**Producer 2**
LowLimit	48	48
InitialCredit	150	100
Personality	calm	calm
ActPrice	57	55

Abbildung 8.17: Die Parametrisierung der Producer-Agenten

Das vorliegende Experiment geht also von zwei ruhigen Produzenten mit moderater Preisbildung aus, die ebenfalls mit einer identischen Preisuntergrenze ausgestattet sind. Unterschiede ergeben sich zwischen den beiden Produzenten allerdings hinsichtlich ihres Startpreises und ihres Vertrauens in die Preisbestimmungsstrategie. Der Produzent 1 startet mit einem Preis von 57 in die erste Transaktionsphase und wird dementsprechend zunächst einmal weniger Marktanteile

für sich verbuchen als Produzent 2, der mit einem Preis von 55 beginnt. Andererseits verfügt der Produzent 1 anfangs über ein höheres Vertrauen in seine Preisbestimmungsstrategie als Produzent 2. Produzent 1 beginnt hinsichtlich seiner Modellgröße Credibility mit einem Wert von 150, während diese Modellgröße bei Produzent 2 mit dem Wert 100 vorbelegt wird.

Auf der Grundlage dieser Starteinstellungen ergibt sich die Frage, welches Modellverhalten zu erwarten ist. Sicherlich werden im Laufe der Preisbildung keine größeren Preissprünge stattfinden, da beide Produzenten als ruhig eingestuft sind und damit Preissignale nicht zum Aktionsrepertoire gehören. Weiterhin lässt sich vermuten, dass Produzent 2 aufgrund des geringeren Vertrauens in seine Preisbildungsstrategie früher zur *Cooperation-Strategie* wechseln wird als Produzent 1. Entsprechend würde dann Produzent 2 im Rahmen der *Cooperation-Strategie* die *Leader*-Rolle übernehmen, da er der erste Produzent wäre, der in dieser Hinsicht die Initiative ergreifen würde. Produzent 1 würde entsprechend nach einer gewissen Zeit bzw. genau dann, wenn er kein ausreichendes Vertrauen mehr in die *Cut-throat-Strategie* hat, als *Cooperation-follower* eine gemeinsame Preissuche unterstützen.

Die zentrale Frage des Experimentes ist nun, ob es gelingt, den in Abbildung 8.1 dargestellten Verlauf der Preise, wie er im Laborexperiment entstanden ist, im Modell anzunähern. Den Preisverlauf, den das Simulationsexperiment tatsächlich hervorbringt, zeigt die Abbildung 8.18.

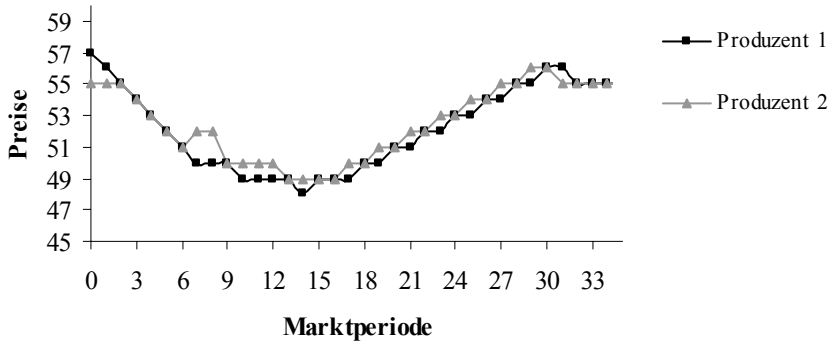

Abbildung 8.18: Der Preisverlauf im Simulationsexperiment

Abbildung 8.18 zeigt, dass beide Produzenten zunächst einmal im Rahmen der *Cut-throat-Strategie* versuchen, ihre Preise zu verringern, um höhere Marktanteile zu erreichen. Zum Zeitpunkt 7 verliert der Produzent 2 zum ersten mal sein Vertrauen in die Wettbewerbsstrategie. Da er von der Annahme ausgeht, dass Produzent 1 immer noch die *Cut-throat-Strategie* verfolgt, übernimmt er zu diesem Zeitpunkt die Rolle des *Cooperation-leaders*. Entsprechend erhöht er zu diesem Zeitpunkt seinen Preis um eine Einheit und signalisiert dadurch seine Kompromissbereitschaft. In der darauf folgenden Marktperiode erkennt Produzent 1 zwar an der Preiserhöhung, dass Produzent 2 zur *Cooperation-Strategie* gewechselt ist, führt aber zunächst noch die Wettbewerbsstrategie fort und genießt

selt ist, führt aber zunächst noch die Wettbewerbsstrategie fort und genießt die dadurch entstehenden, hohen Marktanteile. Produzent 2 hingegen verbleibt gemäß der Kooperationsstrategie zunächst auf dem gewählten Preisniveau und wartet auf eine entsprechende Preisreaktion seines Mitbewerbers. Da Produzent 1 allerdings nicht in der gewünschten Weise agiert, führt Produzent 2 zum Zeitpunkt 9 eine *retaliate*-Aktion aus und setzt damit seinen Preis auf das Niveau von Produzent 1. Produzent 1 reagiert gemäß der *Cut-throat-Strategie* in dieser Situation mit einer Preissenkung und erhöht seinen Marktanteil dadurch erneut, während Produzent 2 in der Folge seinen Preis unverändert lässt. Da Produzent 1 aber auch im zweiten Anlauf nicht auf das Kooperationsangebot von Produzent 2 eingeht, reagiert dieser in Periode 13 erneut mit einer *retaliate*-Aktion. Zum Zeitpunkt 15 verliert schließlich auch Produzent 1 aufgrund der anhaltend geringen Gewinne das Vertrauen in die Wettbewerbsstrategie und nimmt die Rolle des *Cooperation-follower* ein. Von nun an beginnt gemäß der Annahmen der *Cooperation-Strategien* eine gemeinsame Preissuche, die durch Produzent 2 angeführt wird. Die Preissuche endet nach insgesamt 32 Marktperioden auf dem optimalen Preisniveau, das bei dem Wert 55 angesiedelt ist.

Als Fazit kann aus diesem Experiment gezogen werden, dass das Modell für die eingangs geschilderten Randbedingungen das Laborexperiment sehr gut annähert. Der Preisverlauf im Modell deckt sich nahezu mit dem Preisverlauf des Laborexperiments und nach einer vergleichbar langen Interaktionsdauer führen sowohl das Laborexperiment als auch das Simulationsmodell zum optimalen Gleichgewichtspreis. Das Modell kann also für das hier zugrunde gelegte Szenario als gute Näherung der Realität betrachtet werden.

8.4 Zusammenfassung und Fazit

Das in diesem Abschnitt vorgestellte Marktmodell demonstriert in grundsätzlicher Weise, wie agentenbasierte Simulationsmodelle zur Theorievalidierung im Bereich der experimentellen Ökonomie eingesetzt werden können. Grundlage für diese Fallstudie ist eine Arbeit von José Castro Caldas und Helder Coelho (1992). Im Rahmen dieser Arbeit wurden Laborexperimente mit Testpersonen durchgeführt, deren Aufgabe darin bestand, die Rolle von Produzenten auf einem Oligopol-ähnlichen Markt zu übernehmen. Mit Hilfe der in den Experimenten gewonnen Daten wurde eine Theorie über das Preisbildungsverhalten der Testpersonen entwickelt. Um diese Theorie zu validieren, wurden auf deren Grundlage ein agentenbasiertes Simulationsmodell entworfen und Simulationsexperimente durchgeführt, die die Ergebnisse der realen Laborexperimente nachstellen sollten. Ein Vergleich der Daten, die aus den Laborexperimenten und den Simulationsexperimenten gewonnen wurden, zeigte, dass die Theorie von Caldas und Coelho über das Preisbildungsverhalten der Testpersonen durchaus eine gute Näherung der Realität darstellt.

Im Rahmen der vorliegenden Arbeit dient das Marktmodell als Fallstudie, mit deren Hilfe der Einsatz des Referenzmodells *PECS* im Bereich der experimentellen Ökonomie gezeigt wird. Das Marktmodell ist geprägt durch die Agenten, die die Produzenten des Laborexperimentes abbilden, sowie deren Strategien zur Preisbildung. Aufgrund der funktionalen Dekomposition von *PECS*-Agenten in eine Menge interagierender Subkomponenten entsteht in diesem Zusammenhang eine sehr klare und leicht erweiterbare Modellstruktur. Ebenso tragen die gewählten Konzepte zur Beschreibung der diskreten Modelldynamik dazu bei, dass die Verhaltensregeln der Agenten in leicht verständlicher Weise implementiert werden.

Wie die Implementierungsbeschreibung des Marktmodells belegt, eignet sich das Referenzmodell *PECS* im Zusammenspiel mit den in *Simplex3* verankerten, systemtheoretisch fundierten Konzepten zur Modellspezifikation in ausgezeichneter Weise, um agentenbasierte Simulationsmodelle im Anwendungsbereich der experimentellen Ökonomie aufzubauen.

9 FALLSTUDIE III: MODELLIERUNG VON GRUPPENPROZESSEN

Wie in Kapitel 2.1.5 bereits angedeutet wurde, wendet sich die gegenwärtige Sozialforschung in zunehmendem Maße der agentenbasierten Modellbildung und Simulation zu, um künstliche Sozialsysteme zu schaffen und deren Verhalten zu erforschen. In der Fallstudie *Modellierung von Gruppenprozessen* wird demonstriert, wie auf der Grundlage des Referenzmodells *PECS* ein derartiges künstliches Sozialsystem konstruiert werden kann. Als Untersuchungsgegenstand dient dazu die Formation, Entwicklung und Disintegration von Kleingruppen am Beispiel von Studenten einer virtuellen Universität.

Die folgenden Abschnitte geben einen Überblick über das grundsätzliche Modellkonzept, den Aufbau und die Implementierung des Modells auf der Grundlage des Referenzmodells *PECS*, sowie die vorhandenen Experimentiermöglichkeiten.

9.1 Modellbeschreibung

Das Modell *Lerngruppe* geht von einer Gruppe von 36 Studenten einer virtuellen Universität aus, die sich auf bevorstehende Prüfungen vorbereiten. Um einen möglichst großen Lernerfolg zu erzielen, bereiten die Studenten den erforderlichen Lernstoff sowohl in Einzel- als auch in Gruppenarbeit vor. Im Rahmen der Einzelarbeit verschafft sich jeder Student zunächst einmal einen Überblick über eine gewisse Teilmenge des Lernstoffs. Erzielt ein Student in der Einzelarbeit keine signifikanten Fortschritte mehr, so besteht für ihn die Möglichkeit, sich einer bestehenden Lerngruppe anzuschließen bzw. eine neue Lerngruppe zu gründen, falls zum betrachteten Zeitpunkt keine geeignete Lerngruppe existiert. Innerhalb der Lerngruppen diskutieren die Studenten den in der Einzellernphase erarbeiteten Lernstoff und erhalten dadurch ein vertieftes und erweitertes Verständnis. Doch auch die Gruppenarbeit stößt nach einer gewissen Zeit an ihre Grenzen, so daß die Studenten wieder aus den Lerngruppen austreten und sich erneut der Einzelarbeit zuwenden. Der zyklische Wechsel zwischen Einzelarbeit und Gruppenarbeit stellt eine grundlegende Annahme des vorliegenden Modells dar.

9.1.1 Die Individuen

Agenten finden in diesem Modell Verwendung, um die Studenten der virtuellen Universität abzubilden. Die Agenten verfügen als Stellvertreter der Stundenten im Modell über individuelle Persönlichkeitsmerkmale, Grundbedürfnisse und interne Zustände. Die internen Zustände sowie die daran gekoppelten Bedürfnisse ändern sich im Laufe der Zeit in Abhängigkeit der aktuellen Lernphase. Den Agenten steht zudem eine Menge an Aktionen zur Auswahl, die es ihnen ermöglichen, zu lernen, eine bestehende Lerngruppe auszuwählen und sich dieser anzuschließen, eine neue Lerngruppe zu gründen, sowie eine Lerngruppe wieder zu verlassen bzw. aufzulösen. Die Eigenschaften und Verhaltensweisen der Agenten werden in den folgenden Abschnitten im Detail vorgestellt.

9.1.1.1 Die Persönlichkeitsmerkmale der Individuen

Das Modell Lerngruppe umfaßt eine heterogene Population von 36 Individuen. Die einzelnen Individuen werden mit Hilfe von Agenten modelliert und unterscheiden sich durch individuelle Ausprägungen der beiden Persönlichkeitsmerkmale *Intelligenz* und *soziale Veranlagung*. In Anlehnung an Eigenschaftstheorien wie sie beispielsweise bei Allport (1937, 1966) oder Eysenck (1973, 1990) zu finden sind, sollen Persönlichkeitsmerkmale im vorliegenden Modell einzigartige psychologische Merkmale bzw. Traits eines Individuums bezeichnen, die sich auf das Verhalten auswirken und ihm in verschiedenen Situationen sowie zu verschiedenen Zeitpunkten Kohärenz verleihen.

Jedem Persönlichkeitsmerkmal ist eine kontinuierliche Skala von 80 bis 120 Merkmalseinheiten zugeordnet. Um die Ausprägungen der Persönlichkeitsmerkmale für ein bestimmtes Individuum festzulegen, wird auf beiden Skalen ein entsprechender Wert ausgewählt und dem Individuum fest zugeordnet. Die Auswahl erfolgt dabei zufällig nach einer Gauß-Verteilung. Auf diese Weise wird die Mehrzahl der Individuen in bezug auf die Ausprägungen beider Merkmale im mittleren Skalenbereich liegen. Die Persönlichkeitsmerkmale werden nach ihrer Festlegung zu Beginn als unveränderlich angenommen.

Intelligenz

Mit Intelligenz werden gemeinhin die mit dem Verstand verbundenen, geistigen Fähigkeiten bezeichnet (Häcker & Stapf, 1998). Eine elementare Leistung der Intelligenz besteht in der Auffassung und dem Begreifen von Sachverhalten.

Bezogen auf die Individuen im Modell *Lerngruppe* bedeutet Intelligenz die Fähigkeit, Lernstoff aufzunehmen und zu verarbeiten. Je höher die Intelligenz eines Individuums ist, um so schneller wird es in der Lage sein, eine gewisse Menge an Lernstoff zu verinnerlichen. Die Lernkurve zeigt bei intelligenteren Individuen

einen steileren Anstieg als bei weniger intelligenten Individuen (s. Abschnitt 9.1.1.3 – Kognitive Zustandsvariablen).

Soziale Veranlagung

Das Persönlichkeitsmerkmal *soziale Veranlagung* beschreibt die Soziabilität eines Individuums, d. h. seine Geselligkeit wie auch seine Fähigkeit, sich in soziale Verbände und Gruppen einzupassen (Häcker & Stapf, 1998).

Ein Individuum mit einer sehr ausgeprägten sozialen Veranlagung wird sich im vorliegenden Modellierungsansatz sehr schnell einsam fühlen, wenn es alleine arbeitet und keiner Gruppe angehört. Andererseits wird es im Rahmen einer Gruppe auch sehr schnell in der Lage sein, sein Bedürfnis nach sozialen Kontakten zu befriedigen und damit seine soziale Zufriedenheit zu erhöhen (s. Abschnitt 9.1.1.3 – Soziale Zustandsvariablen).

Die Abbildung 9.1 zeigt im Überblick die beiden Persönlichkeitsmerkmale der Individuen mit ihren zugehörigen Einheiten und Initialbelegungen.

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
Intelligenz	[IQ]	[80 .. 120] gaußverteilt	Intelligenz eines Individuums
SozAnlage	[SQ]	[80 .. 120] gaußverteilt	Soziale Veranlagung eines Individuums

Abbildung 9.1: Die Persönlichkeitsmerkmale eines Individuums

9.1.1.2 Die Grundbedürfnisse der Individuen

Neben einfachen Persönlichkeitsmerkmalen besitzen die Agenten zwei elementare Grundbedürfnisse, die ihr Verhalten im wesentlichen bestimmen. Es handelt sich dabei um die Bedürfnisse nach Wissen und nach Sozialkontakten. Die Agenten werden im Rahmen ihrer Verhaltenssteuerung versuchen, ihr Verhalten so einzurichten, daß beide Bedürfnisse so gut wie möglich befriedigt werden können.

Das Bedürfnis nach Wissen

Bei den betrachteten Individuen handelt es sich um Studenten einer virtuellen Universität, die sich in der Prüfungsvorbereitung befinden. Aus diesem Grund geht dieser Modellierungsansatz von der Annahme aus, daß bei den Studenten ein permanentes und auch sehr starkes Bedürfnis nach Wissen existiert. Dieses Bedürfnis führt dazu, daß die Studenten fortwährend lernen und dadurch ihr Wissen erhöhen. Ebenso bewirkt dieses Bedürfnis, daß die Studenten versuchen, so viel Wissen wie möglich anzusammeln. Stoßen die Studenten in einer Lernphase hin-

sichtlich des Lernerfolgs an Grenzen, so werden sie sich bemühen, die Lernphase zu wechseln und damit den Lernerfolg wieder zu erhöhen.

Aufgrund der Annahme, daß das Bedürfnis nach Wissen im gesamten Betrachtungszeitraum konstant auf einem hohen Niveau angesiedelt ist und zu einem fortwährenden Lernen der Studenten führt, wird dieses Bedürfnis nicht explizit im Modell repräsentiert. Es existiert also keine Zustandsvariable oder Konstante, die das Bedürfnis nach Wissen aufnimmt. Vielmehr steht dieses Bedürfnis implizit als Triebkraft im Hintergrund, die die Studenten zum Lernen bewegt.

Das Bedürfnis nach Sozialkontakten

Zum Bedürfnis nach Wissen kommt ein weiteres Bedürfnis hinzu, das sich auf das soziale Umfeld der Individuen bezieht. Es handelt sich dabei um das Bedürfnis nach Sozialkontakten. Dieses Bedürfnis bewirkt, daß sich die Individuen von Zeit zu Zeit aus der isolierten Position des Einzellernens lösen, sich Lerngruppen anschließen, dort mit anderen Individuen in Kontakt treten und dadurch ihre soziale Zufriedenheit erhöhen.

Im Gegensatz zum Bedürfnis nach Wissen verändert dieses Bedürfnis seine Stärke im Laufe der Zeit. Lernt ein Individuum allein, so nimmt sein Bedürfnis nach Sozialkontakten mit der Zeit zu. Erreicht dieses Bedürfnis eine gewisse Stärke und ist der Lernerfolg in der Einzellernphase entsprechend gering geworden, so wird das Individuum versuchen, sich einer Lerngruppe anzuschließen bzw. eine neue Lerngruppe zu gründen. In der Lerngruppe trägt die Interaktion mit anderen Individuen dazu bei, das Bedürfnis nach Sozialkontakten zu befriedigen.

Eine ausführliche Beschreibung der Dynamik für das Bedürfnis nach Sozialkontakten enthalten die Kapitel 9.1.1.4 und 9.1.1.5.

9.1.1.3 Die Zustandsvariablen der Agenten

Die Agenten, die im Modell die Individuen repräsentieren, sind mit einer Menge von Zustandsvariablen ausgestattet, die den internen kognitiven und sozialen Zustand der Individuen beschreiben. Auf die Betrachtung emotionaler und physischer Zustände der Studenten wird in diesem Modell aus Gründen der Einfachheit und Übersichtlichkeit verzichtet.

Der folgende Abschnitt gibt einen Überblick über die Modellgrößen, die den kognitiven und sozialen Zustand der Agenten aufnehmen. Der Schwerpunkt liegt dabei auf der Einführung und Beschreibung der grundlegenden Bedeutung dieser Modellgrößen. Auf eine Betrachtung der dynamischen Veränderungen während der Einzel- und Gruppenlernphase wird an dieser Stelle noch verzichtet. Eine entsprechende Dynamikbeschreibung enthalten die Abschnitte 9.1.1.4 und 9.1.1.5.

Kognitive Zustandsvariablen

Der kognitive Zustand eines Agenten im Modell *Lerngruppe* wird im wesentlichen durch zwei Modellgrößen beschrieben, die angeben, welche Menge an Wissen der Agent insgesamt sowie in der aktuellen Lernphase angesammelt hat.

Das Gesamtwissen. Die Modellgröße *Wissen* speichert die gesamte, im Laufe der Zeit angesammelte Menge an Wissen eines Agenten. Der Wert dieser Zustandsvariable ist zu Beginn mit 0 [WI] initialisiert – jeder Agent beginnt also gänzlich ohne Vorkenntnisse – und steigt mit der Lebensdauer eines Agenten kontinuierlich an. Das Gesamtwissen eines Agenten beeinflußt seine soziale Position, d. h. je mehr Wissen ein Agent anhäuft, um so weiter klettert er in der sozialen Rangfolge nach oben (s. Abschnitt 9.1.1.7).

Das Wissen der aktuellen Lernphase. Innerhalb einer Lernphase kann das Wissen eines Agenten maximal um eine Wissenseinheit ansteigen. Ein Agent kann dieser Annahme entsprechend während einer Lernphase etwa ein bestimmtes Themengebiet erarbeiten. Die aktuell in einer Lernphase angesammelte Menge an Wissen wird in der kognitiven Zustandsvariable *WisAkt* abgelegt. Zu Beginn einer neuen Lernphase wird diese Zustandsvariable mit einem sehr kleinen Anfangswert belegt, der eine gewisse Menge an benötigten Vorkenntnissen repräsentiert. Der Zuwachs des Wissens in der aktuellen Lernphase beeinflußt unmittelbar das Verhalten eines Agenten (s. Abschnitt 9.1.1.4 und 9.1.1.5).

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
Wissen	[WI]	[0.1 .. 0.5] uniform verteilt	Gesamtwissen eines Agenten
WisAkt	[WI]	[0.1 .. 0.5] uniform verteilt	In der aktuellen Lernphase angesammeltes Wissen eines Agenten

Abbildung 9.2: Die kognitiven Zustandsvariablen eines Agenten

Soziale Zustandsvariablen

Neben kognitiven Attributen besitzt ein Agent im Modell *Lerngruppe* auch soziale Eigenschaften. Hierzu zählen die *soziale Kompetenz*, die sich durch die Zusammenarbeit mit anderen Agenten in Lerngruppen im Laufe der Zeit erhöht, sowie die *soziale Zufriedenheit*, die aufgrund der Isolation während der Einzellernphase absinkt und in der Gruppenlernphase durch den Kontakt mit anderen Agenten wieder ansteigt. Umgekehrt proportional zur sozialen Zufriedenheit verläuft das Bedürfnis nach Sozialkontakten.

Die soziale Kompetenz. Soziale Kompetenz beschreibt die Fähigkeit einer Person, sich bei der Annäherung an andere oder in der Interaktion mit anderen erfolgreich zu verhalten (Zimbardo & Gerrig, 1999). Das Modell *Lerngruppe* geht von der Grundannahme aus, daß ein Individuum durch häufige Interaktion mit anderen Individuen im Laufe der Zeit seine soziale Kompetenz steigern kann und daß der Lernfortschritt innerhalb einer Lerngruppe wesentlich von der sozialen Kompetenz ihrer Gruppenmitglieder abhängt. Je höher die soziale Kompetenz der Mitglieder einer Lerngruppe ist, um so effektiver können die Lerninhalte in der Lerngruppe diskutiert werden und um so größer ist der Lernfortschritt.

Die soziale Kompetenz eines Individuums wird im Agenten mit Hilfe der Zustandsvariable *SozKompetenz* modelliert. Zu Beginn wird jedem Agenten ein zufälliger Startwert für die Zustandsvariable *SozKompetenz* zugewiesen, der uniform verteilt aus dem Intervall [0.1 .. 0.3] [SK] gezogen wird. Die soziale Kompetenz eines Agenten wirkt sich nur in der Gruppenlernphase aus und kann auch nur dort ihren Wert verändern.

Die soziale Zufriedenheit. Die soziale Zufriedenheit verläuft umgekehrt proportional zum Bedürfnis nach Sozialkontakten und beschreibt, wie zufrieden ein Individuum mit der gegenwärtigen Situation in bezug auf das soziale Umfeld ist. Lernt ein Student im Modell *Lerngruppe* für eine Weile alleine, so sinkt seine soziale Zufriedenheit ab, d. h. er sehnt sich im Laufe der Zeit immer mehr nach Gesellschaft mit anderen Personen. Andererseits steigt seine soziale Zufriedenheit auch wieder an, sobald er sich innerhalb einer Lerngruppe befindet und sich mit anderen Studenten austauschen kann.

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
SozKompetenz	[SK]	[0.1 .. 0.3] uniform verteilt	Sozialkompetenz eines Agenten
SozAkt	[SoZ]	[0.1 .. 0.9] uniform verteilt	Soziale Zufriedenheit eines Agenten
SozBed	[SoZ]	-	Bedürfnis nach Sozialkontakten eines Agenten
QualitaetI	[IndQ]	-	Soziale Position eines Agenten innerhalb der Gesellschaft

Abbildung 9.3: Die sozialen Zustandsvariablen eines Agenten

Die soziale Zufriedenheit eines Studenten wird im Modell *Lerngruppe* mit Hilfe der Zustandsvariable *SozAkt* des zugehörigen Agenten abgebildet. Der Initialwert für diese Zustandsvariable wird uniform verteilt aus dem Intervall [0.1 .. 0.9] [SoZ] gezogen. Auf diese Weise startet jeder Agent mit einem indivi-

duellen Wert für die soziale Zufriedenheit. Dadurch wird erreicht, daß die Agenten zu verschiedenen Zeitpunkten ihre Einzellernphasen verlassen und Lerngruppen beitreten bzw. diese zu gründen versuchen.

Das Bedürfnis nach Sozialkontakten. Das Bedürfnis nach Sozialkontakten gibt an, mit welcher Dringlichkeit sich ein Student nach Sozialkontakten sehnt (s. Abschnitt 9.1.1.2). Dieses Bedürfnis steht in direktem Zusammenhang mit der sozialen Zufriedenheit eines Studenten. Die abhängige Variable *SozBed* nimmt das Bedürfnis nach Sozialkontakten eines Studenten auf und wird unmittelbar als Funktion der sozialen Zufriedenheit berechnet (s. Abschnitt 9.1.1.4 und 9.1.1.5).

Die soziale Position eines Individuums. Die soziale Position eines Individuums soll im vorliegenden Modell die Stellung eines Individuums innerhalb des Gesamtsystems beschreiben. Dabei wird angenommen, daß die soziale Position eines Individuums unmittelbar in Beziehung zu dem angesammelten Wissen sowie der sozialen Kompetenz steht. Je mehr Wissen und Sozialkompetenz ein Individuum erworben hat, um so höher wird seine soziale Position im Vergleich zu andern Individuen sein. Die soziale Position dient also im wesentlichen als Vergleichsgröße zwischen den einzelnen Individuen, um zu verfolgen, wie sich die Individuen im Laufe der Zeit und im Verhältnis zueinander hinsichtlich ihrer maßgeblichen Eigenschaften entwickeln. Im Modell wird die soziale Position eines Individuums auf Grundlage der abhängigen Variable *QualitaetI* gebildet, die das Gesamtwissens sowie die soziale Kompetenz eines Individuums im Vergleich zum jeweils besten Individuum der Gesellschaft betrachtet.

9.1.1.4 Die Veränderung der Zustandsvariablen in der Einzellernphase

Während der Einzellernphase erarbeitet ein Student ein bestimmtes Stoffgebiet in Individualarbeit. Sein Wissen nimmt dadurch zu. Allerdings sinkt seine soziale Zufriedenheit kontinuierlich ab, da kein Kontakt zu anderen Studenten besteht.

Die Dynamik der kognitiven Zustandsvariablen in der Einzellernphase

Die Zunahme des Wissens in der Einzellernphase wirkt sich in einem Agenten nicht nur auf die Modellgröße *WisAkt* aus, die das in einer Lernphase angesammelte Wissen bezeichnet, sondern auch auf das Gesamtwissen des Agenten, das in dessen Modellgröße *Wissen* abgelegt ist.

Der Anstieg des Wissens in der aktuellen Lernphase wird auf der Grundlage eines logistischen Wachstumsprozesses modelliert. Zu Beginn einer Lernphase fällt der Wissenszuwachs zunächst noch relativ gering aus. Im Laufe der Zeit steigt der Wissenszuwachs allerdings immer mehr an bis schließlich ein Wende-

punkt erreicht wird, ab dem der Wissenszuwachs zum Ende der Lernphase hin wiederum abnimmt. Die Differentialgleichung, die für die Modellierung dieses Sachverhaltes in den Agenten Verwendung findet, ist in Gleichung Gl. 9.1 dargestellt.

$$WisAkt'(t) := a * WisKap(t) * WisNormal * Intelligenz / 100 * WisAkt(t) \quad \text{(Gl. 9.1)}$$

Dabei gilt:

$$WisKap(t) := (WisAktMax - WisAkt(t)) / WisAktMax \quad \text{(Gl. 9.2)}$$

Wie die Gleichung Gl. 9.1 zeigt, hängt die Veränderung des aktuellen Wissens *WisAkt´ (t)* eines Agenten gemäß des logistischen Wachstums nicht nur von dem vorliegenden Wert des aktuellen Wissens *WisAkt (t)* ab, sondern zusätzlich von einer Reihe weiterer Einflußgrößen.

Der Faktor *WisKap (t)* bewirkt den charakteristischen Verlauf der logistischen Wachstumskurve. Er nimmt zu Beginn einer Lernphase aufgrund der noch sehr kleinen Werte für die Zustandsvariable *WisAkt (t)* relativ große Werte an und drängt dadurch zunächst zu einem starken Wachstum des aktuellen Wissens (s. Gl. 9.2). Je stärker sich das aktuelle Wissen im Laufe einer Lernphase an die Obergrenze *WisAktMax* annähert, um so geringer fällt aufgrund des Faktors *WisKap (t)* der Wissenszuwachs aus.

Der Faktor *a* in Gleichung Gl. 9.1 dient im wesentlichen der Anpassung der Einheiten, so daß das Gesamtergebnis für die Lerngeschwindigkeit eines Individuums die gewünschte Einheit *[WI / h]* aufweist.

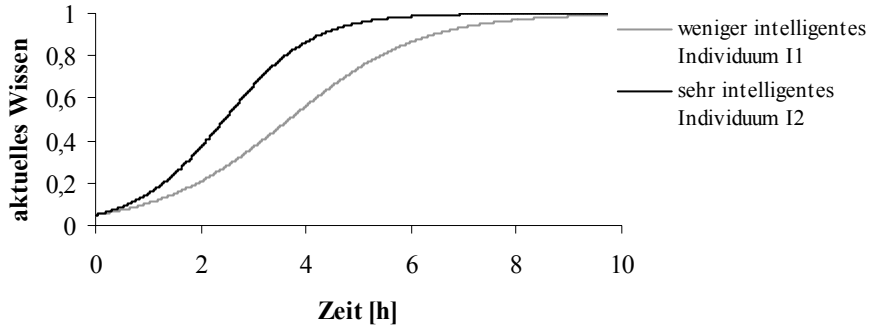

Abbildung 9.4: Der Wissenszuwachs zweier unterschiedlich intelligenter Individuen in der Einzellernphase

Der Faktor *WisNormal* gibt eine „normale“ Wachstumsgeschwindigkeit des Wissens vor und ermöglicht dadurch eine Kalibrierung der Phasendauer. Zusätzlich wird der Wissenszuwachs eines Individuums durch dessen Intelligenz

beeinflußt, die durch die Konstante *Intelligenz* repräsentiert wird. Je höher die Intelligenz eines Individuums ist, um so schneller kann dessen Wissen ansteigen.

Die Abbildung 9.4 zeigt den Anstieg des aktuellen Wissens in der Einzellernphase für zwei Individuen I_1 und I_2 mit unterschiedlicher Intelligenz. Das Individuum I_1 verfügt über eine relativ geringe Intelligenz von 80 [IQ], das Individuum I_2 ist mit einer hohen Intelligenz von 120 [IQ] ausgestattet. Die hohe Intelligenz von Individuum I_2 bewirkt einen sehr steilen Anstieg des aktuellen Wissens, so daß dieses Individuum in der Lage ist, das in einer Lernphase anstehende Lehrmaterial sehr schnell zu erarbeiten. Das Individuum I_1 benötigt für dieselbe Menge an Lehrmaterial bzw. denselben Wissenszuwachs in Höhe von 1 [WI] deutlich mehr Zeit als Individuum I_2, da dessen Wissen aufgrund der geringeren Intelligenz wesentlich langsamer ansteigt.

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
Wissen	[WI]	[0.1 .. 0.5] uniform verteilt	Gesamtwissen eines Agenten
WisAkt	[WI]	0.05 (Initialwert zu Beginn jeder neuen Lernphase)	In der aktuellen Lernphase angesammeltes Wissen eines Agenten
WisAktMax	[WI]	1	Obere Schranke für die Menge an Wissen, das in einer Lernphase erworben werden kann
WisKap	[]	-	Begrenzungsfaktor für das Wachstum des Wissens in Abhängigkeit der oberen Schranke *WisAktMax*
WisNormal	[WI/h]	0.5	„Normale“ Zunahmerate des Wissens
Intelligenz	[IQ]	[80 .. 120] gaußverteilt	Intelligenz eines Individuums
a	[1/(WI*IQ)]	1	Konstante für den Ausgleich der Einheiten

Abbildung 9.5: Die Modellgrößen zur Berechnung der Wissenszunahme in der Einzellernphase

Ebenso wie das aktuelle Wissen *WisAkt* steigt auch das Gesamtwissen eines Agenten in der Einzellernphase an. Die Berechnung des Gesamtwissens *Wissen* folgt dabei denselben Berechnungsvorschriften (s. Gl. 9.1 und Gl. 9.2), die auch für die Bestimmung des aktuellen Wissens *WisAkt* zum Einsatz kommen.

Die Abbildung 9.5 zeigt noch einmal im Überblick die Modellgrößen, die an der Berechnung der Wissenszunahme eines Agenten in der Einzellernphase beteiligt sind.

Die Dynamik der sozialen Zustandsvariablen in der Einzellernphase

In der Einzellernphase sinkt die soziale Zufriedenheit eines Agenten aufgrund der fehlenden Sozialkontakte ab. In gleichem Maße steigt das Bedürfnis nach Sozialkontakten in dieser Phase an. Die Sozialkompetenz, die einen Agenten hinsichtlich seiner Interaktionsfähigkeit mit anderen Agenten auszeichnet, bleibt in der Einzellernphase unverändert.

Die Abnahme der sozialen Zufriedenheit, die in einem Agenten mit Hilfe der Modellgröße *SozAkt* repräsentiert wird, wird auf der Grundlage eines logistischen Wachstumsprozesses berechnet. Zu Beginn einer Einzellernphase sinkt die soziale Zufriedenheit eines Agenten zunächst nur sehr geringfügig ab. Im Laufe der Zeit nimmt die Abnahmerate der sozialen Zufriedenheit allerdings immer stärker zu, bis schließlich ein Wendepunkt erreicht wird, ab dem sich die Abnahmerate der sozialen Zufriedenheit zum Ende der Einzellernphase hin wieder allmählich verringert. Die Differentialgleichung, die für die Modellierung dieses Sachverhaltes Verwendung findet, ist in Gleichung Gl. 9.3 dargestellt.

$$SozAkt'(t) := -b * SozKap(t) * SozNormal * SozAnlage / 100 * SozAkt(t) \quad \text{(Gl. 9.3)}$$

Dabei gilt:

$$SozKap(t) := (SozAktMax - SozAkt(t)) / SozAktMax \quad \text{(Gl. 9.4)}$$

Wie die Gleichung Gl. 9.3 zeigt, hängt die Veränderung der sozialen Zufriedenheit von einer Reihe von Einflußfaktoren ab.

Der Faktor *SozKap (t)* bewirkt den charakteristischen Verlauf der logistischen Wachstumskurve. Er nimmt zu Beginn einer Lernphase aufgrund der noch sehr geringen Differenz zwischen *SozAktMax* und *SozAkt (t)* relativ kleine Werte an und bewirkt dadurch ein geringes Absinken der sozialen Zufriedenheit eines Agenten. Je weiter sich die soziale Zufriedenheit *SozAkt (t)* allerdings vom Maximalwert *SozAktMax* entfernt, um so stärker nimmt die soziale Zufriedenheit eines Agenten ab. Am Ende einer Einzellernphase verändert sich die soziale Zufriedenheit eines Agenten jedoch nur noch geringfügig, da sie bereits auf einem sehr niedrigen Niveau angekommen ist.

Der Faktor *SozNormal* gibt die „normale“ Abnahmegeschwindigkeit der sozialen Zufriedenheit vor und dient der Kalibrierung der Phasendauer. Darüber hinaus hängt die Abnahme der sozialen Zufriedenheit von der sozialen Veranlagung eines Individuums ab. Je höher die soziale Veranlagung eines Individuums ist, d. h. je höher der Wert für die Konstante *SozAnlage* gewählt wird, um so schneller

nimmt die soziale Zufriedenheit eines Individuums in der Einzellernphase ab. Der Faktor *b* in Gleichung Gl. 9.3 dient im wesentlichen der Anpassung der Einheiten, so daß das Gesamtergebnis für die Abnahmegeschwindigkeit der sozialen Zufriedenheit eines Individuums die gewünschte Einheit *[SoZ / h]* aufweist. Die Konstante *SozAktMax* in Gleichung Gl. 9.4 gibt den Maximalwert für die soziale Zufriedenheit eines Individuums vor.

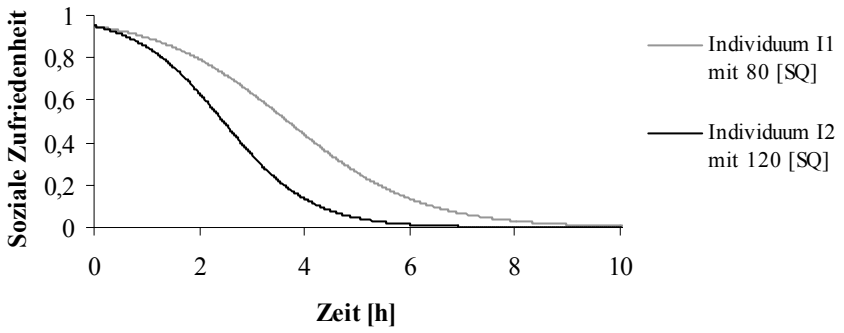

Abbildung 9.6: Die Veränderung der sozialen Zufriedenheit zweier unterschiedlich sozial veranlagter Individuen in der Einzellernphase

Die Abbildung 9.6 zeigt die Abnahme der sozialen Zufriedenheit in der Einzellernphase für zwei Individuen I_1 und I_2 mit unterschiedlicher sozialer Veranlagung. Das Individuum I_1 verfügt über eine relativ gering ausgeprägte soziale Veranlagung von 80 [SQ], das Individuum I_2 ist mit einer hoch ausgeprägten sozialen Veranlagung von 120 [SQ] ausgestattet. Die sehr ausgeprägte soziale Veranlagung von Individuum I_2 bewirkt einen sehr steilen Abfall der sozialen Zufriedenheit, so daß sich dieses Individuum in der dargestellten Lernphase sehr schnell einsam fühlt und dementsprechend auch versuchen wird, einer Lerngruppe beizutreten. Bei Individuum I_1 vollzieht sich der Abfall der sozialen Zufriedenheit wesentlich langsamer, so daß dieses Individuum deutlich länger in der Einzellernphase verbleiben wird, ohne sich einsam zu fühlen. Zu Beginn starten beide Individuen mit einem Wert für die soziale Zufriedenheit, der nahe am Maximalwert *SozAktMax* liegt.

Das Bedürfnis nach Sozialkontakten, das in einem Individuum in der Einzellernphase entsteht, läßt sich mit Hilfe einer abhängigen Variable darstellen, deren Wert unmittelbar mit dem aktuellen Wert für die soziale Zufriedenheit des Individuums korreliert. Zur Berechnung des aktuellen Wertes für das Bedürfnis nach Sozialkontakten findet die algebraische Gleichung Gl. 9.5 Verwendung.

$$SozBed(t) := SozAktMax - SozAkt(t) \qquad \text{(Gl. 9.5)}$$

Die Abbildung 9.7 zeigt den zeitlichen Verlauf für das Bedürfnis nach Sozialkontakten am Beispiel der beiden, bereits oben verwendeten Individuen I_1 und I_2. Bei Individuum I_2, das mit einer sehr stark ausgeprägten sozialen Veranlagung

ausgestattet ist, steigt der aktuelle Wert für das Bedürfnis nach Sozialkontakten in der Einzellernphase wesentlich schneller an als bei Individuum I_1, das eine deutlich schwächere soziale Veranlagung aufweist.

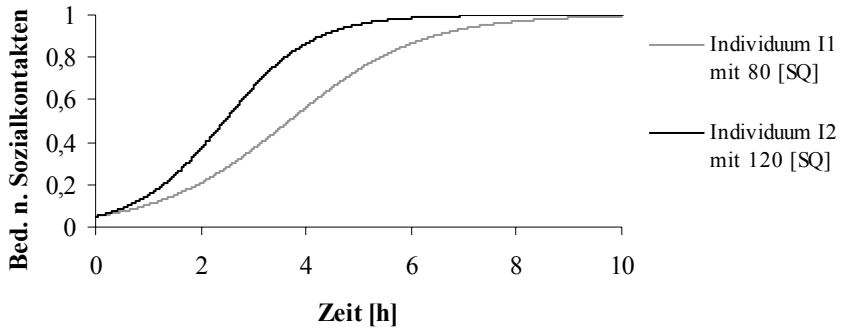

Abbildung 9.7: Der zeitliche Verlauf für das Bedürfnis nach Sozialkontakten bei zwei Individuen mit unterschiedlich stark ausgeprägten sozialen Veranlagungen

Die Sozialkompetenz eines Individuums bleibt während der Einzellernphase unverändert. Ein Individuum ist nur durch die Interaktion mit anderen Individuen in der Lage, seine Sozialkompetenz zu steigern.

Die Abbildung 9.8 zeigt im Überblick alle Modellgrößen, die an der Berechnung des sozialen Zustands eines Individuums in der Einzellernphase beteiligt sind.

9.1.1.5 Die Veränderung der Zustandsvariablen in der Gruppenlernphase

In der Gruppenlernphase arbeiten Studenten zusammen, um den Lehrstoff zu diskutieren und im gemeinsamen Gespräch zu vertiefen. In dieser Phase bauen die Studenten nicht nur ihr Wissen aus, sondern befriedigen gleichzeitig auch ihr Bedürfnis nach Sozialkontakten. Durch die Zusammenarbeit mit anderen Individuen erhöht sich darüber hinaus die soziale Kompetenz der Studenten, die ihnen in späteren Gruppenlernphasen um so mehr zu Gute kommt.

Der vorliegende Modellierungsansatz geht von der grundlegenden Annahme aus, dass der Erfolg der Zusammenarbeit von Individuen in einer Gruppe maßgeblich durch deren Qualität bestimmt wird. Je höher die Qualität einer Lerngruppe ist, um so schneller werden die Individuen innerhalb der Gruppe in der Lage sein, ihr Wissen auszubauen. Ebenso werden die Individuen um so schneller ihre soziale Zufriedenheit erhöhen können.

Der Begriff Qualität bezeichnet dabei ein relativ allgemein gehaltenes Konzept, das ein Maß für die Leistungsfähigkeit einer Gruppe darstellt. Die Leistungsfähigkeit einer Gruppe wird dabei um so höher angenommen, je größer das innerhalb der Gruppe verfügbare Wissen ist, je größer die soziale Kompetenz der einzelnen Gruppenmitglieder ist und je besser die Gruppengröße an die Anforderun-

gen der jeweiligen Ausgangssituation angepasst ist. Eine weiterführende Darstellung des Qualitätskonzeptes, das im Modell Lerngruppe für studentische Arbeitsgruppen zur Anwendung kommt, findet sich in Abschnitt 9.1.2.1.

Bezeichner	Einheit	Initialwert	Erläuterung
SozAkt	[SoZ]	[0.1 .. 0.9] uniform verteilt	Soziale Zufriedenheit eines Agenten
SozAktMax	[SoZ]	1	Maximalwert für die soziale Zufriedenheit eines Agenten
SozKap	[SoZ]	-	Begrenzungsfaktor für den Abfall der sozialen Zufriedenheit in Abhängigkeit der oberen Schranke *SozAktMax*
SozNormal	[SoZ/h]	0.5	„Normale“ Abnahmerate der sozialen Zufriedenheit
SozAnlage	[SQ]	[80 .. 120] gaußverteilt	Soziale Veranlagung eines Individuums
SozBed	[SoZ]	-	Bedürfnis nach Sozialkontakten eines Agenten
SozKompetenz	[SK]	[0.1 .. 0.3] uniform verteilt	Sozialkompetenz eines Agenten
b	[1/(SoZ*SQ)]	1	Konstante für den Ausgleich der Einheiten

Abbildung 9.8: Die Modellgrößen zur Berechnung des sozialen Zustands in der Einzellernphase

Die Dynamik der kognitiven Zustandsvariablen in der Gruppenlernphase

Auch in der Gruppenlernphase liegt die Hauptintention der Studenten darin, ihr Wissen zu vergrößern. Diese Intention kann um so besser verwirklicht werden, je höher die allgemeine Qualität der Lerngruppe angesiedelt ist, in die ein Student als Mitglied aufgenommen wurde.

Der vorliegende Modellierungsansatz geht von der Annahme aus, daß ein Student in der Gruppenlernphase ebenso wie in der Einzellernphase je Lernzyklus eine Wissenseinheit erwerben kann. Der Anstieg des aktuellen Wissens *WisAkt* in der Gruppenlernphase wird analog zur Einzellernphase mit Hilfe eines logistischen Wachstumsprozesses abgebildet. Neben den Parametern *WisAktMax*, *WisNormal* und *Intelligenz* (s. Abschnitt 9.1.1.4), die in der Einzellernphase den Anstieg des Wissens maßgeblich bestimmen, kommt in der Gruppenlernphase der Parameter *QualitaetG* neu hinzu, der die allgemeine Qualität der Gruppe bezeich-

net. Die Zunahme des aktuellen Wissens in einer Gruppenlernphase wird mit Hilfe der folgenden Gleichung modelliert:

$$WisAkt'(t) := a * WisKap(t) * WisNormal * Intelligenz / 100 * QualitaetG(t) * WisAkt(t) \quad \text{(Gl. 9.6)}$$

Hierbei gilt ebenso wie im Falle der Gleichung Gl. 9.1:

$$WisKap(t) := (WisAktMax - WisAkt(t)) / WisAktMax \quad \text{(Gl. 9.7)}$$

Ebenso wie das aktuelle Wissen *WisAkt* steigt auch das Gesamtwissen eines Agenten in der Gruppenlernphase an. Die Berechnung des Gesamtwissens *Wissen* erfolgt dabei nach den Gleichungen Gl. 9.6 und Gl. 9.7.

Die Abbildung 9.9 zeigt den Anstieg des aktuellen Wissens in der Gruppenlernphase für zwei Individuen I_1 und I_2, die zwar sehr ähnliche Persönlichkeitsmerkmale aufweisen und damit alleine vergleichbar schnell lernen würden, aber in zwei unterschiedlich guten Gruppen agieren und damit deutlich in ihrer Lerngeschwindigkeit voneinander abweichen. Das Individuum I_2 steigt in der unten dargestellten Zeitspanne erst ein wenig später in die Gruppenlernphase ein als Individuum I_1, arbeitet aber auch in einer deutlich schlechteren Gruppe und benötigt daher für den Erwerb einer ähnlich großen Menge an Wissen einen wesentlich längeren Zeitraum als das Individuum I_1, das während des Betrachtungszeitraums in eine deutlich bessere Gruppe integriert ist.

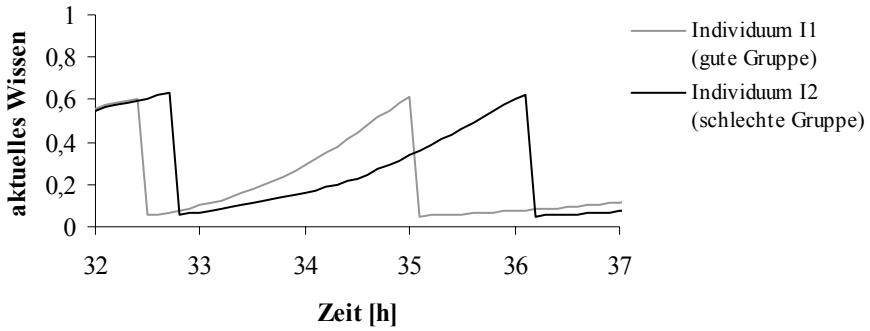

Abbildung 9.9: Der Wissenszuwachs zweier ähnlich veranlagter Individuen in unterschiedlich leistungsfähigen Gruppen

Das starke Absinken der beiden Kurven für Individuum I_1 und I_2 bezeichnet dabei jeweils das Ende einer Lernphase. Am Übergang von einer Lernphase in die nächste wird das aktuelle Wissen eines Agenten wieder auf einen relativ niedrigen Startwert zurückgesetzt. Das Gesamtwissen eines Agenten steigt hingegen fortwährend an.

Die Abbildung 9.10 zeigt noch einmal im Überblick alle im Vergleich zur Einzellernphase (s. Abbildung 9.5) veränderten oder neu hinzugekommenen Mo-

dellgrößen, die an der Berechnung des aktuellen Wissens bzw. Gesamtwissens eines Agenten in der Gruppenlernphase beteiligt sind:

Bezeichner	Einheit	Initialwert	Erläuterung
QualitaetG	[GQ]	-	Qualität einer Lerngruppe
a	[1/(WI*IQ*GQ)]	1	Konstante für den Ausgleich der Einheiten

Abbildung 9.10: Die Modellgrößen zur Berechnung der Wissenszunahme in der Einzellernphase

Die Dynamik der sozialen Zustandsvariablen in der Gruppenlernphase

In der Gruppenlernphase wird die soziale Zufriedenheit der Studenten aufgrund der Interaktion mit anderen Individuen regeneriert. In vergleichbarem Maße sinkt deren Bedürfnis nach Sozialkontakten in dieser Phase ab. Darüber hinaus wird die Sozialkompetenz der Studenten, die in der Einzellernphase keine Rolle spielt, in dieser Phase durch die Gruppenarbeit gefördert.

Auch der Anstieg der sozialen Zufriedenheit wird mit Hilfe einer logistischen Wachstumskurve beschrieben. Im Gegensatz zur Gleichung, die in der Einzellernphase zur Anwendung kommt, ist die Gleichung zur Berechnung der sozialen Zufriedenheit in der Gruppenlernphase mit einem positiven Vorzeichen versehen, so dass in diesem Fall tatsächlich ein positiver Anstieg der sozialen Zufriedenheit erreicht wird. Ebenso spielt wie schon bei der Berechnung des aktuellen Wissens auch hier die Qualität der Gruppe, in der ein Student integriert ist, eine wesentliche Rolle für die Veränderung der sozialen Zufriedenheit. Zur Bestimmung der sozialen Zufriedenheit eines Individuums kommt in der Gruppenlernphase die folgende Gleichung zum Einsatz:

$$SozAkt'(t) := b * SozKap(t) * SozNormal * SozAnlage / 100 * QualitaetG(t) * SozAkt(t) \quad \text{(Gl. 9.8)}$$

Ebenso wie in der Einzellernphase gilt auch hier:

$$SozKap(t) := (SozAktMax - SozAkt(t)) / SozAktMax \quad \text{(Gl. 9.9)}$$

Es ist offensichtlich, daß bei hoher Gruppenqualität – ausgedrückt durch einen großen Wert des Parameters *QualitaetG (t)* in Gleichung Gl. 9.8 – die soziale Zufriedenheit eines Individuums in einer Gruppenlernphase entsprechend schneller ansteigt als bei einer niedrigen Gruppenqualität.

Die Berechnung der Stärke für das Bedürfnis nach Sozialkontakten erfolgt in der Gruppenlernphase nach derselben Vorschrift wie in der Einzellernphase, d. h. es gilt:

$$SozBed\ (t) := SozAktMax - SozAkt\ (t) \qquad \text{(Gl. 9.10)}$$

Da in der Gruppenlernphase die soziale Zufriedenheit *SozAkt (t)* eines Individuums zunimmt, nimmt die Stärke des Bedürfnisses nach Sozialkontakten während dieser Zeit immer mehr ab.

Die Abbildung 9.11 zeigt den Anstieg der sozialen Zufriedenheit in der Gruppenlernphase für zwei Individuen I_1 und I_2, die mit ähnlichen Persönlichkeitsmerkmalen ausgestattet, aber in zwei unterschiedlich leistungsfähige Gruppen integriert sind. Das Individuum I_1 arbeitet in einer Gruppe mit relativ hoher Qualität, so daß es in der Lage ist, seine soziale Zufriedenheit schnell zu steigern. Bei Individuum I_2 bedarf es wesentlich mehr Zeit, einen entsprechenden Anstieg der sozialen Zufriedenheit zu realisieren, da dessen Gruppe im Betrachtungszeitraum weniger leistungsfähig ist. Die negative Steigung am rechten Ende der beiden in Abbildung 9.11 dargestellten Kurven deutet auf einen Wechsel von der Gruppenlernphase in die Einzellernphase hin. Bei Eintritt in die Einzellernphase beginnt die während der Gruppenlernphase regenerierte soziale Zufriedenheit wieder fortwährend abzunehmen.

Abbildung 9.11: Die Veränderung der sozialen Zufriedenheit zweier ähnlich veranlagter Individuen in unterschiedlich leistungsfähigen Gruppen

In Abbildung 9.12. ist der zeitliche Verlauf für die Stärke des Bedürfnisses nach Sozialkontakten am Beispiel der beiden Individuen I_1 und I_2 dargestellt. Aufgrund der Kurvenverläufe lässt sich sehr deutlich die Abhängigkeit der Stärke für das Bedürfnis nach Sozialkontakten von der sozialen Zufriedenheit eines Individuums erkennen. Ebenso wie für Individuum I_1 die soziale Zufriedenheit in Abbildung 9.11 schnell ansteigt, sinkt in Abbildung 9.12 in entsprechender Weise die Stärke für das Bedürfnis nach Sozialkontakten ab. Auch hier dauert es für Individuum I_2 entsprechend länger bis eine Bedürfnisbefriedigung eintritt. Der erneute Anstieg der Bedürfnisstärken korreliert bei beiden Individuen mit dem Wiedereintritt in die Einzellernphase.

Neben der Festigung und Erweiterung ihres Wissens und der Befriedigung sozialer Bedürfnisse erwerben die Studenten durch die Interaktion und Zusammenarbeit mit anderen Studenten in der Gruppenlernphase ein immer höheres

Maß an sozialer Kompetenz. Die Studenten werden also mit einer wiederholten Teilnahme an Gruppenlernphasen aufgrund der bereits gesammelten Erfahrungen im Umgang miteinander in stetig wachsendem Maße auch zur Leistungsfähigkeit einer Gruppe beitragen können. Dieser Einfluß auf die Gruppenleistung ist im vorliegenden Zusammenhang in der Weise abgebildet, daß die soziale Kompetenz der Gruppenmitglieder als wesentlicher Parameter in die Berechnung der Gruppenqualität eingeht (s. Abschnitt 9.1.2.1). Die Gruppenqualität wirkt ihrerseits während der Gruppenlernphasen auf die Entwicklung des Wissens und der sozialen Zufriedenheit ihrer Mitglieder zurück.

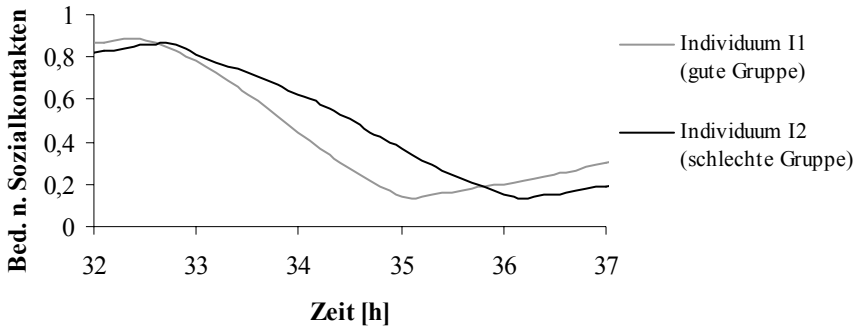

Abbildung 9.12: Die Veränderung der Stärke für das Bedürfnis nach Sozialkontakten zweier ähnlich veranlagter Individuen in unterschiedlich leistungsfähigen Gruppen

Der Anstieg der sozialen Kompetenz eines Individuums während einer Gruppenlernphase wird mit Hilfe der folgenden Gleichung dargestellt:

$$SozKompetenz'(t) := d * KompNormal * SozAnlage / 100 \qquad \text{(Gl. 9.11)}$$

Die soziale Kompetenz eines Individuums nimmt also linear während der Gruppenlernphasen zu. Der Faktor *KompNormal* gibt dabei eine normierte Wachstumsgeschwindigkeit der sozialen Kompetenz vor. Darüber hinaus wird der Anstieg der sozialen Kompetenz durch die soziale Veranlagung eines Individuums beeinflusst. Individuen, die ein besseres Gespür für soziale Situationen mitbringen, werden dementsprechend schneller in der Lage sein, ihre Sozialkompetenz auszubauen und in späteren Gruppensituationen gewinnbringend einzusetzen.

Die Abbildung 9.13 zeigt im Überblick alle im Vergleich zur Einzellernphase veränderten oder neu hinzugekommenen Modellgrößen, die an der Berechnung des sozialen Zustands eines Individuums in der Gruppenlernphase beteiligt sind.

Bezeichner	Einheit	Initialwert	Erläuterung
QualitaetG	[GQ]	-	Qualität einer Lerngruppe
b	[1/(SoZ*SQ*GQ)]	1	Konstante für den Ausgleich der Einheiten
KompNormal	[SK/h]	0.4	„Normale“ Wachstumsrate der sozialen Kompetenz
d	[1/SQ]	1	Konstante für den Ausgleich der Einheiten

Abbildung 9.13: Die Modellgrößen zur Berechnung des sozialen Zustands in der Gruppenlernphase

9.1.1.6 Die Bestimmung der sozialen Position eines Individuums

Das Modell *Lerngruppe* geht von einer leistungsorientierten Gesellschaft aus, in der die soziale Position eines Individuums maßgeblich durch erworbene Fähigkeiten bestimmt wird. Die bis zum betrachteten Zeitpunkt erworbenen Fähigkeiten eines Individuums werden mit Hilfe der Modellgröße *QualitaetI* abgebildet. Für die Bestimmung der „Qualität“ eines Individuums werden sowohl das im Laufe der Zeit angesammelte Wissen, als auch die erworbene soziale Kompetenz herangezogen. Beide Fähigkeiten wirken dabei unabhängig voneinander auf die Qualität eines Individuums, d. h. für die Steigerung der Qualität eines Individuums reicht es bereits aus, wenn eine der beiden Fähigkeiten ausgebaut wird. In die Gesamtberechnung der Qualität eines Individuums gehen beide Eigenschaften bzw. Fähigkeiten mit einem veränderbaren Gewichtungsfaktor *i1* bzw. *i2* ein. Auf diese Weise kann der Stellenwert mit berücksichtigt werden, den eine Gesellschaft einer ganz bestimmten Fähigkeit beimisst. Anhand ihrer Qualität können die Individuen in einem nachfolgenden Berechnungsschritt in eine Reihenfolge gebracht werden. Die Position innerhalb dieser Reihenfolge wird dabei gleichgesetzt mit der sozialen Position eines Individuums.

Die Berechnung der Qualität eines Individuums erfolgt nach Gleichung Gl. 9.12:

$$QualitaetI(t) := i * (i1 * RelWissenI(t) + i2 * RelSozKompetenzI(t)) \quad \text{(Gl. 9.12)}$$

$$RelWissenI(t) := Wissen(t) / WissenMax(t) \quad \text{(Gl. 9.13)}$$

$$RelSozKompetenzI(t) := SozKompetenz(t) / SozKompetenzMax(t) \quad \text{(Gl. 9.14)}$$

$$i := 1 / (i1 + i2) \quad \text{(Gl. 9.15)}$$

RelWissenI (t) bezeichnet dabei das Verhältnis der aktuellen Menge an Wissen eines Individuums im Vergleich zur größten Menge an Wissen, die ein Individuum in der Gesellschaft zum betrachteten Zeitpunkt erworben hat. In analoger Weise stellt die Modellgröße *RelSozKompetenzI (t)* die soziale Kompetenz eines Individuums in Beziehung zu der sozialen Kompetenz des hinsichtlich dieser Eigenschaft besten Individuums der Gesellschaft. Der Faktor *i* bewirkt eine Normierung der Werte für die Qualität der Individuen auf das Intervall [0..1].

Die Abbildung 9.14 zeigt die für die Berechnung der Qualität eines Individuums neu hinzugekommenen Modellgrößen.

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
QualitaetI	[IndQ]	-	Qualität eines Individuums
RelWissenI	-	-	Wissen eines Individuums im Vergleich zum besten Individuum
WissenMax	[WI]	-	Wissen des besten Individuums in der Gesellschaft
RelSozKompetenzI	-	-	soziale Kompetenz eines Individuums im Vergleich zum besten Individuum
SozKompetenzMax	[SK]	-	soziale Kompetenz des besten Individuums in der Gesellschaft
i1	-	1	Gewichtungsfaktor Wissen
i2	-	1	Gewichtungsfaktor soziale Kompetenz
i	[IndQ]	-	Normierungsfaktor

Abbildung 9.13: Die Modellgrößen zur Berechnung des sozialen Zustands in der Gruppenlernphase

9.1.1.7 Das Aktionsrepertoire der Agenten

Die Agenten, die im Modell *Lerngruppe* Individuen repräsentieren, verfügen über verschiedene Klassen von Aktionen, die zu ihrer freien Disposition stehen. Zunächst einmal kann man zwischen impliziten und expliziten Aktionen unterscheiden.

Zu den impliziten Aktionen zählen hierbei das Lernen sowie die Interaktion der Individuen mit anderen Individuen in Lerngruppen. Es wird vereinfachend angenommen, daß diese beiden Aktionen permanent in den entsprechenden Lernphasen ausgeführt werden, ohne daß es dazu eines expliziten Anstoßes im Rah-

men der Verhaltenssteuerung eines Agenten bedarf. Die Abbildung dieser beiden impliziten Aktionen erfolgt im Modell durch die permanente Veränderung des Wissens sowie der sozialen Kompetenz, sobald die Agenten in die entsprechenden Lernphasen eintreten.

Die expliziten Aktionen, die den Agenten im Rahmen des Modells *Lerngruppe* zur Verfügung gestellt werden, stehen in unmittelbarem Zusammenhang mit der Änderung der aktuellen Lernsituation, d. h. mit dem Wechsel von der Einzellernphase in die Gruppenlernphase und wieder zurück. Aktionen, die dieser Klasse angehören, werden im Rahmen der Verhaltenssteuerung explizit ausgewählt und veranlasst. Hierbei kann in Anlehnung an Kapitel 6.2.8 zwischen internen und externen Aktionen unterschieden werden.

Interne Aktionen dienen dem Anstoß kognitiver Prozesse wie etwa die Selektion einer Lerngruppe, der ein Agent im Rahmen des Übergangs von der Einzel- in die Gruppenlernphase gründen möchte.

Externe Aktionen wirken nach außen hin auf die Umgebung der Agenten. Im Modell *Lerngruppe* besteht die Umgebung eines Agenten sowohl aus anderen Agenten als auch aus Lerngruppen, wobei explizit angestoßene Wechselwirkungen auf die Interaktion von Individuen mit Lerngruppen beschränkt sind.

Aktionen im Zusammenhang mit dem Eintritt in eine Gruppenlernphase

Im Rahmen des Wechsels aus einer Einzel- in eine Gruppenlernphase stehen den Agenten im Modell *Lerngruppe* die folgenden Aktionen zur Verfügung:

- Die externe Aktion *Anfrage_verschicken*

  Mit Hilfe der Aktion *Anfrage_verschicken* fordert ein Agent alle zum betrachteten Zeitpunkt bestehenden Lerngruppen auf, ihre aktuelle Qualität mitzuteilen. Diese Information benötigt der Agent, um zu entscheiden, welcher Lerngruppe er bevorzugt beitreten möchte.

- Die externe Aktion *Bewerbung_verschicken*

  Nach Auswahl einer präferierten Lerngruppe teilt ein Agent dieser Lerngruppe auf Grundlage der Aktion *Bewerbung_verschicken* seine Absicht mit, dieser Lerngruppe beitreten zu wollen. Die Lerngruppe reagiert auf eine derartige Anfrage entweder mit einer Zu- oder einer Absage.

- Die interne Aktion *Bestimme_Gruppenagent*

  Existieren in der vorliegenden Situation keine Lerngruppen, denen der Agent beitreten könnte, wird er versuchen, eine neue Lerngruppe zu gründen. Die Ausführung der internen Aktion *Bestimme_Gruppenagent* liefert einem Agenten die Information, welche Gruppen zum gegebenen Zeitpunkt inaktiv sind und damit zur freien Verfügung stehen.

Aktionen im Zusammenhang mit dem Austritt aus einer Gruppenlernphase

Der Austritt aus einer Gruppenlernphase kann aus Sicht eines Agenten entweder freiwillig oder unfreiwillig erfolgen. Entscheidet sich ein Agent aus eigenem Antrieb heraus, eine Lerngruppe zu verlassen, steht ihm zu diesem Zweck die externe Aktion *Gruppe_verlassen* zur Verfügung.

- Die externe Aktion *Gruppe_verlassen*

  Diese Aktion ist an diejenige Gruppe gerichtet, in der der Agent zum betrachteten Zeitpunkt als Mitglied eingebunden ist. Als Folge dieser Aktion registriert die Gruppe den Austritt dieses Mitglieds und der Agent tritt wiederum in eine Einzellernphase ein. Verläßt das letzte noch verbliebene Mitglied durch Ausführung der Aktion *Gruppe_verlassen* eine Gruppe, so wird diese Gruppe schließlich aufgelöst.

Ebenso verfügt eine Lerngruppe über die Möglichkeit, ein bestehendes Mitglied auszuschließen. Dabei wird angenommen, daß ein von einer derartigen Maßnahme betroffener Agent dem Ausschluß Folge leisten muß. Ist der Bedarf an Gruppenarbeit für diesen Agenten zum Zeitpunkt des Ausschlusses noch nicht gedeckt, wird dieser erneut in eine Bewerbungsphase eintreten, um eine andere Lerngruppe zu finden.

9.1.1.8 Die Verhaltenssteuerung der Agenten

Im Rahmen der Verhaltenssteuerung der Agenten wird festgelegt, unter welchen Bedingungen die Agenten im Modell *Lerngruppe* auf die ihnen zur Verfügung stehenden Aktionen zurückgreifen. Die Verhaltenssteuerung ist dabei ganz bewußt sehr einfach gehalten und beschränkt sich auf rein reaktive Mechanismen, die mit Hilfe von vordefinierten Verhaltensregeln beschrieben werden.

Das Ziel der Agenten im vorliegenden Modell besteht darin, eine möglichst große Menge an Wissen zu erwerben, sowie ihr Bedürfnis nach Sozialkontakten möglichst gut zu befriedigen. Dieses Ziel erreichen die Agenten am besten dadurch, daß sie abwechselnd alleine und in Gruppen lernen. Die Entscheidung, in eine Lerngruppe einzutreten bzw. diese zu einem späteren Zeitpunkt wieder zu verlassen, hängt dabei ausschließlich von der Entwicklung des Wissens und der sozialen Zufriedenheit eines Agenten in der gegebenen Situation ab. Abhängig von der Entwicklung dieser beiden Entscheidungsparameter richtet ein Agent also sein Verhalten darauf hin aus, in eine Lerngruppe einzutreten bzw. diese wieder zu verlassen.

Die Verhaltenssteuerung im Zusammenhang mit dem Eintritt in eine Gruppenlernphase

Ein Agent wird versuchen, eine Einzellernphase zu beenden und in eine Lerngruppe einzutreten, wenn die folgende Situation vorliegt:

- Im Rahmen der Einzellernphase ist der gesamte, zur Bearbeitung anstehende Lernstoff weitestgehend durchgearbeitet worden. Eine signifikante und effiziente Steigerung des Wissens ist in dieser Situation nur dadurch möglich, daß die erworbenen Kenntnisse in einer Lerngruppe mit anderen Individuen diskutiert und gefestigt werden.
- Die Stärke des Bedürfnisses nach Sozialkontakten hat eine gewisse Schwelle überschritten, so daß es das Individuum dazu drängt, sich in Gesellschaft mit anderen Individuen zu begeben und durch die soziale Interaktion dieses Bedürfnis zu befriedigen.

Die Bedingung Bed. 9.1 gibt an, unter welchen Umständen nun tatsächlich in einem Agenten eine Kette von Aktionen angetriggert wird mit dem Ziel, eine Gruppenlernsituation herbeizuführen:

$$j * (j1 * WisAkt (t) + j2 * SozBed (t)) > LimitAllein \qquad \text{(Bed. 9.1)}$$

Sowohl der kognitive als auch der soziale Aspekt wirken also auf die Verhaltenssteuerung eines Agenten ein. Beide Einflussfaktoren gehen dabei in Bedingung Bed. 9.1 mit einem spezifischen Faktor *j1* bzw. *j2* ein, so daß über diesen Mechanismus eine Gewichtung der kognitiven und sozialen Einflußgröße im Vergleich zueinander vorgenommen werden kann. Unter den gegebenen Voraussetzungen des Modells erscheint es durchaus plausibel, den kognitiven Aspekt der Entscheidung, eine Gruppenlernsituation herbeizuführen, höher zu gewichten als den sozialen Aspekt.

Ist diese Bedingung im Rahmen der Verhaltenssteuerung eines Agenten erfüllt, so wird ein Agent versuchen, einer Lerngruppe beizutreten bzw. bei Bedarf eine neue Lerngruppe zu gründen. Die Agenten sind dabei als rationale Entscheider konzipiert und wählen Lerngruppen, denen sie beitreten möchten, danach aus, welchen Lernerfolg sie versprechen. Lerngruppen mit einer hohen Qualität erhalten dabei Vorrang vor Lerngruppen mit einer geringeren Qualität. Steht für einen Agenten die Entscheidung fest, in eine Gruppenlernphase einzutreten, wählt er das folgende, prinzipielle Vorgehen:

Zunächst werden mit Hilfe der Aktion *Anfrage_verschicken* Informationen über die Qualität der bestehenden Lerngruppen (s. Abschnitt 9.1.2.1) eingeholt. Daraufhin bringt der Agent die bestehenden Lerngruppen anhand ihrer gemeldeten Qualität in eine Reihenfolge und bewirbt sich, beginnend mit der besten Gruppe, um Aufnahme in die Gruppe. Zur Übermittlung der Bewerbung nutzt der Agent die externe Aktion *Bewerbung_verschicken*, in der als Information für die Gruppe das Wissen sowie die soziale Kompetenz des Agenten mitgeliefert werden. Die betroffene Lerngruppe entscheidet anhand der Fähigkeiten des Bewer-

bers über dessen Aufnahme und meldet diesem ihre Entscheidung zurück. Wird der Agent abgewiesen, stellt er eine weitere Anfrage an diejenige Lerngruppe, die die nächste Position in seiner Präferenzordnung einnimmt. Akzeptiert die Lerngruppe hingegen die Bewerbung des Agenten, wird dieser in die Lerngruppe aufgenommen und tritt in die Gruppenlernphase ein.

Wenn zu dem Zeitpunkt, zu dem ein Agent den Übergang in eine Gruppenlernphase anstrebt, keine Lerngruppe existiert oder wenn alle bestehenden Lerngruppen auf die Bewerbung des Agenten hin mit einer Absage reagieren, wird der Agent versuchen, eine neue Gruppe zu gründen. Hierzu wird er zunächst einmal mit Hilfe der internen Aktion *Bestimme_Gruppenagent* überprüfen, welche Gruppe zur freien Verfügung steht. Danach wendet er sich unter Verwendung der externen Aktion *Bewerbung_verschicken* an die ermittelte, freie Gruppe, versetzt diese damit in einen aktiven Zustand und wird gleichzeitig als erstes Mitglied in dieser Gruppe verzeichnet. In der Folge verbleibt der Agent zunächst einmal für eine gewisse Zeit als einziges Mitglied in dieser Gruppe in der Hoffnung, dass sich seiner neu gegründeten Gruppe bald weitere Mitglieder anschließen. Kommen innerhalb eines vorgegebenen Zeitraums weitere Mitglieder hinzu, ist über diesen Weg eine neue, funktionsfähige Lerngruppe entstanden. Existieren hingegen keine Agenten, die sich dieser Gruppe anschließen, gibt der Agent durch Ausführung der Aktion *Gruppe_verlassen* diese Gruppe wieder auf und versucht unter der Annahme, daß sich die Situation in der Zwischenzeit maßgeblich verändert hat, sich in einem weiteren Bewerbungszyklus einer bereits bestehenden Gruppe anzuschließen. Während des Zeitraums, in dem ein Agent mit dem Ziel, eine neue Gruppe zu gründen, als einziges Mitglied einer Gruppe registriert ist, verändern sich sein Wissen und ebenso seine soziale Kompetenz in Analogie zur Einzellernphase.

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
LimitAllein	-	1.5	Schwellwert für die Beendigung einer Einzellernphase
j1	[1/WI]	2	Gewichtungsfaktor für den kognitiven Aspekt der Entscheidung
j2	[1/SoZ]	1	Gewichtungsfaktor für den sozialen Aspekt der Entscheidung
j	-	-	Normierungsfaktor: *2 / (j1 + j2)*

Abbildung 9.14: Übersicht über die neu eingeführten Modellgrößen, die die Entscheidung eines Agenten, in eine Gruppenlernphase einzutreten, maßgeblich beeinflussen

Die Verhaltenssteuerung im Zusammenhang mit dem Austritt aus einer Gruppenlernphase

Ebenso wie die Einzellernphase unterliegt auch die Gruppenlernphase hinsichtlich ihrer Effektivität gewissen Grenzen. So wird es für einen Agenten nach einer Zeit der intensiven Interaktion mit anderen Agenten auch wieder erforderlich sein, sich zurückzuziehen und weitere Lerninhalte selbständig und alleine zu erarbeiten. Ein Agent wird gemäß dieser Annahme so lange in einer Lerngruppe verbleiben, bis die beiden folgenden Aspekte zum tragen kommen:

- Durch die Interaktion mit anderen Agenten kann kein signifikanter Lernfortschritt mehr erzielt werden. Ein effizienter Ausbau des Wissens kann in dieser Situation nur dadurch erreicht werden, daß neue Lerninhalte selbständig außerhalb einer Gruppe erarbeitet werden.
- Durch die soziale Interaktion mit anderen Agenten hat sich die soziale Zufriedenheit eines Agenten so weit regeneriert, daß kein ausreichend hohes Bedürfnis mehr existiert, in der Gesellschaft anderer Agenten zu sein. Nach einer entsprechenden Phase der sozialen Interaktion ist ein Agent für eine gewisse Zeit lang in der Lage, ohne Kontakt zu anderen Agenten auszukommen.

Beide Einflüsse zusammen können dazu führen, daß ein Agent eine Lerngruppe verlässt und sich wieder in eine Einzellernphase zurückzieht. Die Bedingung 9.2 gibt an, unter welchen Umständen dies nun konkret der Fall sein wird:

$$k * (k1 * WisAkt (t) + k2 * SozAkt (t)) > LimitGruppe \qquad \text{(Bed. 9.2)}$$

Wie schon in Bedingung 9.1 gehen auch in Bedingung 9.2 sowohl der kognitive Aspekt als auch der soziale Aspekt mit einem vordefinierten Gewicht *k1* bzw. *k2* in die Entscheidung ein, eine Lerngruppe zu verlassen.

Die Abbildung 9.15 zeigt im Überblick die für die Formulierung der Austrittsbedingung aus einer Gruppe neu festgelegten Modellgrößen.

Eine besondere Situation tritt dann ein, wenn nach Ausstieg eines Agenten in der betreffenden Gruppe nur noch ein Agent verbleibt. Dieser Agent wird dann für eine gewisse Zeit versuchen, die Gruppe am Leben zu erhalten, und anderen Agenten die Möglichkeit geben, sich dieser Gruppe anzuschließen. Tritt jedoch innerhalb einer vorgegebenen Zeit *TGruppe* kein weiterer Agent dieser Gruppe bei, so verlässt auch der letzte Agent durch Ausführung der Aktion *Gruppe_verlassen* die Gruppe und löst diese damit auf.

Die Entscheidung eines Agenten, als letzter eine Gruppe zu verlassen, wird ausschließlich zeitgesteuert modelliert. Hierzu wird für eine Gruppe in der Modellgröße *TUnechteGruppe* vermerkt, zu welchem Zeitpunkt das vorletzte Mitglied aus der Gruppe ausscheidet. Verstreichen danach weitere *TGruppe* Zeiteinheiten, verlässt auch das letzte Mitglied die Gruppe. Die entsprechende Bedingung für die Ausführung der Aktion *Gruppe_verlassen* des letzten Gruppenmitglieds hat damit die folgende Gestalt:

$T > TUnechteGruppe + TGruppe$ (Bed. 9.3)

Die Modellgröße *T* bezeichnet dabei die aktuelle Zeit im Simulationsmodell.

Bezeichner	Einheit	Initialwert	Erläuterung
LimitGruppe	-	1.5	Schwellwert für die Beendigung einer Gruppenlernphase
k1	[1/WI]	-	Gewichtungsfaktor für den kognitiven Aspekt der Entscheidung *k1 = 1 / j1*
k2	[1/SoZ]	-	Gewichtungsfaktor für den sozialen Aspekt der Entscheidung *k2 = 1 / j2*
k	-	-	Normierungsfaktor: *2 / (1 / j1 + 1 / j2)*

Abbildung 9.15: Übersicht über die neu eingeführten Modellgrößen, die die Entscheidung eines Agenten, eine Gruppenlernphase zu verlassen, maßgeblich beeinflussen

Die Verhaltenssteuerung der Agenten in der Übersicht

Das Ablaufdiagramm in Abbildung 9.16 zeigt noch einmal im Überblick die grundlegende Verhaltenssteuerung, die in den Individualagenten des Modells *Lerngruppe* zum Einsatz kommt.

9.1.2 Die Lerngruppen

Lerngruppen repräsentieren im vorliegenden Modell Zusammenschlüsse von Individuen mit dem Ziel, den Wissenserwerb ihrer Mitglieder auf der Grundlage von gemeinsamen Diskussionen zu fördern wie auch deren Bedürfnis nach Sozialkontakten zu befriedigen. Die Lerngruppen werden dabei als eigenständige Entitäten betrachtet, die zusätzlich zu den Eigenschaften ihrer Gruppenmitglieder auch über eigene Zustände, Zustandsübergänge, Aktionen und Verhaltensweisen verfügen.

Lerngruppen werden dabei mit Hilfe von sog. *Gruppenagenten* modelliert. Dabei gilt die grundlegende, vereinfachende Annahme, daß ein Gruppenagent nach außen hin die ihm zugeordnete Lerngruppe repräsentiert, stellvertretend für die einzelnen Mitglieder der Lerngruppe verschiedene Aufgaben übernimmt und vollständig im Sinne der Gruppenmitglieder agiert. Diese Annahme wurde für das Modell *Lerngruppe* im wesentlichen aus zwei Gründen getroffen:

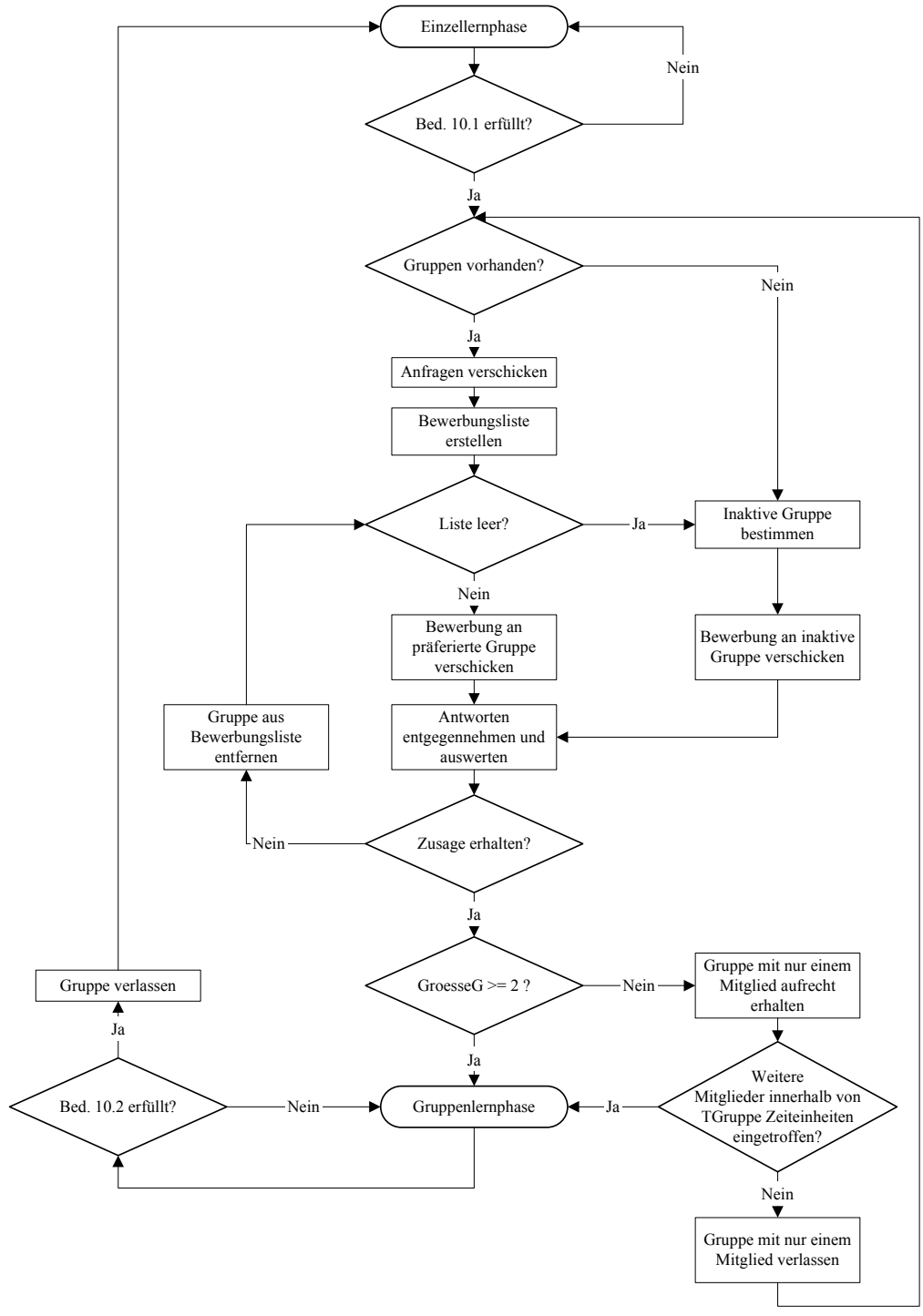

Abbildung 9.16: Übersicht über die Verhaltenssteuerung der Individualagenten

Zum einen existieren für eine Lerngruppe tatsächlich Eigenschaften, wie etwa die Qualität der Lerngruppe, die nur der Gruppe und nicht den einzelnen Mitgliedern der Gruppe zuzuordnen sind. In diesem Sinne stellt das Konstrukt Gruppenagent ein einfaches Trägerobjekt für Informationen dar, die auf der Individualebene nicht zugeordnet werden können. Zum anderen vereinfacht die Einführung von Gruppenagenten die Bereitstellung von Funktionalitäten, die sonst die einzelnen Gruppenmitglieder nach Absprache mit übernehmen müssten, wie etwa die Antwort auf Anfragen potentieller Gruppenmitglieder, die Entscheidung über Bewerbungen neuer Gruppenmitglieder etc. Aufgrund der relativ dynamischen Zusammensetzung der Lerngruppen wäre hier ein permanenter Abstimmungsprozess zwischen den Gruppenmitgliedern erforderlich, welches der Mitglieder zu einem gegebenen Zeitpunkt diese Funktionalitäten wahrnimmt. Da die Abbildung derartiger Aspekte allerdings nicht zu den Kernintentionen des Modells *Lerngruppe* zählt, ist dieser Teil der Modellfunktionalität ganz bewusst einem Abstraktionsprozess unterzogen und dabei in die Gruppenagenten verlagert worden.

Wesentliches Bestimmungsmerkmal einer Lerngruppe ist ihre Qualität. Die Qualität einer Lerngruppe hängt dabei grundlegend von dem Wissen, der sozialen Kompetenz ihrer Mitglieder, sowie ihrer Größe, d. h. von der aktuellen Anzahl an Mitgliedern ab. Die Qualität einer Lerngruppe nimmt Einfluß darauf, wie schnell ihre Mitglieder neues Wissen erwerben und ihre soziale Zufriedenheit steigern können. Ebenso dient sie als wesentlicher Entscheidungsparameter bei der Aufnahme neuer Mitglieder sowie bei Ausschluß bestehender Mitglieder aus der Lerngruppe. Nicht zuletzt findet die Qualität der Lerngruppe auch dafür Verwendung, um ähnlich wie bei den Individuen, eine Rangfolge der Lerngruppen zu ermitteln und den Status von Lerngruppen im Vergleich zu allen anderen Lerngruppen zu bestimmen.

Die folgenden Abschnitte beschäftigen sich nun mit der Bestimmung der Qualität einer Lerngruppe, den Aktionen, die den Gruppenagenten zur Verfügung stehen, sowie mit der Verhaltenssteuerung, die innerhalb der Gruppenagenten zum Einsatz kommt.

9.1.2.1 Die Zustandsvariablen der Gruppenagenten und deren Dynamik

Wie eingangs bereits erwähnt, ist das wesentliche Merkmal, das eine Lerngruppe auszeichnet, ihre Qualität. Die Qualität, die einer Lerngruppe in dieser Hinsicht zugeordnet wird, wird dabei durch die folgenden Faktoren beeinflusst:

- Wissensstand der Gruppenmitglieder

  Die Qualität einer Lerngruppe wird um so höher sein, je größer das Wissen ist, das die einzelnen Mitglieder in die Gruppe hineintragen. Hinter dieser Annahme verbirgt sich der Gedanke, daß eine Gruppe, deren Mitglieder bereits mit einem guten Wissenstand ausgestattet sind, nicht viel Zeit darauf verwenden muß, um grundlegende Sachverhalte zu rekapitulieren, sondern

sofort in weiterführende Diskussionen einsteigen kann, die dazu dienen, das Wissen zu erweitern und zu festigen.

- Soziale Kompetenz der Gruppenmitglieder

  Je höher die soziale Kompetenz der Mitglieder einer Lerngruppe sein wird, um so leichter wird es ihnen fallen, effektiv in der Lerngruppe zusammenzuarbeiten. Dabei erscheint es plausibel, daß solche Gruppen leistungsfähiger sind, deren Mitglieder bereits über ein hohes Maß an Erfahrung in der Teamarbeit verfügen.

- Gruppengröße

  Ebenso wird sich die Größe einer Gruppe ganz entscheidend auf deren Qualität auswirken. Keller (Keller, 1999) nennt in diesem Zusammenhang die Zahl fünf als eine in der Praxis brauchbare Gruppengröße. Bei Gruppen, die noch mehr Mitglieder umfassen, steigen die Reibungsverluste innerhalb der Gruppe an und die Gefahr wächst, daß sich Gruppenmitglieder nicht mit dem erforderlichen Einsatz in die Gruppe einbringen. Die relative Leistungssteigerung einer Gruppe durch Hinzunahme eines weiteren Gruppenmitglieds nimmt daher ab einer gewissen Gruppengröße immer mehr ab. Das Modell *Lerngruppe* geht von einer optimalen Gruppengröße von vier Mitgliedern aus.

  Die Berechnung der Qualität einer Lerngruppe folgt der Gleichung Gl. 9.16:

$$QualitaetG(t) := d + g * (g1 * RelWisGN(t) + g2 * RelSozGN(t) + g3 * QualGroesse(GroesseG(t))) \quad \text{(Gl. 9.16)}$$

Die Modellgröße *RelWisGN* repräsentiert das durchschnittliche Gesamtwissen der Gruppenmitglieder im Verhältnis zum durchschnittlichen Gesamtwissen aller Individuen in der Gesellschaft. Es gilt

$$RelWisGN(t) := RelWisG(t) / RelWisGMax(t) \quad \text{(Gl. 9.17)}$$

sowie

$$RelWisG(t) := \text{durchschn. Wissen der Gruppenmitglieder / durchschn. Wissen aller Individuen} \quad \text{(Gl. 9.18)}$$

Die Modellgröße *RelWisGMax (t)* bezeichnet dabei den größten Wert, der zu einem gegebenen Zeitpunkt über alle Lerngruppen hinweg für deren Modellgröße *RelWisG (t)* auftritt, und dient der Normierung des kognitiven Einflusses im Rahmen der Berechnung der Gruppenqualität.

Die abhängige Größe *RelSozGN (t)* beschreibt die gemittelte soziale Kompetenz der Mitglieder einer Gruppe im Vergleich zur durchschnittlichen sozialen Kompetenz, die sich aus einer Betrachtung über alle Individuen der Gesellschaft

hinweg ergibt. Die Gleichung zur Bestimmung von *RelSozGN (t)* lautet wie folgt:

RelSozGN (t) := RelSozG (t) / RelSozGMax (t) (Gl. 9.19)

sowie

RelSozG (t) := durchschn. soziale Kompetenz der Gruppenmitglieder / durchschn. soziale Kompetenz aller Individuen (Gl. 9.20)

Auch in diesem Fall stellt die Modellgröße *RelSozGMax (t)* den zum Zeitpunkt *t* größten Wert für die Modellgröße *RelSozG (t)* aller bestehender Lerngruppen dar und dient zur Normierung des sozialen Einflusses bei der Festlegung der Gruppenqualität.

Die Größe einer Gruppe, d. h. die zum Zeitpunkt *t* gültige Anzahl an Mitgliedern einer betrachteten Gruppe, wird mit Hilfe der Zustandsvariable *GroesseG* abgebildet. Die Größe einer Gruppe ändert sich ereignisorientiert immer dann, wenn ein neues Mitglied in eine Gruppe aufgenommen wird, ein bestehendes Mitglied eine Gruppe verlässt oder eine Gruppe proaktiv zur Steigerung ihrer Qualität ein Mitglied ausschließt.

Der Einfluß, den die Größe einer Gruppe auf deren Qualität ausübt, wird mit Hilfe einer sog. Tabellenfunktion (Schmidt, 2000) modelliert. Dabei wird jeder möglichen Gruppengröße ein Funktionswert *QualGroesse (GroesseG (t))* zugeordnet, der angibt, mit welchem Betrag die Gruppengröße als Einflussfaktor in die Berechnung der Gruppenqualität eingeht. Es gilt die Annahme, daß bis zu einer Anzahl von insgesamt vier Mitgliedern die Qualität einer Gruppe durch Hinzunahme eines weiteren Mitglieds kontinuierlich ansteigt. Eine Gruppe mit vier Mitgliedern wird als maximal leistungsfähig eingeschätzt. Jedes weitere Mitglied führt zu einem Absinken der Gruppenqualität. Im Modell *Lerngruppe* ist die maximale Größe einer Gruppe auf sechs Mitglieder begrenzt. Die Abbildung 9.17 macht einen Vorschlag für die in der Tabellenfunktion zu hinterlegenden Wertepaare:

GroesseG	1	2	3	4	5	6
QualGroesse (GroesseG (t))	0.45	0.70	0.85	1	0.90	0.65

Abbildung 9.17: Ein Vorschlag für die Einrichtung einer Tabellenfunktion zur Bestimmung des Einflusses der Gruppengröße auf die Gruppenqualität

Die Faktoren *g1*, *g2* und *g3* in Gleichung Gl. 9.16 lassen eine Gewichtung der unterschiedlichen Einflussfaktoren hinsichtlich der Berechnung der Gruppenqualität zu. Bei einer Lerngruppe wird sicherlich der kognitive Aspekt neben der Gruppengröße die wesentliche Rolle spielen.

Mit Hilfe des Faktors *g* wird eine Normierung der Summe aus den drei Einflussfaktoren vorgenommen, so daß sich hier ein Wertebereich von 0 [GQ] bis 2 [GQ] ergibt. Der zusätzliche Summand *d* sorgt dafür, daß der Gesamtwert für die Qualität einer Gruppe nicht unter den Wert 1 [GQ] absinkt.

Die Abbildung 9.18 zeigt im Überblick alle Modellgrößen, die an der Bestimmung der Qualität einer Gruppe beteiligt sind.

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
QualitaetG	[GQ]	-	Qualität der Gruppe
RelWisGN	-	-	normiertes, relatives Wissen der Gruppenmitglieder
RelWisG	-	-	relatives Wissen der Gruppenmitglieder
RelWisGMax	-	-	Maximalwert für *RelWisG* zu einem gegebenen Zeitpunkt über alle Gruppen
RelSozGN	-	-	normierte, relative soziale Kompetenz der Gruppenmitglieder
RelSozG	-	-	relative soziale Kompetenz der Gruppenmitglieder
RelSozGMax	-	-	Maximalwert für *RelSozG* zu einem gegebenen Zeitpunkt über alle Gruppen
GroesseG	-	-	Anzahl an Mitgliedern einer Gruppe
QualGroesse	-	-	Tabellenfunktion zur Bestimmung des Einflusses der Gruppengröße auf die Gruppenqualität
g1	-	2	Gewichtungsfaktor für den Einfluß des Wissens auf die Gruppenqualität
g2	-	1	Gewichtungsfaktor für den Einfluß der sozialen Kompetenz auf die Gruppenqualität
g3	-	1	Gewichtungsfaktor für den Einfluß der Gruppengröße auf die Gruppenqualität
g	[GQ]	-	Normierungsfaktor: *2 / (g1 + g2 + g3)*
d	[GQ]	1	Basiswert für die Qualität einer Gruppe

Abbildung 9.18: Übersicht über die Modellgrößen, die an der Bestimmung der Qualität einer Gruppe beteiligt sind

9.1.2.2 Das Aktionsrepertoire der Gruppenagenten

Gruppenagenten übernehmen, wie eingangs erwähnt, in Stellvertretung der einzelnen Gruppenmitglieder verschiedene Aufgaben und versorgen darüber hinaus die Gruppenmitglieder bei Bedarf mit Informationen über Veränderungen in Bezug auf die Zusammensetzung der Gruppe.

- Die Aktion *Auskunft_verschicken*

  Mit Hilfe der Aktion *Auskunft_verschicken* teilt ein Gruppenagent einem potentiellen Beitrittskandidaten die aktuelle Qualität der Gruppe mit.

- Die Aktion *Zusage_verschicken*

  Die Aktion *Zusage_verschicken* dient der Information eines Bewerbers, daß er in die Lerngruppe aufgenommen wurde.

- Die Aktion *Absage_verschicken*

  Als negatives Pendant zur Aktion *Zusage_verschicken* informiert ein Gruppenagent mit Hilfe der Aktion *Absage_verschicken* einen Bewerber darüber, daß die Gruppe nicht bereit ist, ihn als neues Mitglied mit aufzunehmen.

- Die Aktion *Mitglied_ausschliessen*

  Unterschreitet die Qualität einer Lerngruppe eine gewisse untere Schranke, so besteht die Möglichkeit, ein Gruppenmitglied proaktiv aus der Gruppe auszuschließen. Den Ausschluß teilt die Gruppe dem betroffenen Agenten durch Ausführung der Aktion *Mitglied_ausschließen* mit.

- Die Aktion *Update_verschicken*

  Die Aktion Update_verschicken wird durch einen Gruppenagenten herangezogen, um die Mitglieder seiner Lerngruppe über eine Änderung der Gruppengröße, z. B. nach Aufnahme eines neuen oder Ausschluß eines bestehenden Mitglieds, zu informieren.

Bei den Aktionen der Gruppenagenten handelt es sich also ausnahmslos um externe Aktionen, die stellvertretend für die Mitglieder der Gruppe ausgeführt werden oder zur Verteilung von Informationen auf die Gruppenmitglieder dienen.

9.1.2.3 Die Verhaltenssteuerung der Gruppenagenten

Wie schon bei den Individualagenten ist auch die Verhaltenssteuerung der Gruppenagenten auf rein reaktive Mechanismen beschränkt, d. h. auch in den Gruppenagenten kommt ein vordefinierter Satz an Regeln zum Einsatz, der das Verhalten der Gruppenagenten bestimmt.

Die Verhaltenssteuerung eines Gruppenagenten wird dabei im wesentlichen durch externe Stimuli in Form von Anfragen, die von Individualagenten abgesetzt werden, angestoßen. Dies ist beispielsweise bei der Auslösung der Aktion *Auskunft_verschicken* der Fall, die rein reaktiv als Antwort auf das Eintreffen einer

Anfrage eines Individualagenten ausgeführt wird, oder auch bei der Aktion *Update_verschicken*, die direkt nach einer Änderung der Gruppenzusammensetzung ausgelöst wird.

Die wesentlichen Aktivitäten der Gruppenagenten bestehen darin, im Sinne der Gruppe darüber zu entscheiden, ob ein neuer Bewerber in die Gruppe aufgenommen wird, und, ob ein bestehendes Mitglied aus der Gruppe ausgeschlossen werden kann, um die Qualität der Gruppe zu erhöhen. Als maßgeblicher Entscheidungsparameter kommt hier ein Qualitätsstandard zum tragen, den eine Gruppe immer wieder neu für sich definiert.

Die Definition des Qualitätsstandards einer Gruppe

Das Modell *Lerngruppe* geht von der grundlegenden Annahme aus, daß eine Lerngruppe für sich einen bestimmten Qualitätsstandard festlegt. Dieser Qualitätsstandard wird bei jeder Änderung der Gruppensituation, d. h. immer dann, wenn ein neues Mitglied der Gruppe beitritt oder ein bestehendes Mitglied die Gruppe verlässt bzw. aus dieser ausgeschlossen wird, neu bestimmt. Die Festlegung des Qualitätsstandards *QualitaetGMin* erfolgt dabei auf Grundlage der aktuellen Gruppenqualität *QualitaetG* sowie unter Berücksichtigung einer vordefinierten Toleranz *QualitaetGMinFaktor*. Eine Vorbelegung der Konstante *QualitaetGMinFaktor* mit dem Wert 0.95 würde dann etwa bedeuten, daß eine Gruppe bis zur nächsten Änderung der Gruppensituation ein Absinken ihrer aktuellen Qualität um 5% gerade noch tolerieren würde. Die Berechnung des Qualitätsstandards *QualitaetGMin* einer Gruppe erfolgt unter Verwendung der Gleichung Gl. 9.21.

$$QualitaetGMin(t) := QualitaetG(t) * QualitaetGMinFaktor \qquad \text{(Gl. 9.21)}$$

Die Aufnahme bzw. Ablehnung eines Bewerbers

Eine Lerngruppe wird einen neuen Bewerber in die Gruppe mit aufnehmen, sofern die maximale Gruppengröße dadurch nicht überschritten wird. Zudem wird bei dieser Entscheidung berücksichtigt, inwiefern sich die Qualität der Gruppe durch Aufnahme des Bewerbers verändern würde. Eine Lerngruppe wird nur dann einen neuen Bewerber aufnehmen, wenn dieser Bewerber dazu beitragen kann, die Gruppenqualität anzuheben, oder wenn dadurch die Gruppenqualität zumindest nicht unter den Qualitätsstandard, den die Gruppe für sich vorgegeben hat, absinkt. Die kalkulatorische Gruppenqualität, die sich durch Aufnahme eines Bewerbers für eine Gruppe ergeben würde, wird dabei in der Modellgröße *QualitaetGNeu* hinterlegt.

Die in Abbildung 9.19 dargestellte Entscheidungstabelle zeigt alle für die Aufnahme bzw. Ablehnung eines Bewerbers relevanten Entscheidungsregeln in der Übersicht.

Aufnahme / Ablehnung eines Bewerbers		R1	R2	R3	R4
B1	maximale Gruppengröße überschritten?	N	N	N	J
B2	QualitaetGNeu > QualitaetG	J	-	N	-
B3	QualitaetGNeu > QualitaetGMin	-	J	N	-
A1	*Zusage_verschicken*	X	X		
A2	*Absage_verschicken*			X	X

Abbildung 9.19:Die Entscheidungsregeln für die Aufnahme bzw. Ablehnung eines Bewerbers

Der Ausschluß eines bestehenden Gruppenmitglieds

Die Gruppenagenten übernehmen hinsichtlich der Qualität der Gruppen, die sie repräsentieren, eine gewisse Überwachungsfunktion. In regelmäßigen Zeitabständen überprüfen die Gruppenagenten die aktuelle Gruppenqualität. Liegt bei der Überprüfung die aktuelle Gruppenqualität unter dem Qualitätsstandard der Gruppe, so wird die Gruppe versuchen, ihre Qualität wieder auf ein akzeptables Niveau anzuheben. Zu diesem Zweck wird eine Gruppe in die Lage versetzt, bei Bedarf ein Mitglied aus der Gruppe auszuschließen. Die Gruppe wird dabei versuchen, dasjenige Mitglied aus der Gruppe zu entfernen, das deren Qualität am stärksten negativ beeinflusst bzw. dessen Ausschluß die größte Steigerung der Gruppenqualität bewirken wird. Ein tatsächlicher Ausschluß eines Gruppenmitglieds erfolgt allerdings nur unter der Voraussetzung, daß die Qualität der Gruppe nach Veränderung ihrer Zusammensetzung wieder über dem Qualitätsstandard liegt. Kann eine Steigerung der Qualität über das Mindestniveau des Qualitätsstandards hinaus durch Ausschluß eines Mitglieds nicht erreicht werden, so arbeitet die Lerngruppe in der bestehenden Zusammensetzung weiter.

Die Abbildung 9.20 zeigt im Überblick alle im Zusammenhang mit der Verhaltenssteuerung der Gruppenagenten neu eingeführten Modellgrößen.

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
QualitaetGNeu	[GQ]	-	Qualität der Gruppe nach Änderung der Zusammensetzung
QualitaetGMin	[GQ]	-	Qualitätsstandard einer Gruppe
QualitaetGMinFaktor	[GQ]	-	maximale Toleranz für das Absinken der Gruppenqualität

Abbildung 9.20: Die im Rahmen der Verhaltenssteuerung der Gruppenagenten neu eingeführten Modellgrößen

9.2 Entwurf und Implementierung auf der Grundlage des Referenzmodells *PECS*

Der folgende Abschnitt beschäftigt sich mit dem Implementierungskonzept für das Modell *Lerngruppe*. Ausgehend von der Grundstruktur des Modells werden dazu alle wesentlichen Modellkomponenten hinsichtlich ihres Aufbaus und ihres dynamischen Verhaltens betrachtet. Darüber hinaus wird anhand eines typischen Szenarios das dynamische Zusammenspiel der *PECS*-Komponenten, aus denen die Individualagenten zusammengesetzt sind, beschrieben.

Für die Realisierung des Modells wurde das Referenzmodell *PECS* (s. Kapitel 6) sowie das Simulationssystem *Simplex3* mit der zugehörigen Modellbeschreibungssprache *Simplex-MDL* (Schmidt, 2000) verwendet. *Simplex-MDL* bietet einen systemtheoretischen Ansatz zur Modellbeschreibung und unterstützt einen komponentenorientierten, hierarchischen Modellaufbau. Im Zusammenspiel mit dem Referenzmodell *PECS* wird damit ein komfortabler Aufbau agentenbasierter Simulationsmodelle möglich.

9.2.1 Die Grundstruktur des Modells

Das Simulationsmodell setzt sich in Anlehnung an den komponentenorientierten und hierarchischen Modellaufbau aus einer Menge von unabhängigen und nebenläufig arbeitenden Modellkomponenten zusammen. Das Gesamtmodell entsteht dabei durch eine hierarchische Aggregation von Teilmodellen bzw. Subkomponenten, die sich über insgesamt drei Hierarchieebenen erstrecken. Die Interaktion der Subkomponenten erfolgt auf der Grundlage von kausalen Abhängigkeiten und diskreten Informationsflüssen.

Auf der höchsten Hierarchieebene ist das Gesamtmodell *Lerngruppe* angesiedelt. Das Modell *Lerngruppe* umfasst insgesamt 36 Individualagenten, sowie dieselbe Anzahl an Gruppenagenten. Beide Agententypen weisen eine Unterstruktur auf, die in Anlehnung an die Agentenarchitektur des Referenzmodells *PECS* konzipiert ist. Die Komponente *Connector* dient zur Unterstützung des Informationsaustausches zwischen den Repräsentanten der beiden Agentenklassen. Innerhalb der Komponente *Statistik* werden Informationen sowohl über die Individuen als auch über die aktiven Lerngruppen sowie die Gesamtpopulation berechnet und für die anderen Modellkomponenten zur weiteren Verarbeitung bereitgestellt.

Die Individuen interagieren nicht direkt mit den Gruppen und umgekehrt, sondern bedienen sich zu diesem Zweck der Komponente *Connector*. Die Komponente *Connector* ermöglicht dabei sowohl im Sinne einer zentralen Vermittlungsstelle den gezielten Informationsaustausch zwischen dedizierten Agenten, als auch die Veröffentlichung von Informationen zum asynchronen Zugriff nach dem Vorbild eines Blackboards. Zur Realisierung des direkten Nachrichtenaustausches stellt die Komponente *Connector* für jeden Agenten einen eigenen Ein- und Aus-

gangsbereich für Nachrichten (*EingangI*, *AusgangI*, *EingangG*, *AusgangG*) zur Verfügung, die mit den entsprechenden Nachrichteneinen- und Nachrichtenausgangswarteschlangen in den Agenten verknüpft werden. Zudem legen die Individualagenten auf dem Blackboard der Komponente *Connector* Informationen über ihr aktuelles Wissen (*Wissen*) sowie ihre aktuelle soziale Kompetenz (*SozKompetenz*) ab, die in den Gruppenagenten beispielsweise zur Bestimmung der Gruppenqualität verwendet werden. Die Gruppenagenten stellen ihrerseits mit Hilfe des Blackboards die aktuellen Werte ihrer Gruppenqualität für die Verwendung durch die Individualagenten zur Verfügung. Ebenso vermerken die Gruppenagenten auf dem Blackboard, ob sie zum betrachteten Zeitpunkt aktiv sind, d. h. eine Lerngruppe repräsentieren.

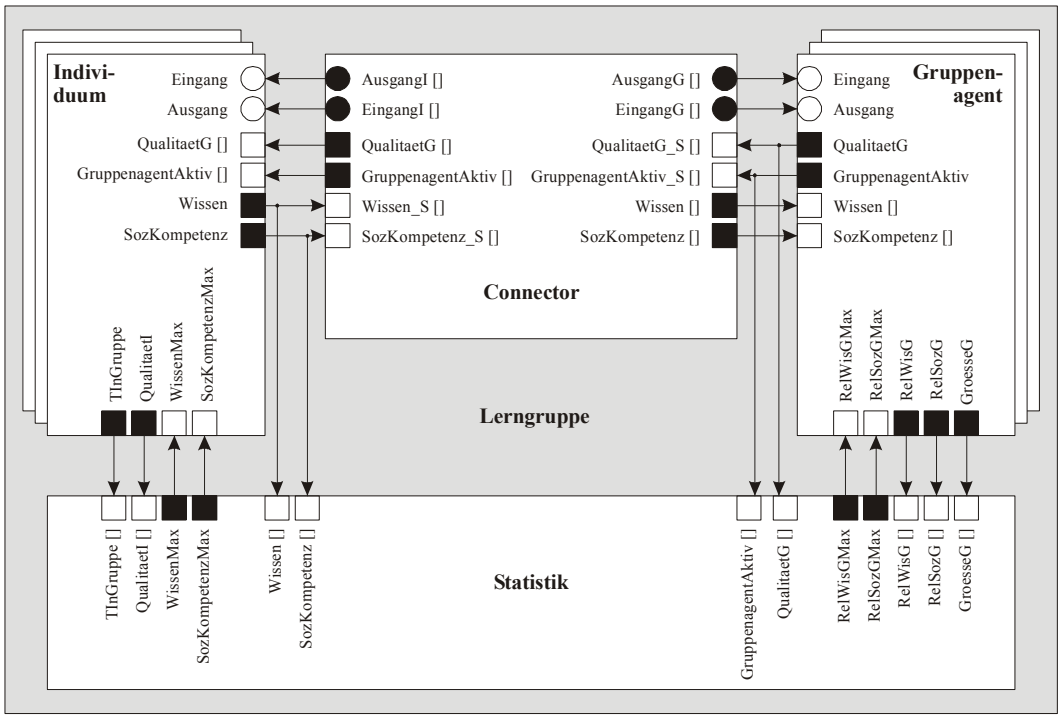

Abbildung 9.21: Die Grundstruktur der Komponente Lerngruppe

Aus der Komponente *Statistik* beziehen die Individualagenten Informationen über die Maximalwerte für das Wissen und die soziale Kompetenz, die über alle Agenten in der Population hinweg auftreten, um ihre eigene Qualität bestimmen zu können. Im Gegenzug stellen die Individualagenten der Komponente *Statistik* die aktuellen Werte ihrer Modellgrößen *Wissen*, *SozKompetenz*, *QualitaetI* und *TInGruppe* zur statistischen Auswertung zur Verfügung, wobei *TInGruppe* die Gesamtzeit bezeichnet, während der sich ein Individualagent in einer Lerngruppe befindet. Ebenso bezieht die Komponente *Statistik* Informationen über die Qualität einer Gruppe, deren relatives Wissen, die relative Sozialkompetenz, sowie deren aktuelle Größe von den aktiven Gruppenagenten. Aus den gelieferten Informa-

tionen werden innerhalb der Komponente *Statistik* entsprechende Maximalwerte über die gesamte Population ermittelt und an die Agenten zurückgemeldet.

Im Vergleich zur grundlegenden Komponentenstruktur des Referenzmodells *PECS* fehlt im Modell *Lerngruppe* die Komponente *Environment*. Auf die Berücksichtigung von Umweltbedingungen, die sich auf das Lernverhalten bzw. die Gruppenbildung von Individuen auswirken, wurde im Modell *Lerngruppe* ganz bewusst verzichtet. Aufgrund des komponentenorientierten Ansatzes des Referenzmodells *PECS* besteht jedoch jederzeit die Möglichkeit, eine derartige Komponente mit einer entsprechenden Dynamikbeschreibung und Wechselwirkungen mit anderen Komponenten in das Modell einzubringen.

Die Abbildung 9.21 zeigt den grundlegenden Aufbau der Strukturkomponente *Lerngruppe*.

9.2.2 Die Architektur der Individualagenten

Die folgenden Abschnitte beschäftigen sich mit dem Aufbau und der Funktionsweise der Agenten, die im Modell *Lerngruppe* die Individuen abbilden.

Der Aufbau der Komponente *Individuum* orientiert sich an der Agentenarchitektur des Referenzmodells *PECS*. Die Komponente *Individuum* verbindet als Strukturkomponente die Subkomponenten *SensorI*, *WahrnehmungI*, *KognitionI*, *StatusI*, *VerhaltenI* und *AktorI*. Da im Modell *Lerngruppe* weder physische noch emotionale Aspekte von Individuen betrachtet werden, fehlen im Vergleich zur Agentenarchitektur von *PECS* in der Architektur der Individuen die Komponenten *Physis* und *Emotion*.

Die Abbildung 9.22 zeigt den grundsätzlichen Aufbau der Strukturkomponente *Individuum*. Von der Komponente *Connector* bezieht die Komponente Individuum die Information, welche Lerngruppen zum betrachteten Zeitpunkt vorhanden sind, Informationen über die Qualität der Lerngruppen, sofern das Individuum Mitglied einer Gruppe ist, sowie Nachrichten, z. B. über die Ablehnung oder die Annahme einer Bewerbung bei einer Lerngruppe. Aus der Komponente *Statistik* werden die Maximalwerte für die soziale Kompetenz sowie das Gesamtwissen über alle Individuen der Population geliefert, die jedes Individuum für die Bestimmung der eigenen Qualität benötigt. Im Gegenzug stellt die Komponente *Individuum* im Falle einer Gruppenlernphase Informationen über das aktuelle Wissen sowie die aktuelle soziale Kompetenz für den betreffenden Gruppenagenten zur Verfügung, um dort eine Berechnung der Gruppenqualität zu ermöglichen. Mit Hilfe von Nachrichten, die über die Location *Ausgang* der Komponente *AktorI* nach außen transportiert werden, setzt die Komponente *Individuum* externe Aktionen in die Außenwelt ab. Alle weiteren Informationen, die ein Individuum nach außen gibt, dienen der statistischen Aufbereitung innerhalb der Komponente *Statistik*.

Auf eine Detaildiskussion der Verbindungen zwischen den einzelnen Unterkomponenten wird an dieser Stelle verzichtet. Entsprechende Ausführungen zu

diesem Aspekt enthalten die folgenden Kapitel, die sich mit dem Aufbau sowie der Funktionsweise der einzelnen Subkomponenten befassen.

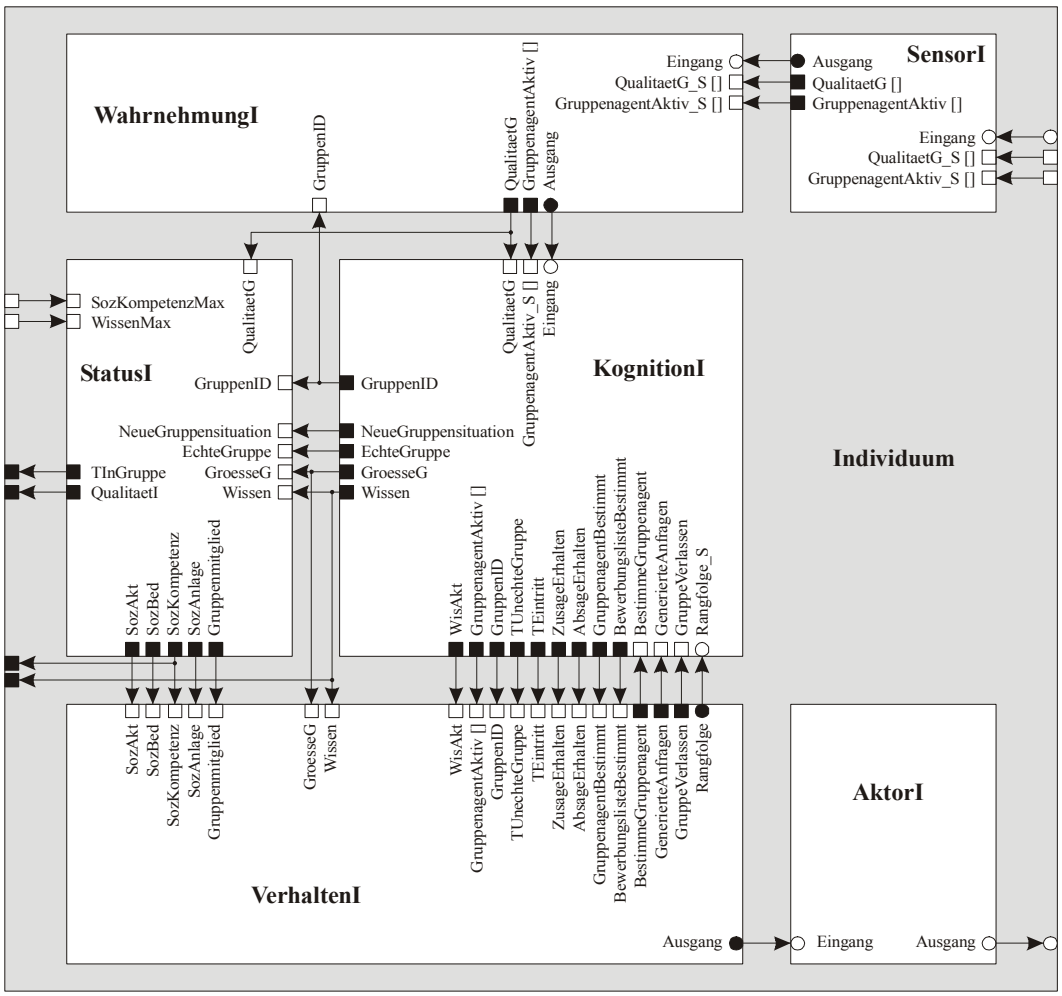

Abbildung 9.22: Der grundlegende Aufbau der Individualagenten

9.2.2.1 Die Komponente *SensorI*

Die Komponente *SensorI* dient der Annahme von Informationen, die von außen an ein Individuum herangetragen werden. Es handelt sich dabei um die zu einem gegebenen Zeitpunkt aktiven Lerngruppen sowie deren Qualität. Beide Informationen werden auf dem Blackboard der Komponente *Connector* bereitgestellt. Zudem nimmt die Komponente *SensorI* Nachrichten der Gruppenagenten entgegen, die ebenfalls über die Komponente *Connector* vermittelt werden. Alle eingehenden Informationen werden unverändert an die Komponente *WahrnehmungI* weitergeleitet.

9.2.2.2 Die Komponente *WahrnehmungI*

Innerhalb der Komponente *WahrnehmungI* findet eine Filterung der Informationen statt, die über die Komponente *SensorI* aus der Umgebung aufgenommen wurden. So wird etwa nicht für alle vorhandenen Lerngruppen deren Qualität zur weiteren Verarbeitung an die internen Komponenten des Agenten weitergeleitet, sondern nur für diejenige Lerngruppe, bei der der betrachtete Agent als Mitglied verzeichnet ist. Die Identifikation der betreffenden Lerngruppe erfolgt dabei mit Hilfe der Modellgröße *GruppenID*, die aus der Komponente *KognitionI* bezogen wird. Die eingehenden Nachrichten, sowie Informationen über aktive Lerngruppen werden den Komponenten *KognitionI* und *StatusI* unverändert zur weiteren Bearbeitung zur Verfügung gestellt.

9.2.2.3 Die Komponente *KognitionI*

Aufgabe und grundlegender Aufbau der Komponente KognitionI

Die Komponente *KognitionI* erfüllt im Rahmen der Agentenarchitektur des Modells *Lerngruppe* eine Reihe von Aufgaben, die im folgenden aufgelistet sind:

- Bereitstellung des kognitiven Persönlichkeitsmerkmals *Intelligenz* eines Individuums
- Berechnung der kognitiven Zustandsvariablen *Wissen* und *WissenAkt* eines Individuums sowohl in der Einzel- als auch in der Gruppenlernphase
- Speicherung und Aktualisierung von Informationen eines Individuums über sich selbst und seine Umgebung
- Ausführung von Denkprozessen als Folge interner Aktionen

Um diese Aufgaben erfüllen zu können, verfügt die Komponente *KognitionI* sowohl über Verbindungen nach außen, mit deren Hilfe auf Modellgrößen, die in anderen Komponenten verwaltet werden, zugegriffen werden kann, als auch über eigene Modellelemente.

Eingangselemente der Komponente *KognitionI*. Die Komponente *KognitionI* steht sowohl mit der Komponente *WahrnehmungI* als auch mit der Komponente *VerhaltenI* in Verbindung. Durch die Komponente *WahrnehmungI* wird die Komponente *KognitionI* über aktive Gruppenagenten (*GruppenagentAktiv_S []*), im Falle der Zugehörigkeit zu einer Lerngruppe über deren Qualität (*QualitaetG*), sowie über eingetroffene Nachrichten, die in der Eingangslocation *Eingang* bereit gestellt werden, informiert. Zudem erhält die Komponente *KognitionI* von der Komponente *VerhaltenI* mit Hilfe der Modellgröße *GenerierteAnfragen* die Information, wie viele Anfragen an Gruppenagenten hinsichtlich ihrer Qualität im Rahmen des Eintritts in eine neue Gruppenlernphase verschickt wurden. Diese Information wird dazu verwendet, um bei der Verarbeitung der eingegangenen

Nachrichten überprüfen zu können, ob die Rückmeldungen aller Gruppenagenten vollständig vorliegen. Zudem meldet die Komponente *VerhaltenI* der Komponente *KognitionI*, wenn eine Lerngruppe verlassen wurde. Auf Grundlage dieser Information aktualisiert die Komponente *KognitionI* die Zustandsvariablen innerhalb der Wissensbasis, die im Zusammenhang mit der Gruppenzugehörigkeit des Individuums stehen (*GroesseG*, *GruppenID*) und setzt das Wissen der aktuellen Lernphase auf den Initialwert zu Beginn einer neuen Lernphase zurück.

Grundlegende Modellelemente der Komponente *KognitionI*. Innerhalb der Komponente *KognitionI* wird eine Reihe von Modellelementen verwaltet, die verschiedenen Kategorien zugeordnet werden können. Zunächst einmal wird innerhalb der Komponente *KognitionI* das Persönlichkeitsmerkmal *Intelligenz* eines Individuums bestimmt. Die Intelligenz eines Individuums bleibt über die Dauer eines Simulationslaufes hinweg konstant und wird zu Beginn mit Hilfe eines Zufallsprozesses bestimmt. Darüber hinaus stellt die Komponente *KognitionI* die kognitiven Zustandsvariablen *Wissen* und *WisAkt* mit einer entsprechenden Dynamikbeschreibung für die Einzel- und die Gruppenlernphase zur Verfügung. Umweltinformationen sowie Informationen, die das Individuum über sich selbst benötigt, werden ebenso in der Komponente *KognitionI* verwaltet. Hierbei handelt es sich im wesentlichen um Informationen zur aktuellen Gruppensituation (*GruppenID*, *GroesseG*, *echteGruppe*, *TEintritt*, *QualitaetG*), falls das Individuum zum gegeben Zeitpunkt Mitglied einer Gruppe ist, sowie um Informationen, die ein Individuum benötigt, um einer Gruppe beizutreten bzw. diese zu gründen (*GruppenagentAktiv*, *Rangfolge*). Die Modellgröße *GruppenagentAktiv* beinhaltet dabei alle zum betrachteten Zeitpunkt aktiven Lerngruppen. Innerhalb der Rangfolge werden die Bewerbungsanfragen bei bestehenden Lerngruppen vorab abgelegt und hinsichtlich ihrer Präferenz sortiert, so daß die nächste Anfrage immer an diejenige Gruppe mit der höchsten Präferenz gerichtet wird.

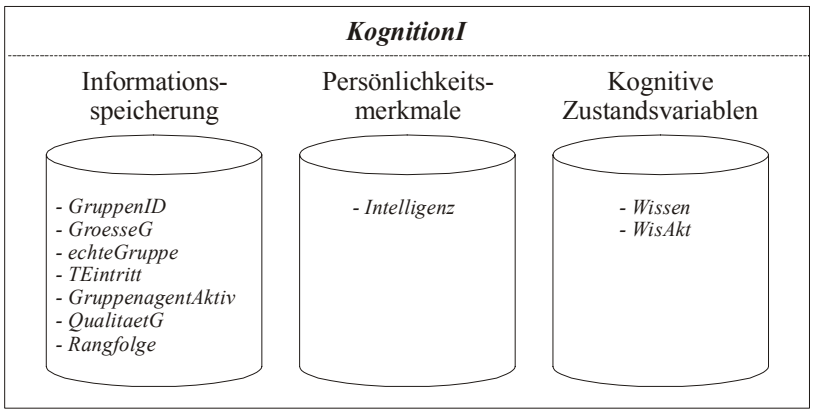

Abbildung 9.23: Der grundlegenden Modellelemente der Komponente KognitionI

Die grundlegende Arbeitsweise der Komponente KognitionI

Zur Beschreibung des dynamischen Verhaltens ihrer Modellelemente bedient sich die Komponente *KognitionI* sowohl kontinuierlicher als auch diskreter Mechanismen. Die zeitliche Veränderung des aktuellen Wissens sowie des Gesamtwissens eines Individuums wird auf Grundlage zweier Differentialgleichungen spezifiziert, deren Struktur den Vorgaben aus Gleichung Gl. 9.6 folgt. Die Unterscheidung des dynamischen Verhaltens zwischen der Einzel- und der Gruppenlernphase wird dabei dadurch gelöst, daß der Faktor *QualitaetG* im Falle der Einzellernphase mit dem Wert 1 belegt ist und sich damit neutral auf das Ergebnis der Gleichung auswirkt, während er in der Gruppenlernphase die tatsächliche Qualität der Gruppe enthält.

Im Rahmen der diskreten Modelldynamik wird die Komponente *KognitionI* sowohl durch die Komponente *WahrnehmungI* als auch durch die Komponente *VerhaltenI* aktiviert. Die Komponente *WahrnehmungI* stellt in ihrer Location *Ausgang* Nachrichten zur weiteren Verarbeitung innerhalb der Komponente *KognitionI* zur Verfügung. Die Komponente *KognitionI* greift diese Nachrichten über die Eingangslocation *Eingang* ab und löst daraufhin verschiedene Bearbeitungsmechanismen aus, die im wesentlichen von den Typen der eingegangenen Nachrichten abhängen.

Die Bearbeitung von Nachrichten des Typs *Auskunft*. Im Rahmen der Bestrebung eines Agenten, einer Lerngruppe beizutreten, werden zunächst einmal Informationen von allen aktiven Gruppenagenten eingeholt. Die Rückmeldung der Gruppenagenten erfolgt in Form von Nachrichten des Typs *Auskunft*. Diese Nachrichten werden zunächst gesammelt bis alle Rückmeldungen vollzählig eingetroffen sind. Liegen alle Rückmeldungen vor, werden die Gruppen anhand ihrer Qualität in eine Reihenfolge gebracht. Auf diese Weise entsteht die Bewerbungsliste der Agenten, aus der die Komponente *VerhaltenI* im Rahmen des Bewerbungsverfahrens jeweils die als nächstes anzufragende Lerngruppe bezieht.

Die Bearbeitung von Nachrichten des Typs *Zusage*. Hat sich ein Agent erfolgreich bei einer Lerngruppe beworben, meldet diese die Aufnahme in die Lerngruppe mit Hilfe einer Nachricht des Typs *Zusage* an den Agenten zurück. Nach Erhalt einer Zusage wird zunächst einmal der Eintrittszeitpunkt in die Lerngruppe (*TEintritt*) protokolliert. Diese Information ist im Rahmen der Verhaltenssteuerung erforderlich, um festzustellen, wann ein Agent nach Aufrechterhaltung einer neu gegründeten Gruppe diese Gruppe wieder auflösen darf, sofern sich kein weiteres Individuum angeschlossen hat. Darüber hinaus wird das aktuelle Wissen des Individuums auf den Initialwert zu Beginn einer neuen Lernphase zurückgesetzt. Im Informationspool des Agenten wird die aktuelle Gruppen-ID sowie die Größe der Gruppe hinterlegt. An die Komponente *StatusI* ergeht die Information, dass

eine neue Gruppensituation vorliegt, um dort die Mitgliedschaft in einer Gruppe zu dokumentieren. Abschließend kann die erarbeitete Bewerbungsliste gelöscht werden, da nach Aufnahme in eine Gruppe keine weiteren Anfragen an andere Gruppen mehr erforderlich sind.

Die Bearbeitung von Nachrichten des Typs *Absage*. Erhält ein Agent auf eine Bewerbung bei einer Gruppe hin eine Absage, wird er eine weitere Bewerbung an diejenige Gruppe mit der nächst niedrigeren Präferenz schicken. Dieser Vorgang wird in der Komponente *VerhaltenI* durchgeführt und durch die Analyse einer Nachricht des Typs *Absage* in der Komponente *KognitionI* ausgelöst.

Die Bearbeitung von Nachrichten des Typs *Ausschluss*. Mit Hilfe einer Nachricht des Typs *Ausschluss* informieren Gruppenagenten Individuen über den Ausschluß aus der Lerngruppe. Innerhalb der Individualagenten wird daraufhin eine Aktualisierung der Zustandsvariablen *GroesseG*, *GruppenID* sowie der kognitiven Zustandsvariable *WisAkt* erforderlich, die in unmittelbarem Zusammenhang mit dem Wechsel der Lernphase stehen. Ebenso wird innerhalb der Komponente *StatusI* der Status der Gruppenmitgliedschaft auf den Wert der Einzellernphase zurückgesetzt.

Die Bearbeitung von Nachrichten des Typs *Update*. Nachrichten des Typs *Update* werden in den Gruppenagenten verwendet, um die Mitglieder der Gruppe über eine Veränderung der Gruppengröße zu informieren. Entsprechend findet bei der Auswertung einer derartigen Nachricht innerhalb der Komponente *KognitionI* eine Aktualisierung der Modellgröße *GroesseG* auf die vorliegende Gruppengröße sowie eine Anpassung der vorliegenden Gruppensituation statt.

Die Aktualisierung des Informationspools. Die Komponente *WahrnehmungI* stellt permanent Informationen über aktive Gruppenagenten sowie im Falle einer Gruppenlernphase auch Informationen über die aktuelle Qualität der Gruppe, in die ein Agent integriert ist, bereit. Diese Informationen werden innerhalb der Komponente *KognitionI* unmittelbar und unverändert zur Aktualisierung des Informationspools des Agenten verwendet. Darüber hinaus liefert die Komponente *VerhaltenI* an die Komponente *KognitionI* eine Information, wenn der Agent im Rahmen seiner Verhaltenssteuerung beschlossen hat, freiwillig eine Gruppe zu verlassen. In dieser Situation ist innerhalb der Komponente *KognitionI* eine Anpassung der Modellgrößen *GroesseG*, *GruppenID* sowie der kognitiven Zustandsvariable *WisAkt* erforderlich, so dass diese wieder die Werte, die zu Beginn einer neuen Einzellernphase zutreffen, erhalten.

Die Ausführung der internen Aktion *BestimmeGruppenagent*. Stellt ein Agent im Rahmen eines Bewerbungsverfahrens fest, dass ein Betritt in eine bestehende Gruppe unmöglich ist, wird er versuchen, eine neue Gruppe zu gründen. Hierzu muß zunächst einmal ermittelt werden, welche Gruppe zum betrachteten Zeitpunkt inaktiv ist, d. h. welcher Gruppenagent für eine erneute Aktivierung zu Verfügung steht. Diese Information kann anhand der Modellgröße *GruppenagentAktiv* der Komponente *KognitionI* ermittelt werden. Ausgelöst wird die Bestimmung eines freien Gruppenagenten innerhalb der Komponente *KognitionI* durch die Komponente *VerhaltenI* unter Verwendung der internen Aktion *BestimmeGruppenagent*. Das Ergebnis der Berechnung steht der Komponente *VerhaltenI* dann für die Generierung einer entsprechenden Bewerbung in der Eingangsgröße *GruppenID* zur Verfügung.

9.2.2.4 Die Komponente *StatusI*

Aufgabe und grundlegender Aufbau der Komponente StatusI

Die Komponente *StatusI* stellt die sozialen Attribute, die einem Individuum im Modell *Lerngruppe* zugeordnet werden, bereit. Es handelt sich dabei um die soziale Veranlagung eines Individuums, die soziale Zufriedenheit, die Sozialkompetenz, die das Individuum durch die Interaktion mit anderen Individuen in Lerngruppen erwirbt, das Bedürfnis nach Sozialkontakten, sowie die Qualität des Individuums, auf deren Grundlage alle Individuen der Population hinsichtlich ihrer insgesamt erworbenen Fähigkeiten in eine Rangfolge gebracht werden können.

Eingangselemente der Komponente *StatusI*. Zur Bestimmung der sozialen Attribute eines Individuums sind Modellgrößen erforderlich, die außerhalb der Komponente *StatusI* verwaltet werden. Die Komponente *StatusI* steht aus diesem Grund mit den Komponenten *WahrnehmungI* und *KognitionI* der Agentenarchitektur, sowie mit der externen Komponente *Statistik* in Verbindung. Aus der Komponente *WahrnehmungI* wird im Falle einer Gruppenlernphase die Qualität der aktuellen Lerngruppe bezogen, um die Entwicklung der sozialen Zufriedenheit des Individuums adäquat berechnen zu können. Die Komponente *KognitionI* liefert gegebenenfalls die aktuelle Größe der Lerngruppe mit dem Ziel der Feststellung, ob eine echte Gruppe, d. h. eine Gruppe mit mindestens zwei Mitgliedern, besteht. Ebenso wird als maßgeblicher Einflußfaktor auf die Bestimmung der Qualität des Individuums die Modellgröße *Wissen* aus der Komponente *KognitionI* gelesen. Als Information über die Gesamtpopulation stellt schließlich die Komponente Statistik die aktuellen Maximalwerte für das Gesamtwissen sowie die soziale Kompetenz als weitere Parameter für die Berechnung der Qualität des Individuums bereit.

Die grundlegende Arbeitsweise der Komponente StatusI

Die Berechnung der sozialen Attribute eines Agenten beruht im wesentlichen auf einer Menge von Differentialgleichungen und algebraischen Gleichungen, die in der Komponente *StatusI* zur Verfügung gestellt werden. Die einzige Ausnahme bildet dabei die Modellgröße *SozAnlage*, die als Persönlichkeitsmerkmal eines Individuums über die gesamte Dauer eines Simulationslaufes hinweg als unveränderlich angenommen wird. Die soziale Veranlagung eines Individuums wird zu Beginn eines Simulationslaufes auf der Grundlage einer Zufallsvariable einmalig diskret belegt.

Die Berechnung der sozialen Zufriedenheit innerhalb der Komponente *StatusI* folgt den Differentialgleichungen Gl. 9.3 und Gl. 9.8. der Modellbeschreibung. Unmittelbar abhängig von der sozialen Zufriedenheit eines Individuums ist dessen Bedürfnis nach Sozialkontakten. Die Berechnung der Stärke für dieses Bedürfnis erfolgt auf der Grundlage einer algebraischen Gleichung (s. Gl. 9.5 und Gl. 9.10), in die die aktuelle soziale Zufriedenheit des Individuums als wesentlicher Parameter mit eingeht. Die soziale Kompetenz eines Individuums kann nur während einer Gruppenlernphase gesteigert werden. Der Anstieg der sozialen Kompetenz wird dabei als linear angenommen und entsprechend den Vorgaben der Gleichung Gl. 9.11 modelliert. Gleichzeitig mit dem Eintritt in eine neue Gruppenlernphase wird aus statistischen Gründen mit protokolliert, wie lange sich ein Individuum bis zum betrachteten Zeitpunkt bereits in Lerngruppen aufgehalten hat. Ebenso wird innerhalb der Komponente *StatusI* die Qualität des Individuums kontinuierlich neu berechnet. Hierfür findet eine algebraische Gleichung nach dem Schema der Gleichung Gl. 9.12 Verwendung, sowie die Größen *Wissen*, *WissenMax* und *SozKompetenzMax*, die aus den Komponenten *KognitionI* sowie *Statistik* gelesen werden.

9.2.2.5 Die Komponente *VerhaltenI*

In der Komponente *VerhaltenI* werden die grundlegenden Verhaltensweisen, mit denen ein Agent im Modell *Lerngruppe* ausgestattet ist, ausgewählt und zur Ausführung durch die Komponente *AktorI* angeordnet. Die Verhaltenssteuerung basiert auf einer Menge vordefinierter Regeln, die in der Komponente *VerhaltenI* hinterlegt sind. In den Bedingungsteil dieser Regeln gehen eigene Modellgrößen der Komponente *VerhaltenI*, sowie externe Modellgrößen der Komponenten *KognitionI* und *StatusI* ein (s. Abbildung 9.22). Auszuführende Aktionen werden der Komponente *AktorI* mit Hilfe von mobilen Komponenten des Typs *AusfuehrungsAOI* (Ausführungsanordnung eines Individuums) mitgeteilt, wobei innerhalb der Ausführungsanordnungen bei Bedarf Parameter mitgeliefert werden, die die Ausführungsanordnung näher spezifizieren. Grundsätzlich wird zwischen den drei Arten *Anfrage_verschicken*, *Bewerbung_verschicken* und *Gruppe_verlassen* von Ausführungsanordnungen unterschieden, die in den entsprechenden Situationen im Rahmen der Verhaltenssteuerung zur Ausführung ausgewählt werden.

Die Verhaltenssteuerung eines Individualagenten zerfällt grundsätzlich in zwei Abschnitte. Der erste Abschnitt beschäftigt sich mit Verhaltensweisen im Zusammenhang mit dem Eintritt in eine Gruppenlernphase. Der zweite Abschnitt spezifiziert Aktivitäten im Rahmen des Austritts aus einer Gruppenlernphase.

Die Verhaltenssteuerung im Zusammenhang mit dem Eintritt in eine Gruppenlernphase. Mit Erfüllung der in Bed. 9.1 angegebenen Bedingung wird innerhalb der Komponente *VerhaltenI* und gegebenenfalls auch im Zusammenspiel mit der Komponente *KognitionI* eine Kette von Ereignissen angestoßen, die das Ziel verfolgen, eine Gruppenlernsituation herbeizuführen. Der Agent tritt in eine Anfragephase ein. Hierbei wird zunächst überprüft, ob zum aktuellen Zeitpunkt Lerngruppen vorhanden sind. Ist dies der Fall, so fordert die Komponente *VerhaltenI* mit Hilfe einer entsprechenden Anzahl an Ausführungsanordnungen der Kategorie *Anfrage_verschicken* die Komponente *AktorI* dazu auf, an alle aktiven Gruppenagenten eine Anfrage hinsichtlich ihrer Qualität zu stellen. Zudem meldet die Komponente *VerhaltenI* an die Komponente *KognitionI* die Anzahl an abgeschickten Anfragen zur Überprüfung der Vollständigkeit der eingehenden Antworten.

```
# Ereignis zur Erzeugung einer Ausführungsanordnung des
# Typs Bewerbung_verschicken
# ---------------------------------
WHENEVER Bewerben AND NUMBER(Rangfolge) > 0
DO
   Ausgang^ : ADD 1 NEW AusfuehrungsAOI
      CHANGING
         Typ^          := 'Bewerbung_verschicken';
         ID^           := Rangfolge:Gruppe[1].ID;
         Wissen^       := Wissen;
         SozKompetenz^ := SozKompetenz;
      END
   Bewerben^ := FALSE;
END
```

Abbildung 9.24: Das Ereignis zur Erzeugung einer Ausführungsanordnung des Typs Bewerbung_verschicken

Nachdem alle Antworten eingetroffen sind und die Präferenzordnung der Gruppen hergestellt wurde, beginnt für den Agenten die Bewerbungsphase. Hierzu veranlasst die Komponente *VerhaltenI* mit Hilfe einer Ausführungsanordnung der Kategorie *Bewerbung_verschicken*, daß an den betreffenden Gruppenagenten eine Bewerbung um Aufnahme in die Gruppe geschickt wird. Der Ausführungsanordnung werden dabei das aktuelle Wissen sowie die aktuelle soziale Kompetenz des Agenten als Parameter mitgegeben. Zudem enthält die Ausführungsanordnung die Identifikation der Gruppe, an die die Bewerbung gerichtet ist, damit im Rahmen der Zustellung der Nachricht der korrekte Empfänger festgestellt werden kann. Das Codebeispiel in Abbildung 9.24 zeigt das Ereignis, das innerhalb

der Komponente *VerhaltenI* für die Erzeugung einer Ausführungsanordnung des Typs *Bewerbung_verschicken* zuständig ist.

Auf eine Bewerbung hin erhält ein Individualagent von der Gruppe entweder eine Zusage oder ein Absage. Im Falle einer Zusage werden alle Informationen in der Bewerbungsliste gelöscht, da diese nicht mehr länger benötigt werden, und eine neue Gruppenlernphase mit Initialisierung der entsprechenden Modellgrößen eingeleitet. Ist hingegen eine Absage eingetroffen, wird zunächst das erste Element der Bewerbungsliste, an das die letzte Bewerbung ohne Erfolg geschickt wurde, entfernt und dann mit Hilfe des oben dargestellten Mechanismus eine weitere Bewerbung an die nächste, in Frage kommende Gruppe auf den Weg gebracht.

Liegt im Falle einer Absage oder auch grundsätzlich keine Gruppe in der Bewerbungsliste vor, so wird im Zusammenspiel mit der Komponente *KognitionI* ein freier Gruppenagent bestimmt (interne Aktion *BestimmeGruppenagent*). Durch Zusendung einer Bewerbungsanfrage wird dieser Gruppenagent dann aktiviert, so daß auf diesem Wege eine neue Gruppe gegründet wird.

Die Verhaltenssteuerung im Zusammenhang mit dem Austritt aus einer Gruppenlernphase. Den freiwilligen Austritt aus einer regulären Gruppe beschließt ein Agent nach Maßgabe der Bedingung Bed. 9.2 oder wenn sich innerhalb einer gewissen Zeit nach Gründung einer neuen Gruppe kein weiteres Mitglied eingefunden hat (s. Bed. 9.3). Tritt eine dieser beiden Situationen ein, so wird im Rahmen der Verhaltenssteuerung eines Agenten eine Ausführungsanordnung des Typs *Gruppe_verlassen* erzeugt und mit Hilfe der Komponenten *AktorI* und *Connector* zur weiteren Bearbeitung an den betreffenden Gruppenagenten übermittelt. Zusätzlich ergeht eine Information über die neue Situation an die Komponente *KognitionI*, damit dort die entsprechenden Modellgrößen an die neue Situation angepasst werden können.

9.2.2.6 Die Komponente *AktorI*

Die Hauptaufgabe der Komponente *AktorI* besteht darin, die Ausführungsanordnungen, die im Rahmen der Verhaltenssteuerung durch die Komponente *VerhaltenI* erzeugt werden, tatsächlich zur Ausführung zu bringen. Im Falle des Modells *Lerngruppe* dienen alle externen Aktionen, die ein Individualagent auszuführen in der Lage ist, der Interaktion mit Gruppenagenten. Aus diesem Grund beschränkt sich die Funktionalität, die innerhalb der Komponente *AktorI* zur Verfügung gestellt werden muß, auf den Empfang von Ausführungsanordnungen aus der Komponente *VerhaltenI* sowie deren Umsetzung in Nachrichten, die anschließend mit dem Ziel der Weiterleitung an die entsprechenden Gruppenagenten an die Komponente *Connector* delegiert werden.

Als grundlegende Modellelemente weist die Komponente *AktorI* zwei Warteschlangen *Eingang* und *Ausgang* auf. Innerhalb der Warteschlange *Eingang* nimmt die Komponente *AktorI* Ausführungsanordnungen der Komponente *VerhaltenI* entgegen. In der Warteschlange *Ausgang* werden erzeugte Nachrichten abgelegt und von dort aus durch die Komponente *Connector* zur weiteren Verarbeitung abgezogen.

Die Komponente *AktorI* nimmt ihre Tätigkeit auf, sobald in der Warteschlange *Eingang* neue Ausführungsanordnungen vorhanden sind. Grundsätzlich werden im Zuge der Verarbeitung der Ausführungsanordnungen neue Nachrichten, d. h mobile Komponente der Klasse *Nachricht*, generiert, wobei für die Spezifikation sowohl des Typs als auch der Inhalte der Nachrichten die Informationen verwendet werden, die den Ausführungsanordnungen als Parameter angefügt sind. Die neu generierten Nachrichten werden anschließend ohne Verzug in den Nachrichtenausgang gestellt und somit nach außen hin verfügbar gemacht. Einzige Ausnahme bildet hierbei die Abwicklung von Anfragen an Gruppenagenten hinsichtlich ihrer aktuellen Gruppenqualität. In diesem Fall werden zunächst einmal für alle Ausführungsanordnungen der Komponente *VerhaltenI* entsprechende Nachrichten des Typs *Anfrage_verschicken* erzeugt und intern zwischengespeichert. Erst dann, wenn alle zugehörigen Ausführungsanordnungen vollständig abgearbeitet sind, werden die generierten Nachrichten gleichzeitig im Nachrichtenausgang zur Verfügung gestellt. Für die Generierung und Zwischenspeicherung dieser Anfragen wird innerhalb der Komponente *AktorI* eine weitere, nur intern sichtbare Warteschlange *Nachrichten* verwendet.

Alle Ausführungsanordnungen werden grundsätzlich unmittelbar nach deren Abarbeitung in der Komponente *AktorI* vernichtet.

Hinsichtlich der Konzeption von Aktionen bleibt anzumerken, daß im Modell *Lerngruppe* aus Gründen der Einfachheit die Ausführung von Aktionen mit keinen zeitverbrauchenden Vorgängen verbunden ist. Die Kommunikation der Agenten untereinander ist also ohne jeglichen Zeitverzug vorgesehen. Entsprechende Mechanismen können jedoch aufgrund des komponentenorientierten Ansatzes durch eine lokale Modifikation der Komponente *AktorI* in einfacher Weise eingeführt werden.

9.2.3 Die Architektur der Gruppenagenten

Gruppenagenten repräsentieren im vorliegenden Modellierungsansatz die Lerngruppen, zu denen sich die Individuen bei Bedarf zusammenschließen. Sie sind, ebenso wie die Individualagenten, nach dem Vorbild von *PECS*-Agenten konzipiert und stellen in dieser Hinsicht hierarchisch strukturierte Modellkomponenten dar. Auf der übergeordneten Ebene befindet sich die Strukturkomponente *Gruppenagent*, die auf der darunter liegenden Hierarchieebene die Subkomponenten *SensorG*, *WahrnehmungG*, *KognitionG*, *StatusG*, *VerhaltenG* und *AktorG* instanziiert und miteinander verbindet.

Wesentlich für die Funktionsfähigkeit der Gruppenagenten sind hierbei die Komponenten *KognitionG*, *StatusG* und *VerhaltenG*. Die internen Komponenten *KognitionG* und *StatusG* aggregieren die kognitiven und sozialen Eigenschaften der einzelnen Gruppenmitglieder und stellen eigenständige Modellgrößen, wie etwa die Qualität der Lerngruppe, die eine Gruppe als Ganzes auszeichnen, nach außen sowohl zu den Gruppenmitgliedern als auch zur Komponente *Statistik* hin zur Verfügung. Die Komponente *VerhaltenG* realisiert die Verhaltenssteuerung der Gruppenagenten, die stellvertretend für die einzelnen Gruppenmitglieder bestimmte Aufgaben zu erfüllen haben. Hierzu zählen etwa die Reaktion auf Nachrichten der Individuen, die Entscheidung über die Aufnahme neuer Gruppenmitglieder sowie die Überwachung der Gruppenqualität mit der Option, bei Bedarf bestehende Gruppenmitglieder aus der Gruppe auszuschließen.

Die Komponenten *SensorG* und *WahrnehmungG* dienen in Analogie zu den Individualagenten der Aufnahme von Informationen aus der Umgebung und leiten diese Informationen unverändert an die internen Komponenten des Gruppenagenten weiter. Ebenso erfüllt auch in den Gruppenagenten die Komponente *AktorG* die Aufgabe, im Rahmen der Verhaltenssteuerung festgelegte Ausführungsanordnungen in entsprechende Nachrichten an die betroffenen Individuen umzusetzen. Aufgrund der großen Ähnlichkeit, die die Funktionalitäten der einander entsprechenden Komponenten in den Individualagenten und den Gruppenagenten aufweisen, soll an dieser Stelle auf eine erneute Detaildiskussion der Komponenten *SensorG*, *WahrnehmungG* und *AktorG* im Zusammenhang mit den Gruppenagenten verzichtet werden. Die folgenden Abschnitte beschäftigen sich daher in ausführlicherer Weise nur mit den Komponenten *KognitionG*, *StatusG* und *VerhaltenG*.

9.2.3.1 Die Komponente *KognitionG*

Aufgabe und grundlegender Aufbau der Komponente KognitionG

Die Komponente *KognitionG* erfüllt im wesentlichen die folgenden Aufgaben:

- Bereitstellung elementarer Informationen über die Zusammensetzung der Lerngruppe sowie wesentliche Eigenschaften der Gruppenmitglieder
- Berechnung der Gruppenqualität
- Auswertung von Nachrichten der Individualagenten

Um diese Aufgaben erfüllen zu können, verfügt die Komponente *KognitionG* sowohl über Verknüpfungen zu anderen Komponenten als auch über eigene Modellgrößen.

Eingangselemente der Komponente *KognitionG*. Die Komponente *KognitionG* bezieht Eingangsinformationen von den Komponenten *WahrnehmungG*, *Statistik*

und *VerhaltenG*. Über die Komponente *WahrnehmungG* werden Informationen zur weiteren Verarbeitung an die Komponente *KognitionG* weitergeleitet, die außerhalb der Agenten in der Komponente *Connector* bereitgestellt werden. Hierzu zählen insbesondere die Nachrichten der Individualagenten an die Gruppenagenten sowie das Wissen und die soziale Kompetenz der Gruppenmitglieder als wesentliche Einflüsse auf die Gruppenqualität. Von der Komponente *Statistik* werden statistische Informationen über die Gesamtpopulation eingeholt, die ebenso als Parameter in die Bestimmung der Gruppenqualität mit eingehen. Die Komponente *VerhaltenG* liefert Rückmeldungen über zur Ausführung angeordnete Aktionen an die Komponente *KognitionG* zurück, um den Informationspool des Gruppenagenten an die neu entstandene Situation anzupassen.

Grundlegende Modellelemente der Komponente *KognitionG.* Die Komponente *KognitionG* stellt wesentliche Verwaltungsinformationen über die dem Gruppenagenten zugeordnete Lerngruppe bereit. Ausschlaggebend sind hierbei die Mitgliedsliste, in der die aktuellen Mitglieder der Lerngruppe verzeichnet sind, sowie Listen, die die wesentlichen Bestimmungsmerkmale Wissen und soziale Kompetenz aller Gruppenmitglieder enthalten. Darüber hinaus ist in der Komponente *KognitionG* permanent die Größe der Lerngruppe abzulesen.

Die grundlegende Arbeitsweise der Komponente KognitionG

Die Komponente *KognitionG* verfügt sowohl über zeitkontinuierliche als auch zeitdiskrete Zustandsübergänge. Im Rahmen ihrer kontinuierlichen Dynamik erfolgt die Aktualisierung der Gruppenqualität. Hierbei ist zu unterscheiden, ob zum betrachteten Zeitpunkt eine echte Gruppe, d. h. eine Gruppe mit mehr als einem Mitglied, vorliegt, oder ob die Gruppe nach Ausstieg eines bestehenden Mitglieds oder etwa nach einer Neugründung nur mit einem Mitglied besetzt ist. Im Falle einer echten Gruppe mit mindestens zwei Mitgliedern folgt die Berechnung der Gruppenqualität der Gleichung Gl. 9.16. Im Falle einer Gruppe mit nur einem Mitglied erhält die Gruppenqualität den Wert 1 und wirkt somit neutral auf das betreffende Individuum ein, als befände es sich immer noch in einer Einzellernsituation. Um die Berechnung der Gruppenqualität adäquat durchführen zu können, sind in der Komponente *KognitionG* eine Reihe von Funktionen hinterlegt, die auf den Daten der Gruppenmitglieder aufsetzen und entsprechende statistische Werte generieren. Ebenso stellt die Komponente *KognitionG* die Tabellenfunktion (s. Abbildung 9.17) bereit, die den Einfluß der Gruppengröße auf die Qualität der Lerngruppe beschreibt.

Im Rahmen ihrer zeitdiskreten Dynamik wird die Komponente KognitionG sowohl durch die Komponente WahrnehmungG als auch durch die Komponente VerhaltenG aktiviert. Trifft in der Eingangslocation Eingang aus der Komponente WahrnehmungG eine neue Nachricht eines Individualagenten ein, so schalten in

der Komponente KognitionG unmittelbar eine Reihe von Ereignissen, die sich mit der Auswertung dieser Nachrichten beschäftigen. Die resultierenden Aktivitäten sind dabei nach dem Typ der eingetroffenen Nachricht zu unterscheiden.

Die Bearbeitung von Nachrichten des Typs *Anfrage*. Trifft im Rahmen des Bewerbungsverfahrens eines Individualagenten eine Nachricht des Typs *Anfrage* ein, so legt der Gruppenagent in der Komponente *KognitionG* zunächst die Identifikation des Bewerbers zur späteren Verwendung ab und signalisiert der Komponente *VerhaltenG*, daß eine externe Anfrage existiert, die beantwortet werden muß.

Die Bearbeitung von Nachrichten des Typs *Bewerbung*. Ebenso wie im Falle einer Anfrage wird auch bei einer Bewerbung eines Individualagenten zunächst in der Komponente *KognitionG* dessen Identifikation abgelegt. Anschließend werden sofort die wesentlichen Kriterien für die Aufnahme des Bewerbers in die Gruppe bestimmt. Hierzu erfolgt eine Überprüfung der aktuellen Gruppengröße. Wird dabei festgestellt, daß die maximale Gruppengröße noch nicht erreicht ist, hängt die Aufnahme des Bewerbers davon ab, wie sich die Gruppenqualität nach dessen Aufnahme verändern wird. Um diesen Einflußfaktor greifbar zu machen, wird die Gruppenqualität hypothetisch neu bestimmt unter der Annahme, daß der Bewerber in die Gruppe aufgenommen wurde. Dadurch ergibt sich eine potentielle, neue Gruppenqualität, die zusammen mit der Information, dass eine neue Bewerbung vorliegt, im nächsten Schritt zum Zweck der Entscheidungsfindung an die Komponente *VerhaltenG* weitergeleitet wird (vgl. hierzu auch Abschnitt 9.1.2.3).

Die Bearbeitung von Nachrichten des Typs *Ausstieg*. Mit Hilfe einer Nachricht des Typs *Ausstieg* gibt ein Individualagent seinen Entschluss, die Gruppe zu verlassen, bekannt. Innerhalb der Komponente *KognitionG* des Gruppenagenten wird daraufhin das Mitgliederverzeichnis um den entsprechenden Individualagenten gekürzt und die Komponente *VerhaltenG* darüber informiert, daß eine neue Gruppenzusammensetzung entstanden ist.

Die Bearbeitung von Rückmeldungen aus der Komponente *VerhaltenG*. Die Komponente *KognitionG* steht in enger Wechselwirkung mit der Komponente *VerhaltenG*. So liefert die Komponente *VerhaltenG* einerseits Rückmeldungen über getroffene Entscheidungen zur Aktualisierung der Wissensbasis an die Komponente *KognitionG*. Andererseits erfordern Entscheidungen, die in der Kompo-

nente *VerhaltenG* getroffen werden, Vorarbeiten innerhalb der Komponente *KognitionG*.

Wird in der Komponente *VerhaltenG* die Entscheidung getroffen, daß ein Bewerber in die Gruppe aufgenommen wird, wird die Komponente *KognitionG* darüber in Kenntnis gesetzt, um daraufhin das Mitgliederverzeichnis zu aktualisieren. Zum selben Zweck erhält die Komponente *KognitionG* auch eine entsprechende Information, wenn eine Entscheidung über den proaktiven Ausschluss eines Gruppenmitglieds getroffen wurde.

Entscheidungsunterstützend wirkt die Komponente *KognitionG* aber bereits im Vorfeld des Ausschlusses eines bestehenden Gruppenmitglieds. Hierbei wird in der Komponente *KognitionG* dasjenige Gruppenmitglied bestimmt, das die Gruppenqualität am stärksten negativ beeinflusst. Dieses Gruppenmitglied wird dann zusammen mit der aus dem Ausschluss resultierenden, neuen Gruppenqualität an die Komponente *VerhaltenG* gemeldet.

9.2.3.2 Die Komponente *StatusG*

Die Komponente *StatusG* verwaltet den Aktivierungsstatus eines Gruppenagenten. Mit Beitritt des ersten Gruppenmitglieds wird ein Gruppenagent in den Status *aktiv* versetzt. Verlässt hingegen das letzte Mitglied eine Gruppe, so wird der zugehörige Gruppenagent im Rahmen der Dynamik seiner Komponente *StatusG* wieder deaktiviert und steht damit für eine erneute Aktivierung durch andere Individualagenten, die eine Gruppe bilden möchten, bereit.

9.2.3.3 Die Komponente *VerhaltenG*

Die Komponente *VerhaltenG* steuert in enger Zusammenarbeit mit der Komponente *KognitionG* die Verhaltensweisen der Gruppenagenten. Auch für die Gruppenagenten beschränkt sich das Verhaltensrepertoire dabei auf rein reaktive Mechanismen, selbst wenn ein Gruppenagent in der Lage ist, in Abhängigkeit der aktuellen Gruppenqualität über den proaktiven Ausschluss eines Mitglieds zu entscheiden. Die Komponente *VerhaltenG* verfügt also über eine Menge von Ereignissen, die die Verhaltensregeln eines Gruppenagenten beschreiben. Auszuführende Aktionen werden der Komponente *AktorG* mit Hilfe von Ausführungsanordnungen, d. h. mobilen Komponenten des Typs *AusfuehrungsAOG*, mitgeteilt.

Reaktionen auf Nachrichten der Individualagenten. Individualagenten fragen bei den Gruppenagenten die Qualität der zugehörigen Lerngruppe an, bewerben sich um Aufnahme in die Gruppe und informieren die Gruppen über deren Ausstieg.

Anfragen von Individualagenten bewirken im Rahmen der Verhaltenssteuerung der Gruppenagenten die Erzeugung einer Ausführungsanordnung des Typs

Auskunft_verschicken. Im Rahmen dieser Ausführungsanordnung wird die Komponente *AktorG* angewiesen, dem Absender der Anfrage, der in der Komponente *KognitionG* hinterlegt ist, eine entsprechende Antwort zu schicken, in der die aktuelle Qualität der Lerngruppe enthalten ist.

Von größerer Bedeutung für eine Gruppe ist die Entscheidung, einen Bewerber in die Gruppe aufzunehmen oder dessen Bewerbung abzulehnen. Die Abbildung 9.19 zeigt die Regeln, die dieser Entscheidung im Gruppenagenten zugrunde liegen. Sind die Qualitätsanforderungen der Gruppe nach Aufnahme des Bewerbers erfüllt, so entscheidet sich der Gruppenagent für die Aufnahme des Bewerbers in die Gruppe und fordert die Komponente *AktorG* mit Hilfe einer Ausführungsanordnung des Typs *Zusage_verschicken* dazu auf, den Bewerber über die Aufnahme in die Gruppe zu informieren. Im Rahmen der Zusage wird dem Bewerber darüber hinaus die aktuelle Gruppengröße mitgeteilt. Ebenso wird durch die Komponente *VerhaltenG* im Falle einer Zusage eine Aktualisierung des Mitgliederverzeichnisses in der Komponente *KognitionG* angestoßen.

Ergibt sich nach einer näheren Betrachtung der resultierenden Situation, daß ein Bewerber nicht in die Gruppe mit aufgenommen werden kann, so wird die Komponente *AktorG* mit Hilfe einer Ausführungsanordnung des Typs *Absage_verschicken* dazu aufgefordert, eine entsprechende Ablehnung an den betroffenen Individualagenten zu versenden.

Die Mitteilung eines Individualagenten über den freiwilligen Ausstieg aus der Gruppe bedarf im Rahmen der Verhaltenssteuerung der Gruppenagenten keiner Behandlung, die zu einer externen Aktion führen wird. Ausstiegsmitteilungen werden durch den Gruppenagenten der Einfachheit halber nicht bestätigt und führen lediglich zu einer Aktualisierung der Komponente *KognitionG*, sowie zu einer Neuberechnung des Qualitätsstandards der Gruppe.

Festlegung des Qualitätsstandards der Gruppe. Immer dann, wenn sich durch Hinzunahme bzw. Ausstieg eines Mitglieds die vorliegende Gruppensituation ändert, wird im Modell *Lerngruppe* der Qualitätsstandard einer Gruppe neu festgelegt. Der Qualitätsstandard einer Gruppe berechnet sich dabei gemäß den Vorgaben aus Gleichung Gl. 9.21. Ebenso ergeht eine Mitteilung an alle Gruppenmitglieder, daß und in welcher Weise sich die Zusammensetzug der Gruppe verändert hat.

Überwachung der Gruppenqualität. Den Gruppenagenten im Modell *Lerngruppe* kommt eine wesentliche Kontrollfunktion zu. In regelmäßigen Zeitabständen überprüfen die Gruppenagenten im Rahmen ihrer Verhaltenssteuerung die Einhaltung des Qualitätsstandards der Gruppe. Liegt zum Zeitpunkt der Überprüfung die aktuelle Gruppenqualität unter dem vorgegebenen Standard, so wird die Komponente *KognitionG* damit beauftragt, einen Ausschlußkandidaten zu bestimmen. Wenn ein derartiger Ausschlußkandidat festgestellt werden konnte

und nach Ausschluß des betroffenen Individualagenten die Gruppenqualität wieder über die Höhe des Qualitätsstandards ansteigen würde, so wird der Ausschlußkandidat tatsächlich aus der Gruppe verwiesen. Der Ausschluß wird der Komponente *AktorG* mit Hilfe einer Ausführungsanordnung des Typs *Mitglied_ausschliessen* mitgeteilt. Ebenso wird innerhalb der Komponente *KognitionG* eine entsprechende Aktualisierung des Mitgliederverzeichnisses angestoßen.

9.2.4 Die Komponente *Connector*

Die Komponente *Connector* ist sehr eng nach den Vorgaben des Referenzmodells *PECS* konzipiert, die in Abschnitt 6.3.1 dargestellt sind. Sie dient als Kommunikationsplattform zwischen den Individual- und den Gruppenagenten. Dabei wird sowohl von direkter Kommunikation als auch von indirekter Kommunikation auf der Grundlage von gemeinsam nutzbaren Datenbereichen (Blackboards) Gebrauch gemacht. Als Einschränkung gilt hierbei allerdings, daß aufgrund der Annahmen des Modells *Lerngruppe* eine direkte Kommunikation nur zwischen Agenten unterschiedlicher Klassen, d. h. zwischen Individual- und Gruppenagenten, stattfindet.

9.2.4.1 Direkte Kommunikation im Modell *Lerngruppe*

Ein direkte Kommunikation findet im Modell *Lerngruppe*, wie eingangs erwähnt, nur zwischen Agenten unterschiedlicher Klassen statt. Zur Unterstützung dieser Kommunikationsform bietet die Komponente *Connector* für jeden Agenten einen eigenen Ein- und Ausgangsbereich für Nachrichten an. Im Eingangsbereich legen die Agenten ihre ausgehenden Mitteilungen ab. Im Ausgangsbereich nehmen die Agenten die eingetroffenen Nachrichten in Empfang. Die Modellierung der Ein- und Ausgangsbereiche, in denen Nachrichten abgelegt werden, erfolgt auf der Grundlage von Warteschlangen bzw. Locations.

Im Falle der direkten Kommunikation arbeitet die Komponente *Connector* ereignisorientiert. Sobald eine neue Nachricht im Eingangsbereich der Komponente *Connector* eintrifft, wird diese zunächst in eine interne Verteilstation übernommen. Im Rahmen der Nachrichtenverteilung wird zunächst anhand des Nachrichtenkopfes der Empfänger der Nachricht festgestellt. Anhand der Empfängerinformation wird dann die Nachricht in den adressierten Ausgangsbereich zugestellt, aus dem der angesprochene Agent anschließend seine neue Nachricht abziehen kann.

Nachrichten bestehen im Modell *Lerngruppe* aus zwei Teilen, aus dem Nachrichtenkopf und dem Nachrichtenrumpf. Im Nachrichtenkopf werden Meta-Daten gespeichert, die die korrekte Funktionsweise der Nachrichtenübermittlung sicherstellen. Hierzu zählen der Absender und der Empfänger einer Nachricht, sowie der Nachrichtentyp. Die Empfänger-Information wird innerhalb des Verteilmecha-

nismus der Komponente *Connector* genutzt, um die Nachricht der korrekten Ausgangswarteschlange zuzuordnen. Der Absender sowie der Typ der Nachricht dienen dem Empfänger als Grundlage für deren Interpretation. Die im Nachrichtenrumpf hinterlegten Nutzinformationen variieren je nach Typ der Nachricht. Im Falle einer Auskunft eines Gruppenagenten an einen Individualagenten besteht die Nutzinformation etwa aus der Qualität der Lerngruppe, die durch den Gruppenagenten vertreten wird.

9.2.4.2 Blackboard-Kommunikation im Modell *Lerngruppe*

Im Modell *Lerngruppe* stellt die Komponente *Connector* neben der direkten Kommunikation auch einen gemeinsam zugreifbaren Datenbereich, d. h. ein Blackboard, für die freie Nutzung durch die Agenten bereit. Auf diesem Blackboard werden Informationen angeboten, die permanent von mehreren Agenten benötigt werden. Hierzu zählen etwa die Qualität der aktiven Lerngruppen oder der Aktivierungsstatus von Gruppenagenten, wobei diese Kategorie von Informationen durch die Gruppenagenten auf dem Blackboard veröffentlicht und als Eingangsinformation in den Individualagenten genutzt wird. Ebenso stellen die Individualagenten Informationen über ihr Gesamtwissen und ihre soziale Kompetenz auf dem Blackboard zur Verfügung und liefern damit den Gruppenagenten während der Gruppenlernphase die Grundlage für die Berechnung der Gruppenqualität. Das Blackboard, das im Modell *Lerngruppe* zum Einsatz kommt, weist also hinsichtlich seiner Nutzung durch die beiden Agentenklassen zwei unterschiedliche Regionen auf.

Eine Besonderheit im Modell *Lerngruppe* besteht darin, dass die Agenten Informationen auf dem Blackboard nicht ereignisorientiert, sondern kontinuierlich im Rahmen von lesenden Zugriffen auf einen Teil ihrer Modellgrößen zur Verfügung stellen. Dieser Mechanismus bietet sich im vorliegenden Fall an, da sich sämtliche Informationen auf dem Blackboard mit in den Agenten vorhandenen Modellgrößen decken. In dieser Hinsicht macht der hier gewählte Implementierungsansatz in sehr nützlicher Weise Gebrauch von dem Glas-Box-Konzept der Modellbeschreibungssprache *Simplex-MDL* (Schmidt, 2000).

9.2.5 Die Komponente *Statistik*

Innerhalb der Komponente *Statistik* werden statistische Daten sowohl über die Individuen als auch die Lerngruppen des Modells generiert. Die Komponente *Statistik* steht zu diesem Zweck mit allen Repräsentanten beider Agentenklassen in Verbindung und greift wesentliche Merkmale, wie beispielsweise das Gesamtwissen der Agenten oder auch die Qualität der Lerngruppen als Eingangsinformationen ab. Diese Eingangsdaten werden im Rahmen von mathematischen Funktionen und algebraischen Gleichungen statistisch aufbereitet. Die generierten Daten wer-

den im Gegenzug den Agenten zur weiteren Verwendung zur Verfügung gestellt. Ebenso fließen die statistischen Werte als Basisdaten in die Animation des Modells *Lerngruppe* ein und dienen dort der graphischen Darstellung der Modelldynamik.

9.3 Simulationsexperimente

Das Modell *Lerngruppe* bietet vielfältige Ansatzpunkte für die Durchführung von Simulationsexperimenten. Im Rahmen von Modelluntersuchungen werden in der Regel ausgewählte Modellgrößen mit bestimmten Initialwerten versehen, das Simulationsmodell bis zu einem vorgewählten Zeitpunkt betrieben und anschließend die Zeitverläufe sowie die Endzustände charakteristischer Modellgrößen näher untersucht. Um derartige Modelluntersuchungen auch Anwendern ohne detaillierte Kenntnisse der Tiefenstruktur des Modells zugänglich zu machen, wurde das Modell *Lerngruppe* mit einer graphischen Benutzungsoberfläche versehen. Mit der Grundkonzeption dieser graphischen Benutzungsoberfläche beschäftigt sich der folgende Abschnitt 9.3.1. Das Kapitel 9.3.2 zeigt daraufhin in exemplarischer Weise einige grundlegende Experimentier- und Auswertungsmöglichkeiten des Modells *Lerngruppe*.

9.3.1 Die graphische Benutzungsoberfläche

Die graphische Benutzungsoberfläche des Modells *Lerngruppe* verfolgt das Ziel, ein einfaches Experimentieren mit dem Modell zu ermöglichen. Im Rahmen der Grundfunktionalität der graphischen Benutzungsoberfläche besteht die Möglichkeit, Initialwerte einiger ausgewählter Modellgrößen zu Beginn eines Simulationslaufes einzustellen, die Durchführung von Simulationsläufen anzustoßen, sowie im Anschluß daran im Rahmen einer Animation den Ablauf der Simulation zu analysieren.

Für die Menge der veränderbaren Modellgrößen wurde im Rahmen der graphischen Benutzungsoberfläche der Einfachheit halber bereits eine Vorauswahl getroffen. Änderungsmöglichkeiten sind hierbei im wesentlichen für diejenigen Parameter vorgesehen, die die Belegung der Persönlichkeitsmerkmale sowie die Veränderung der kognitiven und sozialen Zustandsvariablen der Individualagenten betreffen:

- Verteilung der Intelligenz / sozialen Veranlagung

  Zu Beginn eines Simulationslaufes werden allen Individualagenten zufällige Startwerte für ihre Intelligenz sowie ihre soziale Veranlagung zugewiesen. Mit Hilfe dieser beiden Parameter können die hierfür verwendeten Verteilungsfunktionen angegeben werden, so daß auf diese Weise Populationen ge-

neriert werden können, die sich grundlegend hinsichtlich der Ausprägung der Persönlichkeitsmerkmale ihrer Individuen voneinander unterscheiden.

- Gewichtung des Lernens / der sozialen Zufriedenheit

  Die Entscheidung eines Individuums, in eine Lerngruppe einzutreten, hängt sowohl von kognitiven als auch von sozialen Aspekten ab (s. Bed. 9.1). Beide Aspekte gehen dabei in die Entscheidung eines Individuums jeweils mit einer spezifischen Gewichtung ein, die über die graphische Benutzungsoberfläche modifiziert werden kann. Diese Einstellungen bieten also die Möglichkeit, den Fokus der Individuen bei ihrer Entscheidung, in eine Gruppe einzutreten, auf eher kognitive oder eher soziale Gesichtspunkte zu verlagern.

- Normales Wachstum des Wissens

  Mit Hilfe dieses Parameters kann die durchschnittliche Lerngeschwindigkeit der Individuen beeinflusst werden. Indirekt wird sich durch eine Veränderung dieser Grundeinstellung auch eine Veränderung der Dauer der Einzel- sowie der Gruppenlernphase ergeben.

- Normale Entwicklung der sozialen Zufriedenheit

  Dieser Parameter erlaubt eine Veränderung der durchschnittlichen Zu- bzw. Abnahmerate der sozialen Zufriedenheit in der jeweiligen Lernphase.

- Normales Wachstum der sozialen Kompetenz

  Die soziale Kompetenz der Individuen nimmt innerhalb der Gruppenlernphasen linear zu. Dieser Parameter erlaubt eine Modifikation des normalen Anstiegs der sozialen Kompetenz während der Gruppenlernphasen.

- Toleranz der Lerngruppen

  Die Toleranz der Lerngruppen ist ausschlaggebend für die Berechnung des Qualitätsstandards der Gruppen. Mit Hilfe dieses Parameters kann gesteuert werden, bis zu welcher Untergrenze hinsichtlich ihrer Gruppenqualität eine Gruppe gerade noch bereit ist, ein neues Mitglied aufzunehmen.

Nach der Vorbelegung der wesentlichen Modellgrößen kann ein Simulationslauf mit dem Modell *Lerngruppe* gestartet werden. Anschließend stehen die im Rahmen des Simulationslaufes erzeugten Modelldaten zur graphischen Darstellung in der Animation zur Verfügung. Den Aufbau der hierfür verwendeten Animationsoberfläche zeigt die Abbildung 9.25.

Die Benutzungsoberfläche des Modells *Lerngruppe* gliedert sich grundsätzlich in drei Bereiche. Am oberen Rand befinden sich die Menüleiste, eine Reihe von Schaltflächen zur Steuerung des Ablaufs der Animation sowie zum Öffnen eines weiteren Fensters, das statistische Informationen zum Simulationslauf zeigt, und die Anzeige der aktuellen Simulationszeit. Darunter sind zwei Regionen angesiedelt, die mit *Poolbereich* und *Gruppenbereich* beschriftet sind.

Im Poolbereich sind diejenigen Individuen dargestellt, die sich zum betrachteten Zeitpunkt in einer Einzellernphase befinden. Jedes Individuum wird dabei

durch ein Rechteck visualisiert, in dem die Nummer des Individuums, sowie zwei Balken enthalten sind, die das Wissen der aktuellen Lernphase sowie den aktuellen Wert der sozialen Zufriedenheit des Individuums repräsentieren.

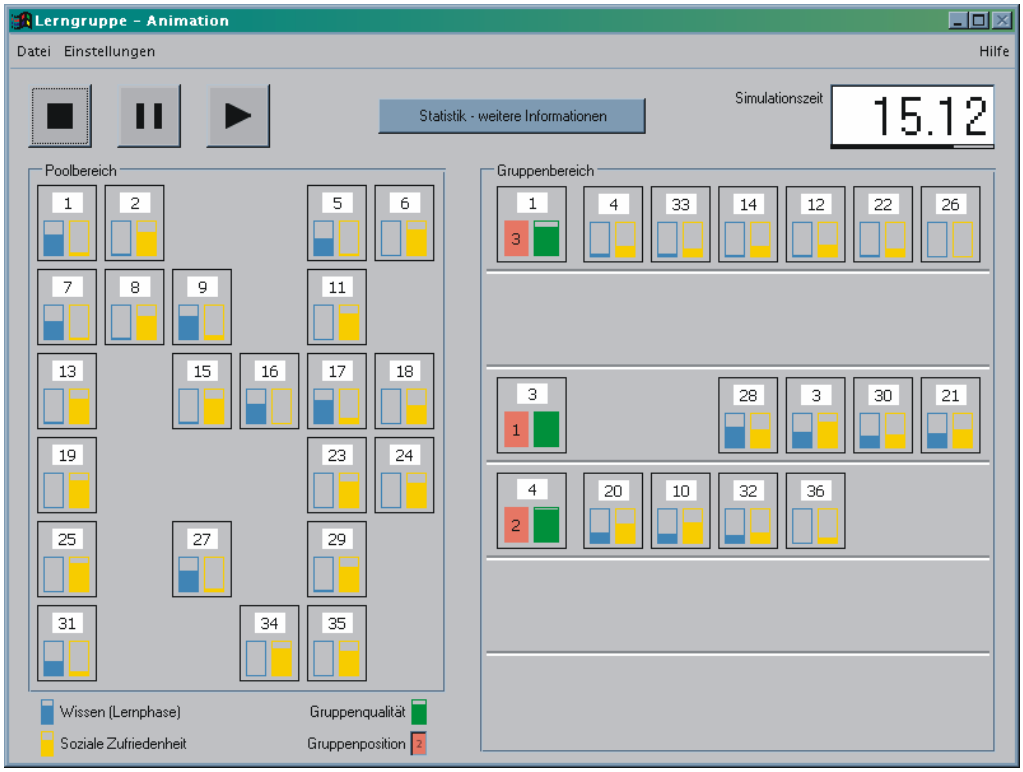

Abbildung 9.25: Die Benutzungsoberfläche der Animation für das Modell Lerngruppe (Kraus, 1999)

Der Gruppenbereich ist in mehrere Zeilen aufgeteilt. In jeder Zeile wird eine eigene Lerngruppe dargestellt. Der erste Eintrag in einer Zeile repräsentiert dabei jeweils den Gruppenagenten, der die zugehörige Lerngruppe vertritt, und zeigt die Nummer der Lerngruppe, deren Qualität sowie die Position der Lerngruppe in der Rangfolge der Gruppen. Hinter dem Gruppenagenten sind jeweils die der Gruppe angehörigen Individuen mit ihren wesentlichen Eigenschaften in Analogie zum Poolbereich dargestellt. Zusätzlich ist für jedes Individuum ein weiteres Dialogfenster vorgesehen, das eine detaillierte Sicht auf die aktuelle Belegung wesentlicher Modellgrößen bietet.

9.3.2 Experimentiermöglichkeiten mit dem Modell *Lerngruppe*

Aufgrund der Vielzahl an vorhandenen Modellgrößen und Parametern bietet das Modell *Lerngruppe* ein breites Spektrum an Experimentiermöglichkeiten. Exem-

plarisch soll nun anhand einer vergleichenden Modelluntersuchung gezeigt werden, welche Resultate bzw. Aussagen das Modell *Lerngruppe* grundsätzlich zu liefern im Stande ist. Den Ausgangspunkt hierfür liefert ein Standardlauf, der auf der Grundeinstellung des Modells basiert. Verglichen werden die Ergebnisse dieses Standardlaufes im nächsten Schritt mit einem Simulationslauf, bei dem ein wesentlicher Parameter zu Beginn verändert wurde. Es handelt sich dabei um die Toleranz der Lerngruppen. Mit Hilfe dieses Parameters kann gesteuert werden, welche Verringerung ihrer Qualität eine Lerngruppe bei Aufnahme eines neuen Gruppenmitglieds gerade noch zu akzeptieren bereit ist. Im Standardlauf ist dieser Parameter mit dem Wert 0.95 belegt. Eine Gruppe akzeptiert damit eine Qualitätsminderung um bis zu 5%. Der Vergleichslauf geht hingegen von einer stark leistungsorientierten Population aus, in der die Lerngruppen nur noch eine 1%-ige Abweichung von ihrer Qualität tolerieren werden. In diesem Fall wird die Toleranz der Lerngruppen mit dem Wert 0.99 vorbelegt.

Ein Vergleich der beiden Simulationsläufe soll dabei im wesentlichen hinsichtlich der folgenden Fragen stattfinden:

- Wie hoch ist die mittlere Anzahl an Lerngruppen über den gesamten Simulationslauf hinweg?
- Aus wie vielen Mitgliedern bestehen die Lerngruppen im Mittel?
- Wie entwickelt sich das Wissen der Individuen im Mittel und wie groß ist dessen Schwankungsbreite in der Population?
- Wie entwickelt sich die Sozialkompetenz der Individuen im Mittel und wie groß ist deren Schwankungsbreite in der Population?

Parameter	**Belegung im Standardlauf**	**Belegung im Vergleichslauf**
Verteilung der Intelligenz	Gauß-Verteilung	
Verteilung der sozialen Veranlagung	Gauß-Verteilung	
Gewichtung des Lernens	2	
Gewichtung der sozialen Zufriedenheit	1	
Normales Wachstum des Wissens	0.5 [WI/h]	
Normale Entwicklung der sozialen Zufriedenheit	0.5 [SoZ/h]	
Normales Wachstum der sozialen Kompetenz	0.4 [SK/h]	
Toleranz der Lerngruppen	0.95	0.99

Abbildung 9.26: Initialbelegung der Modellgrößen für das Vergleichsexperiment mit dem Modell Lerngruppe

Als Vorbelegung der grundlegenden Modellparameter werden in den beiden Simulationsläufen die folgenden Werte verwendet (s. Abbildung 9.26):

Wie aus Abbildung 9.26 noch einmal deutlich hervorgeht, weichen beide Simulationsläufe lediglich in der Vorbelegung der Toleranz der Lerngruppen voneinander ab.

Mit beiden Parametereinstellungen wurde jeweils ein Simulationslauf über 500 Zeiteinheiten durchgeführt. Die Ergebnisse der beiden Simulationsläufe sind in Abbildung 9.27 in der Übersicht dargestellt. Die betrachteten Ergebnisgrößen resultieren dabei unmittelbar aus den eingangs eingeführten Fragen, die diesem Vergleichsexperiment zugrunde gelegt wurden.

Ergebnisgröße	**Wert im Standardlauf**	**Wert im Vergleichslauf**
mittlere Anzahl an Lerngruppen	2.39	2.97
mittlere Gruppengröße	4.05	3.20
mittleres Gesamtwissen	64.82	64.72
Wertebereich des Gesamtwissens	[50.39 .. 76.04]	[46.66 .. 76.57]
mittlere soziale Kompetenz	52.10	51.13
Wertebereich der sozialen Kompetenz	[42.69 .. 61.27]	[41.86 .. 60.12]

Abbildung 9.27: Ergebnisse des Vergleichsexperiments mit dem Modell Lerngruppe

Die in Abbildung 9.27 dargestellten Ergebnisse des Vergleichexperiments können in der folgenden Weise interpretiert werden:

Im Standardlauf ergibt sich eine mittlere Anzahl von 2.39 Lerngruppen über die Gesamtdauer des Simulationslaufes hinweg. Im Vergleich dazu steigt im Rahmen der leistungsorientierten Population dieser Wert sehr deutlich auf 2.97 Gruppen im Mittel an. Dieser Anstieg begründet sich darin, daß aufgrund ihrer geringeren Toleranz sehr restriktiv bei der Aufnahme neuer Gruppenmitglieder vorgehen und dadurch mehr Individuen gezwungen werden, neue Lerngruppen zu gründen, nachdem ihre Bewerbungsversuche bei bestehenden Lerngruppen gescheitert sind. Ebenso wurde im Modell *Lerngruppe* die optimale Gruppengröße bei 4 Individuen angesetzt. Auch hier wird eine Lerngruppe aufgrund der gesunkenen Toleranz ein fünftes oder gar sechstes Gruppenmitglied nur noch dann aufnehmen, wenn es aufgrund eines ausgezeichneten Wissens oder einer besonders ausgeprägten sozialen Kompetenz den Verlust an Qualität durch die über das Optimum hinaus ansteigende Gruppengröße kompensieren kann. Diesen Erklärungsansatz bestätigt auch die mittlere Gruppengröße, die im Vergleich zu 4.05 Individuen im Standardlauf auf 3.20 Individuen in der leistungsorientierten Population zurückgeht.

Hinsichtlich des durchschnittlichen Gesamtwissens, das sich über die gesamte Population betrachtet ergibt, sind zwischen den beiden Läufen keine signifikanten

Unterschiede erkennbar. Stärker abweichend stellen sich hingegen die auftretenden Schwankungsbreiten im Gesamtwissen der Individuen dar. Im Vergleich zum Standardlauf wird in der leistungsorientierten Gesellschaft die untere Schranke für das Gesamtwissen eines Individuums nach unten und die obere Schranke nach oben verschoben. Dieser Sachverhalt deutet darauf hin, daß in der leistungsorientierten Gesellschaft eine stärkere Differenzierung hinsichtlich der Leistungsfähigkeit der Individuen stattfindet. Die Ursache dieser Entwicklung liegt darin, daß schwächere Individuen in der Regel nur noch in schwache Gruppen aufgenommen werden und der Lernfortschritt dadurch zusätzlich abgebremst wird. Sehr leistungsfähige Individuen profitieren allerdings von der höheren Selektivität der Lerngruppen und entwickeln sich schneller und besser weiter als im Standardlauf.

Sowohl hinsichtlich der mittleren sozialen Kompetenz als auch der zugehörigen Randwerte ist in der leistungsorientierten Gesellschaft im Vergleich zum Standardlauf ein Absinken deutlich erkennbar. Die Ursache hierfür liegt darin begründet, daß die Individuen in der leistungsorientierten Gesellschaft wesentlich mehr Zeit für die Gründung neuer Gruppen aufwenden müssen als die Individuen im Standardlauf und damit wertvolle Zeit für das eigentliche Lernen in Gruppen verlieren. Ein Individuum im Standardlauf verbringt im Mittel etwa 25.76% der Zeit in Lerngruppen, während der Vergleichswert in der leistungsorientierten Gesellschaft sehr deutlich auf 25.27% abfällt.

9.4 Zusammenfassung und Fazit

Die Fallstudie *Modellierung von Gruppenprozessen* demonstriert den Einsatz agentenbasierter Methoden zur Unterstützung der Modellbildung und Simulation im Bereich der Sozialwissenschaften. Es wird ein künstliches Sozialsystem konstruiert, das aus 36 Individuen sowie einer wechselnden Anzahl an Kleingruppen besteht. Untersuchungsgegenstand ist dabei die Formation, Entwicklung und Disintegration von Gruppen am Beispiel von Studenten einer virtuellen Universität, die sich in der Prüfungsvorbereitungsphase befinden.

Das Ziel der vorliegenden Fallstudie besteht darin, in grundsätzlicher Weise zu zeigen, daß künstliche Sozialsysteme, in denen sowohl Individuen als auch Zusammenschlüsse von Individuen zu Gruppen eine Rolle spielen, in strukturierter und übersichtlicher Weise auf der Grundlage der Referenzmodells *PECS* modelliert werden können. Um dieses grundsätzliche Ziel zu fokussieren, sind die Agenten des Modells *Lerngruppe* ganz bewusst nur mit einem Minimalmaß an Eigenschaften und Verhaltensweisen ausgestattet, die gerade ausreichen, um die Mechanismen der Gruppenbildung, der Gruppenarbeit und der Auflösung von Gruppen darstellen zu können. Es zeigt sich jedoch, daß trotz der strikten Beschränkung der Funktionalität innerhalb der Agenten und der angrenzenden Komponenten bereits ein Modell mit beachtlicher Komplexität entsteht. In dieser Hin-

sicht erfüllt das Modell *Lerngruppe* ohne Zweifel die Anforderungen an ein praxisrelevantes Einsatzszenario für das Referenzmodell *PECS*.

Wie auch das Modell *Adam* deutlich zum Ausdruck bringen wird, stellen sowohl der systemtheoretische als auch der komponentenorientierte Ansatz, die dem Referenzmodell *PECS* technologisch zugrunde liegen, eine solide Basis für den Aufbau agentenbasierter Simulationsmodelle zur Verfügung. Der systemtheoretische Ansatz ermöglicht eine sehr komfortable Beschreibung von Zuständen und Zustandsübergängen, die sowohl im Inneren der Agenten als auch im Rahmen der Interaktion der Agenten untereinander und mit ihrer Umgebung eine Rolle spielen. Ganz besonders kommt dieser Aspekt beim Aufbau künstlicher Sozialsysteme zum tragen, da in diesem Fall alle oben genannten Arten von Zuständen und Zustandsübergängen auftreten werden. Der komponentenorientierte Ansatz führt zu einer klaren Strukturierung von Simulationsmodellen und bietet sich ebenso ganz besonders für die Konzeption und Implementierung künstlicher Sozialsysteme an, die sich sehr häufig durch eine größere Anzahl interagierender Entitäten auszeichnen. Durch eine klare Strukturierung derartiger Modelle auf der Grundlage eines komponentenorientierten Ansatzes kann sichergestellt werden, dass selbst große und komplexe Modelle überschaubar und wartbar bleiben. In dieser Hinsicht stellt sich das Referenzmodell *PECS*, das diese beiden grundsätzlichen Ansätze für den Aufbau agentenbasierter Simulationsmodelle vereint, als solide Grundlage für die Konzeption und Implementierung agentenbasierter Simulationsmodelle im Bereich der Sozialwissenschaften heraus.

10 FALLSTUDIE IV: MODELLIERUNG MENSCHLICHER HANDLUNGSREGULATIONSMECHANISMEN

Die Fallstudie *Modellierung menschlicher Handlungsregulationsmechanismen* soll in exemplarischer Weise demonstrieren, wie das Referenzmodell *PECS* zur Unterstützung der psychologischen Modellbildung eingesetzt werden kann.

Den Ausgangspunkt für diese Fallstudie bildet die Ψ-Theorie von Dietrich Dörner (Dörner, 1999). In dieser Theorie wird der Versuch unternommen, das komplexe Zusammenspiel unterschiedlicher Einflüsse und Prozesse, die im Zusammenhang mit der menschlichen Handlungsregulation von Bedeutung sind, auf der Grundlage einer einheitlichen psychologischen Theorie zu beschreiben und zu erklären. Dörner verfolgt mit dieser Theorie das Ziel, menschliches Verhalten und Erleben in umfassender Weise abzubilden und sieht seinen Ansatz gewissermaßen als einen „Bauplan für eine Seele“ (Dörner, 1999). Mit dieser Sichtweise ist ein sehr hoher Anspruch verbunden, der zu einem hohen Maß an Komplexität und Detailreichtum in der Ψ-Theorie führt.

Diesen Anspruch teilt das Modell Adam, das in diesem Kapitel vorgestellt wird, ganz bewußt nicht. Wie eingangs bereits erwähnt, besteht das Ziel, das mit dem Modell Adam verfolgt wird, im wesentlichen darin, zu zeigen, daß psychische Zustände und Prozesse, sowie deren Wechselwirkungen auf der Grundlage des Referenzmodells *PECS* in strukturierter und übersichtlicher Weise modelliert werden können. Aus diesem Grund wird im Rahmen dieser Fallstudie auf eine besonders reichhaltige Ausstattung des Agenten Adam mit vielfältigen Eigenschaften und Fähigkeiten verzichtet. Adam soll keinen wirklichen Menschen nachbilden. Vielmehr beschränkt sich die Ausstattung von Adam auf ein Minimalmaß an Eigenschaften und Fähigkeiten, die unabdingbar erscheinen, um das grundsätzliche Zusammenspiel physischer, emotionaler und kognitiver Prozesse im Rahmen der menschlichen Verhaltenssteuerung auf der Grundlage des Referenzmodells *PECS* untersuchen zu können.

Die wesentlichen Unterschiede zwischen dem Modell Adam und der Ψ-Theorie von Dörner bestehen in den verwendeten kognitiven Strukturen und der Modellierung der Emotionen. In Hinblick auf die kognitiven Strukturen geht die Ψ-Theorie von einem Protokollgedächtnis aus, das in Anlehnung an neuronale

Strukturen des menschlichen Gehirns gestaltet ist. Adam hingegen nutzt als grundlegenden Informationsspeicher eine kognitive Landkarte seiner Umgebung, die im Laufe der Zeit mit konkreten Erfahrungen gefüllt wird. Emotionen werden in der Ψ-Theorie als abgeleitete Größen betrachtet. Sie werden durch vier Modulatoren determiniert und mit bestimmten Verhaltensweisen von Ψ gleichgesetzt. Das Modell Adam verfolgt auch hier eine andere Sichtweise und geht davon aus, daß Emotionen eigenständige Zustände eines Organismus sind, die psychische Prozesse und auch das Verhalten des Organismus beeinflussen.

In den folgenden Abschnitten soll nun ein Überblick über die grundlegende Funktionsweise, die Implementierung sowie die Experimentiermöglichkeiten des Modells Adam vermittelt werden.

10.1 Modellbeschreibung

Das Modell Adam geht von einem Agenten namens Adam aus, der versucht, in einer zunächst für ihn unbekannten Umgebung Erfahrungen zu sammeln und zu überleben. Die Umwelt ist in Anlehnung an Standardszenarien in der Künstlichen Intelligenz als zweidimensionales Gitter konzipiert, wobei den einzelnen Feldern des Gitters unterschiedliche Bedeutungen zugeordnet werden (vgl. hierzu beispielsweise die Wumpus-Welt nach Russel und Norvig, 1995). Im Mittelpunkt dieser Modelluntersuchung steht die Konzeption der Verhaltenssteuerung des Agenten Adam, wobei physische, emotionale und kognitive Einflüsse Berücksichtigung finden.

10.1.1 Die Umwelt des Agenten Adam

Die Umwelt beschreibt den Lebensraum, der den Agenten Adam umgibt, und umfaßt alle Einflüsse, die für das Verhalten von Adam bestimmend sind. Adams Lebensraum ist beschränkt. Er wird durch ein zweidimensionales Gitter modelliert, das nach allen Seiten hin begrenzt ist und aus 12×12, d. h. also insgesamt 144, Feldern besteht. Die Felder können dabei unterschiedliche Bedeutungen tragen. Es existieren neutrale Felder, Nahrungsfelder sowie Gefahrenfelder.

Neutrale Felder haben keine besondere Funktion. Allerdings kostet Adam das Betreten eines neutralen Feldes, wie jede andere Aktion, eine gewisse Menge an Energie. Nahrungsfelder bieten Adam die Möglichkeit, Nahrung aufzunehmen und seinen Energiehaushalt zu regenerieren. Beim Betreten von Gefahrenfeldern nimmt die physische Konstitution von Adam Schaden. Man kann sich Gefahrenfelder beispielsweise als Sümpfe oder Fallgruben vorstellen, aus denen sich Adam nur mit großer Mühe wieder befreien kann. Mit der Befreiung aus einer Gefahrenstelle ist ein großer Energieverlust verbunden, der durchaus zu einer ernsthaften Bedrohung für Adam führen kann.

Die Abbildung 10.1 zeigt eine mögliche Konstellation für Adams Umwelt. Felder, die mit einem Apfel gekennzeichnet sind, markieren Nahrungsfelder. Mit einem Blitz versehene Felder symbolisieren Gefahrenfelder. Neutrale Felder tragen keine besondere Kennzeichnung. Der aktuelle Aufenthaltsort von Adam wird durch das Strichmännchen angedeutet.

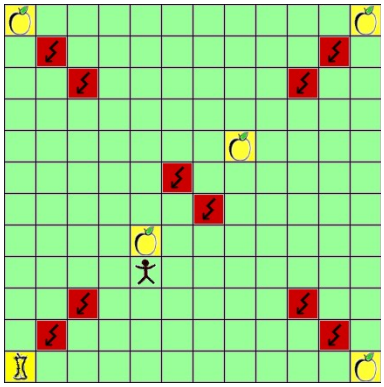

Abbildung 10.1: Das Grundkonzept der Umwelt im Modell Adam

Adams Umwelt unterliegt dynamischen Veränderungen. Die Bedeutungen der jeweiligen Felder bleiben zwar für die Dauer eines Simulationslaufes konstant, jedoch können sich die Zustände der einzelnen Felder im Laufe der Zeit sowohl infolge von Aktionen, die Adam ausführt, als auch eigendynamisch verändern.

Durch Ausführung der Aktionen *Gehen* oder *Befreien* ist Adam in der Lage, seine Position innerhalb der Umwelt und damit den Zustand der betroffenen Felder zu beeinflussen. Die Aktion *Essen* gibt Adam die Möglichkeit, Nahrung, die auf einem Nahrungsfeld zur Verfügung steht, zu sich zu nehmen. Durch Ausführung der Aktion *Essen* wird die auf einem Nahrungsfeld vorhandene Nahrungsmenge auf ein Minimum reduziert.

Nahrungsfelder können sich im Laufe der Zeit wieder aus eigener Kraft regenerieren, nachdem sie von Adam abgeerntet wurden. Der Einfachheit halber wird im vorliegenden Modell für alle Nahrungsfelder ein identisches dynamisches Verhalten unterstellt, das einem logistischen Wachstum folgt. Dieser Sachverhalt kann für jedes Nahrungsfeld mit Hilfe der folgenden Differentialgleichung beschrieben werden:

$$\begin{aligned} Nahrungsmenge'(t) := {} & NahrungsFaktor * Nahrungsmenge(t) * \\ & (NahrungsMax - Nahrungsmenge(t)) / \\ & NahrungsMax \end{aligned} \qquad \text{(Gl. 10.1)}$$

Der Faktor *(NahrungsMax – Nahrungsmenge) / NahrungsMax* bewirkt dabei eine Annäherung der auf einer Nahrungsstelle verfügbaren Nahrungsmenge an einen Maximalwert *NahrungsMax*.

Die in Gleichung 10.1 verwendeten Bezeichner haben die folgende Bedeutung:

Bezeichner	Einheit	Initialwert	Erläuterung
Nahrungsmenge	[EnE]	20	Nahrungsmenge auf einem Nahrungsfeld
NahrungsFaktor	[1/h]	0.15	Wachstumsfaktor der Nahrung
NahrungsMax	[EnE]	100	Maximalwert für die Nahrungsmenge auf einem Nahrungsfeld

Abbildung 10.2: Die Variablen für die Berechnung der Nahrungsmenge auf Nahrungsfeldern

Die Abbildung 10.3 veranschaulicht den normalen Wachstumsverlauf der Nahrung auf den Nahrungsfeldern in Adams Umwelt.

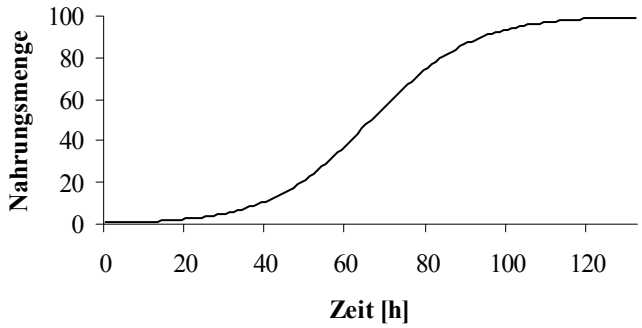

Abbildung 10.3: Der normale Wachstumsverlauf der Nahrung auf Nahrungsfeldern

Trotz ihrer Einfachheit bietet die in dieser Weise konzipierte Umwelt bereits vielfältige Möglichkeiten. Sie umfaßt Einflüsse, die sich sowohl in positiver, als auch negativer Weise auf die Verfassung von Adam auswirken. Zudem ändert sich Adams Umwelt dynamisch und verfügt über eine ausreichende Größe, so daß Adam zu keinem Zeitpunkt der Simulation auf vollständige Informationen über den Zustand seiner Umwelt zurückgreifen kann.

10.1.2 Der Agent Adam

Die folgenden Abschnitte beschäftigen sich mit den Eigenschaften und Fähigkeiten des Agenten Adam. Es werden über die Zeit veränderliche Zustandsvariablen eingeführt, die den physischen, emotionalen und kognitiven Zustand von Adam beschreiben. Darüber hinaus wird die Menge der Aktionen, eingeteilt in interne und externe Aktionen, spezifiziert, die für Adam zur Disposition stehen. Abschließend werden die Grundmechanismen vorgestellt, die für Adams Verhaltenssteuerung Verwendung finden. Adam kann bei der Auswahl seiner Verhaltensweisen sowohl reaktiv als auch deliberativ vorgehen.

10.1.2.1 Adams Modellelemente

Adam verfügt über eine Menge an Zustandsvariablen, die seinen internen physischen, emotionalen und kognitiven Zustand beschreiben sowie die Auswahl und die Ausführung seiner Verhaltensweisen beeinflussen.

Physische Zustandsvariablen

Der physische Zustand des Agenten Adam wird durch die Menge an Energie bestimmt, die ihm zu einem gegebenen Zeitpunkt zur Verfügung steht. Die Energie stellt dabei ein Maß für Adams Vitalität dar und spiegelt seine Leistungsfähigkeit wider. Wenn Adams Energie einen vorgegebenen Wert *EnergieMin* unterschreitet, endet sein Leben.

Adams Energieniveau ändert sich dynamisch in der Zeit. Es hängt im wesentlichen von den Aktionen ab, die Adam im Rahmen seiner Verhaltenssteuerung zur Ausführung auswählt. Adam kann einerseits seinen Energiehaushalt regenerieren, indem er die Aktion *Essen* auf einem Nahrungsfeld ausführt. Andererseits führt er jedoch auch Aktionen aus, die mit einem Verbrauch an Energie verbunden sind. Die Menge an Energie, die zur Ausführung einer Aktion nötig ist, hängt dabei vom Typ der jeweiligen Aktion ab. Adam verbraucht beispielsweise für die Aktion *Befreien*, die ein mühevolles Herausklettern aus einer Falle modelliert, wesentlich mehr Energie als für die Aktion *Planen*, die einen Denkprozeß abbildet. In Abbildung 10.4 wird für jede einzelne Aktion quantifiziert, welche Konsequenzen ihre Ausführung für den Energiehaushalt von Adam hat.

Die Veränderung von Adams Energie kann mit Hilfe von einfachen Differentialgleichungen beschrieben werden. Die Gleichung 10.2 gibt an, welcher Energiezuwachs sich durch Ausführung der Aktion *Essen* einstellt. Der Faktor *Energie_Essen* ist dabei mit einem positiven Wert belegt und bewirkt einen konstanten Anstieg des Energieniveaus. Adams Energieniveau ist nach oben hin durch eine maximale Obergrenze *Energie_Max* beschränkt.

$$Energie'(t) := Energie\_Essen \qquad \text{(Gl. 10.2)}$$

Der durch die Ausführung von energieverbrauchenden Aktionen bedingte Energieverlust wird durch die Gleichung 10.3 modelliert.

$$Energie'(t) := EFaktor * Energie\_Aktion \qquad \text{(Gl. 10.3)}$$

Der Faktor *EFaktor* repräsentiert einen gewissen Grundverbrauch an Energie, der gleichermaßen für alle energieverbrauchenden Aktionen angenommen wird. Mit Hilfe dieses Faktors kann angegeben werden, ob Adam grundsätzlich einen vergleichsweise hohen oder niedrigen Verbrauch an Energie bei der Ausführung unterschiedlicher Aktionen hat. Dieser Grundverbrauch wird sicherlich die Überlebensfähigkeit von Adam in seiner Umwelt beeinflussen.

Der Faktor *Energie_Aktion* gibt den aktionsspezifischen Verbrauch an Energie an. Der Teilname *Aktion* bezeichnet dabei die jeweils ausgeführte Aktion aus der Menge der Aktionen *{ Planen*, *Explorieren*, *Prüfen*, *Gehen*, *Befreien }*. Bis auf die Aktion *Essen* bewirken alle Aktionen, die Adam ausführt, einen Verlust an Energie. Aus diesem Grund nimmt der Faktor *Energie_Aktion* für alle betreffenden Aktionen einen negativen Wert an (s. Abbildung 10.4).

Die Variablen, die für die Bestimmung von Adams Energieniveau von Bedeutung sind, sind in der Abbildung 10.4 noch einmal im Überblick zusammengestellt.

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
Energie	[EnE]	100	Aktuelles Energieniveau
EFaktor	[1/h]	4	Grundverbrauch an Energie
Energie_Planen	[EnE]	-0,2	Zusatzverbrauch an Energie für die Aktion *Planen*
Energie_Explorieren	[EnE]	-0,3	Zusatzverbrauch an Energie für die Aktion *Explorieren*
Energie_Prüfen	[EnE]	-1,0	Zusatzverbrauch an Energie für die Aktion *Prüfen*
Energie_Gehen	[EnE]	-0,8	Zusatzverbrauch an Energie für die Aktion *Gehen*
Energie_Befreien	[EnE]	-4,0	Zusatzverbrauch an Energie für die Aktion *Befreien*
Energie_Essen	[EnE]	20	Zuwachs an Energie bei Ausführung der Aktion *Essen*
EnergieMax	[EnE]	100	Maximalwert der Energie
EnergieMin	[EnE]	0	Minimalwert der Energie

Abbildung 10.4: Die Variablen für die Berechnung von Adams Energie

Die Abbildung 10.5 zeigt beispielhaft einen möglichen Kurvenverlauf für die Zustandsvariable *Energie*. Zu Beginn erkundet Adam seine Umwelt und verbraucht dadurch nur sehr wenig Energie. Kurz nach dem Zeitpunkt 12 gerät Adam in eine Falle und muß sich mit einem hohen Energieaufwand daraus befreien. Dieser Sachverhalt äußert sich durch den starken Abfall der Kurve zwischen den Zeitpunkten 12 und 14. Im Anschluß daran fährt Adam zunächst mit der Erkundung seiner Umwelt fort und trifft dabei etwa zum Zeitpunkt 15 auf eine Nahrungsstelle. Dort füllt er seinen Energievorrat durch Ausführung der Aktion *Essen* bis zur Obergrenze *Energie_Max* auf und geht danach wieder zur weiteren Erkundung seines Lebensraumes über.

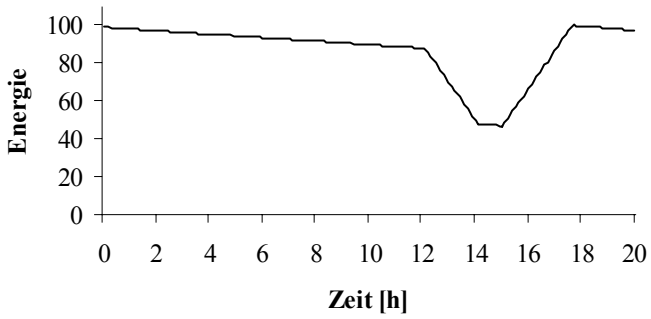

Abbildung 10.5: Ein exemplarischer Verlauf der Zustandsvariable Energie

Die verfügbare Energie nimmt einen entscheidenden Einfluß auf das Verhalten von Adam. Ein geringes Energieniveau löst bei Adam ein Bedürfnis nach Nahrung aus. Dieses Bedürfnis führt schließlich dazu, daß Adam zur Nahrungssuche motiviert wird. Das zugrundeliegende Motiv *Hunger* wird dabei um so stärker, je größer das Bedürfnis nach Nahrung ist (s. dazu auch Kapitel 10.1.2.3).

Emotionale Zustandsvariablen

Neben den physischen Zustandsvariablen verfügt der Agent Adam auch über Zustandsvariablen, die seinen emotionalen Zustand beschreiben. Um das Modell möglichst einfach und übersichtlich zu halten, beschränken wir uns allerdings auch hier ganz bewußt auf die Betrachtung einer einzigen Emotion, nämlich der Angst.

Die Modellierung der emotionalen Dynamik von Adam beruht im wesentlichen auf der psychoevolutionären Emotionstheorie des amerikanischen Psychologen Robert Plutchik (Plutchik, 1994). Plutchik geht von der Grundannahme aus, daß Emotionen im Rahmen der Phylogenese zur Bewältigung von grundlegenden Anpassungsproblemen entstanden sind. Er identifiziert Emotionen mit komplexen Abfolgen von Reaktionen auf bestimmte Reize und nennt unter anderem kognitive Bewertungen, Veränderungen im subjektiven Erleben, Handlungsimpulse, sowie konkrete Verhaltensweisen als wesentliche Bestimmungsstücke von Emotionen (Plutchik, 1984).

Die Abbildung 10.6 zeigt die Aktualgenese von Emotionen gemäß der Theorie von Plutchik (Plutchik, 1993). Am Beginn einer Sequenz von emotionalen Reaktionen steht ein auslösendes Ereignis. Das Ereignis wird zunächst wahrgenommen und einer kognitiven Bewertung unterzogen. Die auf diese Weise gewonnene Einschätzung bewirkt daraufhin eine Modifikation des emotionalen sowie des physiologischen Zustands. Als Konsequenz dieser Zustandsänderung werden emotionsspezifische Handlungstendenzen und schließlich ganz konkrete, beobachtbare Verhaltensweisen aktiviert, deren Auswirkungen zu einem in der Theorie

nicht näher spezifizierten Gleichgewichtszustand zwischen Person und Situation führen sollen.

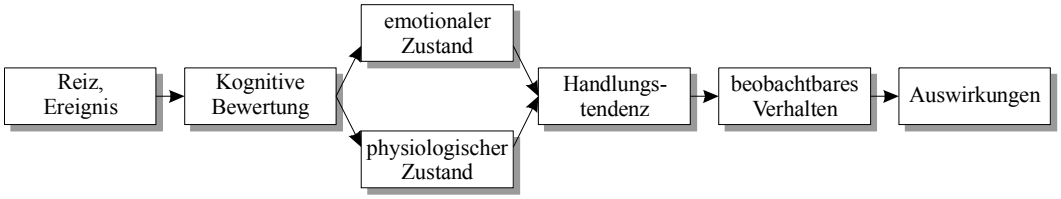

Abbildung 10.6: Die Aktualgenese von Emotionen nach Robert Plutchik (1993)

Wie eingangs schon erwähnt, soll der Agent Adam mit der Emotion Angst ausgestattet werden. Gemäß der zugrundeliegenden, psychoevolutionären Theorie muß nun also angegebenen werden, unter welchen Umständen die Emotion Angst entsteht, wie sie im Modell repräsentiert werden soll und welche Konsequenzen sie für die Handlungen des Agenten Adam haben soll.

Die Emotion Angst entsteht, wenn der Agent Adam bei seinen Erkundungen der Umwelt oder bei der Nahrungssuche auf ein Gefahrenfeld gerät. Nachdem Adam wahrgenommen hat, daß er sich auf einem Gefahrenfeld befindet, schätzt er diese Situation als ernste Bedrohung für seinen körperlichen Zustand ein, da es ihm nur mit großer Mühe und großem Energieaufwand gelingen wird, sich wieder aus der Gefahrenstelle zu befreien. Als unmittelbare Folge dieser Einschätzung wird Adam ängstlich.

Die Emotion Angst wird im Modell durch die Zustandsvariable *Angst* repräsentiert. Je höher der aktuelle Wert der Zustandsvariable *Angst* zu einem betrachteten Zeitpunkt ist, um so ängstlicher ist Adam. Immer dann, wenn Adam kognitiv realisiert, daß er sich in einer Gefahrenstelle befindet, wird der aktuelle Wert der Zustandsvariable *Angst* diskret um einen konstanten Wert *AngstZunahme* erhöht. Allerdings ist der Wert für die Zustandsvariable *Angst* nach oben hin durch einen Maximalwert *AngstMax* begrenzt. Die Gleichung 10.4 wird im Modell verwendet, um den neuen Wert der Zustandsvariable *Angst* nach der kognitiven Realisierung einer Bedrohung zu berechnen:

$$Angst\ (t_{n+1}) := Min\ (Angst\ (t_n) + AngstZunahme,\ AngstMax) \qquad \text{(Gl. 10.4)}$$

Dabei bezeichnet *Min (x, y)* eine Funktion, die das Minimum der beiden Zahlen *x* und *y* berechnet.

Von physiologischen Korrelaten von Emotionen, wie z. B. einer erhöhten Aktivität des autonomen Nervensystems bei Angst, die Plutchik in seiner Theorie anführt, soll in unserem Modell der Einfachheit halber abstrahiert werden.

Emotionale Zustände gehen gemäß der Theorie von Plutchik (1980) mit Dispositionen zu bestimmten Verhaltensweisen einher. So verstärkt Angst beispielsweise die Tendenz eines Organismus, Verhaltensweisen auszuwählen, die dem eigenen Schutz zuträglich sind. Im Modell Adam berücksichtigen wir diesen Sachverhalt in der Weise, daß der Agent Adam vorsichtiger handelt, wenn er sich in

einem ängstlichen Zustand befindet. Übersteigt der aktuelle Wert der Zustandsvariable *Angst* einen vorgegebenen Schwellwert *AngstSchranke*, so unterzieht der Agent Adam die Felder, die er zu betreten plant, zunächst einer eingehenden Untersuchung aus der Ferne, bevor er sie tatsächlich betritt. Auf diese Weise kann Adam verhindern, daß er auf Gefahrenfelder gerät und dadurch seine Physis erneut Schaden nimmt. Die Abbildung 10.7 stellt noch einmal die Entstehung sowie die Auswirkungen der Emotion Angst im Überblick dar.

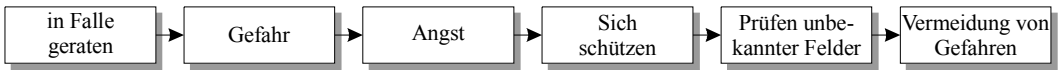

Abbildung 10.7: Entstehung und Auswirkungen der Emotion Angst

Adam wird nicht für den gesamten Rest seines virtuellen Lebens ängstlich bleiben und damit vorsichtig handeln. Vielmehr klingt Adams Angst im Laufe der Zeit wieder ab. Dieser Sachverhalt wird im Modell durch einen kontinuierlichen, negativ exponentiellen Abstieg der Werte für die Zustandsvariable *Angst* repräsentiert. Die Gleichung 10.5 zeigt die in diesem Zusammenhang verwendete Differentialgleichung. Die Variable *AngstAbnahme* beschreibt dabei einen konstanten Abnahmefaktor.

$$Angst'(t) := AngstAbnahme * Angst(t) \qquad \text{(Gl. 10.5)}$$

Für die Werte der Zustandsvariable *Angst* ergibt sich damit typischerweise ein Verlauf, wie er in 10.8 dargestellt ist. Der sprunghafte Anstieg der Kurve zwischen den Zeitpunkten 0 und 20 ist dadurch bedingt, daß Adam in diesem Zeitraum auf ein Gefahrenfeld gerät. Im Laufe der Zeit nimmt Adams Angst jedoch in Anlehnung an Gleichung 10.5 wieder ab. Die Angst wirkt sich allerdings so lange auf das Verhalten von Adam aus, wie der aktuelle Wert der Zustandsvariable *Angst* höher als der Schwellwert *AngstSchranke* liegt, der in Abbildung 10.8 durch die horizontale Linie auf Höhe des Wertes 50 angedeutet ist.

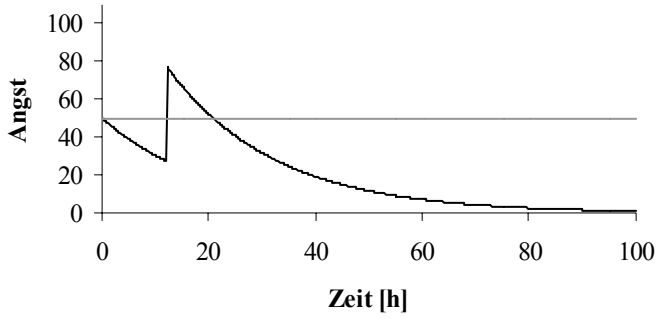

Abbildung 10.8: Typischer Verlauf der Zustandsvariable Angst in der Zeit

Abschließend sollen noch einmal alle für die Berechnung des emotionalen Zustands von Adam relevanten Größen im Überblick dargestellt werden (s. Abbildung 10.9):

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
Angst	[MStE]	50	Zustandsvariable Angst
AngstZunahme	[MStE]	60	Zunahme der Angst nach dem Erkennen einer Gefahr
AngstMax	[MStE]	100	Maximalwert der Zustandsvariable *Angst*
AngstSchranke	[MStE]	50	Grenzwert für die Zustandsvariable Angst, ab dem sich Adams Angst auf sein Verhalten auswirkt
AngstAbnahme	[1/h]	-0.05	Abnahmefaktor für die Zustandsvariable *Angst*

Abbildung 10.9: Die Variablen für die Berechnung des emotionalen Zustands des Agenten Adam

Kognitive Zustandsvariablen und Prozesse

Der Agent Adam benötigt, um sich in seiner Welt zurechtzufinden, eine gewisse Menge an Informationen und kognitiven Fähigkeiten. Die Informationen, auf die Adam bei der Bestimmung seiner Verhaltensweisen zurückgreifen kann, werden dabei zustandsorientiert modelliert. Der interne Zustand unseres Agenten wird also nun um Zustandsvariablen erweitert, die seinen kognitiven Zustand beschreiben. Zudem werden kognitive Prozesse, wie beispielsweise Lernen oder Vergessen, eingeführt, die auf dem kognitiven Zustand von Adam operieren und diesen in geeigneter Weise modifizieren.

Permanentes Wissen. Adams Wissen setzt sich aus permanenten, d. h. von Beginn an vorhandenen und überdauernden, wie auch aus dynamischen Anteilen, die im Laufe der Zeit durch perzeptive und informationsverarbeitende Prozesse erworben werden, zusammen.

Eine gewisse Menge an permanenten Informationen wird dem Agenten Adam von Beginn an zur Verfügung gestellt. So weiß Adam etwa, daß sich die Nahrungsmenge auf abgeernteten Nahrungsfeldern im Laufe der Zeit wieder regenerieren wird und daß die Regeneration für alle Nahrungsfelder mit der gleichen Geschwindigkeit erfolgt. Darüber hinaus kennt Adam die Dimension seiner Umwelt

und auch die Menge und Konsequenzen der Aktionen, die er auszuführen in der Lage ist.

Dynamisches Wissen. Adams Wissen ist nicht auf statische Inhalte beschränkt. Vielmehr sammelt Adam bei den Streifzügen durch seine Umwelt Erfahrungen, die er nutzt, um eine dynamische mentale Repräsentation verschiedener Sachverhalte aufzubauen.

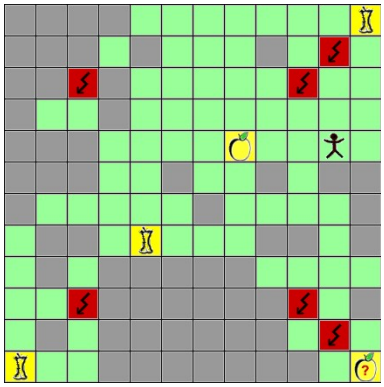

Abbildung 10.10: Momentaufnahme der kognitiven Landkarte des Agenten Adam

Von zentraler Bedeutung ist in dieser Hinsicht eine Repräsentation der Umwelt, die als kognitive Landkarte bezeichnet werden kann. In dieser Landkarte hinterlegt Adam Informationen über erkundete Felder in seiner Umwelt, damit er in zukünftigen Entscheidungs- und Planungssituationen auf diese Informationen zurückgreifen kann. Zu den im Umweltmodell gespeicherten Informationen zählen beispielsweise die Feldart, die gegebenenfalls auf einem Feld festgestellte Nahrungsmenge, oder auch der Zeitpunkt, zu dem Adam auf einem bestimmten Feld zum letzten mal gegessen hat. Die mentale Repräsentation der Umwelt baut Adam dabei strukturähnlich zu seiner tatsächlichen Umwelt auf, d. h. die kognitive Landkarte hat ebenso wie die Umwelt die Gestalt eines zweidimensionalen Gitters. Die Informationen im Umweltmodell werden erneuert, sobald Adam ein Feld erkundet. Allerdings können Informationen im Umweltmodell im Laufe der Zeit auch durch Vergessensprozesse verloren gehen.

Die Abbildung 10.10 zeigt eine Momentaufnahme von Adams kognitiver Landkarte. Die dunkel dargestellten Felder bezeichnen Positionen in Adams Umwelt, über die er noch keine Kenntnisse erworben bzw. deren Bedeutung er in der Zwischenzeit bereits wieder vergessen hat. Die helleren Felder symbolisieren bekannte neutrale Felder. Die mit einem Blitz markierten Felder repräsentieren Gefahrenstellen und die helleren, mit einem Apfel versehenen Felder weisen auf Nahrungsfelder hin.

Die Menge an Wissen über seine Umwelt, das Adam im Laufe der Zeit angesammelt hat, wird mit Hilfe der kognitiven Zustandsvariable *Wissensstand* quantifiziert. Adams Wissensstand korrespondiert dabei mit der Anzahl an bekannten

Feldern in seinem Umweltmodell und beeinflußt in entscheidender Weise auch die Auswahl seiner Verhaltensweisen (s. Kapitel 10.1.2.3).

Weiterhin gehören zu Adams dynamischem Wissen Informationen über seine aktuelle Position in der Umwelt, sowie über den Zustand von Nahrungsfeldern, d. h. wieviel Nahrung Adam zu einem gegebenen Zeitpunkt auf einem Nahrungsfeld vermutet (s. Kapitel 10.1.1). In Abbildung 10.10 sind Nahrungsfelder, von denen Adam sicher weiß, daß sie Nahrung bieten, mit Hilfe eines unversehrten Apfels dargestellt. Nahrungsfelder, die noch leer sind, werden mit einem verspeisten Apfel gekennzeichnet. Schließlich kann es auch vorkommen, daß Adam nicht ganz sicher ist, ob sich auf einem Feld bereits wieder Nahrung befindet. Derartige Felder werden mit Hilfe eines Apfels, der mit einem Fragezeichen versehen ist, kenntlich gemacht.

Neben Informationen über seine Umwelt enthält Adams Kognition auch ein einfaches Element der Selbstreflektion. Adam wird mit einem Kompetenzempfinden (Dörner, 1999) ausgestattet, das in einfacher Weise den Sachverhalt modellieren soll, wie leistungsfähig Adam sich selbst einschätzt. Das Kompetenzempfinden spiegelt also das subjektive Zutrauen von Adam in seine eigenen Fähigkeiten wider. In unserem Modell bezieht Adam die Kompetenz auf sein Geschick bei der Nahrungssuche. Ist Adam bei der Nahrungssuche erfolgreich, so erhöht sich sein Kompetenzempfinden. Sinken wird seine Kompetenz hingegen, wenn die Nahrungssuche erfolglos bleibt. Adams Einschätzung seiner eigenen Fähigkeiten wirkt sich auf seine Verhaltenssteuerung und insbesondere auf seine Planung aus. Schätzt Adam in einer gegebenen Situation seine Kompetenz sehr gering ein, wird er bei der Auswahl seiner Verhaltensweisen vorsichtiger vorgehen (s. Kapitel 10.1.2.3: Die Auswahl einer Handlungsalternative). Ist Adam allerdings selbstbewußt und fühlt sich kompetent, so traut er sich in Entscheidungssituationen mit unsicherem Ausgang auch riskantere Strategien zu.

Nachdem in den vorangegangenen Abschnitten die wichtigsten kognitiven Eigenschaften und Strukturen von Adam eingeführt wurden, wenden wir uns nun den Gedächtnisfunktionen bzw. kognitiven Prozessen zu, die diese Eigenschaften und Strukturen modifizieren. Zu diesen Prozessen zählen Lernen, Behalten, Vergessen und Erinnern. Weiterhin verfügt Adam über die Fähigkeit, seine Handlungen auf der Grundlage seines Wissens über sich selbst und seine Umwelt im voraus zu planen. Diese Fähigkeit wird zu einem späteren Zeitpunkt im Zusammenhang mit Adams Verhaltenssteuerung (s. Kapitel 10.1.2.3) noch ausführlicher diskutiert.

Lernen. Lernen wird in der Psychologie gemeinhin als Sammelbegriff für eine Vielzahl von Prozessen verwendet, die zum Erwerb oder der Veränderung von Wissen und damit auch zu einer Veränderung von Fähigkeiten und Verhaltensweisen führen (Plötzner, 1996). Auf der Grundlage dieser Definition könnte bereits der Aufbau und die Anpassung von Adams Umweltmodell als Lernprozess angesehen werden. Allerdings ist Adam in der Lage, noch einen weiteren Sach-

verhalt über seine Umwelt zu lernen. Durch sukzessives Erkunden seiner Umwelt und entsprechende Auswertung der dadurch erhaltenen Informationen kann Adam im Laufe der Zeit lernen, wie lange es dauert, bis auf einem von ihm abgeernteten Nahrungsfeld wieder Nahrung verfügbar sein wird. Diese Information kann Adam auf alle Nahrungsfelder in seiner Umwelt übertragen, da für alle Nahrungsfelder der Einfachheit halber ein identischer Regenerationsprozeß zugrunde gelegt wird.

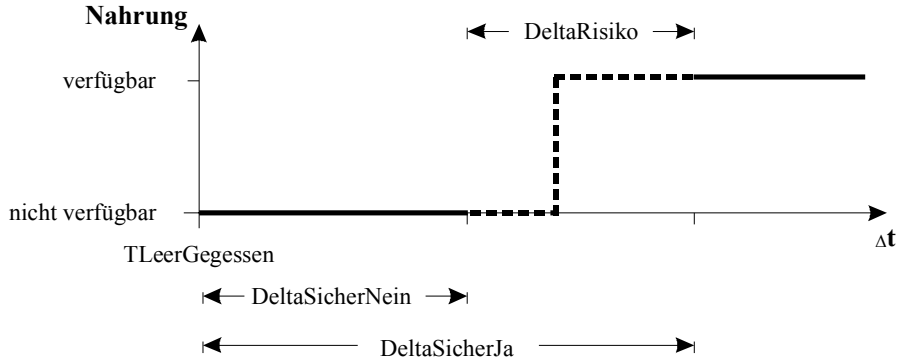

Abbildung 10.11: Die Lernschranken für die Regeneration von Nahrungsstellen

Um entsprechende Informationen über das zeitliche Verhalten der Nahrungsfelder zu erhalten, geht Adam in der folgenden Weise vor: sobald Adam auf einem Nahrungsfeld die vorhandene Nahrung aberntet, vermerkt er bei dem entsprechenden Feld in seinem Umweltmodell den zugehörigen Zeitpunkt *TLeerGegessen*. Wenn Adam dann zu einem späteren Zeitpunkt *T* wieder zu diesem Feld zurückkehrt, kann er durch eine Untersuchung dieses Feldes feststellen, ob bereits wieder Nahrung vorhanden ist. Ist dies nicht der Fall, so kann Adam den Schluß ziehen, daß sich nach *(T – TLeerGegessen)* Zeiteinheiten das entsprechende Nahrungsfeld noch nicht wieder regeneriert hat. Die Information über die Zeitspanne, in der sicher noch keine Nahrung nachgewachsen ist, legt Adam in der Variable *DeltaSicherNein* ab. Falls Adam jedoch auf dem untersuchten Feld zum Zeitpunkt *T* bereits wieder Nahrung vorfindet, kann er daraus schließen, daß es maximal *(T – TLeerGegessen)* Zeiteinheiten bedarf, bis auf einem abgeernteten Nahrungsfeld wieder Nahrung vorhanden ist. Entsprechend kann er die obere Lernschranke *DeltaSicherJa* aktualisieren, die die zugehörige Zeitspanne bezeichnet. Im Laufe von mehreren Beobachtungen wird Adam in der Lage sein, die untere Lernschranke *DeltaSicherNein* immer weiter nach hinten zu korrigieren und die obere Lernschranke *DeltaSicherJa* immer weiter nach vorne zu verschieben. Dadurch erhält er eine immer genauere Information darüber, ab welchem Zeitpunkt auf abgeernteten Nahrungsfeldern wieder Nahrung verfügbar ist. Insbesondere kann Adam das Zeitintervall *DeltaRisiko*, über das noch keine genaueren Informationen vorliegen und das durch die beiden Lernschranken *DeltaSicherJa* und *DeltaSicherNein* beschränkt wird, immer weiter schrumpfen. Adams Prognosen gewinnen also im Laufe der Zeit immer mehr an Präzision und steigern entsprechend seine Überlebensfähigkeit.

In Abbildung 10.11 sind Adams Informationen über das Verhalten der Nahrungsfelder noch einmal im Überblick dargestellt. Die durchgezogenen, dicken Linienabschnitte bezeichnen dabei die Bereiche, in denen Adam sichere Informationen über den Zustand der Nahrungsfelder zur Verfügung hat. Der gestrichelte Abschnitt stellt nur einen von mehreren möglichen Kurvenverläufen dar. Da Adam im Zeitintervall *DeltaRisiko* noch auf keine Erfahrungen mit dem tatsächlichen Verhalten der Nahrungsfelder zurückgreifen kann, weiß er in dieser Situation noch nicht, wann exakt im Zeitraum *DeltaRisiko* der Anstieg der Nahrung von *nicht verfügbar* auf *verfügbar* erfolgt.

Behalten und Vergessen. Behalten und Vergessen werden in der Psychologie als komplementäre Prozesse angesehen (Dörner & Selg, 1996). Das Behalten bezeichnet dabei den Transport von gelernten Informationen über die Zeit. Informationen, die nicht behalten werden, werden vergessen.

Grundsätzlich ist Adam in der Lage, verschiedene Informationen zu speichern und damit zu behalten. Auf diese Informationen kann Adam dann zu späteren Zeitpunkten wieder zurückgreifen und sie im Rahmen seiner Verhaltenssteuerung einsetzen. Allerdings kommt es auch vor, daß Adam Informationen wieder vergißt.

Informationen über die im letzten Abschnitt beschriebenen Lernschranken bleiben für unbegrenzte Zeit in Adams Gedächtnis erhalten. Aufgrund ihrer Wichtigkeit wird Adam diese Informationen nicht vergessen. Anders sieht es dagegen mit den Informationen über die einzelnen Felder seiner Umwelt aus. Adam verliert das Wissen über Felder seiner Umwelt aus dem Gedächtnis, sofern es für längere Zeit nicht benötigt bzw. aktualisiert wird.

Dieser Sachverhalt wird mit Hilfe einer festen Zeitspanne *DeltaVergessen* modelliert. Wird in Adams Gedächtnis eine Information über ein Feld in der Umwelt nicht spätestens nach *DeltaVergessen* Zeiteinheiten (durch Ausführung der Aktion *Explorieren* auf diesem Feld) aufgefrischt, so wird sie aus Adams Gedächtnis entfernt. Derartige Informationen haben also grundsätzlich eine Latenzzeit von *DeltaVergessen* Zeiteinheiten. Diese Zeitspanne verlängert sich allerdings, sofern Adam noch bevor die Informationen vergessen wird das zugehörige Feld erneut exploriert. In diesem Fall verlängert sich die Latenzzeit der Information um weitere *DeltaVergessen* Zeiteinheiten.

Diese Art der Modellierung fußt zwar der Einfachheit halber nicht unmittelbar auf einer psychologischen Theorie. Jedoch erinnern die sich einstellenden Effekte sehr stark an das bei Dörner und Selg (1996) angesprochene Peters-Prinzip. Diesem Prinzip zufolge werden emotional positiv gefärbte Gedächtnisinhalte in der Regel länger behalten als emotional negative und diese wiederum länger als emotional neutrale. Als emotional positiv getönte Gedächtnisinhalte können im Modell Adam die Nahrungsfelder interpretiert werden, selbst wenn zunächst keine Emotion vorgesehen ist, die explizit mit der Nahrungsaufnahme korreliert ist. Gefahrenstellen können als emotional negativ eingestuft werden und schließlich

neutrale Felder als emotional neutral. Dadurch, daß Adam bei seiner Nahrungssuche häufig auf bereits besuchte Nahrungsfelder trifft, werden Informationen über Nahrungsfelder auch sehr häufig in Adams Gedächtnis aufgefrischt. Bei jeder Aktualisierung eines Feldes im Gedächtnis verlängert sich die Latenzzeit *TVergessen* der Information um die Zeitspanne *DeltaVergessen*. Bei Nahrungsfeldern wandert der Zeitpunkt des Vergessens also automatisch immer weiter in die Zukunft. Ein ähnlicher Zusammenhang gilt auch für Gefahrenfelder, die Adam mehrmals exploriert, um festzustellen, ob er sich innerhalb der Gefahrenstelle oder bereits außerhalb befindet. Auch hier verlängert sich die Latenzzeit der Informationen im Vergleich zu Informationen über neutrale Felder, die weniger oft untersucht werden. In dieser Hinsicht läßt sich in bezug auf die Konsequenzen also durchaus eine Korrespondenz mit dem Peters-Prinzip erkennen.

Erinnern. Adam hat im Gegensatz zum Menschen zu jedem Zeitpunkt die Möglichkeit, auf bestehende Gedächtnisinhalte zurückzugreifen. Das Problem des Erinnerns von Informationen aus dem Gedächtnis bleibt also aus Gründen der Übersichtlichkeit und Einfachheit für unseren Agenten unberücksichtigt.

In Abbildung 10.12 sind abschließend noch einmal alle Modellgrößen in einer Übersicht zusammengestellt, die die kognitiven Eigenschaften und Prozesse des Agenten Adam beschreiben.

Bezeichner	Einheit	Initialwert	Erläuterung
Feldart	[]	‚Unbekannt'	Art eines Feldes der Umwelt
Nahrungsmenge	[EnE]	0	Nahrungsmenge auf einem Feld
Wissensstand	[WiSE]	0	Anzahl der bekannten Felder
XAdam	[]	5	x-Koordinate von Adam in seiner kognitiven Landkarte
YAdam	[]	5	y-Koordinate von Adam in seiner kognitiven Landkarte
InFalle	[]	FALSE	Indikator, der anzeigt, ob sich Adam in einer Falle befindet
Kompetenz	[KoE]	0	Kompetenzempfinden
KompetenzPlus	[KoE]	10	Kompetenzzuwachs bei erfolgreicher Nahrungssuche
KompetenzMinus	[KoE]	20	Kompetenzverlust bei erfolgloser Nahrungssuche
KompetenzMin	[KoE]	0	Untergrenze für Kompetenz
KompetenzMax	[KoE]	100	Obergrenze für Kompetenz
KompetenzSchranke	[KoE]	50	Schwellwert für die Berücksichtigung der Kompetenz im Rahmen der Verhaltenssteuerung (s. Abschnitt 10.1.2.3)
TLeerGegessen	[h]	0	Zeitpunkt, zu dem ein bestimmtes Nahrungsfeld abgeerntet wurde
DeltaSicherNein	[h]	0	Untere Lernschranke für das Nahrungswachstum
DeltaSicherJa	[h]	10000	Obere Lernschranke für das Nahrungswachstum
DeltaVergessen	[h]	50	Standardwert für die Verfügbarkeitsdauer eines Gedächtniseintrages
TVergessen	[h]	0	Zeitpunkt, an dem ein Eintrag aus dem Gedächtnis entfernt wird

Abbildung 10.12: Die Variablen für die Berechnung des kognitiven Zustands des Agenten Adam

10.1.2.2 Adams Aktionsrepertoire

Der Agent Adam verfügt über ein Repertoire an unterschiedlichen Aktionen, mit deren Hilfe er sowohl auf den Zustand seiner Umwelt (externe Aktionen) als auch auf seinen eigenen, internen Zustand (interne Aktionen) Einfluß nehmen kann. Die einzelnen Aktionen stehen dabei zu Adams freier Disposition und werden im Rahmen seiner Verhaltenssteuerung ausgewählt. Der Einfachheit halber gehen wir in diesem Modell von der Annahme aus, daß Adam immer aktiv ist, d. h. zu jedem Zeitpunkt eine Aktion ausführt und nie ruht. Die Ausführung von Aktionen ist mit einem spezifischen Verbrauch an Energie verbunden, der bereits im Abschnitt *Physische Zustandsvariablen* näher erläutert wurde.

Interne Aktionen

Mit Hilfe von internen Aktionen steuert Adam zum einen seinen Wahrnehmungsapparat und damit die Aufnahme von externen Informationen aus der Umwelt. Zum anderen stößt er auf deren Grundlage auch Planungsprozesse an, die dazu dienen, sein Handeln im Hinblick auf ein bestimmtes Ziel für eine gewisse Zeit im voraus zu strukturieren. Adams Repertoire an internen Aktionen umfaßt die Aktionen *Planen*, *Explorieren* und *Prüfen*, die nun in den folgenden Abschnitten etwas genauer betrachtet werden.

Die Aktion Planen. Mit Hilfe der Aktion *Planen* initiiert Adam einen Denkprozeß, der in verschiedenen Situationen zur Neuorganisation seines Verhaltens führen soll (s. Kapitel 4.2.3.2). Er geht dabei von dem aktuell vorliegenden, handlungsleitenden Motiv aus. In Abhängigkeit dieses Motivs und der im Gedächtnis gespeicherten Informationen legt Adam zunächst einmal ein geeignetes Handlungsziel fest. Im nächsten Schritt konstruiert Adam einen Handlungsplan, der ihn zum gesteckten Ziel führen soll. Im vorliegenden Fall besteht ein derartiger Handlungsplan aus einer Folge von Teilzielen, die Adam durch die Ausführung von geeigneten Aktionen sukzessive zu erreichen versucht. Bei der Konstruktion des Handlungsplans greift Adam auf Information, wie beispielsweise die kognitive Landkarte seiner Umwelt, zurück, die in seinem Gedächtnis gespeichert sind. Ebenso legt er den erzeugten Handlungsplan nach Fertigstellung in seinem Gedächtnis ab, damit er in der Folgezeit für die Ausführung zur Verfügung steht. Eine detailliertere Beschreibung der Verhaltenssteuerung und Planungsfähigkeiten des Agenten Adam erfolgt in Kapitel 10.1.2.3.

Die Aktion Explorieren. Die Aktion *Explorieren* dient der genaueren Untersuchung des Feldes in der Umwelt, auf dem sich Adam zum gegebenen Zeitpunkt gerade befindet. Als Resultat der Aktion *Explorieren* erhält Adam die folgenden

Informationen, die er für eine spätere Verwendung im Rahmen der Verhaltensauswahl in seinem Gedächtnis aufbewahrt.

Adam erfährt mit Hilfe der Aktion *Explorieren* zunächst einmal, an welcher Stelle er sich gerade in seiner Umwelt befindet. Diese Information nutzt er, um seine eigene Position, angegeben durch die kognitiven Zustandsvariablen *XAdam* und *YAdam* (vgl. Abbildung 10.12), in seinem Gedächtnis zu aktualisieren.

Zudem stellt Adam fest, welcher Art das explorierte Feld ist. Er kann also in seinem Gedächtnis (Variable *Feldart*) vermerken, ob es sich bei dem vorliegenden Feld um ein neutrales Feld, ein Gefahrenfeld oder ein Nahrungsfeld handelt.

Befindet sich Adam auf einem Gefahrenfeld, so stellt er mit Hilfe der Aktion *Explorieren* zusätzlich fest, ob er noch in der Falle sitzt oder sich bereits aus der Gefahrenstelle befreit hat. Auf der Grundlage dieser Information kann er die kognitive Zustandsvariable *InFalle* an die aktuellen Gegebenheiten anpassen.

Falls Adam auf ein Nahrungsfeld gestoßen ist, erkundet er, welche Menge an Nahrung das vorliegende Nahrungsfeld bereithält, und legt diese Information in der Variable *Nahrungsmenge* des jeweiligen Nahrungsfeldes ab. Hier gilt allerdings die Einschränkung, daß Adam sehr kleine Mengen an Nahrung nicht wahrnehmen kann. Sein Wahrnehmungsapparat arbeitet selektiv. Adam kann also Reize erst dann wahrnehmen und weiterverarbeiten, wenn sie über einer gewissen, vorgegebenen Wahrnehmungsschwelle liegen. In bezug auf die Nahrungsmenge ist die Wahrnehmungsschwelle *NahrungWahrnehmungMin* bei 10 Energieeinheiten angesiedelt. Liegt also zum Zeitpunkt des Explorierens die tatsächlich auf einem Nahrungsfeld verfügbare Nahrungsmenge unter dem Wert 10, so geht Adam von der falschen Annahme aus, daß das Nahrungsfeld noch leer ist und keine Nahrung beinhaltet.

Die Aktion Prüfen. Neben der Aktion *Explorieren* steht dem Agenten Adam eine weitere Aktion zur Verfügung, mit deren Hilfe er Informationen über seine Umwelt sammeln kann. Es handelt sich dabei um die Aktion *Prüfen*, die Adam ausführt, wenn er sich in einem ängstlichen Zustand befindet. Um Gefahren zu vermeiden, inspiziert Adam mit Hilfe der Aktion *Prüfen* unbekannte Felder, die in seinem Handlungsplan vorgesehen sind, bevor er diese tatsächlich betritt. Er stellt dabei fest, ob sich hinter dem geprüften, unbekannten Feld ein Gefahrenfeld verbirgt, und erhält dadurch die Möglichkeit, potentiellen Gefahrenstellen aus dem Weg zu gehen. Stößt Adam beim *Prüfen* auf eine Gefahrenstelle, so verwirft er seinen bisherigen Handlungsplan und führt eine erneute Planung durch, wobei die Information über die erkannte Gefahrenstelle bereits berücksichtigt wird. Stellt sich ein Feld beim Prüfen als neutrales oder Nahrungsfeld heraus, so fährt Adam mit seinen geplanten Aktionen fort.

Die Aktion *Prüfen* liefert nur eine eingeschränkte Information über ein Feld in der Umwelt. Benötigt Adam weitere Informationen über das Feld, die durch die Aktion *Prüfen* nicht bereitgestellt werden, so muß er zu einem späteren Zeitpunkt zusätzlich die Aktion *Explorieren* auf dem entsprechenden Feld ausführen.

Externe Aktionen

Der Agent Adam ist in der Lage, mit Hilfe von externen Aktionen den Zustand seiner Umwelt, sowie seine eigene Position innerhalb der Umwelt zu verändern. Zu Adams Repertoire an externen Aktionen gehören die Aktionen *Gehen*, *Befreien* und *Essen*.

Die Aktion Gehen. Durch Ausführung der Aktion *Gehen* erhält Adam die Möglichkeit, sich in seiner Umwelt zu bewegen. Die auf diese Weise gewonnene Mobilität versetzt den Agenten in die Lage, seine Umwelt zu erkunden und darin nach Nahrung zu suchen.

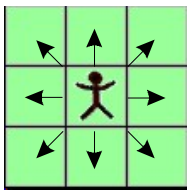

Abbildung 10.13: Bewegungsmöglichkeiten des Agenten Adam durch Ausführung der Aktion Gehen

Für die Bewegung stehen Adam insgesamt acht Möglichkeiten zur Verfügung (s. Abbildung 10.13). Als Zielfelder der Aktion *Gehen* kommen alle unmittelbar angrenzenden Nachbarfelder desjenigen Feldes in Frage, auf dem sich Adam zum betrachteten Zeitpunkt gerade befindet. Eingeschränkt werden Adams Bewegungsmöglichkeiten allerdings am Rand seiner Umwelt, da diese im vorliegenden Fall begrenzt und nicht, wie beispielsweise in Fallstudie I, als Torus konzipiert ist.

Die Aktion Befreien. Falls Adam in eine Gefahrenstelle geraten ist, kann er sich, wenn auch nur mit einem sehr hohen Energie- und Zeitaufwand, wieder aus dieser Gefahrenstelle befreien. Durch Ausführung der Aktion *Befreien* verbleibt Adam auf demselben Feld in seiner Umwelt, jedoch ändert sich seine Position innerhalb dieses Feldes. Er befindet sich danach nicht mehr innerhalb der Gefahrenstelle dieses Feldes, sondern außerhalb und kann wieder mit geplanten Verhaltensweisen fortfahren.

Die Aktion Essen. Mit Hilfe der Aktion *Essen* ist Adam in der Lage, verbrauchte Energie zu regenerieren. Hat Adam ein Nahrungsfeld erreicht, das zum betreffenden Zeitpunkt eine ausreichende Menge an Nahrung bereitstellt, kann er seiner Physis durch Ausführung der Aktion *Essen* neue Energie zuführen.

Die Dauer der Aktion *Essen* ist im Gegensatz zu den anderen Aktionen variabel. Adam kann je Zeiteinheit nur eine gewisse Maximalmenge an Energie zu sich

nehmen. Je größer also die Menge an Nahrung ist, die Adam konsumieren möchte, um so länger dauert der Vorgang des Essens.

Die beim Essen aufgenommene Energie wird in kontinuierlicher Weise der physischen Zustandsvariable *Energie* zugerechnet. Sofern auf einem Nahrungsfeld eine ausreichende Menge an Nahrung, gemessen in Energieeinheiten [EnE], verfügbar ist, füllt Adam seinen Energievorrat bis zur Obergrenze *EnergieMax* (vgl. Abbildung 10.4) auf. Sollte nach dem Essen noch eine gewisse Restmenge an Nahrung auf dem Nahrungsfeld verblieben sein, so verdirbt dieser Rest, so daß der Wachstumsprozeß der Nahrung in der Umwelt in jedem Fall beim Minimalwert *NahrungsMin* startet. Wenn ein Nahrungsfeld hingegen nicht genügend Nahrung bietet, um Adams Hunger gänzlich zu stillen, muß Adam nach dem Essen unter Umständen noch weitere Nahrungsfelder aufsuchen.

Dauer der Aktionen

Die Ausführung von Aktionen ist mit einer entsprechenden Zeitdauer, sowie mit Ausnahme der Aktion *Essen* mit einem gewissen Energieverbrauch verbunden. Während eine Aktion ausgeführt wird, nimmt Adams Energie zusätzlich zum Grundverbrauch kontinuierlich mit dem für die jeweilige Aktion angegebenen Verbrauchsfaktor (vgl. Abbildung 10.4) ab. Ebenso ist jede Aktion mit einer individuellen Zeitdauer versehen, die angibt, welchen Zeitraum die Ausführung einer Aktion in Anspruch nimmt. In Abbildung 10.14 sind die spezifischen Zeitdauern für Adams interne und externe Aktionen im Überblick dargestellt.

Aktion	**Dauer [h]**
Planen	0.2
Explorieren	0.1
Prüfen	0.2
Gehen	0.2
Befreien	2.0
Essen	variabel, abhängig von der Nahrungsmenge

Abbildung 10.14: Zeitdauern für Adams interne und externe Aktionen

10.1.2.3 Adams Verhaltenssteuerung

Der Agent Adam verfügt sowohl über reaktive als auch über deliberative Verhaltensweisen. Die Konzeption der Verhaltenssteuerung orientiert sich dabei grundlegend an den Ansätzen von Gollwitzer (1990, 1991) und Heckhausen (1989). Die

menschliche Verhaltenssteuerung wird hier im wesentlichen als mehrstufiger Prozeß aufgefaßt, der bei Motiven, also inneren psychischen Kräften, ansetzt und letztendlich dazu dienen soll, die mit Motiven verbundenen Zielvorstellungen durch geschicktes Handeln zu erreichen.

Am Beginn dieses Prozesses stehen unterschiedliche Motive. Durch eine Auswahlstrategie wird zunächst eines dieser Motive selektiert, das dann für eine gewisse Zeit das Handeln des Organismus bestimmt. In Abhängigkeit des ausgewählten Motivs wird daraufhin eine entsprechende Zielsetzung bzw. ein geeignetes Handlungsziel festgelegt. Um das Handlungsziel realisieren zu können, muß nun eine Handlungsstrategie, d. h. eine spezifische Art des Vorgehens, gewählt und eine Planung der auszuführenden Handlungen durchgeführt werden. Liegt schließlich ein brauchbarer Handlungsplan vor, so wird mit der Umsetzung des Handlungsplans begonnen, indem entsprechende Aktionen der Reihe nach ausgeführt werden. Nach der vollständigen Abarbeitung eines Handlungsplans erfolgt eine abschließende Bewertung der erzielten Ergebnisse des Handelns und eine Wiederaufnahme von gegebenenfalls vorliegenden, zwischenzeitlich unterbrochenen Handlungen.

Die folgenden Abschnitte werden sich nun ausführlich mit den einzelnen Phasen, die für die Verhaltenssteuerung von Bedeutung sind, befassen.

Zustände, Bedürfnisse, Motive und Zielzustände

Wie in Kapitel 4 bereits eingeführt, werden Motive als psychische Kräfte angesehen, die das menschliche Handeln in Gang setzen, für eine gewisse Zeit aufrecht erhalten und auf ein bestimmtes Ziel hin ausrichten. Adam ist mit zwei derartigen Motiven ausgestattet: *Hunger* und *Wissenserwerb*. Die Motive korrelieren dabei über eine Kette von Abhängigkeiten mit den Bedürfnissen des Agenten, sowie mit seinen internen Zuständen.

Am Anfang der Wirkkette stehen die Zustände des Agenten. Maßgeblich sind dabei die physische Zustandsvariable *Energie*, sowie die kognitive Zustandsvariable *Wissensstand*. Ist Adams Energieniveau gering, so entwickelt er ein *Bedürfnis nach Nahrung*. Ebenso entsteht ein kognitives *Bedürfnis nach Wissen* in Abhängigkeit von Adams Wissensstand. Beiden Bedürfnissen wird eine Bedürfnisstärke zugeordnet, die deren Intensität beschreibt.

An diese beiden Bedürfnisse werden im nächsten Schritt Adams Motive gekoppelt. Aus dem Bedürfnis nach Nahrung entsteht das Motiv *Hunger*. Es handelt sich dabei in Anlehnung an Maslow (1955) um ein physiologisches Grundbedürfnis. Adams Bedürfnis nach Wissen bedingt das kognitive Motiv *Wissenserwerb*. Beide Motive werden mit einer spezifischen Motivstärke versehen, die angibt, mit welcher Dringlichkeit das jeweilige Motiv bei der Verhaltensauswahl berücksichtigt werden muß. Grundsätzlich werden die Handlungen des Agenten von demjenigen Motiv bestimmt, das zum betrachteten Zeitpunkt die höchste Motivstärke aufweist. Allerdings gewährt Adam bei vergleichbarer Intensität dem Motiv *Hun-*

ger Vorrang vor dem Motiv *Wissenserwerb*. Diese Annahme basiert auf der Bedürfnishierarchie nach Maslow (1955), die davon ausgeht, daß ein Organismus zunächst einmal versuchen wird, die physiologischen Grundbedürfnisse zu befriedigen, um seine Lebensfunktionen zu erhalten, bevor er sich den höheren und insbesondere kognitiven Bedürfnissen zuwendet.

Mit der Festlegung des handlungsleitenden Motivs wird schließlich ein Zielzustand aktiviert, den es anschließend durch Planung und Ausführung geeigneter Handlungen zu erreichen gilt. Bis zur Bestimmung des für die späteren Phasen der Verhaltenssteuerung gültigen Zielzustandes ergibt sich also eine Folge von Beeinflussungen, wie sie in Abbildung 10.15 noch einmal im Überblick dargestellt ist.

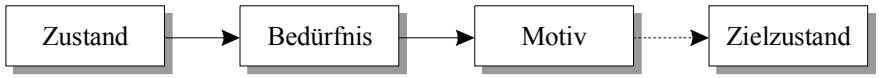

Abbildung 10.15: Wirkkette für die Entstehung von Motiven und Zielzuständen

Die Modellierung des Motivs Hunger. Für die Modellierung des Motivs *Hunger* sind, wie eingangs dieses Abschnittes bereits erwähnt, die physische Zustandsvariable *Energie*, sowie Adams Bedürfnis nach Nahrung *BedNahrung* von Bedeutung.

Adams Energieverbrauch hängt im wesentlichen von seinem Grundbedarf an Energie ab, ebenso wie von einem spezifischen Zusatzbedarf, der durch die Ausführung von Aktionen bedingt ist (s. Kapitel 10.1.2.1). Die Zustandsvariable *Energie* ändert sich also dynamisch in der Zeit gemäß den Gleichungen Gl. 10.2 und Gl. 10.3.

Unmittelbar aus dem aktuellen Wert für die Zustandsvariable *Energie* resultiert Adams Bedürfnis nach Nahrung *BedNahrung*. Dabei liegt die Annahme zugrunde, daß das Bedürfnis nach Nahrung um so größer sein wird, je weniger Energie Adam zur Verfügung hat. Dieser Sachverhalt läßt sich mit Hilfe der folgenden Gleichung beschreiben:

$$BedNahrung(t) := (EnergieMax - Energie(t)) / (Energie(t) + 1) \qquad \text{(Gl. 10.6)}$$

Die Modellgröße *EnergieMax* bezeichnet dabei den Sättigungspunkt für Adams Energie und ist mit 100 [EnE] vorbesetzt (s. Abschnitt 10.1.2.1). Ist Adams Energieniveau sehr hoch, d. h. nimmt die Zustandsvariable *Energie* einen Wert an, der sehr nahe am Maximalwert *EnergieMax* liegt, so tendiert Adams Bedürfnis nach Nahrung gegen den Wert 0. Besteht jedoch ein sehr großes Defizit an Energie, ausgedrückt durch eine große Differenz zwischen dem Istwert *Energie* und dem Sollwert *EnergieMax*, so steigt Adams Bedürfnis nach Nahrung entsprechend an. Der Wertebereich für die Modellgröße *BedNahrung* liegt gemäß dieser Modellierung im Intervall [0, 100] (s. Abbildung 10.16).

Die Bedürfnisstärke für das Bedürfnis nach Nahrung beeinflußt Adams Motivation zur Nahrungssuche. Je größer Adams Bedürfnis nach Nahrung ist, um so

höher wird auch seine Motivation sein, nach Nahrung zu suchen. In Anlehnung an Dörner (1999) wird für die Modellierung dieses Sachverhalts ein logarithmischer Zusammenhang zwischen der Bedürfnisstärke *BedNahrung* und der daraus resultierenden Motivstärke *MStHunger* zugrunde gelegt. Dadurch läßt sich erreichen, daß mit der Entstehung eines Bedürfnisses die Motivation zum Handeln sehr schnell ansteigt, auf der anderen Seite jedoch das Wachstum der Motivstärke auf einem hohen Niveau nur noch gemäßigt erfolgt.

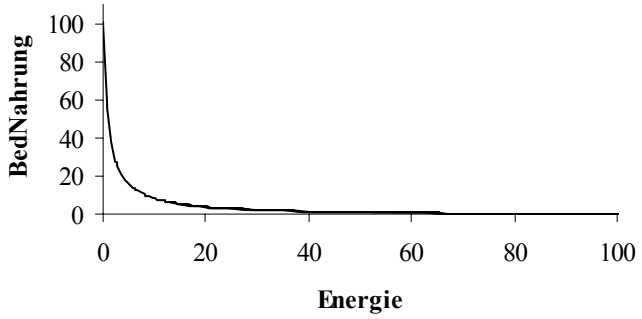

Abbildung 10.16: Der Verlauf des Bedürfnisses nach Nahrung in Abhängigkeit von Adams Energie

Um eine Priorisierung von Motiven vornehmen zu können, beispielsweise um Motiven, die mit existentiellen Grundbedürfnissen zusammenhängen, bei der Verhaltenssteuerung eine höhere Bedeutung einzuräumen, wird für die Berechnung der Motivstärken im Modell Adam neben der zugrundeliegenden Bedürfnisstärke ein motiv-spezifischer Gewichtungsfaktor berücksichtigt. Im Falle des Motivs *Hunger* handelt es sich dabei um den Faktor *HungerFaktor*, der mit dem Initialwert 3 belegt ist.

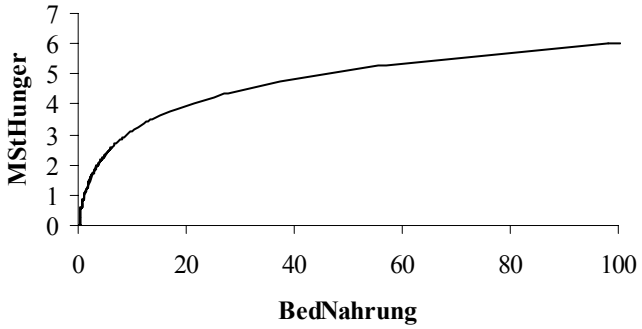

Abbildung 10.17: Der Verlauf der Motivstärke für das Motiv Hunger in Abhängigkeit von der Stärke des Bedürfnisses nach Nahrung

Damit ergibt sich eine Berechnung der Motivstärke für das Motiv *Hunger* aus der Stärke des Bedürfnisses nach Nahrung nach der folgenden Gleichung (der Ausdruck log_{10} bezeichnet dabei den dekadischen Logarithmus):

$$MStHunger\ (t) := HungerFaktor * log_{10}\ (BedNahrung\ (t) + 1) \quad \text{(Gl. 10.7)}$$

Adam wird sich nach einem erfolgreichen Eßvorgang zunächst einmal für eine gewisse Zeit in einem Zustand der Sättigung befinden und keine weiteren Handlungen ausführen, die der Nahrungssuche und –aufnahme dienen. Dies bedeutet insbesondere, daß für eine gewisse Zeit das Motiv *Hunger* deaktiviert wird und erst dann wieder mit dem Motiv *Wissenserwerb* in Konkurrenz um die handlungsleitende Position tritt, wenn die zugrundeliegende Stärke des Bedürfnisses nach Nahrung einen vorgegebenen Schwellwert *BedNahrungSchranke* überschreitet.

Die Abbildung 10.18 zeigt noch einmal im Überblick alle Modellgrößen, die an der Bestimmung der Motivstärke für das Motiv *Hunger* direkt oder indirekt beteiligt sind.

Bezeichner	**Einheit**	**Initialwert**	**Erläuterung**
Energie	[EnE]	100	Energie
EnergieMin	[EnE]	0	Minimalwert an Energie
EnergieMax	[EnE]	100	Maximalwert an Energie
BedNahrung	[NBE]	0	Stärke des Bedürfnisses nach Nahrung
MStHunger	[MStE]	0	Motivstärke des Motivs *Hunger*
HungerFaktor	[MStE]	3	Gewichtungsfaktor für die Motivstärke des Motivs *Hunger*
BedNahrungSchranke	[NBE]	0.2	Aktivierungsschranke für das Motiv *Hunger*

Abbildung 10.18: Die Modellgrößen für die Berechnung der Motivstärke des Motivs Hunger

Die Modellierung des Motivs Wissenserwerb. Adam verfügt neben dem Motiv *Hunger*, das auf ein elementares physiologisches Bedürfnis zurückgeht, auch über ein höheres kognitives bzw. informationelles Motiv, das ihn zur Erkundung seiner Umwelt drängt. Immer dann, wenn Adam nicht auf Nahrungssuche ist, versucht er seine Umwelt zu erforschen.

Das kognitive Motiv *Wissenserwerb* basiert ursprünglich auf dem aktuellen Wissensstand von Adam, der in der gleichnamigen Modellgröße verwaltet wird. Zu Beginn verfügt Adam über keinerlei Informationen über die Felder in seiner

Umwelt. Seine kognitive Landkarte ist anfangs leer. Im Rahmen der Streifzüge durch seine Umwelt beginnt sich Adams Umweltmodell allmählich zu füllen und sein Wissensstand erhöht sich. Im Gegenzug vergisst Adam allerdings auch nach einiger Zeit wieder Sachverhalte über seine Umwelt, so daß sein Wissensstand letztendlich ständigen Veränderungen sowohl nach unten als auch nach oben unterliegt (vgl. dazu auch Abschnitt 10.1.2.1: Kognitive Zustandsvariablen und Prozesse).

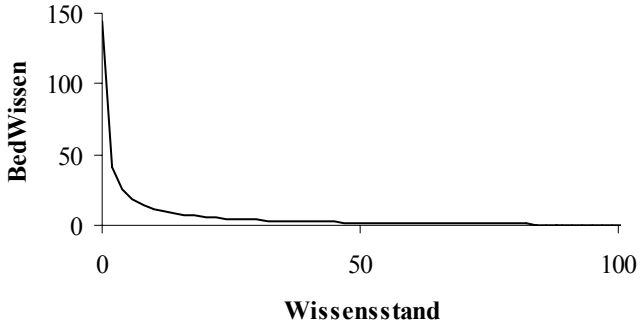

Abbildung 10.19: Verlauf der Stärke für Adams Bedürfnis nach Wissen in Abhängigkeit von seinem Wissensstand

Aus der Menge der vorhandenen Informationen über seine Umwelt resultiert Adams Bedürfnis nach Wissen *BedWissen*. Je geringer Adams Wissensstand ist, um so höher wird sein Bedürfnis nach Wissen ausfallen. Da Adam zu Beginn nur sehr wenig über seine Umwelt weiß und sein Wissensstand noch sehr gering ist, wird in dieser Phase sein Bedürfnis nach Wissen sehr groß sein. Entsprechend sinkt die Stärke des Bedürfnisses nach Wissen immer mehr ab, je mehr Informationen Adam über seine Umwelt in Erfahrung bringt. Dieser Zusammenhang zwischen Adams Wissensstand und der daraus resultierenden Stärke für das Bedürfnis nach Wissen wird im Modell vereinfacht mit Hilfe des Verhältnisses der unbekannten Felder zu den bekannten Feldern dargestellt. Die Anzahl der unbekannten Felder läßt sich dabei sofort aus der Gesamtanzahl der Felder, die aus der Größe der Umwelt *XYMAX* folgt, sowie aus Adams Wissensstand ermitteln. Insgesamt ergibt sich also für die Modellierung der Stärke des Bedürfnisses nach Wissen die folgende Gleichung:

$$BedWissen(t) := ((XYMAX * XYMAX) - Wissensstand(t)) / (Wissensstand(t) + 1) \quad \text{(Gl. 10.8)}$$

Die Abbildung 10.19 zeigt den funktionalen Zusammenhang zwischen Adams Wissensstand und der Stärke seines Bedürfnisses nach Wissen. Die Anzeige ist dabei zur Steigerung der Übersichtlichkeit in bezug auf die Modellgröße *Wissensstand* auf das Intervall [0, 100] beschränkt.

Das Bedürfnis nach Wissen löst bei Adam eine Motivation zur Erforschung seiner Umgebung aus. Die Intensität dieser Motivation ist dabei um so höher, je größer Adams Bedürfnis nach Wissen ist. Ebenso wie beim Motiv *Hunger* wird auch für das Motiv *Wissenserwerb* ein logarithmischer Zusammenhang zwischen der Bedürfnisstärke und der daraus resultierenden Motivstärke angenommen. Weiterhin geht auch in die Berechnung der Motivstärke für das Motiv *Wissenserwerb* ein konstanter Gewichtungsfaktor *WissensFaktor* ein, der im Rahmen der Motivauswahl eine beliebige Priorisierung einzelner Motive ermöglicht (s. Gl. 10.9).

$$MStWissenserwerb\ (t) := WissensFaktor * log_{10}\ (BedWissen\ (t) + 1) \qquad \text{(Gl. 10.9)}$$

Damit ergibt sich für die Motivstärke des Motivs *Wissenserwerb* der folgende Kurvenverlauf:

Abbildung 10.20: Verlauf der Motivstärke für das Motiv Wissenserwerb in Abhängigkeit der Stärke des Bedürfnisses nach Wissen

Bezeichner	Einheit	Initialwert	Erläuterung
Wissensstand	[WiSE]	0	Anzahl der bekannten Felder in Adams kognitiver Landkarte
BedWissen	[WiBE]	0	Stärke des Bedürfnisses nach Wissen
MStWissenserwerb	[MStE]	0	Motivstärke des Motivs *Wissenserwerb*
WissensFaktor	[MStE]	1	Gewichtungsfaktor für die Motivstärke des Motivs *Wissenserwerb*

Abbildung 10.21: Die Modellgrößen für die Berechnung der Motivstärke des Motivs Wissenserwerb

Die Abbildung 10.21 zeigt noch einmal in der Gesamtschau alle Modellgrößen, die für die Bestimmung der Stärke des Motivs *Wissenserwerb* von Bedeutung sind.

Motivselektion und Festlegung des Handlungsziels. Im Rahmen von Adams Verhaltenssteuerung kann zu einem Zeitpunkt immer nur ein Motiv wirksam sein. Aus diesem Grund besteht die Notwendigkeit, aus der Menge der vorhandenen Motive eines auszuwählen, das dann für eine gewisse Zeit Adams Handeln bestimmt. Das ausgewählte Motiv soll aufgrund seiner Funktion als *handlungsleitendes Motiv* bezeichnet werden. Der Vorgang der Motivauswahl wird in Anlehnung an Dörner (1999) auch *Motivselektion* genannt.

Im vorliegenden Modell wird durch die Motivselektion eine Entscheidung getroffen, welches der beiden Motive *Hunger* oder *Wissenserwerb* handlungsleitend werden soll. Ausschlaggebend für die Auswahlentscheidung ist dabei die aktuelle Motivstärke, die dem jeweiligen Motiv zugeordnet ist. Dieses Vorgehen basiert auf der Annahme, daß immer dasjenige Motiv die Handlungen eines Organismus bestimmt, das die höchste Dringlichkeit besitzt. Hierbei handelt es sich um eine stark vereinfachte Form der sogenannten Erwartung-Wert-Theorien (Brandstätter & Gollwitzer, 1996), wie sie in der psychologischen Modellbildung häufig zum Einsatz kommen. Der Wert-Anteil der Auswahlstrategie kann dabei mit der aktuellen Motivstärke identifiziert werden. Von der Erfolgserwartung, die gemäß dieser Theorien als zusätzlicher Einfluß auf die Auswahlentscheidung einwirkt, wird im vorliegenden Fall der Einfachheit halber abstrahiert.

Neben der Motivstärke manifestiert sich in Adams Motivselektion zudem der Maslowsche Ansatz (Maslow, 1955), daß physiologische Grundbedürfnisse befriedigt sein müssen, bevor höhere und insbesondere kognitive Bedürfnisse zum Zuge kommen können. Die Motivselektion gewährt also dem auf dem physiologischen Nahrungsbedürfnis gründenden Motiv *Hunger* Vorrang vor dem kognitiven Motiv *Wissenserwerb*. Für die Konzeption der Motivselektion bedeutet dies, daß, nachdem das Motiv *Hunger* handlungsleitend geworden ist, ein Motivwechsel zugunsten von *Wissenserwerb* erst dann wieder stattfinden kann, wenn Adams Nahrungsbedürfnis durch Nahrungsaufnahme vollständig gestillt ist. Ein Motivwechsel von *Hunger* auf *Wissenserwerb* findet auch dann nicht statt, wenn die Motivstärke des Motivs *Wissenserwerb* zwischenzeitlich einen höheren Wert angenommen hat als die Motivstärke für das Motiv *Hunger*, jedoch noch ein Restbedürfnis nach Nahrung vorhanden ist. Im Gegensatz dazu kann das Motiv *Wissenserwerb* jederzeit durch ein stärkeres Motiv *Hunger* aus der handlungsleitenden Position verdrängt werden, auch wenn es Adam zum betrachteten Zeitpunkt noch nicht geschafft hat, alle Felder seiner Umwelt zu erkunden, und damit noch ein Restbedürfnis nach Wissen besteht.

Die Motivselektion findet nicht fortwährend, sondern nur in ganz bestimmten Situationen statt, um Adams Verhalten eine gewisse Kontinuität zu verleihen und zu vermeiden, daß Adam zwischen unterschiedlichen Verhaltensweisen wechselt

ohne sie jeweils zu Ende zu bringen. In der Regel tritt also ein Motivwechsel erst dann in Kraft, wenn die mit dem bestehenden handlungsleitenden Motiv verbundenen Handlungen abgeschlossen sind oder unvorhergesehene Ereignisse, wie beispielsweise das Betreten einer Gefahrenstelle, eintreten, die ohnehin eine Neuorganisation des Verhaltens erfordern.

Mit der Festlegung eines handlungsleitenden Motivs wird gleichzeitig ein Zielzustand aktiviert, den Adam durch Ausführung geeigneter Handlungen zu erreichen versucht. Wird das Motiv *Hunger* handlungsleitend, so strebt Adam einen Zielzustand an, in dem sein Energievorrat bis zur maximalen Obergrenze aufgefüllt ist. Erhält das Motiv *Wissenserwerb* den Vorrang bei der Motivselektion, so wird Adam versuchen, unbekannte Felder zu erkunden und dadurch die Anzahl der bekannten Felder in seiner kognitiven Landkarte zu erhöhen.

Aufgrund dieser Annahmen wird Adams Verhalten im Laufe der Zeit abwechselnd durch das Motiv *Hunger* und das Motiv *Wissenserwerb* geprägt. Zu Beginn, wenn Adam auf einen vollen Energiespeicher zurückgreifen kann, wird zunächst einmal das Motiv *Wissenserwerb* die Führungsrolle übernehmen, so daß sich Adam der Erkundung seiner Umwelt zuwenden wird. Im Rahmen seiner Streifzüge durch die Umwelt lernt Adam immer mehr unbekannte Felder kennen, so daß sich sein Wissensstand im Laufe der Zeit erhöht und letztlich seine Motivation zum Wissenserwerb abnimmt. Gleichzeitig kosten die Aktionen, die Adam ausführt, Energie. Adams Energievorrat nimmt also im Laufe der Zeit ab und die Motivstärke für *Hunger* steigt entsprechend immer mehr an. Sobald dann die Motivstärke für *Hunger* die Motivstärke für *Wissenserwerb* übersteigt, übernimmt das Motiv *Hunger* die handlungsleitende Funktion, so daß Adam in der Folgezeit auf Nahrungssuche gehen wird. Schafft es Adam, durch das Aufsuchen eines Nahrungsfeldes sein Nahrungsbedürfnis zu befriedigen, wechselt das handlungsleitende Motiv wiederum auf *Wissenserwerb*. Die Abbildung 10.21 zeigt einen exemplarischen Zeitverlauf der Motivstärken für das Motiv *Hunger* und das Motiv *Wissenserwerb*.

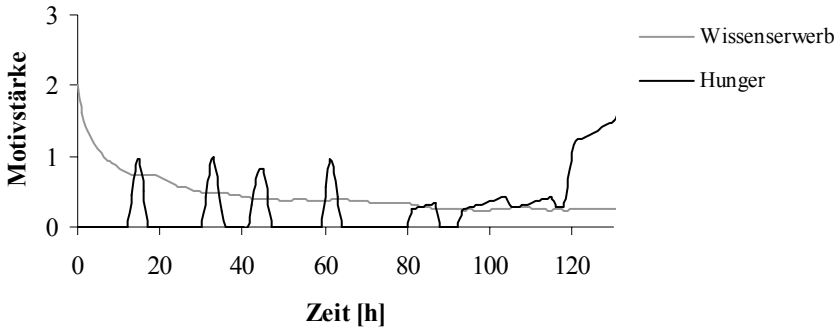

Abbildung 10.21: Die zeitliche Entwicklung der Motivstärken für das Motiv Hunger und das Motiv Wissenserwerb

Die Auswahl einer Handlungsalternative

Für der Realisierung seiner Ziele stehen Adam (zumindest im Falle des Motivs *Hunger*) verschiedene Handlungsalternativen zur Verfügung. Nachdem mit Hilfe der Motivselektion ein handlungsleitendes Motiv bestimmt worden ist, muß Adam also eine Entscheidung treffen, nach welcher Handlungsalternative er vorgehen wird, um das gesteckte Handlungsziel zu erreichen.

Übernimmt das Motiv *Hunger* die Kontrolle über Adams Verhalten, so kann er zwischen einer eher konservativen (*Nahrungssuche_konservativ*) und einer eher wagemutigen (*Nahrungssuche_wagemutig*) Handlungsalternative auswählen. Konservativ wird Adam dann vorgehen, wenn sein Zutrauen in seine eigenen Fähigkeiten, modelliert mit Hilfe der kognitiven Zustandsvariable *Kompetenz* (s. Kapitel 10.1.2.1: Kognitive Zustandsvariablen und Prozesse), eher gering ausfällt. Unterschreitet also der aktuelle Wert der Zustandsvariable *Kompetenz* zum betrachteten Zeitpunkt den Schwellwert *KompetenzSchranke*, so versucht Adam bei der Nahrungssuche nur solche Felder zu berücksichtigen, von denen er sicher weiß, daß sie eine ausreichende Menge an Nahrung bereitstellen werden, um seinen Hunger zu stillen. Ist hingegen Adams *Kompetenz* sehr hoch, so traut er sich riskantere Strategien zu und steuert bei der Nahrungssuche auch solche Felder an, über deren aktuellen Zustand er nur unzureichende Informationen hat. Es handelt sich dabei um Nahrungsfelder, die sich in bezug auf Adams Lernschranken im Intervall *DaltaSicherNein* und *DeltaSicherJa* (s. Kapitel 10.1.2.1: Kognitive Zustandsvariablen und Prozesse) befinden. Adam geht damit das Risiko ein, bei seiner Ankunft auf einem derartigen Feld keine Nahrung vorzufinden.

Falls das Motiv *Wissenserwerb* handlungsleitend wird, besteht für Adam in bezug auf die Handlungsalternative keine Wahlmöglichkeit. In diesem Fall tritt automatisch die Handlungsalternative *Erkunden* in Kraft.

Die Auswahl von Handlungszwischenzielen

Handlungszwischenziele legen Meilensteine für Adams Handeln fest. Sie müssen erreicht werden, damit übergeordnete Handlungsziele realisiert werden können, und werden in Abhängigkeit der aktuellen Handlungsalternative ausgewählt. Im vorliegenden Fall können Handlungszwischenziele mit dem Erreichen eines ganz bestimmten Zielfeldes in der Umwelt identifiziert werden. Befindet sich Adam auf einem derartigen Zielfeld, so kann er entsprechende Endhandlungen ausführen, die das angestrebte Handlungsziel herbeiführen oder seinen aktuellen Bedürfniszustand zumindest um ein gewisses Maß verbessern sollen.

Adam verfügt, wie im vorangegangenen Abschnitt erläutert, über die drei Handlungsalternativen *Erkunden*, *Nahrungssuche_konservativ* und *Nahrungssuche_wagemutig*. Im Falle der Handlungsalternative *Erkunden* wählt Adam als Handlungszwischenziel ein derartiges Feld in seiner Umgebung aus, das in seiner kognitiven Landkarte zum gegebenen Zeitpunkt noch unbekannt ist. Ist eine der beiden Alternativen zur Nahrungssuche aktiv, so können nur Felder, auf denen

Adam Nahrung vermutet, zu Handlungszwischenzielen werden, es sei denn, Adam hat zum betrachteten Zeitpunkt noch keine Informationen über Nahrungsfelder in seiner kognitiven Landkarte zur Verfügung. In diesem Fall geht Adam wie bei der Alternative *Erkunden* vor, in der Hoffnung, bei der Erforschung der Umwelt zufällig auf eine Nahrungsstelle zu stoßen. Sowohl bei der Erkundung seiner Umwelt als auch bei der Nahrungssuche bevorzugt Adam, um schonend mit seinen Energiereserven umzugehen, in Frage kommende Felder in seiner unmittelbaren Umgebung.

Adam greift zur Bestimmung der Zielfelder bzw. Handlungszwischenziele auf seine kognitive Landkarte zurück und wendet dabei ein einfaches Suchverfahren an. Ausgehend von seiner aktuellen Position in der Umwelt betrachtet Adam in einem Suchschritt alle Felder, die denselben Abstand zu seiner aktuellen Position aufweisen, und überprüft, ob eines dieser Felder als Handlungszwischenziel geeignet ist. Er beginnt dabei mit einem Abstand von einem Feld zu seiner aktuellen Position, d. h. also mit den unmittelbar benachbarten Feldern, und tastet sich bei Bedarf ringweise zu weiter entfernt liegenden Feldern vor, wobei von Ring zu Ring der Abstand jeweils um ein Feld anwächst. Werden in einem Suchschritt mehrere mögliche Zielfelder gefunden, so wird eines davon zufällig ausgewählt und als Handlungszwischenziel festgelegt.

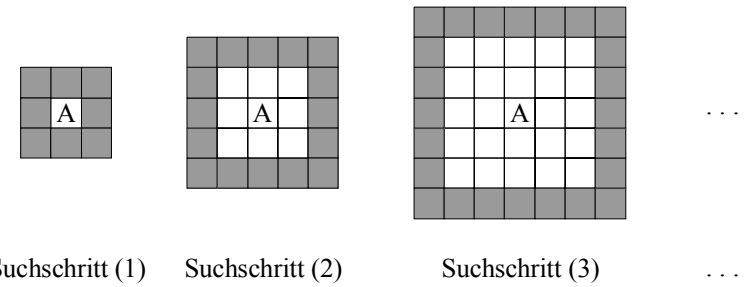

Abbildung 10.22: Das Suchverfahren für die Bestimmung der Handlungszwischenziele

Die Abbildung 10.22 verdeutlicht die einfache Vorgehensweise dieses Suchverfahrens. Die grau hinterlegten Felder bezeichnen dabei die Felder, die während eines Suchschrittes betrachtet werden. Das mit einem „A“ markierte Feld deutet Adams aktuelle Position an.

Die Handlungsplanung

Im Rahmen der Handlungsplanung versucht Adam sein Handeln zunächst einmal in strategischer Weise vorzustrukturieren, bevor er mit der Ausführung konkreter Aktionen beginnt. Er legt dabei im Vorgriff fest, welche grundsätzliche Vorgehensweise er wählen wird, um das vorgegebene Handlungszwischenziel und später auch das übergeordnete Handlungsziel zu erreichen. Adam konstruiert dazu eine Folge von Teilzielen, die die Funktion von Meilensteinen übernehmen und Adam ausgehend von der aktuellen Situation immer näher an das gesteckte Hand-

lungszwischenziel heranführen sollen. Ein fertiger Handlungsplan besteht dann aus einer Folge von Teilzielen, deren letztes Element mit dem vorgegebenen Handlungszwischenziel übereinstimmt. Sobald ein brauchbarer Handlungsplan vorliegt, kann sich Adam dessen Umsetzung zuwenden. Dazu betrachtet er der Reihe nach jedes einzelne Teilziel des Handlungsplans und versucht dieses durch Ausführung einer geeigneten, an die aktuelle Situation angepaßten Folge von Aktionen zu erreichen.

Die Erstellung eines Handlungsplans beschränkt sich im Falle von Adam auf die Konstruktion einer Folge von Feldern in der Umwelt, durch deren schrittweises Betreten Adam von seinem aktuellen Feld zum Zielfeld der bestehenden Handlungsalternative gelangt. Es handelt sich dabei also um eine Form von Wegeplanung, die Adam auf der Grundlage eines einfachen Algorithmus nach dem Greedy-Prinzip (s. Russel & Norvig, 1995) vornimmt. Der verwendete Algorithmus liefert stets einen der kürzesten Wege von Adams aktueller Position zu einem vorgegebenen Zielfeld und bezieht dabei bestehende Informationen über aktuelle Gefahrenfelder in Adams Umwelt mit ein. Allerdings wird vereinfachend angenommen, daß der Algorithmus nur dann in der Lage ist, Wege ohne bekannte Gefahrenstellen zu planen, wenn nicht mehrere Gefahrenstellen unmittelbar waagrecht oder senkrecht nebeneinander platziert sind.

Das folgende kleine Beispiel soll verdeutlichen, wie ein Handlungsplan, der für die Verhaltenssteuerung des Agenten Adam Verwendung findet, konkret aufgebaut ist. Den Ausgangspunkt soll dazu eine Situation bilden, in der sich Adam auf Feld (2,2) seiner Umwelt befindet und aufgrund seines handlungsleitenden Motivs sowie der ausgewählten Handlungsalternative das Zielfeld (5,7) anstrebt.

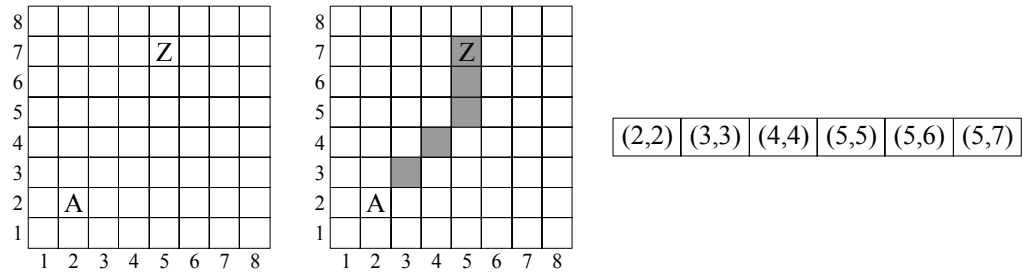

Abbildung 10.23: Die Erstellung eines Handlungsplans für eine gegebene Situation

Im Rahmen der Handlungsplanung wird nun ein Weg vom Startfeld (2,2) zum Zielfeld (5,7) entwickelt, der über die Felder (3,3), (4,4), (5,5) und (5,6) schließlich zum Zielfeld (5,7) führt. Dieser Weg wird dann als gültiger Handlungsplan für die Verhaltensauswahl von Adam zugrunde gelegt.

Die Erstellung und Ausführung von Aktionsfolgen

Im Rahmen der Handlungsplanung strukturiert Adam in deliberativer Weise sein Vorgehen und legt Teilziele, d. h. also einzelne Etappen seines Handelns fest, ohne an dieser Stelle bereits Entscheidungen darüber zu treffen, welche Aktionen konkret ausgeführt werden müssen, damit die Teilziele realisiert werden können. Mit dieser Aufgabe befaßt sich die nächste Stufe in Adams Verhaltenssteuerung.

Adam betrachtet nach der Fertigstellung eines Handlungsplans der Reihe nach die einzelnen Teilziele und wählt in Abhängigkeit seines gesamten internen physischen, emotionalen und kognitiven Zustands Aktionen aus, deren Ausführung zur Erreichung des jeweiligen Teilziels führen soll. Der Auswahlvorgang erfolgt dabei reaktiv und situationsabhängig und liefert im allgemeinen Fall eine Aktionsfolge, d. h. eine Sequenz von Beschreibungen der auszuführenden Aktionen einschließlich ihrer gegebenenfalls benötigten Parameter.

Die in einer Aktionsfolge befindlichen Aktionen werden in der Reihenfolge ihrer Erzeugung zur Ausführung gebracht. Im Normalfall wird eine Aktionsfolge vollständig abgearbeitet und damit auch das zugrundeliegende Teilziel erreicht.

Zudem können jedoch auch unvorhergesehene Situationen eintreten, in denen die Ausführung einer bestehenden Aktionsfolge unterbrochen werden muß. Stellt sich in Anbetracht der gegebenen Situation eine noch nicht fertig bearbeitete Aktionsfolge als nachteilig heraus, so kann diese verworfen werden und gegebenenfalls durch eine neue, der Situation angemessene Aktionsfolge ersetzt werden. Dies kann beispielsweise dann der Fall sein, wenn Adam mit Hilfe der Aktion *Prüfen* festgestellt hat, daß er im Begriff ist, eine Gefahrenstelle zu betreten. In dieser Situation wird er die als nächstes anstehende Aktion *Gehen* nicht zur Ausführung bringen, sondern die vorliegende Aktionsfolge verwerfen und durch eine neue Aktionsfolge ersetzen, die die Aktion *Planen* enthält und damit zu einer Neuorganisation seines Handelns führt.

Ebenso kann es vorkommen, daß Adam nach der vollständigen Abarbeitung einer Aktionsfolge nicht sofort zur Bearbeitung des nächsten Teilziels übergeht und dafür eine neue Aktionsfolge konstruiert, sondern zuvor noch eine andere Aktionsfolge zwischenschiebt, die in der vorliegenden Situation als nützlich erachtet wird. Nach Abarbeitung der kurzfristig eingefügten Aktionsfolge fährt Adam dann mit der regulären Bearbeitung des nächsten Teilziels fort. Eine situationsabhängig eingeschobene Aktionsfolge wird etwa dann erzeugt, wenn Adam bei seiner Nahrungssuche zufällig auf ein Nahrungsfeld stößt, das nicht dem geplanten Zielfeld des Handlungsplans entspricht. Gemäß seines Handlungsplans würde Adam in einem derartigen Fall einfach zum nächsten geplanten Feld in seiner Umwelt voranschreiten. Um sich allerdings die auf diesem Feld vorhandene Nahrung nicht entgehen zu lassen, baut Adam in dieser Situation unabhängig von seinem Handlungsplan eine Aktionsfolge auf, die die Aktion *Essen* enthält und dadurch die Aufnahme der entdeckten Nahrung ermöglicht. Ist sein Hunger anschließend noch nicht gänzlich gestillt, setzt er seinen ursprünglich erstellten Handlungsplan in unveränderter Weise mit der Bearbeitung des nächsten Teilziels fort.

Um die Konsequenzen seines Handelns in Erfahrung zu bringen, muß Adam seine Umwelt gezielt beobachten. Zu diesem Zweck führt Adam nach jeder externen Aktion automatisch eine Orientierungsaktion aus. In den Aktionsfolgen, die das reaktive Verhalten von Adam bestimmen, werden die externen Aktionen *Gehen*, *Essen* und *Befreien* daher stets durch die Aktion *Explorieren* ergänzt, die den aktuellen Zustand der Umwelt für Adams Informationsverarbeitung zugänglich macht.

In der folgenden Entscheidungstabelle (Balzert, 2000) sind die wichtigsten Regeln mit ihren zugehörigen Bedingungen und Konsequenzen aufgeführt, die Adams reaktives Verhalten steuern.

Adams reaktives Verhalten		R1	R2	R3	R4	R5	R6	R7	R8
B1	Gefahrenfeld beim Prüfen entdeckt?	J	N	N	N	N	N	N	N
B2	Adam in Falle?	-	N	N	J	N	N	N	N
B3	HL Motiv gewechselt?	-	J	N	-	N	N	N	N
B4	Aktionsfolge leer?	-	-	J	-	J	J	J	N
B5	HL Motiv ist Hunger und Nahrungsmenge des aktuellen Feldes > 0	-	-	N	-	J	N	N	N
B6	Handlungsplan leer?	-	-	J	-	-	N	N	-
B7	Angst > 0 und nächstes Teilziel unbekannt?	-	-		-	-	J	N	-
A1	AF *Planen*	X	X	X					
A2	AF *Befreien – Explorieren*				X				
A3	AF *Essen – Explorieren*					X			
A4	AF *Prüfen – Gehen – Explorieren*						X		
A5	AF *Gehen – Explorieren*							X	
A6	Bestehende AF nicht verändern								X

Abbildung 10.24: Die Entscheidungsregeln für Adams reaktives Verhalten

Die Entscheidungstabelle gibt dabei an, unter welchen Umständen Adam welche Aktionsfolgen zur Ausführung freigibt. Im Bedingungsteil der Entscheidungstabelle gibt das Symbol „J“ an, daß die zugehörige Bedingung erfüllt sein muß,

„N“, daß die entsprechende Bedingung nicht erfüllt sein darf, und „-“, daß diese Bedingung für die betrachtete Aktionsfolge nicht von Belang ist.

Die Regeln R1 bis R3 befassen sich mit der Erzeugung der Aktionsfolge *Planen*. Adam wird seine Handlungen neu planen, wenn Adam beim *Prüfen* eines Feldes entdeckt hat, daß es sich dabei um ein Gefahrenfeld handelt (R1). Er entwirft in diesem Fall einen neuen Handlungsplan, um das Gefahrenfeld zu umgehen. Zudem wird ein Planungsprozeß erforderlich, wenn das handlungsleitende Motiv gewechselt hat (R2). Adam muß in diesem Fall einen Handlungsplan entwerfen, dessen Ausführung eine Befriedigung der dem Motiv zugrunde liegenden Bedürfnisse herbeiführt. Schließlich wird Adam einen Planungsprozeß auch dann anstoßen, wenn ein Handlungsplan vollständig abgearbeitet ist und keine weiteren Aktivitäten mehr anstehen (R3).

Die Aktionsfolge *Befreien – Explorieren* (R4) wird erzeugt, wenn Adam in eine Falle geraten ist. Er wird zunächst versuchen, mit Hilfe der Aktion *Befreien* aus der Falle zu entkommen, um dann durch Ausführung der Aktion *Explorieren* festzustellen, ob sein Versuch erfolgreich war.

Die Regel R4 gibt an, daß Adam die Aktion *Essen* ausführen wird, wenn *Hunger* sein handlungsleitendes Motiv ist und er ein Feld in seiner Umwelt erreicht hat, auf dem sich Nahrung befindet.

Die Regeln R6 und R7 beschreiben Adams Bewegungen in seiner Umwelt. In beiden Fällen wählt Adam die Aktionen *Gehen* und *Explorieren* aus, sobald ein neues Teilziel zur Bearbeitung freigegeben wurde. Befindet sich Adam in der vorliegenden Situation im Zustand *ängstlich*, so schaltet er dem tatsächlichen Betreten des nächsten Feldes die Aktion *Prüfen* vor, um dieses Feld zunächst einmal auf Gefahren hin zu untersuchen. Ist er nicht ängstlich, verzichtet er auf die Überprüfung des Zielfeldes und gibt die Aktionsfolge *Gehen – Explorieren* frei.

Adam fährt mit der Ausführung einer teilweise abgearbeiteten Aktionsfolge fort, sofern keine der vorstehend beschriebenen Regeln in der gegebenen Situation zutreffen (R8).

Das folgende Beispiel (s. Abbildung 10.25) zeigt für eine einfache Situation, wie Adam von einem Handlungsplan über eine Aktionsfolge zu einer auszuführenden Aktion gelangt. Den Ausgangspunkt dafür bildet der Handlungsplan aus Abbildung 10.23, der die Teilziele (2,2), (3,3), (4,4), (5,5), (5,6), sowie das Zwischenziel (5,7) enthält. Als erstes wendet sich Adam dem Teilziel (2,2) zu und erzeugt die Aktionsfolge *Prüfen – Gehen – Explorieren*, da wir exemplarisch davon ausgehen, daß er sich in einem ängstlichen Zustand befindet. Um das Teilziel zu erreichen, beginnt Adam anschließend damit, die Aktionsfolge der Reihe nach abzuarbeiten und greift sich als erstes die Aktion *Prüfen* mit den Koordinaten (2,2) für das zu überprüfende Teilziel heraus. Sind alle Aktionen der erstellten Aktionsfolge bearbeitet, wird das nächste Teilziel aus dem Handlungsplan entnommen, eine entsprechende Aktionsfolge erzeugt und ebenso zur Ausführung gebracht. Dieser Prozeß endet in der Regel dann, wenn der gesamte Handlungsplan vollständig abgearbeitet werden konnte oder bestimmte Umstände eintreten, die zu einem vorzeitigen Abbruch des Handlungsplans führen.

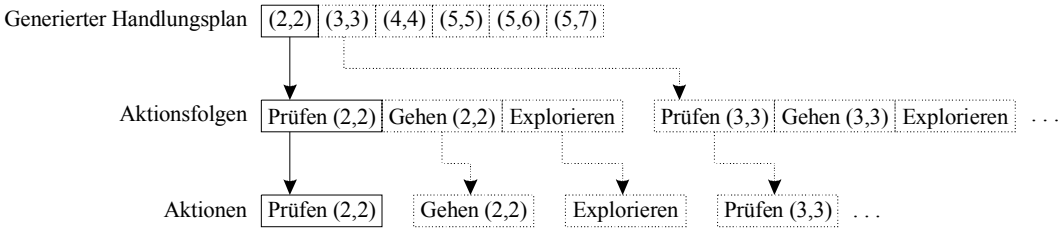

Abbildung 10.25: Handlungspläne, Aktionsfolgen und Aktionen

Die Bewertung der Resultate ausgeführter Aktionen

Wie im vorangegangenen Abschnitt bereits angedeutet, führt Adam im Anschluß an externe Aktionen stets Orientierungsaktionen aus, die ihm die Möglichkeit geben, die Resultate ausgeführter Aktionen in Erfahrung zu bringen. Im wesentlichen handelt es sich bei diesen Resultaten um Informationen hinsichtlich des Erfolgs oder Mißerfolgs von Aktionen.

Da sowohl die Aktionen, die Adam ausführen kann, als auch seine Umwelt bewußt sehr einfach gehalten sind, besteht nur in Ausnahmefällen die Gefahr, daß Adams Bemühungen nicht zu den gewünschten Resultaten führen. So wird Adam nach Ausführung der Aktion *Gehen* vereinfachend immer auf dem Feld landen, das er auch tatsächlich zu erreichen versucht hat. Ebenso resultiert aus einer Ausführung der Aktion *Befreien* stets ein Folgezustand, in dem sich Adam außerhalb der auf einem Feld existierenden Gefahrenstelle befindet.

Eine andere Situation ergibt sich allerdings bei der Nahrungssuche. Hier kann es durchaus vorkommen, daß sich Adam aufgrund der wagemutigen Handlungsalternative zu einem Zielfeld begibt, dessen Nahrungsangebot zum Zeitpunkt seiner Ankunft noch nicht wieder regeneriert ist. Dadurch ist Adam entgegen seiner Erwartung nicht in der Lage, sein Nahrungsbedürfnis an dieser Stelle zu befriedigen, und wertet diesen Sachverhalt als Kompetenzverlust. Existiert also an einer Stelle, an der Adam Nahrung zu finden hoffte, tatsächlich keine Nahrung, so führt dieser Umstand zu einer Verringerung des aktuellen Wertes seiner kognitiven Zustandsvariable *Kompetenz* und damit gegebenenfalls zu einem vorsichtigeren Vorgehen bei zukünftigen Unternehmungen. Bei erfolgreicher Nahrungssuche erhöht sich hingegen der aktuelle Wert für Adams Kompetenz und führt bei Überschreiten eines vorgegebenen Schwellwertes zu mutigeren Verhaltensweisen. Stößt Adam bei der Nahrungssuche zufällig auf ein Nahrungsfeld, so bleibt seine Kompetenz von diesem glücklichen Umstand unberührt, da er sein Kompetenzempfinden nur auf geplante und absichtlich angeordnete Verhaltensweisen bezieht.

10.2 Entwurf und Implementierung auf der Grundlage des Referenzmodells *PECS*

Der folgende Abschnitt beschäftigt sich mit dem Implementierungskonzept für das vorstehend beschriebene Modell. Ausgehend von der Grundstruktur des Modells werden dazu nacheinander alle wesentlichen Modellkomponenten hinsichtlich ihres Aufbaus und ihres dynamischen Verhaltens betrachtet. Darüber hinaus wird anhand eines typischen Szenarios das dynamische Zusammenspiel der Komponenten, aus denen der Agent Adam zusammengesetzt ist, beschrieben.

Für die Realisierung des Modells wurde das Referenzmodell *PECS* (s. Kapitel 6) sowie das Simulationssystem *Simplex3* mit der zugehörigen Modellbeschreibungssprache *Simplex-MDL* (Schmidt, 2000) zugrunde gelegt. *Simplex-MDL* bietet einen systemtheoretischen Ansatz zur Modellbeschreibung und unterstützt einen komponentenorientierten, hierarchischen Modellaufbau, so daß im Zusammenspiel mit dem Referenzmodell *PECS* ein komfortabler Aufbau agentenbasierter Simulationsmodelle möglich wird.

10.2.1 Die Grundstruktur des Modells

Das Simulationsmodell setzt sich in Anlehnung an den komponentenorientierten und hierarchischen Modellaufbau aus einer Menge von unabhängigen und nebenläufig arbeitenden Modellkomponenten zusammen. Das Gesamtmodell entsteht dabei durch eine hierarchische Aggregation von Teilmodellen bzw. Subkomponenten, die sich über insgesamt drei Hierarchieebenen erstreckt. Die Interaktion der Subkomponenten erfolgt auf der Grundlage von kausalen Abhängigkeiten und diskreten Informationsflüssen.

Auf der obersten Hierarchieebene ist das Gesamtmodell *WorldOfAdam* angesiedelt, das die beiden Subkomponenten *Adam* und *Environment* enthält (s. Abbildung 10.26). Hinter der Komponente *Environment* verbirgt sich die Modellierung der Umwelt des Agenten. Die Komponente *Adam* ist für die Beschreibung des Agenten Adam zuständig und auf der untersten Hierarchieebene aus einer Menge von weiteren Subkomponenten (*Sensor*, *Perception*, *Physis*, *Emotion*, *Cognition, Behaviour* und *Actor*) zusammengesetzt, die durch die Agentenarchitektur des Referenzmodells *PECS* vorgegeben sind. Die Substruktur der Komponente *Adam* ist in Abbildung 10.26 aus Gründen der Übersichtlichkeit noch nicht dargestellt. Sie wird allerdings in Kapitel 10.2.3 im Zusammenhang mit der Beschreibung des Agenten nachgereicht.

Der Agent Adam ist mit Hilfe von externen Aktionen in der Lage seine Umwelt zu verändern. Zu diesem Zweck werden Aktionen, die Adam zur Ausführung freigibt und an die Umwelt gerichtet sind, als diskrete Informationspakete an die Komponente *Environment* übermittelt. Im Modell wird daher in der Komponente *Environment* eine Eingangslocation *DatVonACT* eingerichtet, die von dieser Stel-

le aus den Zugriff auf externe Aktionen des Agenten ermöglicht. Im Gegenzug liefert die Komponente *Environment* Informationen über ihren aktuellen Zustand an die Komponente *Adam*, die den internen Zustand sowie das Verhalten des Agenten beeinflussen. Die Umweltinformationen werden der Komponente *Adam* mit Hilfe der Eingangsvariablen *Nahrungsmenge*, *Feldart*, *InFalle*, *xAdam* und *yAdam* zugänglich gemacht.

Im Vergleich zur Komponentenstruktur des Referenzmodells *PECS* fehlen im vorliegenden Modell sowohl die Komponente *Connector* auf der mittleren Hierarchieebene, als auch die Komponente *Social Characteristics* in der Agentenarchitektur (s. Abschnitt 10.2.3). Dieser Sachverhalt rührt daher, daß in diesem Modell nur ein einziger Agent mit seinen internen Zuständen und Verhaltensweisen betrachtet wird und damit weder Kommunikation, noch soziale Merkmale eine Rolle spielen.

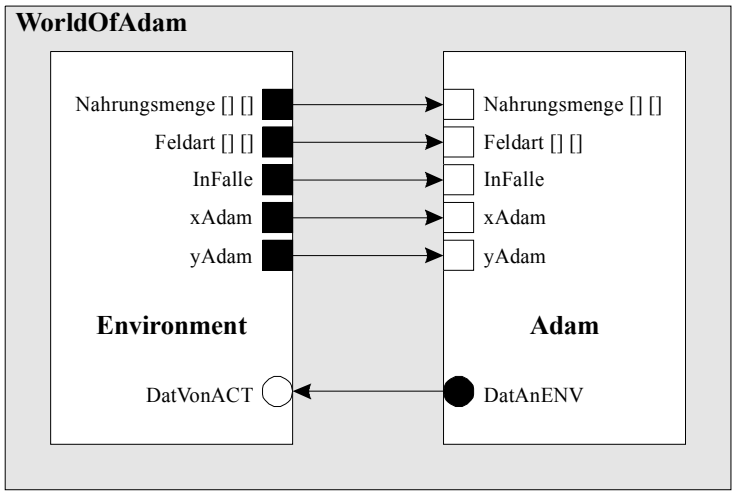

Abbildung 10.26: Die Grundstruktur der Komponente WorldOfAdam

10.2.2 Die Basiskomponente *Environment*

10.2.2.1 Aufgabe und Aufbau der Komponente *Environment*

Die Basiskomponente *Environment* (s. Abbildung 10.26) verwaltet alle Modellgrößen, die den Zustand der Umwelt des Agenten Adam bestimmen. Sie umfaßt zwei zweidimensionale Arrays, in denen der Typ der jeweiligen Felder in der Umwelt (*Feldart*) sowie gegebenenfalls die aktuelle Nahrungsmenge (*Nahrungsmenge*), falls es sich bei dem betrachteten Feld um ein Nahrungsfeld handelt, gespeichert werden. Zudem wird mit Hilfe eines ganzzahligen Koordinatentupels (*xAdam*, *yAdam*) die aktuelle Position von Adam zur Verfügung gestellt und eine weitere Modellgröße *InFalle* verwaltet, die anzeigt, ob sich Adam zum betrachteten Zeitpunkt innerhalb oder außerhalb der unter Umständen auf einem Feld existierenden Gefahrenstelle befindet. Da alle dieser Modellgrößen für die Bestim-

mung der Verhaltensweisen des Agenten Adam von Bedeutung sind, werden sie mit Hilfe des Glass-Box Konzepts (s. Kapitel 5 und Schmidt, 2000) nach außen hin sichtbar gemacht.

Zusätzlich zu den eben geschilderten Modellgrößen verfügt die Komponente *Environment* auch über eine Eingangslocation *DatVonACT*, mit deren Hilfe sie Aktionen, die der Agent Adam an sie richtet, entgegennehmen kann.

10.2.2.2 Die grundlegende Arbeitsweise der Komponente *Environment*

Die Komponente *Environment* verfügt sowohl über eine Eigendynamik als auch über eine von außen, durch die Aktionen des Agenten Adam induzierte Dynamik.

Die Eigendynamik bezieht sich auf das Wachstum der Nahrung auf den Nahrungsfeldern. Für jedes Feld der Umwelt ist eine Differentialgleichung gemäß Gl. 10.1 vorgesehen, die ein logistisches Wachstum der Nahrungsmenge in der Zeit bewirkt. Allerdings führen die Differentialgleichungen tatsächlich nur auf solchen Feldern zu einem Nahrungswachstum, die mit einer gewissen Grundmenge an Nahrung initialisiert werden. Durch Angabe von Startparametern für die Nahrungsmenge auf den einzelnen Feldern kann somit eine beliebige Verteilung der Nahrungsfelder in Adams Umwelt erreicht werden.

Die Komponente *Environment* nimmt zudem ihre Arbeit auf, wenn Adam eine externe Aktion an sie richtet. Eine Aktion wird dabei als mobiles Datenelement auf einer Location abgelegt, die von der Komponente *Environment* aus mit Hilfe der Eingangslocation *DatVonACT* zugreifbar ist. Stellt die Komponente *Environment* also fest, daß auf der Eingangslocation *DatVonACT* eine neue Aktion eingetroffen ist, werden entsprechend des Typs sowie der Parameter der Aktion diskrete Zustandsänderungen in der Komponente *Environment* vorgenommen. Sind alle mit einer Aktion einhergehenden Zustandsänderungen ausgeführt, wird das mobile Datenelement, mit dessen Hilfe die Aktion übermittelt worden ist, vernichtet.

Führt Adam die externe Aktion *Gehen* aus, so werden in der Komponente *Environment* die Koordinaten *xAdam* und *yAdam* auf die neue Position, die durch die Parameter der Aktion vorgegeben sind, gesetzt. Handelt es sich bei der neuen Position, die Adam in der Umwelt einnimmt, um ein Gefahrenfeld, so ändert zusätzlich die boolesche Zustandsvariable *InFalle* ihren Wert auf *true*. Wenn es Adam schafft, mit Hilfe der Aktion *Befreien* aus der Falle zu entkommen, erhält die Zustandsvariable *InFalle* wieder den Wert *false*. Schließlich bewirkt die Ausführung der Aktion *Essen* einen Zustandsübergang in der Komponente *Environment*, durch den die auf dem betroffenen Feld verfügbare Nahrungsmenge auf einen Minimalwert zurückgesetzt wird.

Der Einfachheit halber geht die Komponente *Environment* von der Annahme aus, daß alle Aktionen, die der Agent Adam ausführt, korrekt und auch realisierbar sind. Aus diesem Grund wird in der vorliegenden Fassung noch auf eine eingehende Überprüfung der Vorbedingungen von Aktionen verzichtet.

10.2.3 Die Architektur des Agenten Adam

Der Komponentenstruktur von Adam liegt die Agentenarchitektur des Referenzmodells *PECS* zugrunde. Die Komponente *Adam* stellt daher eine Strukturkomponente dar und aggregiert die miteinander in Verbindung stehenden Subkomponenten *Sensor*, *Perception*, *Physis*, *Emotion*, *Cognition, Behaviour* und *Actor* (s. Abbildung 10.27). Die im Rahmen der Agentenarchitektur von *PECS* vorgegebene Komponente *Social Characteristics* findet im vorliegenden Modell keine Verwendung, da Adam der einzige Bewohner seiner virtuellen Welt ist und damit soziale Aspekte nicht von Bedeutung sind.

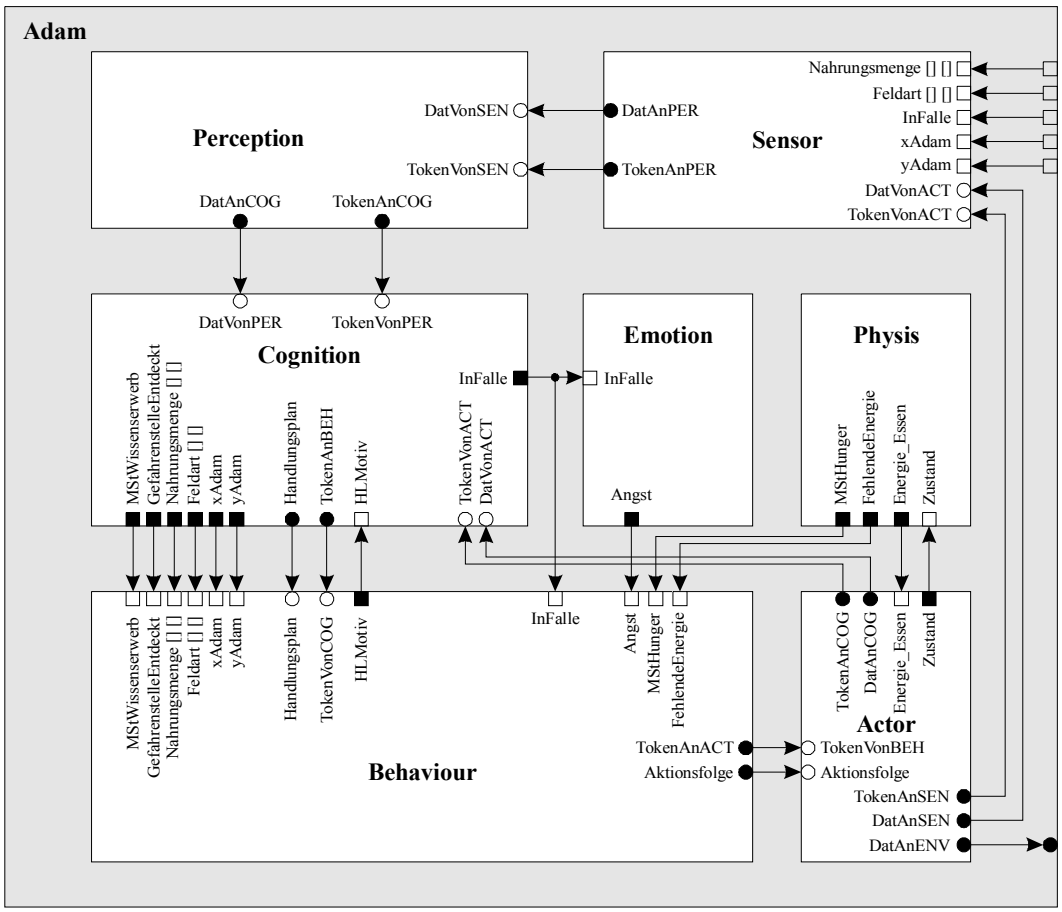

Abbildung 10.27: Die Modellstruktur der Komponente Adam

Auf die in Abbildung 10.27 dargestellten Verbindungen zwischen den einzelnen Komponenten des Agenten Adam soll an dieser Stelle noch nicht eingegangen werden. Entsprechende Ausführungen dazu finden sich in den nachfolgenden Abschnitten bei der Beschreibung der jeweiligen Subkomponenten.

10.2.3.1 Das Token-Konzept zur Synchronisation der Subkomponenten

Die Subkomponenten eines *PECS*-Agenten arbeiten, sofern keine weiteren Vorkehrungen getroffen werden, nebenläufig (s. Kapitel 5). Für die korrekte Funktionsweise des Agenten Adam ist es jedoch erforderlich, die Aktivitäten einzelner Komponenten der Agentenarchitektur aufeinander abzustimmen. Adam kann beispielsweise erst dann situationsabhängig Entscheidungen über die Auswahl bestimmter Verhaltensweisen treffen, wenn entsprechende Situationsparameter sensorisch, perzeptiv und kognitiv vorverarbeitet worden sind. Ebenso ist Adam nicht in der Lage Aktionen auszuführen, wenn diese Aktionen nicht vorher erzeugt und zur Ausführung freigegeben worden sind. In dieser Hinsicht besteht also die Notwendigkeit, die nebenläufigen Aktivitäten voneinander abhängiger Komponenten durch geeignete Mechanismen einzuschränken, um auf diese Weise ein koordiniertes Zusammenspiel dieser Komponenten zu erhalten.

Im vorliegenden Modell bilden die Komponenten *Sensor*, *Perception*, *Cognition*, *Behaviour* und *Actor* eine Wirkkette, die von der Verarbeitung sensorischer Informationen über die Speicherung von Wissensinhalten bis hin zur Auswahl und Ausführung situationsspezifischer Verhaltensweisen und Aktionen reicht. Zwischen diesen Komponenten bestehen Datenabhängigkeiten, die eine unkontrollierte und nebenläufige Ausführung der Komponenten unmöglich machen und daher einen geeigneten Koordinationsmechanismus erfordern.

Die Grundlage für die Koordination dieser Komponenten bildet das sogenannte *Token-Konzept*. Muß eine Komponente hinsichtlich bestimmter Berechnungsprozesse in koordinierter Weise mit anderen Komponenten zusammenarbeiten, so darf diese Komponente für diese Prozesse nur dann aktiv werden, wenn eine entsprechende Synchronisationsmarke bzw. ein entsprechendes Token in der Komponente vorhanden ist. Befindet sich kein Token auf dem dafür vorgesehenen Aufenthaltsort, muß die Komponente mit ihren Berechnungen warten, bis ihr ein geeignetes Token von einer anderen Komponente zugeteilt worden ist. Auf diese Weise kann eine komponentenübergreifende Sequentialisierung von Berechnungsprozessen erreicht werden.

Die Anzahl der im System befindlichen Tokens richtet sich grundsätzlich nach der Anzahl der nebenläufig auszuführenden Berechnungsprozesse in der Agentenarchitektur. Die Subkomponenten des Agenten Adam kommen mit einem einzigen Token zur Koordination ihrer Aktivitäten aus.

Die Weitergabe des Tokens von Komponente zu Komponente erfolgt dezentral im Rahmen der Dynamik der einzelnen Komponenten. Neben der Dynamikbeschreibung zur Berechnung der eigentlichen Zustandsübergänge ist in den einzelnen Komponenten der Agentenarchitektur daher auch eine Dynamik zur Weitergabe des Tokens angelegt. Nach Abschluß ihrer Aktivitäten schiebt eine Komponente das Token an diejenige Komponente weiter, die im Rahmen des betrachteten Gesamtprozesses als nächste an der Reihe ist. Es existiert also keine zentrale Steuerungsinstanz, die für die Koordination der Subkomponenten des Agenten zuständig ist.

10.2.3.2 Die Komponente *Sensor*

Aufgabe und Aufbau der Komponente Sensor

Die Komponente *Sensor* (s. Abbildung 10.27) dient der Aufnahme von Informationen aus der Umwelt des Agenten. Sie verfügt zu diesem Zweck über lesenden Zugriff auf Modellgrößen in der Komponente *Environment* und wird mit Hilfe der internen Aktionen *Prüfen* und *Explorieren*, die in der Komponente *Actor* erzeugt und angestoßen werden, aktiviert.

In bezug auf den Eingang von Informationen steht die Komponente *Sensor* mit der Komponente *Environment* sowie der Komponente *Actor* in Verbindung. Aus der Komponente *Environment* werden mit Hilfe der Eingangsgrößen *Narungsmenge*, *Feldart*, *InFalle*, *xAdam* und *yAdam* Informationen über den aktuellen Zustand der Umwelt wie auch Adams Position abgelesen und für die weitere Verarbeitung durch nachgeschaltete Komponenten der Agentenarchitektur verfügbar gemacht. Die Verbindung der Komponente *Sensor* zur Komponente *Actor* dient der Übermittlung von internen Aktionen, für deren Bearbeitung sich die Komponente *Sensor* verantwortlich zeichnet. In der Eingangslocation *TokenVonACT* findet die Komponente *Sensor* die Synchronisationsmarke vor. Die Eingangslocation *DatVonACT* ermöglicht den Zugriff auf die nächste zur Ausführung anstehende, interne Aktion.

Die grundlegende Arbeitsweise der Komponente Sensor

Grundsätzlich nimmt die Komponente *Sensor* ihre Arbeit auf, wenn sie durch die Komponente *Actor* zur Ausführung einer internen Aktion aufgefordert wird. Dies ist der Fall, wenn in den beiden Eingangslocations sowohl die Synchronisationsmarke als auch eine interne Aktion vorliegen.

Handelt es sich bei der internen Aktion um die Aktion *Explorieren*, so liest die Komponente *Sensor* die Werte für die Modellgrößen, die den aktuellen Aufenthaltsort des Agenten charakterisieren, aus der Komponente *Environment* aus, speichert diese Werte auf einem mobilen Datenelement *DatenTransfer* und stellt dieses Datenelement in der Location *DatAnPER* für die weitere Verarbeitung durch die Komponente *Perception* zur Verfügung. Wird die Komponente *Sensor* mit Ausführung der Aktion *Prüfen* beauftragt, so wird an die Komponente *Perception* nicht der gesamte Umfang der grundsätzlich zur Verfügung stehenden Informationen weitergereicht, sondern nur ein Hinweis, ob das zu überprüfende Feld in der Umwelt ein Gefahrenfeld ist. Nach Abschluß aller Aktivitäten zur Aktionsausführung stellt die Komponente *Sensor* die Synchronisationsmarke in die Location *TokenAnPER* ein und aktiviert damit die Komponente *Perception*.

10.2.3.3 Die Komponente *Perception*

Die Komponente *Perception* dient der Modellierung der Wahrnehmungsfunktionen des Agenten Adam. Wie in Abschnitt 10.1.2.2 bei der Beschreibung der Aktion *Explorieren* bereits angedeutet, tritt Adams selektive Wahrnehmung in Aktion, wenn auf Nahrungsfeldern nur geringe Nahrungsmengen vorrätig sind. Befindet sich auf einem explorierten Nahrungsfeld eine Menge an Nahrung, die Adams Wahrnehmungsschwelle unterschreitet, so kann er die verfügbare Nahrung noch nicht als solche erkennen und liefert an die Komponente *Cognition* die Information weiter, daß das betreffende Feld in der Umwelt noch keine Nahrung zur Verfügung stellt. Alle anderen Informationen, die von der Komponente *Sensor* an die Komponente *Perception* zur weiteren Verarbeitung herangetragen werden, werden ohne weitere Modifikationen an die Komponente *Cognition* geleitet.

Die Aktivierung der Komponente *Perception* erfolgt ereignisgesteuert. Liegen sowohl das Token in der Location *TokenVonSEN* als auch ein Datenpaket, das sensorische Informationen umfaßt, in der Location *DatVonSEN* vor, so nimmt die Komponente *Perception* ihre Arbeit auf. Sie verschiebt dazu die mobilen Datenelemente auf die Location *DatAnCog*, von der sie durch die Komponente *Cognition* zugegriffen werden können, und modifiziert gegebenenfalls nach vorstehend beschriebenem Mechanismus die Information über die verfügbare Nahrungsmenge. Das Token wandert zur Aktualisierung der Umweltinformationen des Agenten an die Komponente *Cognition*.

10.2.3.4 Die Komponente *Cognition*

Aufgaben und grundlegender Aufbau der Komponente Cognition

Die Komponente *Cognition* ist die umfangreichste aller Komponenten der Agentenarchitektur. Sie erfüllt eine Reihe von Aufgaben, die im folgenden aufgelistet sind:

- Verwaltung der Umweltinformationen
- Modellierung des Lernprozesses in bezug auf das Nachwachsen von Nahrung
- Modellierung des Vergessens bestehender Umweltinformationen
- Verwaltung der kognitiven Zustandsvariable *Kompetenz*
- Verwaltung des kognitiven Motivs *Wissenserwerb*
- Bestimmung einer Handlungsalternative in Abhängigkeit des handlungsleitenden Motivs
- Festlegung von Handlungszwischenzielen
- Erstellung eines Handlungsplans zur Realisierung des ausgewählten Handlungszwischenziels

Aufgrund der Vielfalt an Aufgaben, die die Komponente *Cognition* zu erfüllen hat, umfaßt diese Komponente eine Reihe von Modellelementen, die ihren in-

ternen Zustand aufnehmen und beschreiben. Ebenso wird in der Komponente *Cognition* eine Menge von Eingangselementen benötigt, mit deren Hilfe Verbindungen zu den anderen Komponenten der Agentenarchitektur hergestellt werden.

Eingangselemente der Komponente *Cognition*. Die Komponente *Cognition* steht in bezug auf ihre Inputs mit den Komponenten *Perception*, *Behaviour* und *Actor* in Verbindung. Von der Komponente *Perception* erhält die Komponente *Cognition* mit Hilfe der Eingangslocation *DatVonPER* Informationen über wahrgenommene, sensorische Daten, ebenso wie die Synchronisationsmarke, die in der Eingangslocation *TokenVonPER* abgelegt wird. Aus der Komponente *Behaviour* bezieht die Komponente *Cognition* mit Hilfe der Eingangsgröße *HLMotiv* das aktuell gültige, handlungsleitende Motiv. Die Komponente *Actor* beauftragt die Komponente *Cognition* mit der Ausführung der internen Aktion *Planen*, indem sie in der Location *DatAnCog*, die in der Komponente *Cognition* mit Hilfe der Eingangslocation *DatVonACT* zugegriffen werden kann, entsprechende mobile Datenelemente ablegt. Die Synchronisationsmarke wird von der Komponente *Actor* kommend in der Eingangslocation *TokenVonACT* zugänglich gemacht.

Grundlegende Modellelemente der Komponente *Cognition*. Die Komponente *Cognition* umfaßt eine Reihe von Modellelementen, die im wesentlichen der Informationsspeicherung, der Verwaltung kognitiver Zustände, sowie der Realisierung der Handlungsplanung dienen.

Im Zusammenhang mit der Speicherung von Informationen verwaltet die Komponente *Cognition* eine Menge von Modellelementen, die zur Aufnahme von Informationen über Adams Umwelt, der Modellierung der Prozesse Lernen und Vergessen, sowie der Verwaltung von Adams Kompetenzempfinden dienen. Zu diesen Modellelementen zählen im wesentlichen

- das zweidimensionale Array *Feldart*, das den Typ für bekannte Felder in der Umwelt aufnimmt,
- das zweidimensionale Array *Nahrungsmenge*, das die beobachtete Nahrungsmenge für bekannte Felder in der Umwelt speichert,
- die Zustandsvariable *InFalle*, die anzeigt, ob sich Adam zum betrachteten Zeitpunkt in einer Falle befindet,
- die beiden Koordinaten *xAdam* und *yAdam* für Adams aktuelle Position,
- die beiden Lernschranken *DeltaSicherJa* und *DeltaSicherNein*, die den Zeitpunkt der Verfügbarkeit von Nahrung auf den Nahrungsfeldern eingrenzen,
- die Location *Gedächtnis*, die Gedächtniseinträge zur Modellierung des Vergessens bestehender Informationen aufnimmt; die Elemente in dieser Warteschlange sind dabei aufsteigend nach den Werten ihrer Zustandsvariable

TVergessen sortiert, die den Zeitpunkt angibt, zu dem ein Gedächtnisinhalt aus dem Gedächtnis verlorengeht,

- die Zustandsvariable *DeltaVergessen*, die die reguläre Latenzzeit einer Information im Gedächtnis vorgibt,
- sowie die Zustandsvariable *Kompetenz*, die den aktuellen Wert für Adams Kompetenzempfinden beinhaltet und damit ein einfaches Element der Selbstreflexion darstellt.

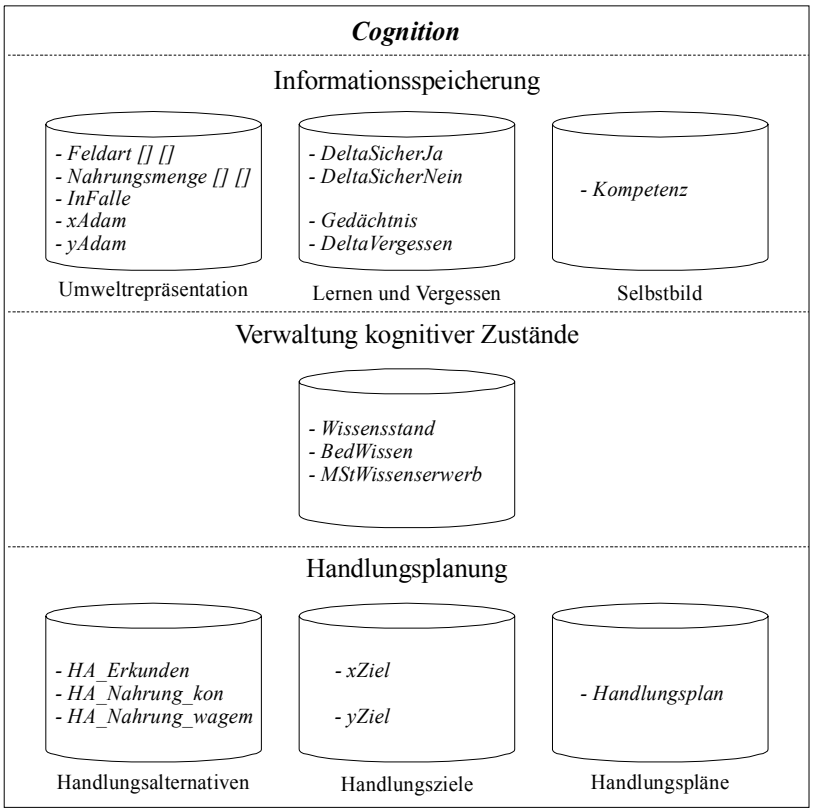

Abbildung 10.28: Wesentliche Modellelemente der Komponente Cognition

Zudem verwaltet die Komponente *Cognition* eine Reihe von Modellgrößen, die an der Berechnung der Motivstärke des kognitiven Motivs *Wissenserwerb* beteiligt sind. Es handelt sich dabei um

- die Zustandsvariable *Wissensstand*, die Aufschluß darüber gibt, wie viele Felder seiner Umwelt der Agent Adam kennt,
- die abhängige Variable *BedWissen*, die Adams Bedürfnis nach Wissen modelliert,
- und die abhängige Variable *MStWissenserwerb*, die die Motivstärke für das Motiv *Wissenserwerb* bereitstellt.

Im Rahmen der Handlungsplanung werden Modellgrößen benötigt, die die aktuelle Handlungsalternative, das Handlungszwischenziel, sowie einen gültigen Handlungsplan aufnehmen. Die ausgewählte Handlungsalternative wird in der Komponente *Cognition* mit Hilfe von Indikatoren angezeigt. Sie bewirkt eine entsprechende Selektion des Handlungszwischenziels, das mit Hilfe des Koordinatentupels (*xZiel, yZiel)* für die Erstellung des Handlungsplans abgelegt wird. Ein Handlungsplan wird als Folge von mobilen Datenelementen vom Typ *Daten-Transfer* erzeugt und in die Location *Handlungsplan* eingestellt, die von der Komponente *Behaviour* aus lesend zugegriffen werden kann.

Die Abbildung 10.28 zeigt die wesentlichen Modellelemente der Komponente *Cognition* im Überblick.

Die grundlegende Arbeitsweise der Komponente Cognition

Wie im Einführungsabschnitt bereits angedeutet, verfügt die Komponente *Cognition* aufgrund der vielfältigen Aufgaben, die sie zu erfüllen hat, über ein umfangreiches dynamisches Verhalten. Eingeschlossen sind darin Prozesse zur Aktualisierung und Entfernung von Wissensinhalten, zur Verwaltung kognitiver Zustandsvariablen, sowie zur Durchführung der Handlungsplanung des Agenten Adam. Zustandsänderungen in der Komponente *Cognition* erfolgen dabei sowohl eigendynamisch etwa im Rahmen des Vergessens bestehender Information, als auch durch Anregung von außen, beispielsweise aufgrund einer internen Aktion, deren Ausführung der Komponente *Cognition* angetragen wird, oder eines Perzepts, das in die Wissensbasis des Agenten integriert werden muß.

Die Berechnung der Motivstärke des kognitiven Motivs *Wissenserwerb*. Die Motivstärke für das kognitive Motiv *Wissenserwerb* geht ursprünglich auf die Menge an Informationen zurück, die Adam über seine Umgebung zur Verfügung hat (s. Abschnitt 10.1.2.3). Die Aktualisierung des Wissens und damit der kognitiven Zustandsvariable *Wissensstand* erfolgt bei der Integration von Perzepten in das Wissen des Agenten und wird im folgenden Abschnitt genauer dargestellt. Das aus dem Wissensstand resultierende Bedürfnis nach Wissen wird in der Komponente *Cognition* mit Hilfe einer algebraischen Gleichung modelliert, die auf Gleichung Gl. 10.8 aufsetzt. Ebenso ist eine weitere algebraische Gleichung für die Bestimmung der aktuellen Motivstärke des Motivs *Wissenserwerb* vorgesehen, die der Gleichung Gl. 10.9 folgt. Die Motivstärke des Motivs *Wissenserwerb* verändert sich also mit jeder Änderung der Anzahl an bekannten Feldern in Adams Umgebung.

Die Integration von Perzepten aus der Komponente *Perception*. Die Komponente *Perception* übermittelt wahrgenommene Sinneseindrücke, die aus der Aus-

führung der internen Aktionen *Prüfen* und *Explorieren* resultieren, zur weiteren Verarbeitung an die Komponente *Cognition*. Die Perzepte werden dazu in der Eingangslocation *DatVonPer* abgelegt und können bei Verfügbarkeit der Synchronisationsmarke von dort aus durch die Komponente *Cognition* abgeholt werden. Für die vollständige Integration der Perzepte sind innerhalb der Komponente *Cognition* eine Reihe von Verarbeitungsschritten nötig (s. Abbildung 10.29).

Zunächst wird zur Verarbeitung der Perzepte geprüft, ob sie durch Ausführung der internen Aktion *Prüfen* oder *Explorieren* gewonnen wurden.

Im Falle der Aktion *Prüfen* ist eine Veränderung von Adams Gedächtnisstrukturen nur dann nötig, wenn beim Prüfen eine Gefahrenstelle entdeckt wurde. In diesem Fall wird die neu gewonnene Information über die Art des geprüften Feldes in Adams kognitiver Landkarte eingetragen, die kognitive Zustandsvariable *Wissensstand* um den Betrag eins erhöht und zugleich Adams Gedächtnis um einen Eintrag erweitert, damit die neu hinzugekommene Information auch dem Prozeß des Vergessens unterworfen werden kann.

Liegt dem zu verarbeitenden Perzept die interne Aktion *Explorieren* zugrunde, so sind in der Komponente *Cognition* abhängig von der vorliegenden Situation verschiedene Aktualisierungsschritte durchzuführen:

- Aktualisierung des Handlungsplans

  Falls das explorierte Feld mit dem aktuellen Teilziel des Handlungsplans übereinstimmt, wurde dieses Teilziel erreicht und kann damit als erledigt aus dem Handlungsplan entfernt werden.

- Aktualisierung der Umweltrepräsentation

  Durch die Exploration eines Feldes liegen Informationen über die Umwelt vor, die in Adams mentale Repräsentation integriert werden können. Zu dem explorierten Feld werden demnach die festgestellte Nahrungsmenge und die Art des Feldes gespeichert. Darüber hinaus wird Adams Information über seine aktuelle Position, die in den Koordinaten *xAdam* und *yAdam* abgelegt ist, erneuert, sowie der Indikator *InFalle* gesetzt, wenn sich Adam in einer Gefahrenstelle wiederfindet.

- Aktualisierung der Zustandsvariable *Wissensstand*

  Hat Adam mit Hilfe der Aktion *Explorieren* ein Feld erkundet, das ihm zum betrachteten Zeitpunkt noch unbekannt war oder dessen Bedeutung er über die Zeit hinweg wieder vergessen hatte, so erweitert sich sein aktueller Wissensstand um ein Feld.

- Aktualisierung des Gedächtnisses

  Analog zur Aktion *Prüfen* werden auch durch die Aktion *Explorieren* gewonnene Informationen mit einer entsprechenden Latenzzeit in die Location *Gedächtnis* eingeordnet und damit dem Prozeß des Vergessens unterworfen.

Abbildung 10.29: Die grundsätzlichen Verarbeitungsschritte bei der Integration von Perzepten

- Aktualisierung der Lernschranken

  Handelt es sich bei dem explorierten Feld um ein bereits bekanntes Nahrungsfeld, so muß überprüft werden, ob aufgrund der neu hinzugekommenen Informationen eine Anpassung der Lernschranken für das Nachwachsen der Nahrung möglich ist. Liegt auf dem Nahrungsfeld bereits wieder Nahrung vor, so kann gegebenenfalls die Lernschranke *DeltaSicherJa* nach unten korrigiert werden. Befindet sich noch keine Nahrung auf dem Nahrungsfeld, so ist unter Umständen eine Verschiebung der Lernschranke *DeltaSicherNein* möglich.

- Aktualisierung der Zustandsvariable *Kompetenz*

  Wurde mit Hilfe der Aktion *Explorieren* ein Nahrungsfeld erkundet und hat Adam dieses Feld gezielt mit der Absicht der Nahrungsaufnahme aufgesucht, so kann anschließend eine Bewertung des Erfolgs seiner Aktivitäten vorgenommen und dadurch die kognitive Zustandsvariable *Kompetenz* angepaßt werden. Findet Adam auf dem explorierten Feld Nahrung vor, so führt dies zu einer Erhöhung seiner Kompetenz, andernfalls zu einer Verringerung.

- Weitergabe der Synchronisationsmarke an die Komponente *Behaviour*

  Nach Abschluß aller Tätigkeiten zur Aktualisierung der kognitiven Strukturen wird die Synchronisationsmarke an die Komponente *Behaviour* weitergegeben, die mit der Festlegung der auszuführenden Verhaltensweisen fortfährt.

Das Vergessen bestehender Gedächtnisinhalte. Das Vergessen von Gedächtnisinhalten gehört zur Eigendynamik der Komponente *Cognition* und ist als eigenständiger Prozeß fortwährend in der Komponente *Cognition* aktiv. Alle Felder, die Adam einer genaueren Untersuchung unterzieht, bleiben nicht für immer, sondern nur für einen gewissen Zeitraum in Adams Gedächtnis erhalten. Um diesen Sachverhalt adäquat abzubilden, werden alle Informationen über die Umwelt, die in der Komponente *Cognition* ankommen, mit einem Zeitstempel versehen und, repräsentiert durch ein mobiles Datenelement, in der Warteschlange *Gedächtnis* abgelegt. Der Zeitstempel gibt dabei den Zeitpunkt an, zu dem der Gedächtniseintrag wieder vergessen wird, d. h. aus der Warteschlange entfernt werden muß.

Um die Verwaltung dieser Warteschlange so einfach wie möglich zu halten, sind ihre Elemente nach aufsteigenden Austrittszeitpunkten geordnet abgelegt. Das Element, das als nächstes die Warteschlange verlassen wird, steht also an erster Stelle. Erreicht die Simulationszeit den Austrittszeitpunkt des ersten Elements in der Warteschlange, so wird dieses Element aus der Warteschlange entfernt und ebenso alle anderen Informationen in der Komponente *Cognition*, die durch diesen Gedächtniseintrag repräsentiert werden, gelöscht. Der Wissensstand des Agenten Adam verringert sich jeweils um ein Feld, wenn ein Eintrag aus der Warteschlange *Gedächtnis* entfernt wird.

Die Realisierung der internen Aktion *Planen*. Neben der Komponente *Perception* ist auch die Komponente *Actor* in der Lage, die Dynamik der Komponente *Cognition* in Gang zu setzen. Mit Hilfe der internen Aktion *Planen* kann die Komponente *Actor* die Komponente *Cognition* damit beauftragen, einen neuen Handlungsplan für die vorliegende Situation zu entwickeln. Sie übermittelt der Komponente *Cognition* zu diesem Zweck sowohl ein mobiles Datenelement vom Typ *DatenTransfer*, das den Auftrag *Planen* enthält, als auch die Synchronisationsmarke, um anzuzeigen, daß alle Vorarbeiten für die Planung abgeschlossen sind. Liegen beide Datenelemente in den Eingangslocations *DatVonACT* und *TokenVonACT* vor, so nimmt die Komponente *Cognition* den mehrstufigen Planungsprozess in Angriff.

Als erstes werden dazu gegebenenfalls noch verbliebene und nicht abgearbeitete Teilziele eines abgebrochenen Handlungsplans von der Location *Handlungsplan* entfernt. Im Anschluß daran wird gemäß des in Kapitel 10.1.2.3 geschilderten, grundsätzlichen Ablaufs eine der Situation angemessene Handlungsalternative bestimmt. Für diese Auswahlentscheidung wird das aktuell gültige, handlungsleitende Motiv als maßgebliche Größe herangezogen. Ist das Motiv *Wissenserwerb* handlungsleitend, so folgt daraus unmittelbar die Handlungsalternative *Erkunden*. Im Falle des Motivs *Hunger* begibt sich der Agent Adam auf Nahrungssuche, allerdings entscheidet dabei der aktuelle Wert der kognitiven Zustandsvariable *Kompetenz*, ob Adam die eher konservative oder eher wagemutige Alternative wählt.

Liegt die Handlungsalternative fest, wird ein Zielfeld in der Umwelt bestimmt, das als Zwischenziel für die Handlungsplanung angesetzt wird. Die Kriterien für die Zielfeldauswahl, sowie der an dieser Stelle verwendete Suchalgorithmus wurden bereits im Rahmen der Modellbeschreibung in Kapitel 10.1.2.3 dargelegt. Als Resultat liefert die Zielfeldbestimmung ein Koordinatentupel (*xZiel*, *yZiel*), das die Position des anzustrebenden Feldes in der Umwelt angibt.

Im nächsten Schritt wird die Planung des Weges von Adams aktueller Position zum Zielfeld des Handlungsplans vorgenommen. Das Planungsverfahren basiert, wie in Kapitel 10.1.2.3 bereits beschrieben auf einer Suche nach dem Greedy-Prinzip, wobei Informationen über bestehende Gefahrenstellen in die Suche mit einbezogen werden. Am Ende der Planung steht in der Location *Handlungsplan* ein neuer Handlungsplan zur Verfügung, der aus einer Folge von Teilzielen, d. h. von Feldern, die Adam sukzessive durch Ausführung geeigneter Verhaltensweisen anzusteuern versucht, besteht. Nach diesem Schritt ist die Handlungsplanung abgeschlossen, so daß mit der Umsetzung des Handlungsplans, für die die Komponente *Behaviour* zuständig ist, begonnen werden kann. Zu diesem Zweck wird die Synchronisationsmarke von der Komponente *Cognition* an die Komponente *Behaviour* übertragen.

10.2.3.5 Die Komponente *Emotion*

Die Komponente *Emotion* verwaltet Modellgrößen, die den emotionalen Zustand eines Agenten beschreiben. Im Falle von Adam besteht der emotionale Zustand der Einfachheit halber nur aus einer Zustandsvariable *Angst* (s. Kapitel 10.1.2.1), die angibt, wie ängstlich Adam zu einem gegebenen Zeitpunkt ist. Der aktuelle Wert der Zustandsvariable *Angst* beeinflußt die Auswahl der Verhaltensweisen von Adam.

Neben der Zustandsvariable *Angst* existiert in der Komponente Emotion eine Eingangsgröße *InFalle*, die lesenden Zugriff auf die gleichnamige Zustandsvariable in der Komponente *Cognition* ermöglicht. Appraisal-Ansätzen zur Erzeugung von Emotionen folgend (s. Abschnitt 10.1.2.1), erhöht sich der aktuelle Wert der Zustandsvariable *Angst* immer dann, wenn mit Hilfe der Eingangsgröße *InFalle* festgestellt wird, daß Adam in eine Gefahrenstelle geraten ist. Die Erhöhung erfolgt dabei um einen konstanten Betrag bis zu einer maximalen Obergrenze *AngstMax*. Bei der Generierung der Emotion *Angst* handelt es sich also um eine von außen induzierte Dynamik.

Ebenso verfügt die Komponente *Emotion* über eine Eigendynamik, die dafür sorgt, daß Adam nicht für unbegrenzte Zeit ängstlich bleibt, nachdem er in eine Gefahrenstelle getappt ist. Das Abklingen der Angst wird in der Komponente *Emotion* mit Hilfe einer Differentialgleichung modelliert, die Gleichung Gl. 10.5 folgt.

10.2.3.6 Die Komponente *Physis*

Aufgabe und Aufbau der Komponente Physis

Die Komponente *Physis* enthält und verwaltet alle Modellelemente, die den physischen Zustand des Agenten Adam beschreiben. Insbesondere werden in dieser Komponente alle Modellgrößen zur Verfügung gestellt, die an der Generierung des physischen Motivs *Hunger* beteiligt sind. Es handelt sich dabei um

- die Zustandsvariable *Energie*, die das aktuelle Energieniveau von Adam beschreibt,
- die abhängige Variable *BedNahrung*, die die Stärke von Adams Bedürfnis nach Nahrung modelliert,
- sowie die abhängige Variable *MStHunger*, die für die Bereitstellung der Motivstärke des Motivs *Hunger* zuständig ist.

Zudem stellt die Komponente Physis eine abhängige Variable *FehlendeEnergie* bereit, die angibt, wieviel Energie zum betrachteten Zeitpunkt benötigt wird, um Adams Energiehaushalt vollständig zu regenerieren. Dieser Wert wird an die Komponente *Behaviour* gemeldet und dient dort zur Parametrisierung der externen Aktion *Essen*.

Da Adams Energieverbrauch in entscheidender Weise nicht nur von der Zeit, sondern auch von seinen ausgeführten Aktionen abhängt, benötigt die Komponente *Physis* zudem lesenden Zugriff auf den aktuellen Ausführungszustand der Komponente *Actor*, der angibt, welche Aktion zum betrachteten Zeitpunkt in dieser Komponente ausgeführt wird.

Die grundlegende Arbeitsweise der Komponente Physis

In der Komponente *Physis* finden überwiegend zeitkontinuierliche Zustandsübergänge statt, die mit Hilfe von Differentialgleichungen und algebraischen Gleichungen beschrieben werden.

Den Ausgangspunkt für die Berechnung der Motivstärke des Motivs *Hunger* bildet die physische Zustandsvariable *Energie*. Der Zeitverlauf dieser Modellgröße wird mit Hilfe zweier Typen von Differentialgleichungen beschrieben, die einerseits die Zunahme an Energie durch Ausführung der Aktion *Essen* und andererseits die Abnahme an Energie durch Ausführung der übrigen internen und externen Aktionen angeben. Gemäß des aktuellen Ausführungszustands der Komponente *Actor*, der in der Eingangsgröße *Zustand* zur Verfügung steht, wird in der Komponente *Physis* mit Hilfe einer Fallunterscheidung entweder die Differentialgleichung zur Beschreibung der Energiezunahme (Gl. 10.2) oder eine der ausgeführten Aktion entsprechende Differentialgleichung zur Beschreibung der Energieabnahme (Gl. 10.3) aktiviert.

Die Berechnung der Werte für die abhängigen Variablen *BedNahrung* und *MStHunger* erfolgt auf der Grundlage der beiden algebraischen Gleichungen Gl. 10.6 und Gl. 10.7.

Falls Adams Energiehaushalt zur Neige geht und schließlich den Wert *EnergieMin [EnE]* unterschreitet, wird dieser Sachverhalt als „virtueller Tod“ von Adam interpretiert. Die Abfrage für den Wert der Zustandsvariable *Energie* auf Unterschreitung der vorgegebenen Untergrenze erledigt eine einfache Regel in der Komponente *Physis*, die bei Aktivierung den Simulationslauf beendet.

10.2.3.7 Die Komponente *Behaviour*

Aufgabe und Aufbau der Komponente Behaviour

Die Komponente *Behaviour* dient zur Steuerung des reaktiven Verhaltens des Agenten Adam. Sie enthält eine Menge von vordefinierten Verhaltensregeln, die im wesentlichen die Auswahl des handlungsleitenden Motivs, die Auswahl und Erstellung von Aktionsfolgen, sowie die Weitergabe der Synchronisationsmarke an die Komponente *Actor* festlegen (s. Abbildung 10.30).

Die Bedingungen der Verhaltensregeln nehmen Bezug auf die Modellgrößen der Komponenten *Physis*, *Emotion* und *Cognition*, in denen der interne Zustand des Agenten abgelegt ist. In der Komponente *Behaviour* müssen relevante Mo-

dellgrößen dieser Komponenten also mit Hilfe entsprechender Eingangsgrößen verfügbar gemacht werden. In Abbildung 10.30 sind diese Eingangsgrößen mit Hilfe der nicht ausgefüllten Quadrate und Kreise angedeutet.

Als Resultat der Auswertung der Verhaltensregeln wird in der Komponente *Behaviour* das handlungsleitende Motiv festgelegt und in der Zustandsvariable *HLMotiv* für die weitere Verarbeitung in der Komponente *Cognition* verfügbar gemacht. Ebenso wird eine an die gegebene Situation angepaßte Aktionsfolge erstellt, die zur Realisierung des aktuellen Teilziels des Handlungsplans dient. Die Aktionsfolge wird in der Location *Aktionsfolge* abgelegt und dadurch der Komponente *Actor* zur Ausführung übergeben. Die Aktionsbeschreibungen, die in der Location *Aktionsfolge* hinterlegt werden, werden dabei mit Hilfe von mobilen Datenelementen des Typs *DatenTransfer* modelliert und mit entsprechenden Kennungen für die auszuführende Aktion sowie gegebenenfalls weiteren Parametern zur näheren Bestimmung der Aktion versehen.

Da im Zusammenhang mit Adams Verhaltenssteuerung im Anschluß an die Komponente *Behaviour* immer die Komponente *Actor* ihre Arbeit aufnehmen muß, stellt die Komponente *Behaviour* eine Location *TokenAnACT* bereit, in der die Synchronisationsmarke zugunsten der Komponente *Actor* abgelegt werden kann.

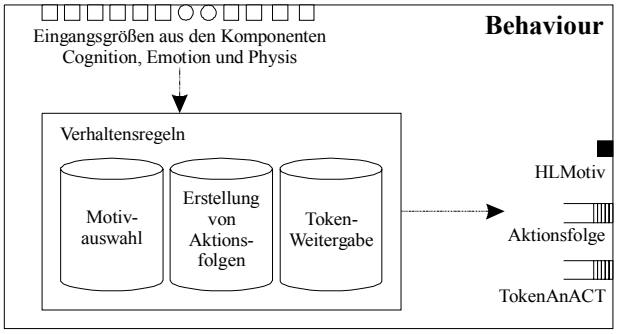

Abbildung 10.30: Die Grundstruktur der Komponente Behaviour

Die grundlegende Arbeitsweise der Komponente Behaviour

Die Dynamik der Komponente *Behaviour* basiert auf ausschließlich zeitdiskreten Zustandsübergängen, die mit Hilfe von Regeln bzw. Produktionen modelliert werden. Die Gesamtmenge an Regeln, die die Komponente *Behaviour* umfaßt, kann dabei in drei Teilmengen partitioniert werden, deren Elemente

- zur Bestimmung des handlungsleitenden Motivs,
- zur Auswahl und Erzeugung von Aktionsfolgen,
- sowie zur Weitergabe der Synchronisationsmarke an die Komponente Actor dienen.

Die Bestimmung des handlungsleitenden Motivs. Für die Realisierung der Motivselektion sind im vorliegenden Modell zwei voneinander unabhängige Regeln notwendig. Die erste Regel signalisiert den Wechsel des handlungsleitenden Motivs von *Wissenserwerb* auf *Hunger* und schaltet, sobald die Motivstärke des Motivs *Hunger* die Motivstärke des Motivs *Wissenserwerb* erreicht. Die zweite Regel beschreibt den Motivwechsel in der anderen Richtung. Sie bewirkt, daß das handlungsleitende Motiv von *Hunger* auf *Wissenserwerb* wechselt, und schaltet erst dann, wenn Adams Nahrungsbedürfnis vollständig gestillt werden konnte. Auf diese Weise erhält das Motiv *Hunger* aufgrund seiner größeren Bedeutung für Adams Überlebensfähigkeit eine bevorzugte Stellung in seiner Verhaltensauswahl. Der tatsächliche Wechsel des handlungsleitenden Motivs findet allerdings nicht sofort mit dem Schalten der entsprechenden Regel statt, sondern tritt erst dann in Kraft, wenn die Komponente *Behaviour* wieder die Möglichkeit hat, eine neue Aktionsfolge zu bestimmen. Dadurch wird sichergestellt, daß die Aktion, die Adam zu einem gegebenen Zeitpunkt ausführt, immer konform mit seinem handlungsleitenden Motiv ist.

Die Auswahl und Erzeugung von Aktionsfolgen. Im Rahmen der Verhaltenssteuerung des Agenten Adam wird ein Handlungsplan nach seiner Erzeugung in der Komponente *Cognition* an die Komponente *Behaviour* zur Ausführung übergeben. Die Komponente *Behaviour* arbeitet dann in Kooperation mit den anderen Komponenten der Agentenarchitektur die einzelnen Teilziele des Handlungsplans der Reihe nach ab. Für die Erreichung eines Teilziels reicht zumeist eine einzige Aktion nicht aus. In der Regel werden mehrere Aktionen in Folge benötigt, deren Auswahl und Erzeugung Teil der Dynamik der Komponente *Behaviour* ist.

Die Auswahl und Erzeugung von Aktionsfolgen geschieht in der Komponente *Behaviour* aus Gründen der Übersichtlichkeit in zwei Schritten. Im ersten Schritt wird eine für die vorliegende Situation geeignete Aktionsfolge ausgewählt. Im zweiten Schritt werden die Aktionsbeschreibungen, die zu der ausgewählten Aktionsfolge gehören, tatsächlich erzeugt, gegebenenfalls mit Parametern versehen und schließlich in der Location *Aktionsfolge*, die für die Komponente *Actor* sichtbar ist, abgelegt.

Die Regeln zur Auswahl einer geeigneten Aktionsfolge sind in der Entscheidungstabelle in Abbildung 10.24 zusammengefaßt. Eine Spalte der Entscheidungstabelle repräsentiert dabei grundsätzlich eine eigene Regel in der Komponente *Behaviour*. Der Bedingungsteil einer derartigen Regel setzt sich aus einer Konjunktion der in den Zeilen annotierten Bedingungen zusammen. Führen mehrere Regeln auf dieselbe Aktionsfolge, so können diese Regeln zusammengefaßt werden, indem eine Gesamtbedingung gebildet wird, die durch eine Disjunktion der Einzelbedingungen entsteht. Die ausgewählte Aktionsfolge wird mit Hilfe eines spezifischen Indikators angezeigt, der im nächsten Schritt zur Generierung der zugehörigen Aktionsfolge führt.

Grundsätzlich besteht eine Aktionsfolge aus einer Sequenz von Aktionsbeschreibungen, die mit Hilfe von mobilen Datenelementen des Typs *DatenTransfer* abgebildet werden. Bei der Generierung einer Aktionsfolge wird eine entsprechende Anzahl an Datenelementen erzeugt und mit Informationen über die auszuführende Aktion sowie gegebenenfalls mit Parametern, die die Aktion näher bestimmen, bestückt. Als Parameter können dabei etwa bei den Aktionen *Prüfen* und *Gehen* die Koordinaten des zu erreichenden Feldes oder bei der Aktion *Essen* die Menge an Nahrung, die aufgenommen werden soll, angegeben werden.

```
# Regel zur Auswahl der Aktionsfolge
# ----------------------------------
WHENEVER
   NUMBER ( Aktionsfolge ) = 0
   AND Feldart [Handlungsplan:DatenTransfer[1].x]
               [Handlungsplan:DatenTransfer[1].y]
       = ´Unbekannt´
   AND Angst > AngstSchranke
DO
   SIGNAL AF_PrGeEx;
END

# Regel zur Erzeugung und Parametrisierung der Aktionen
# -----------------------------------------------------
ON AF_PrGeEx
DO
   Aktionsfolge^: ADD 1 NEW DatenTransfer
      CHANGING
         Aktion^ := ´Pruefen´;
         x^ := Handlungsplan:DatenTransfer[1].x;
         y^ := Handlungsplan:DatenTransfer[1].y;
      END

   Aktionsfolge^: ADD 1 NEW DatenTransfer
      CHANGING
         Aktion^ := ´Gehen´;
         x^ := Handlungsplan:DatenTransfer[1].x;
         y^ := Handlungsplan:DatenTransfer[1].y;
      END

   Aktionsfolge^: ADD 1 NEW DatenTransfer
      CHANGING
         Aktion^ := ´Explorieren´;
      END
END
```

Abbildung 10.31: Die Regeln zur Erzeugung der Aktionsfolge Prüfen – Gehen – Explorieren

Der Codeausschnitt in Abbildung 10.31 zeigt exemplarisch und in vereinfachter Form die beiden Regeln der Komponente *Behaviour*, die für die Auswahl und Erzeugung der Aktionsfolge *Prüfen – Gehen – Explorieren* zuständig sind. Diese Aktionsfolge wird ausgewählt, wenn sich Adam in einem ängstlichen Zustand be-

findet und er keine Informationen über den Typ des als nächstes zu betretenden Feldes zur Verfügung hat. Er wird dadurch das angestrebte Feld in seiner Umwelt zunächst einmal auf eine Gefahrenstelle hin überprüfen, bevor er das Feld tatsächlich betritt. Nach der Aktion *Gehen* wird er dann die Aktion *Explorieren* ausführen, um Informationen über das erreichte Feld zu gewinnen.

Die Funktion *NUMBER* aus *Simplex-MDL* (Schmidt, 2000) liefert dabei die Anzahl der aktuell auf einer Location befindlichen Elemente. Die Anweisung *SIGNAL* setzt eine Indikation, die in den Bedingungen von Regeln abgefragt werden kann, und das *CHANGING*-Konstrukt kapselt Änderungen, die an Attributen aktuell ausgewählter, mobiler Datenelemente vorgenommen werden sollen.

Im Falle einer Neuplanung kann die Notwendigkeit entstehen, vor der Generierung einer neuen Aktionsfolge Reste einer noch nicht zu Ende gebrachten Aktionsfolge zu entfernen. Findet etwa ein Motivwechsel von *Wissenserwerb* nach *Hunger* statt, bevor eine aus dem Wissenserwerb resultierende Aktionsfolge vollständig abgearbeitet werden konnte, so befindet sich der verbliebene Rest der Aktionsfolge zunächst noch in der Location *Aktionsfolge*. Da diese Aktionen nach dem Motivwechsel keine Gültigkeit mehr haben, müssen sie vor der Erstellung einer neuen Aktionsfolge, die dem Motiv *Hunger* entspringt, aus der Location *Aktionsfolge* entfernt werden.

Die Weitergabe der Synchronisationsmarke. Nachdem in der Komponente *Behaviour* entweder eine neue Aktionsfolge erstellt worden ist oder die Entscheidung getroffen worden ist, daß eine bestehende Aktionsfolge weiter ausgeführt werden soll, wird die Komponente *Actor* durch Weitergabe der Synchronisationsmarke dazu aufgefordert, ihre Arbeit aufzunehmen. Eine einfache Regel innerhalb der Komponente *Behaviour* sorgt dabei für die Verschiebung der Synchronisationsmarke von der Eingangslocation *TokenVonCOG* an die Location *TokenAnACT*.

10.2.3.8 Die Komponente *Actor*

Aufgabe und Aufbau der Komponente Actor

Die Komponente *Actor* ist für die Ausführung der Aktionen des Agenten Adam zuständig. Im Rahmen dieser Komponente werden interne wie auch externe Aktionen, die durch die Verhaltenssteuerung in der Komponente *Behaviour* ausgewählt werden, in konkrete Aktionsbefehle umgesetzt.

Die Komponente *Actor* arbeitet zeitdiskret nach dem Prinzip einer einfachen Bedieneinheit. Die Abarbeitung einer anstehenden Aktion wird dabei als zeitverbrauchender Vorgang aufgefaßt. Sobald eine Aktion zur Bearbeitung ansteht, wird ein zeitverbrauchender Vorgang eingeleitet, der die Ausführungsdauer der Aktion nachbildet. Nachdem die Bearbeitungszeit der Aktion verstrichen ist, wird die Berechnung der Konsequenzen der ausgeführten Aktion angestoßen. Dazu

verschickt die Komponente *Actor* eine Benachrichtigung an diejenige Komponente, deren Zustand durch die Ausführung der Aktion beeinflußt wird.

Im vorliegenden Modell werden der Komponente *Actor* sowohl interne als auch externe Aktionen, die zur Ausführung anstehen, durch die Komponente *Behaviour* in der Eingangslocation *Aktionsfolge* zur Verfügung gestellt. Diese Vorgehensweise stellt zwar eine gewisse Einschränkung hinsichtlich der parallelen Ausführbarkeit interner und externer Aktionen dar, bleibt aber im vorliegenden Modell ohne Bedeutung, da vereinfachend angenommen wird, daß der Agent Adam zu einem gegebenen Zeitpunkt nur eine Aktion ausführen kann.

Die Zeitdauern für die Ausführung der unterschiedlichen Aktionen (s. Abbildung 10.14) sind in der Komponente *Actor* mit Hilfe von Konstanten hinterlegt. Eine Ausnahme bildet dabei die Aktion *Essen*, deren Dauer abhängig von der Menge an Energie ist, die Adam zu sich nehmen wird. Je größer die aufzunehmende Menge an Energie ist, um so länger dauert die Aktion *Essen*. Konkret wird die Zeitdauer für die Aktion *Essen* dabei als Quotient aus der Gesamtmenge der aufzunehmenden Energie (Parameter *Nahrungsmenge* des mobilen Datenelements *DatenTransfer*) und der Menge an Energie, die pro Zeiteinheit konsumiert werden kann (Eingangsgröße *Energie_Essen* aus der Komponente *Physis*), errechnet.

Nach Ablauf des mit einer Aktion verknüpften, zeitverbrauchenden Vorgangs beauftragt die Komponente *Actor* diejenige Modellkomponente, an die die Aktion gerichtet ist, mit der Berechnung der entsprechenden Zustandsübergänge. Zu diesem Zweck stellt sie für jede Modellkomponente, die von einer internen oder externen Aktion betroffen sein kann, eine eigene Location zu Verfügung, in der entsprechende Aktionsbeschreibungen abgelegt werden. Um eine Komponente mit der Berechnung der aus einer ausgeführten Aktion resultierenden Zustandsübergänge zu beauftragen, verschiebt die Komponente *Actor* die Aktionsbeschreibung, die in ihrer Eingangslocation *Aktionsfolge* zur Verfügung steht, in die Location der entsprechenden Komponente und kann von dort aus zur Bearbeitung abgeholt werden.

Die Komponenten *Cognition* und *Sensor* sind für die Ausführung der Zustandsübergänge der internen Aktionen *Planen*, *Prüfen* und *Explorieren* zuständig. Entsprechende Aktionsbeschreibungen für diese beiden Komponenten werden in den Locations *DatAnCOG* und *DatAnSEN* abgelegt. Die Aktionen *Gehen*, *Befreien* und *Essen* richten sich an die Komponente *Environment*. Sie erhält die zugehörigen Aktionsbeschreibungen in der Location *DatAnENV*.

Neben den Aktionsbeschreibungen ist für eine Aktivierung der internen Komponenten der Agentenarchitektur auch die Weitergabe der Synchronisationsmarke erforderlich. Zu diesem Zweck enthält die Komponente *Actor* zwei weitere Locations *TokenAnCOG* und *TokenAnSEN*, mit deren Hilfe die Synchronisationsmarke an die Komponenten *Cognition* und *Sensor* übermittelt werden kann.

Die Komponente *Actor* verwaltet darüber hinaus eine Zustandsvariable *Zustand*, die angibt, welche Aktion zu einem gegebenen Zeitpunkt ausgeführt wird. Diese Information wird zur Berechnung des Energieverbrauchs der jeweiligen Aktion an die Komponente *Physis* weitergeleitet.

Die grundlegende Arbeitsweise der Komponente Actor

Für die Erledigung der im vorangegangenen Abschnitt geschilderten Aufgaben sind in der Komponente *Actor* im wesentlichen zwei diskrete Regeln zuständig.

```
# Regel zum Start der Bearbeitung einer Aktion
# ---------------------------------------------
WHENEVER ( NUMBER ( TokenVonBEH ) > 0 ) AND
         ( NaechsteAktion = TRUE )
DO
   IF ( Aktionsfolge:DatenTransfer[1].Aktion = ´Planen´)
   DO
      Zustand^ := ´Planend´;
      TNext^ := T + TPlanen;
   END
   ...
   ELSIF ( Aktionsfolge:DatenTransfer[1].Aktion = ´Essen´)
   DO
      Zustand^ := ´Essend´;
      TNext^ := T + 1.0 [h] *
              ( Aktionsfolge:DatenTransfer[1].Nahrungsmenge /
                Energie_Essen );
   END
   NaechsteAktion^ := FALSE;
END

# Regel zum Abschluss der Bearbeitung einer Aktion
# ------------------------------------------------
ON ^T >= TNext^
DO
   IF ( Zustand = ´Planend´ )
   DO
      DatAnCOG^: FROM Aktionsfolge GET DatenTransfer[1];
      SIGNAL TokenAnCognition;
   END
   ELSIF ( Zustand = ´Explorierend´ OR Zustand = ´Pruefend´ )
   DO
      DatAnSEN^: FROM Aktionsfolge GET DatenTransfer[1];
      SIGNAL TokenAnSensor;
   END
   ELSIF ( Zustand = ´Gehend´ OR Zustand = ´Befreiend´ OR
           Zustand = ´Essend´ )
   DO
      DatAnENV^: FROM Aktionsfolge GET DatenTransfer[1];
      NaechsteAktion^ := TRUE;
   END
END
```

Abbildung 10.32: Die Regeln zur Ausführung von Aktionen in der Komponente Actor

Die erste Regel (s. Abbildung 10.32) startet den mit der Ausführung einer Aktion einhergehenden, zeitverbrauchenden Vorgang und belegt die Zustandsvariable *Zustand* mit einem entsprechenden Wert (*Planend*, *Explorierend*, *Pruefend*,

Gehend, *Befreiend*, *Essend*). Diese Regel schaltet, wenn die Komponente *Actor* durch Bereitstellung der Synchronisationsmarke in der Eingangslocation *Token-VonBEH* zur Berechnung ihrer Zustandsübergänge aufgefordert wird, d. h. also durch die Komponente *Behaviour* eine Aktionsfolge zur Ausführung freigegeben worden ist, und die Komponente *Actor* nicht gerade mit der Ausführung einer Aktion beschäftigt ist (*NaechsteAktion = TRUE*). Die Informationen über die als nächste anstehende Aktion bezieht die Komponente *Actor* aus dem führenden Element der Eingangslocation *Aktionsfolge*. Die Zustandsvariable *TNext* bezeichnet den Zeitpunkt, zu dem die Ausführung der aktuellen Aktion beendet ist.

Die zweite Regel in der Komponente *Actor* sorgt dafür, daß nach Ablauf des zeitverbrauchenden Vorgangs, der die Ausführungsdauer einer Aktion modelliert, die Berechnung der resultierenden Zustandsübergänge angestoßen wird. Gemäß des Typs der ausgeführten Aktion wird dazu die Aktionsbeschreibung, die an erster Stelle der Aktionsfolge liegt, in die Location derjenigen Komponente verschoben, deren Zustand durch die Ausführung der Aktion modifiziert werden muß. Zudem wird in dieser Regel die Weitergabe der Synchronisationsmarke angeregt (*SIGNAL*-Anweisungen), falls es sich bei der betroffenen Komponente um eine Subkomponente des Agenten Adam handelt, sowie die Zustandsvariable *NaechsteAktion* auf den booleschen Wert *true* gesetzt, um anzuzeigen, daß die Komponente *Actor* wieder bereit ist, eine neue Aktion in Angriff zu nehmen.

10.2.3.9 Das dynamische Zusammenspiel der *PECS*-Komponenten

Nachdem in den vorangegangenen Abschnitten ein Überblick über die Struktur und das dynamische Verhalten der einzelnen Modellkomponenten gegeben wurde, soll in diesem Abschnitt nun ein Eindruck über das Zusammenspiel der Komponenten vermittelt werden. Im Mittelpunkt stehen also die Komponenten der Agentenarchitektur und deren Interaktion zur Steuerung der Aktivitäten des Agenten Adam.

Grundsätzlich arbeiten alle Komponenten der Agentenarchitektur nebenläufig. Eine Ausnahme bilden dabei Teile der Dynamik der Komponenten *Sensor*, *Perception*, *Cognition, Behaviour* und *Actor*, deren Aktivitäten mit Hilfe des Token-Konzepts aufeinander abgestimmt sind (s. Kapitel 10.2.3.1). Die Komponenten *Emotion* und *Physis*, die wesentliche Bestandteile des internen Zustands von Adam abbilden, können unabhängig vom Token-Konzept betrieben werden, da sie nicht für die Durchführung von Zwischenschritte zur Berechnung von Adams Verhaltensweisen verantwortlich sind. Sie liefern jedoch Zustandsvariablen, die in entscheidender Weise die Verhaltensauswahl des Agenten beeinflussen.

Die Interaktion der Komponenten soll nun exemplarisch anhand eines einfachen Szenarios aufgezeigt werden. Gegeben sei dazu die folgende Situation:

- Adam befindet sich auf Feld (2,2) seiner Umwelt.
- Adams Kompetenz ist sehr hoch.

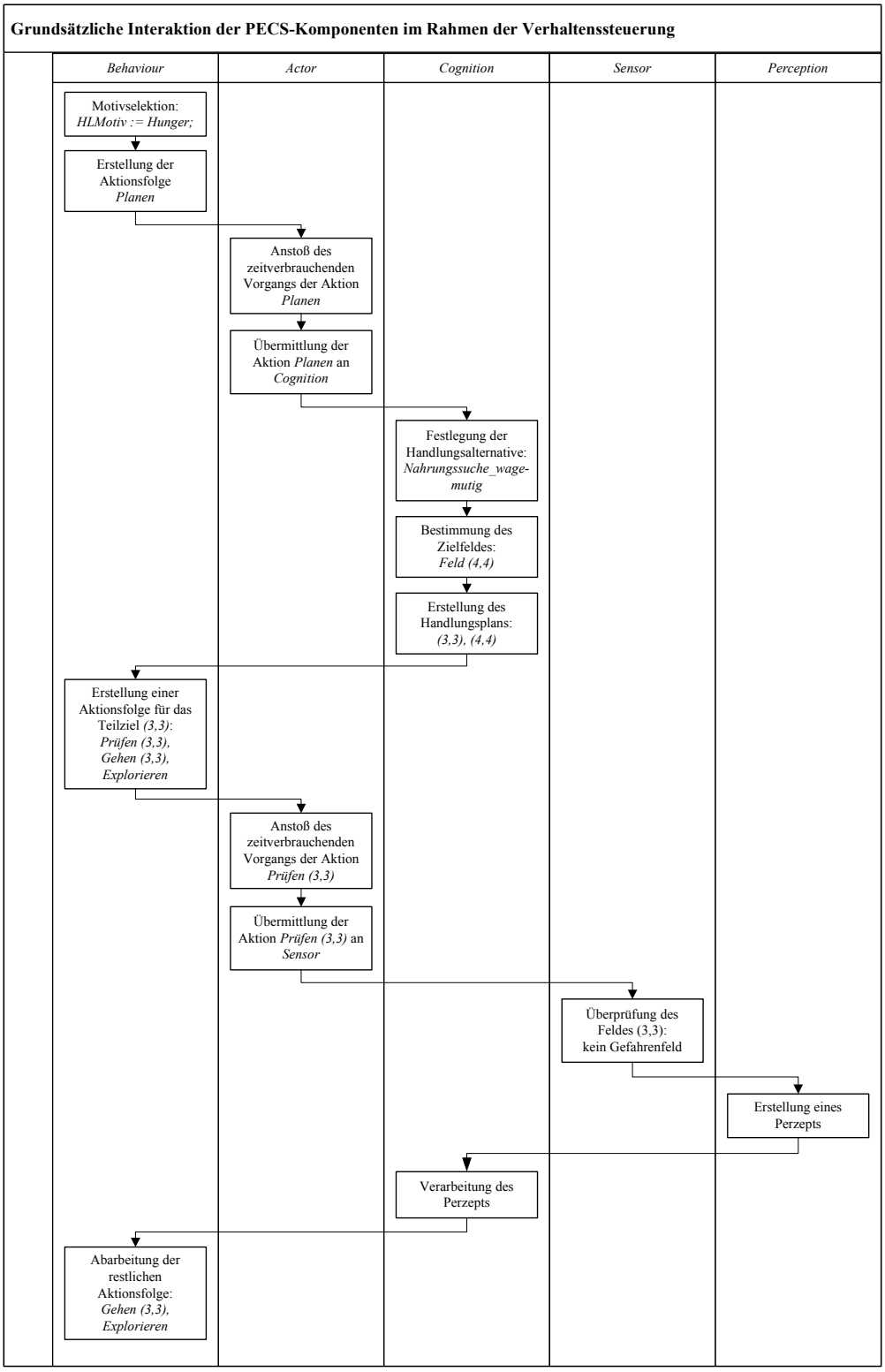

Abbildung 10.33: Die Interaktion der Komponenten der Agentenarchitektur

- Adam ist sehr ängstlich, baut allerdings im Rahmen der kontinuierlichen Eigendynamik in der Komponente *Emotion* seine Angst im Laufe der Zeit ab.
- Die Motivstärke für das Motiv *Hunger* hat zum betrachteten Zeitpunkt die Motivstärke des Motivs *Wissenserwerb* erreicht.
- Die Synchronisationsmarke befindet sich im Eingangsbereich der Komponente *Behaviour*.

Die Kette zur Berechnung der einzelnen Zwischenschritte im Rahmen von Adams Verhaltenssteuerung beginnt in der Komponente *Behaviour* mit der Auswahl des handlungsleitenden Motivs. Auf der Grundlage des Vergleichs der beiden Motivstärken wird das Motiv *Hunger* zum handlungsleitenden Motiv bestimmt. Aufgrund des Motivwechsel wird eine Neuplanung von Adams Verhalten nötig und deshalb die einelementige Aktionsfolge *Planen* erzeugt. Durch Übergabe der Synchronisationsmarke wird daraufhin die Komponente *Actor* aufgefordert, mit der Bearbeitung der ausgewählten Aktionsfolge zu beginnen.

Die Komponente *Actor* startet dazu einen zeitverbrauchenden Vorgang zur Modellierung der Ausführungsdauer der Aktion *Planen* und setzt ihren internen Zustand auf *Planend*. Gleichzeitig verbraucht Adam durch die Ausführung dieser Aktion Energie. Der Energieverbrauch läuft gleichzeitig mit dem zeitverbrauchenden Vorgang der Komponente *Actor* in der Komponente *Physis* ab. Nach Ablauf der Ausführungsdauer der Aktion *Planen* stellt die Komponente *Actor* sowohl die Synchronisationsmarke als auch die Aktionsbeschreibung der Komponente *Cognition* für den nächsten Berechnungsschritt zur Verfügung.

In der Komponente *Cognition* wird nun der deliberative Anteil von Adams Verhaltenssteuerung in Gang gesetzt. Zunächst einmal wird mit der Festlegung einer geeigneten Handlungsalternative begonnen. Da Adam zum betrachteten Zeitpunkt gemäß unserer Ausgangssituation über eine sehr hohe Kompetenz verfügen soll, wird er zur Befriedigung seines Nahrungsbedürfnisses die wagemutige Alternative der Nahrungssuche auswählen. Gemäß dieser Handlungsalternative sucht Adam in seiner kognitiven Landkarte nach einem geeigneten Zielfeld und stößt dabei etwa auf das Feld (4,4), das für das vorliegende Beispiel den in dieser Situation gültigen Auswahlkriterien entsprechen soll. Um zu diesem Zielfeld zu gelangen, muß Adam im Rahmen seiner Handlungsplanung einen Weg konstruieren, der ihn von seiner aktuellen Ausgangsposition am Feld (2,2) zum ausgewählten Feld (4,4) führt. Der generierte Weg soll über das Feld (3,3) führen, so daß sich insgesamt ein Handlungsplan mit dem Teilziel (3,3) und dem Zielfeld (4,4) ergibt. Damit ist die deliberative Dynamik in der Komponente *Cognition* abgeschlossen, so daß das Token an die Komponente *Behaviour* übergeben werden kann.

In der Komponente *Behaviour* wird als nächstes eine Aktionsfolge erstellt, die zur Erreichung des ersten Teilziels des Handlungsplans führen soll. Da Adam gemäß der Annahmen dieses Szenarios in der gegebenen Situation ängstlich ist und er keinerlei Informationen über das als nächstes zu betretende Feld zur Verfügung

hat, wählt er eine vorsichtige Vorgehensweise und entscheidet sich für die Aktionsfolge *Prüfen (3,3) - Gehen (3,3) - Explorieren*.

Die Komponente *Actor* erhält daraufhin sowohl die erzeugte Aktionsfolge, als auch die Synchronisationsmarke und kann somit mit der Ausführung der ersten Aktion *Prüfen (3,3)* beginnen. Dazu wird als erstes wiederum der zeitverbrauchende Vorgang eingeleitet und der interne Zustand auf *Prüfend* gesetzt. Nach Ablauf der entsprechenden Zeitdauer wird die Komponente *Sensor* mit der Ausführung der durch die Aktion *Prüfen (3,3)* vorgegebenen Transitionen beauftragt.

Die Komponente *Sensor* stellt gemäß der Aktion *Prüfen* fest, ob es sich bei dem zu betretenden Feld (3,3) um ein Gefahrenfeld handelt. Für das vorliegende Beispiel soll davon ausgegangen werden, daß sich das geprüfte Feld (3,3) als neutrales Feld herausstellt und daher gefahrlos betreten werden kann. Diese Information wird zusammen mit dem Token an die Komponente *Perception* weitergegeben.

Innerhalb der Komponente *Perception* werden in diesem Fall keine Transformationen an den eingegangenen Informationen vorgenommen, so daß diese in unveränderter Weise als Perzepte an die Komponente *Cognition* gehen.

Die Komponente *Cognition* stell anhand der übermittelten Informationen fest, daß es sich bei dem überprüften Feld um kein Gefahrenfeld handelt. Aus diesem Grund sind in dieser Komponente keine weiteren Aktualisierungen erforderlich (s. Abbildung 10.29), so daß das Token unmittelbar an die Komponente *Behaviour* weitergegeben werden kann.

In der Komponente *Behaviour* ist keine Änderung des handlungsleitenden Motivs möglich, da Adams Nahrungsbedürfnis noch nicht gestillt werden konnte. Aus diesem Grund fährt Adam mit der Abarbeitung der restlichen Aktionsfolge fort, die nunmehr aus den Aktionen *Gehen (3,3)* und *Explorieren* besteht.

Die Sequenz der eben geschilderten Zustandsübergänge in den einzelnen Komponenten ist in Abbildung 10.33 anhand eines Aktivitätsdiagramms (s. Balzert, 2000) zusammengefaßt. Die Aktivitäten der einzelnen Komponenten finden sich dabei in den jeweiligen „Swim Lanes“ wieder und sind mit Hilfe von Rechtecken dargestellt. Die Pfeile in Abbildung 10.33 bezeichnen Kontrollflüsse zwischen den Komponenten der Agentenarchitektur.

10.3 Simulationsexperimente

Das in den vorangegangenen Abschnitten geschilderte Simulationsmodell bietet vielfältige Ansatzpunkte für die Durchführung von Experimenten. Grundsätzlich werden bei Simulationsexperimenten ausgewählte Modellgrößen mit entsprechenden Initialwerten versehen, das Modell für einen gewissen Zeitraum betrieben und anschließend die Auswirkungen der Parametermodifikationen anhand der Zeitverläufe und Endwerte signifikanter Modellgrößen beobachtet.

Der folgende Abschnitt beschäftigt sich mit den wesentlichen Eigenschaften der graphischen Benutzungsoberfläche, die für dieses Modell entwickelt wurde, um ein komfortables Experimentieren mit dem Modell zu ermöglichen. Der Abschnitt 10.3.2 stellt ein ausgewähltes Simulationsexperiment im Überblick dar, das das Ziel verfolgt, die Auswirkungen von Adams Ängstlichkeit auf seine Überlebensfähigkeit in einer vorgegebenen Umgebung zu untersuchen.

10.3.1 Die graphische Benutzungsoberfläche

Für das vorliegende Modell wurde eine graphische Benutzungsoberfläche entwickelt, um die Durchführung von Modellexperimenten in komfortabler Weise und ohne Kenntnis der internen Modellstruktur zu ermöglichen. Die graphische Benutzungsoberfläche bietet dabei im wesentlichen die Möglichkeit, Initialbelegungen ausgewählter Modellgrößen vorzugeben, Simulationsläufe bis zu einem frei wählbaren Endezeitpunkt durchzuführen, sowie die Ergebnisse der Simulation als Animation darzustellen.

Für die Modellgrößen, die zu Beginn eines Experimentes mit individuellen Startwerten versehen werden können, wurde in der graphischen Benutzungsoberfläche eine Vorauswahl getroffen. Die veränderbaren Modellgrößen ermöglichen eine Anpassung wesentlicher interner Eigenschaften des Agenten Adam sowie seiner Umwelt.

In der Benutzungsoberfläche werden für den Agenten Adam die folgenden Parameter zur Modifikation angeboten:

- Gedächtnisleistung

  Die Gedächtnisleistung bezeichnet die Latenzzeit einer Information in Adams Gedächtnis. Je höher Adams Gedächtnisleistung ist, um so mehr Informationen über seine Umwelt wird er im Laufe der Zeit in seinem Gedächtnis ansammeln können.

- Ängstlichkeit

  Der Parameter Ängstlichkeit gibt vor, wie schnell Adam nach dem Betreten einer Gefahrenstelle seine Angst wieder abbauen kann. Dieser Parameter ist in zweifacher Hinsicht von Bedeutung. Adams Angst verhindert auf der einen Seite, daß er unvorsichtig in gefährliche Situationen gerät, die mit einem hohen Energieverbrauch verbunden sind. Andererseits führt sie aber auch selbst zu einem höheren Verbrauch an Energie, da stets vor dem Betreten eines unbekannten Feldes zusätzlich die Aktion *Prüfen* ausgeführt werden muß.

- Verbrauch an Energie beim Befreien

  Dieser Parameter führt im Modell zu einem veränderten Wert für den Energieverbrauch der Aktion *Befreien*. Auf diese Weise kann angegeben werden, wie bedrohlich es für Adam sein wird, in eine Falle zu geraten.

- Grundverbrauch an Energie

  Der Grundverbrauch an Energie geht als Faktor in die Berechnung des Energieverbrauchs aller Aktionen (mit Ausnahme der Aktion *Essen*) ein. Ein hoher Grundverbrauch führt dazu, daß Adam häufiger Nahrungsstellen aufsuchen muß, um seinen Energievorrat zu regenerieren.

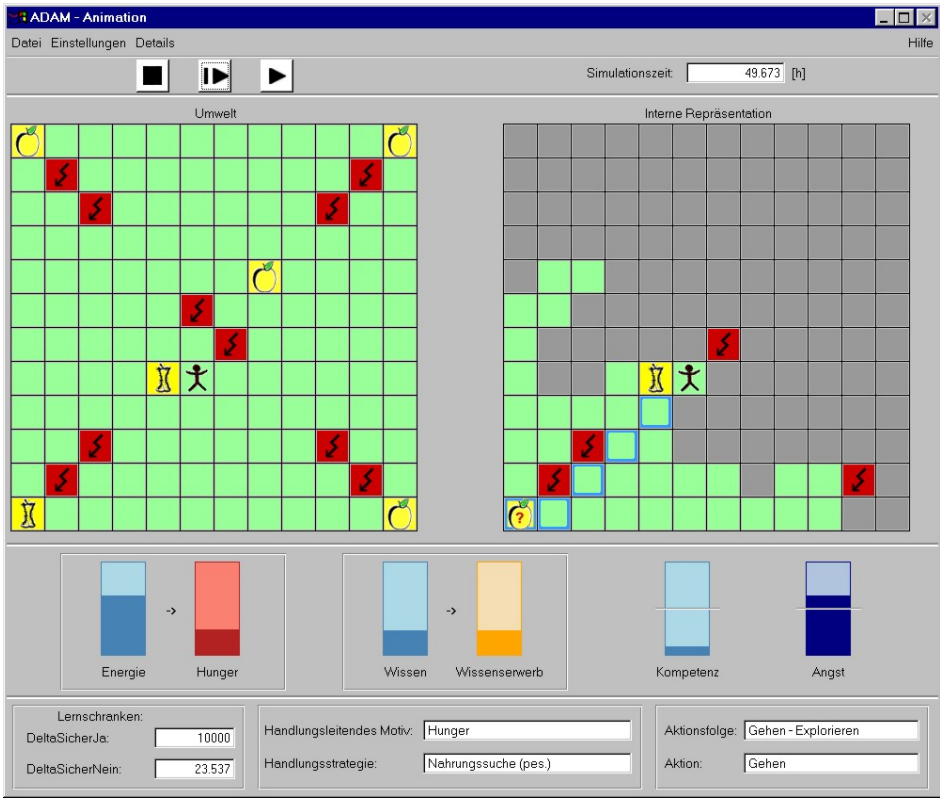

Abbildung 10.34: Die Benutzungsoberfläche der Animation für das Modell Adam (Schwarz, 2000)

Auch die Eigenschaften von Adams Umwelt können beeinflußt werden. Hierzu stehen die folgenden Möglichkeiten zur Verfügung:

- Regenerationsrate der Nahrungsstellen

  Die Regenerationsrate der Nahrungsstellen gibt an, wie schnell auf den einzelnen Nahrungsfeldern in Adams Umwelt Nahrung nachwächst, nachdem sie von Adam abgeerntet wurde. Eine höhere Regenerationsrate bewirkt eine höhere Verfügbarkeit von Nahrung und führt dadurch zu freundlicheren Umweltbedingungen für Adam.

- Editor zur Erstellung einer Umwelt

  Die graphische Benutzungsoberfläche umfaßt einen Editor, der die Erstellung neuer Welten für den Agenten Adam unterstützt. Er basiert auf einem zweidimensionalen Gitter, das die einzelnen Felder der Umwelt beinhaltet und ermöglicht eine beliebige Zuordnung von Feldarten (neutrales Feld, Nahrungs-

feld, Gefahrenfeld). Ebenso kann Adams Startposition in der Umwelt festgelegt werden.

Nachdem alle Parameter des Modells mit geeigneten Startwerten versehen sind, kann ein Simulationslauf bis zu einem frei wählbaren Endezeitpunkt durchgeführt werden. Die Ergebnisse der Simulation können anschließend mit Hilfe einer Animation visualisiert werden. Die Abbildung 10.34 zeigt einen Bildschirmauszug der Animationsoberfläche.

Die Animationsoberfläche ist im wesentlichen in vier horizontale Bereiche untergliedert. Am oberen Rand sind die Menüleiste sowie die Bedienelemente zur Steuerung der Animation angesiedelt. Unmittelbar darunter befinden sich zwei Matrizen, die den aktuellen Zustand der Umwelt und der mentalen Repräsentation der Umwelt im Agenten darstellen. Die dritte Schicht umfaßt die wesentlichen internen Zustände von Adam und die beiden Motivstärken für die Motive *Hunger* und *Wissenserwerb*. Informationen über die Fortschritte in Adams Lernprozeß sowie wesentliche Angaben zum aktuellen Zustand seiner Verhaltenssteuerung finden sich am unteren Rand des Animationsfensters.

10.3.2 Das Experiment *Angst und Überlebensfähigkeit*

Stellvertretend für eine Reihe unterschiedlicher Experimente, die auf der Grundlage des Modells *Adam* durchgeführt werden können, wird an dieser Stelle ein charakteristisches Experiment dargestellt, das sich mit der Untersuchung des Einflusses von Adams Ängstlichkeit auf seine Überlebensfähigkeit befaßt. Den Ausgangspunkt für dieses Experiment bildet die grundsätzliche, evolutionspsychologische Frage, inwieweit Emotionen, die sich im Laufe der Phylogenese einer Spezies entwickelt haben, die Überlebensfähigkeit dieser Spezies beeinflussen. Plutchik (Meyer, 1993) behauptet etwa, daß Emotionen der Bewältigung von Anpassungsproblemen dienlich sind und damit durchaus hilfreich für das Überleben einer Spezies sein können. Mit Hilfe dieses Simulationsexperimentes soll nun untersucht werden, welche Wirkung die Emotion *Angst* auf die Überlebensfähigkeit des Agenten Adam in einer vorgegebenen Umwelt hat.

10.3.2.1 Grundsätzliche Vorüberlegungen zum Experiment

Um die Überlebensfähigkeit des Agenten Adam untersuchen zu können, muß eine Ausgangssituation geschaffen werden, in der die Möglichkeit besteht, daß Adams Energiereserven vollständig aufgebraucht werden und er damit stirbt. Die Umwelt darf also nicht zu freundlich sein. Darüber hinaus müssen die Parameter, die den internen Zustand von Adam beeinflussen, so eingestellt werden, daß ein Überleben in der gegebenen Umwelt nicht zu einfach wird.

Grundsätzlich sind in Adams Umwelt sowohl Gefahrenstellen als auch Nahrungsfelder vorgesehen. Die Gefahrenstellen bewirken beim Betreten die Entste-

hung der Emotion *Angst*. Es ist zu erwarten, daß sich in einer Umwelt mit vielen Gefahrenstellen eine hohe Ängstlichkeit positiv auf Adams Überlebensfähigkeit auswirken wird, da sie dazu beiträgt, diese Gefahren zu vermeiden. In einer ungefährlichen Umwelt wird die Angst eher hemmend wirken. Nahrungsfelder dürfen in Adams Umwelt nicht zu üppig vorgesehen werden, da sonst keinerlei Notwendigkeit besteht, schonend mit Energiereserven umzugehen.

Angst führt bei Adam ab einer gewissen Intensität zu einer Vermeidung von Gefahrenstellen. Der Vorteil, der sich daraus ergibt, wird um so höher sein, je bedrohlicher das Betreten einer Gefahrenstelle für Adam sein wird. Der Grundverbrauch an Energie sowie der Energieverbrauch für die Aktion *Befreien* müssen daher so eingestellt werden, daß es sich lohnt, Gefahrenstellen aus dem Wege zu gehen. Andererseits ist auch für die Aktion *Prüfen* ein entsprechender Energie- und Zeitverbrauch vorzusehen, um dem zusätzlichen Aufwand des vorsichtigeren Vorgehens Rechnung zu tragen.

Für die Modellierung von Adams Ängstlichkeit stehen grundsätzlich drei alternative Ansätze zur Verfügung:

- Modifikation der Konstante *AngstSchranke*

  Die Konstante *AngstSchranke* legt fest, ab welchem Wert der Zustandsvariable *Angst* Adam unbekannte Felder vor dem Betreten prüft. Je niedriger diese Schranke gewählt wird, um so länger befindet sich Adam in einem ängstlichen Zustand und um so höher ist seine Ängstlichkeit.

- Modifikation der Konstante *AngstZunahme*

  Die Konstante *AngstZunahme* legt fest, welchen Zuwachs die Zustandsvariable *Angst* erfährt, nachdem Adam eine Gefahrenstelle betreten hat. Je größer dieser Konstante ist, um so größer wird der absolute Wert der Zustandsvariable *Angst* und um so länger ist Adam ängstlich.

- Modifikation der Konstante *AngstAbnahme*

  Die Konstante *AngstAbnahme* bezeichnet die Geschwindigkeit der Verringerung von Adams Angst. Je geringer die Abnahme von Adams Angst ist, um so länger bleibt Adam in einem ängstlichen Zustand und um so größer ist seine Ängstlichkeit.

Alle Alternativen bewirken, daß bei entsprechenden Parametereinstellungen Adam für einen kürzeren oder längeren Zeitraum in einem ängstlichen Zustand bleibt, und sind damit für das vorliegende Experiment als gleichwertig zu betrachten. Als veränderliche Größe soll exemplarisch etwa die Konstante *AngstAbnahme* angenommen werden, die in der Benutzungsoberfläche (s. Abschnitt 10.3.1) durch den Parameter *Ängstlichkeit* repräsentiert wird. Der Parameter *Ängstlichkeit* und die Konstante *AngstAbnahme* im Modell sind dabei indirekt proportional.

Eine interessante Fragestellung ergibt sich durch eine Betrachtung der Konsequenzen unterschiedlicher Einstellungen für Adams Ängstlichkeit. Wird Adam mit einer hohen Ängstlichkeit ausgestattet, so geht er meist sehr vorsichtig vor,

prüft unbekannte Felder, bevor er sie betritt und kann auf diese Weise Gefahrenstellen sehr gut vermeiden. Andererseits verbraucht er durch das Prüfen unbekannter Felder relativ viel Zeit und auch Energie, so daß er nur sehr langsam in seiner Umwelt vorwärts kommt. Wird Adams Ängstlichkeit sehr gering initialisiert, so agiert er weniger konservativ, verliert dadurch nicht viel Zeit und Energie, setzt sich allerdings der Gefahr aus, häufig auf Felder zu geraten, auf denen er Schaden nimmt. Die Vermutung liegt also nahe, daß Adam die größten Überlebenschancen haben wird, wenn seine Ängstlichkeit in einem mittleren Bereich angesiedelt ist. Dadurch wird Adam für eine gewisse Zeit nach negativen Erfahrungen vorsichtiger agieren und sich damit vor weiteren Gefahren schützen, aber dann wieder recht schnell seine Angst verlieren und zügig mit normalen Aktivitäten fortfahren.

10.3.2.2 Das Setup des Experiments

Im Rahmen dieses Modellexperiments wurden insgesamt 55 Simulationsläufe über eine Dauer von jeweils 500 Zeiteinheiten durchgeführt. Als Zielgröße wurde bei jedem Lauf die Lebensdauer des Agenten Adam ermittelt, wobei durch die Simulationsdauer eine maximale Lebensspanne von 500 Zeiteinheiten vorgegeben wird.

Parameter	**Wert in der Benutzungsoberfläche**	**Wert im Modell**
Ängstlichkeit	1, 10, 20, ... , 100	-2, -0.2, -0.1, ... , -0.02
Gedächtnisleistung	50	50
Grundverbrauch Energie	55	6
Energieverbrauch Befreien	40	-4
Regenerationsrate Nahrungsstellen	45	0.15

Abbildung 10.35: Initialwerte der Modellparameter in der Benutzungsoberfläche

Variiert wurden in den jeweiligen Simulationsläufen die Einstellungen für Adams Ängstlichkeit sowie der Zufallszahlengeneratoren. Adams Ängstlichkeit wurde in Zehnerschritten ausgehend von einer minimalen Ängstlichkeit von 1 sukzessive auf den maximalen Wert 100 erhöht. Zu jedem Wert für Adams Ängstlichkeit wurden jeweils fünf Simulationsläufe mit unterschiedlichen Einstellungen für die verwendeten Zufallszahlengeneratoren durchgeführt. Zufallszahlen spielen bei Adams Wegeplanung eine Rolle, wenn mehrere Felder mit gleicher Präferenz existieren.

Die Modellgrößen, die in der Benutzungsoberfläche zur Parametrisierung des Modells angeboten werden, erhalten für das vorliegende Experiment Initialwerte gemäß der Abbildung 10.35. Die Umwelt des Agenten Adam wurde für dieses Experiment mit sieben Nahrungsfeldern und sieben Gefahrenstellen ausgestattet. Die geringe Zahl an Nahrungsfeldern zwingt den Agenten dazu, seine Umwelt zu erkunden, um nicht zu verhungern. Adam befindet sich zu Beginn eines Simulationslaufes auf einem Nahrungsfeld. Dadurch soll erreicht werden, daß Adam zumindest ein Nahrungsfeld in seiner Umwelt von Beginn an kennt und nicht frühzeitig verhungern muß, nur weil er nicht zufällig bei der Exploration seiner Umwelt auf ein Nahrungsfeld gestoßen ist. Die Abbildung 10.36 zeigt die für alle Simulationsläufe dieses Experiments gültige Konfiguration der Umwelt.

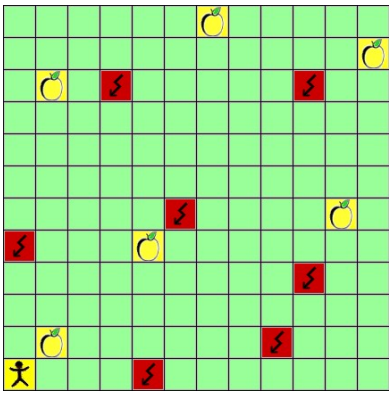

Abbildung 10.36: Die Konfiguration der Umwelt für das Experiment Angst und Überlebensfähigkeit

10.3.2.3 Ergebnisse des Experiments

Das auf der Grundlage der oben geschilderten Annahmen durchgeführte Simulationsexperiment bestätigt die Vermutung, daß Adam die größten Überlebenschancen besitzt, wenn seine Ängstlichkeit in einem mittleren Bereich angesiedelt ist. In diesem Fall existiert ein brauchbarer Kompromiß zwischen vorsichtigem Abtasten seiner Umwelt und ungestümem Tatendrang.

Die Abbildung 10.37 zeigt den durch das Experiment ermittelten Zusammenhang zwischen Adams Ängstlichkeit und seiner Lebensdauer in der vorgegebenen Umwelt. Die kurze Lebensdauer für kleine Werte der Ängstlichkeit rührt daher, daß Adam aufgrund der fehlenden Vorsicht sehr häufig auf Gefahrenfelder gerät und damit schnell an Energie verliert. Ebenso ist Adams Lebensdauer auch für hohe Werte seiner Ängstlichkeit relativ kurz. Eine Erklärung für diesen Sachverhalt ergibt sich dadurch, daß Adam aufgrund der umfassenden Prüfung unbekannter Felder nur sehr langsam in seiner Umwelt vorwärts kommt und dadurch nur sehr wenig Informationen über seine Umwelt und insbesondere Nahrungsstellen aufsammeln kann. Nachdem Adam alle bekannten Nahrungsfelder abgeerntet hat,

verhungert er relativ bald, da er nur einen kleinen Teil seiner Umwelt kennt und bei ihrer Erkundung nur sehr geringe Fortschritte macht.

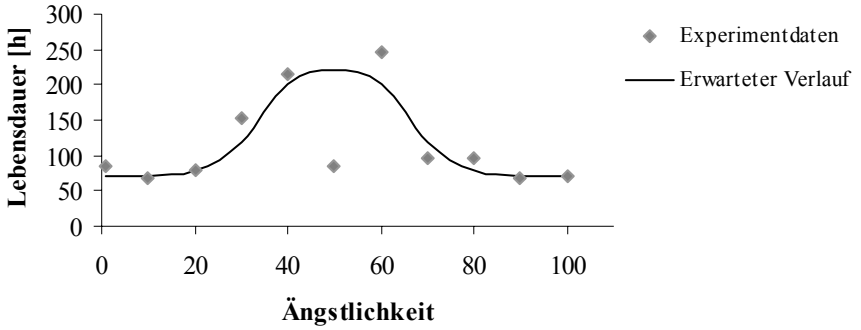

Abbildung 10.37: Adams Lebensdauer in Abhängigkeit seiner Ängstlichkeit

Die Resultate, die dieses Simulationsexperiment liefert, gehen konform mit Plutchiks evolutionspsychologischer Annahme, daß Emotionen förderlich für das Überleben einer Spezies sind. Mit steigender Ängstlichkeit nimmt die Überlebensfähigkeit des Agenten Adam zu. Allerdings wirkt sich Adams Ängstlichkeit auch negativ auf seine Lebensdauer aus, wenn sie zu dominant für seine Verhaltensweisen wird.

10.4 Zusammenfassung und Fazit

Die Fallstudie *Modellierung menschlicher Handlungsregulationsmechanismen* exemplifiziert den Einsatz agentenbasierter Methoden zur Unterstützung der Modellbildung und Simulation in der Psychologie. Das Ziel dieser Fallstudie besteht im wesentlichen darin zu zeigen, daß psychische Zustände und Prozesse, sowie deren Wechselwirkungen in strukturierter und übersichtlicher Weise auf der Grundlage des Referenzmodells *PECS* modelliert werden können. Um dieses Ziel zu erreichen, besteht keine Notwendigkeit, den Protagonisten des Modells, in unserem Fall also den Agenten Adam, besonders reichhaltig mit vielfältigen Eigenschaften und Fähigkeiten auszustatten, wie es beispielsweise Dörner in seinem Ψ-Ansatz (Dörner, 1999) tut. Adam soll keinen wirklichen Menschen nachbilden. Vielmehr beschränkt sich die Ausstattung von Adam auf ein Minimalmaß an Eigenschaften und Fähigkeiten, die unabdingbar erscheinen, um das grundsätzliche Zusammenspiel physischer, emotionaler und kognitiver Prozesse im Rahmen der menschlichen Verhaltenssteuerung untersuchen zu können.

Das Modell Adam zeigt einerseits, daß sich durch den Einsatz formaler Methoden der Modellbildung und Simulation die Präzision psychologischer Theorien deutlich erhöhen läßt. Zum anderen bringt diese Fallstudie eine Reihe von Vortei-

len zum Ausdruck, die sich durch den Einsatz von Referenzmodellen für die Konzeption von Simulationsmodellen ergeben.

Der systemtheoretische Ansatz des Referenzmodells *PECS* ermöglicht eine adäquate Beschreibung der internen Zustände und Zustandsübergänge des Agenten Adam. Ebenso können auf der Grundlage von kausalen Abhängigkeiten die Einflüsse des internen Zustands auf Adams Verhaltensauswahl in einfacher Weise modelliert werden.

Aufgrund des komponentenorientierten Ansatzes läßt sich das Gesamtmodell in eine Menge interagierender Komponenten aufteilen und dadurch ein hohes Maß an Übersichtlichkeit, sowie eine gute Erweiterbarkeit des Modells erreichen. Dieser Gesichtspunkt spielt in diesem Zusammenhang eine ganz wesentliche Rolle, da, wie die vorliegende Fallstudie zeigt, psychologische Modelle trotz einfacher Grundannahmen oftmals eine sehr komplexe Struktur aufweisen. An dieser Stelle bietet das Referenzmodell *PECS* eine ausgezeichnete Grundlage für die Konzeption der Struktur sowie der internen Dynamik agentenbasierter Modelle in der Psychologie.

11 EVALUATION BESTEHENDER AGENTEN-ARCHITEKTUREN UND MODELLKONZEPTE

In den vorangegangenen Abschnitten wurde die Grundkonzeption des Referenzmodells *PECS* beschrieben, das als domänenunabhängiges Konstruktionsschema den Entwurf agentenbasierter Simulationsmodelle, in denen menschliches Handeln, Entscheiden und Verhalten abgebildet werden, unterstützt. Ebenso wurde im Rahmen von vier Fallstudien aus unterschiedlichen Anwendungsgebieten der Einsatz des Referenzmodells in der Praxis demonstriert.

Der nun folgende Abschnitt beschäftigt sich mit der grundsätzlichen Frage, inwiefern bestehende Architekturansätze aus dem Bereich der Künstlichen Intelligenz und angrenzenden Gebieten ebenso in der Lage wären, die Anforderungen zu erfüllen, die bei einer domänen- und theorieunabhängigen Modellierung menschlichen Handelns, Entscheidens und Verhaltens im Vordergrund stehen. Zu diesem Zweck wird zu Beginn dieses Kapitels zunächst einmal diskutiert, in welcher Weise eine Bewertung von Agentenarchitekturen stattfinden kann und welche Kriterien im Umfeld der vorliegenden Arbeit dafür herangezogen werden müssen. Weiterhin werden einige ausgewählte und als besonders relevant erachtete Agentenarchitekturen im Überblick dargestellt und anhand der zugrunde gelegten Kriterien diskutiert. Den Abschluss dieses Abschnitts bildet eine Zusammenfassung der wesentlichen Ergebnisse aus der Betrachtung der bestehenden Architekturen.

11.1 Grundsätzliches Vorgehen

Der folgende Abschnitt beschäftigt sich mit der grundlegenden Frage, in welcher Weise eine Bewertung von Agentenarchitekturen bzw. agentenbasierten Modellkonzepten vorgenommen werden kann. Ausgehend vom aktuellen Stand der Forschung werden die Vorgehensweise sowie die Kriterien erläutert, auf deren Grundlage eine Diskussion der bestehenden Architekturansätze stattfinden soll.

11.1.1 Evaluation von Agentenarchitekturen – Stand der Forschung

Die integrative Betrachtung verschiedenster Fähigkeiten, die bei Lebewesen und insbesondere beim Menschen zu beobachten sind, und die damit verbundene Erforschung von Agentenarchitekturen gewinnt im Bereich der Künstlichen Intelligenz immer mehr an Bedeutung. Diese Entwicklung ist nicht zuletzt anhand der großen Anzahl an Veröffentlichungen erkennbar, die sich mit dem Entwurf und der Implementierung von Agentenarchitekturen auseinandersetzen. Die Konstruktion derartiger Architekturen wird dabei durch eine Vielzahl von Faktoren, wie etwa die Zielsetzung des jeweiligen Forschungsansatzes oder die systemtechnischen Voraussetzungen, unter denen die Architektur zum Einsatz kommen soll, beeinflusst. Entsprechend groß sind die Unterschiede zwischen den vorhandenen Architekturen und entsprechend schwierig stellt sich auch eine Evaluation vorhandener Agentenarchitekturen dar.

Zusätzlich erschwert wird die Evaluation von Agentenarchitekturen dadurch, dass hierfür zumindest bislang keine allgemein anerkannten Kriterien oder Vorgehensweisen existieren. In der jüngsten Literatur finden sich zwar erste Ansätze, die sich darum bemühen, Forschungsaktivitäten im Bereich der Agentenarchitekturen zusammenzuführen. Hierzu zählt etwa die Arbeit von Scheutz und Sloman (2002), die das Ziel verfolgt, ein methodologisches Framework zu schaffen, in dem unterschiedliche Kategorien von Agentenarchitekturen abgebildet und miteinander verglichen werden können. Diese Aktivitäten befinden sich allerdings noch in einer sehr frühen Phase. Ebenso existieren im Bereich der Software-Agenten erste Arbeiten, die sich mit der Evaluation von Agentenarchitekturen befassen. Die erste Veröffentlichung in diesem Zusammenhang stammt aus dem Jahre 1998 und wurde von Jörg P. Müller verfasst. Müller unternimmt dabei den Versuch, Anwender verschiedener Domänen bei der Auswahl geeigneter Agentenarchitekturen zu unterstützen, indem er sowohl Anwendungsdomänen als auch Agentenarchitekturen klassifiziert und anschließend Faustregeln formuliert, die festhalten, welche Architekturansätze in welchen Situationen zu bevorzugen sind. Spätere Arbeiten (z.B. Wallace & Laird, 1999) geben bereits Kriterien für die Bewertung von Agentenarchitekturen vor, fokussieren dabei aber im wesentlichen technische Aspekte wie etwa die Performance agentenbasierter Programme.

Zusammenfassend muss also festgestellt werden, dass bislang weder im Bereich der künstlichen Intelligenz noch in angrenzenden Gebieten geeignete Methoden existieren, die im Rahmen dieser Arbeit als Vorlage für die Evaluation bestehender Agentenarchitekturen eingesetzt werden können. Im nun folgenden Abschnitt wird daher zunächst kurz die Vorgehensweise vorgestellt, die für die Evaluation der Architekturen Verwendung finden soll. Ebenso werden die wesentlichen Kriterien dargestellt, die hinsichtlich der Zielsetzung der vorliegenden Arbeit und für die Konstruktion einer domänen- und theorieunabhängigen Referenzarchitektur als unerlässlich erachtet werden.

11.1.2 Vorgehen und Kriterien zur Evaluation der Agentenarchitekturen und Modellkonzepte

Die nachfolgenden Abschnitte befassen sich mit einer Reihe von aktuellen Agentenarchitekturen und Modellierungsansätzen, die aus dem Bereich der Künstlichen Intelligenz sowie anderen angrenzenden Gebieten entnommen sind. Aufgrund der Vielzahl an existierenden Architekturen ist an dieser Stelle eine erschöpfende Analyse aller vorhandenen Architekturen unmöglich. Vielmehr bezieht sich die Betrachtung auf einige ausgewählte Ansätze, die sich entweder über die Jahre hinweg in der Künstlichen Intelligenz als grundlegend erwiesen und damit eine sehr weite Verbreitung gefunden haben, oder hinsichtlich der Zielsetzung der vorliegenden Arbeit als besonders relevant erachtet werden. Zur zweiten Kategorie gehören im wesentlichen Modellierungsansätze, die im Grenzgebiet zwischen Künstlicher Intelligenz, Kognitionswissenschaften und der Emotionsforschung angesiedelt sind.

Für den Entwurf agentenbasierter Referenzmodelle, die eine anwendungs- und theorieunabhängige Abbildung menschlicher Akteure in Simulationsmodellen ermöglichen sollen, wurden in den Kapiteln 3 und 4 dieser Arbeit eine Reihe von grundlegenden Anforderungen formuliert. Diese Anforderungen beziehen sich sowohl auf strukturelle als auch funktionale Aspekte der Modellierung und insbesondere der Agentenarchitektur. Die funktionalen Anforderungen sind dabei sehr abstrahiert dargestellt und beziehen sich auf unterschiedliche Kategorien von Funktionalitäten, die grundsätzlich bereitgestellt werden müssen, um menschliche Akteure flexibel in unterschiedlichen Anwendungsszenarien in Simulationsmodellen zu berücksichtigen. Diese Anforderungen sollen nun als grundlegende Evaluationskriterien für die Agentenarchitekturen und Modellierungsansätze dienen, die in den folgenden Abschnitten diskutiert werden. Dabei steht die Frage im Vordergrund, inwiefern der jeweils betrachtete Ansatz in der Lage ist, diese Kriterien zu erfüllen. Die Evaluation bezieht sich also auf eine kategorische Betrachtung struktureller und funktionaler Eigenschaften der Architekturen und Modellierungsansätze im Vergleich zu den spezifizierten Anforderungen.

Welche Kriterien nun im einzelnen verwendet werden, zeigt der nachfolgende Kriterienkatalog. Die Kriterien decken sich, wie eingangs bereits erwähnt, mit den wesentlichen Anforderungen aus Kapitel 3 und 4. Sie werden aus diesem Grund in Abbildung 11.1 nur noch einmal aufgelistet und nicht mehr ausführlich diskutiert.

Das Hauptkriterium 1 der Domänen- und Theorieunabhängigkeit (s. Abbildung 11.1) soll bewirken, dass eine Modellarchitektur möglichst unabhängig von Anforderungen spezifischer Anwendungsgebiete und Theorien spezifiziert wird, um ein breites Anwendungsspektrum zu adressieren. Das Kriterium 2 zielt darauf ab, dass für eine Architektur ein modularer Aufbau, beispielsweise auf der Grundlage eines komponentenorientierten und hierarchischen Ansatzes, gewählt wird, um eine gute Adaptierbarkeit der Architektur, sowie eine langfristige Wart- und Erweiterbarkeit sicherzustellen. Das Hauptkriterium 3 befasst sich mit dem Auf-

bau und den Fähigkeiten der Agenten. Sensorische und aktuatorische Fähigkeiten gehören dabei zur obligatorischen Ausstattung von Agenten, um mit ihrer Umwelt in Kontakt zu treten und diese auch zu verändern. Komplexe innere Zustände und Prozesse zeichnen Agenten allerdings in der Regel erst dann aus, wenn sie auch komplexere Aufgaben zu erfüllen haben oder dazu dienen, Lebewesen nachzubilden. Um menschliches Handeln und Verhalten zu modellieren, wird in Anlehnung an moderne Handlungstheorien vorausgesetzt, dass ein Agent über physische, kognitive, emotionale und soziale Eigenschaften und zugehörige Prozesse verfügen muss. Ebenso komplex stellt sich die Verhaltenssteuerung der Agenten dar, wenn Menschen in Simulationsmodellen abgebildet werden sollen. Je nach Zielsetzung und Aufgabenstellung der Modells reicht das Spektrum möglicher Verhaltensweisen, die betrachtet werden müssen, von einfachen Reaktionen über deliberatives Verhalten mit Planungsprozessen bis hin zu reflektiven Aktivitäten, in denen ein Agent Schlüsse auf der Grundlage seines Selbstmodells zieht. Nicht zuletzt sollten die Modellierungsansätze auch Grundmechanismen für die Interaktion bzw. Kommunikation der Agenten vorsehen, sowie Ansätze für die Modellierung der Umwelt und die Interaktion der Agenten mit ihrer Umwelt enthalten.

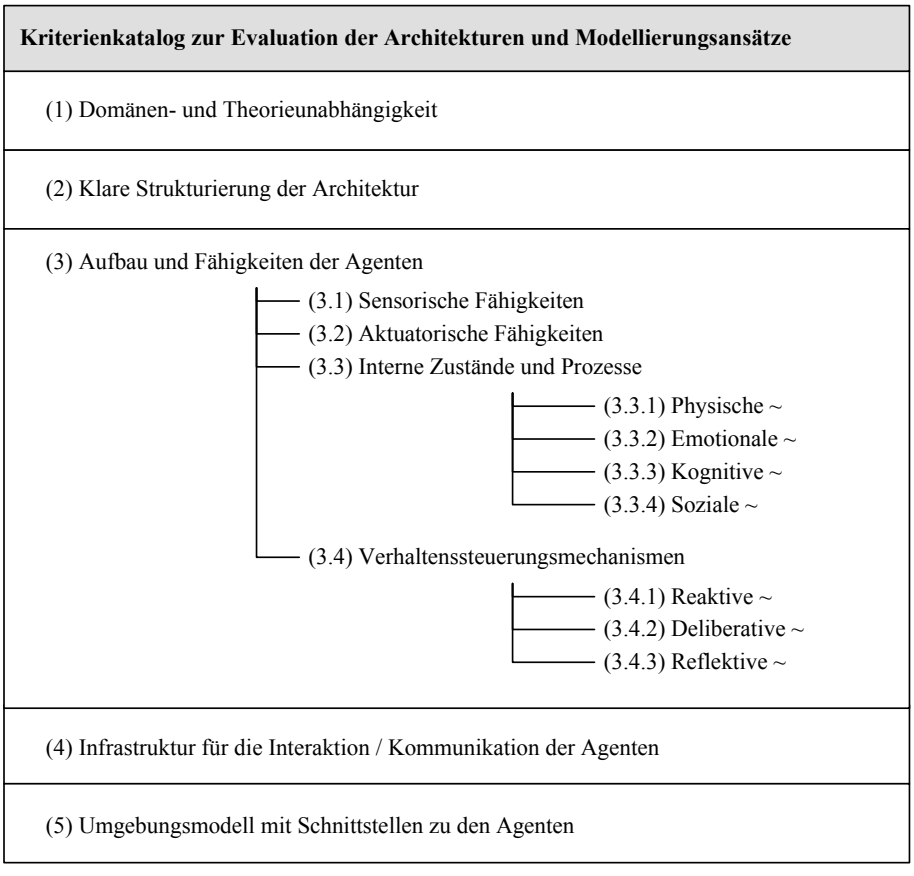

Abbildung 11.1: Übersicht über die Evaluationskriterien der Agentenarchitekturen und Modellierungsansätze

11.2 Ausgewählte Architekturen und Modellkonzepte im Überblick

In den folgenden Abschnitten werden einige ausgewählte Architekturen und Modellkonzepte dargestellt und kritisch diskutiert. Die Architekturansätze entstammen zum einen dem Bereich der Künstlichen Intelligenz. Hierbei handelt es sich um die „klassischen" Architekturansätze deliberativer, reaktiver und hybrider Agenten sowie die BDI-Architekturen. Zum anderen werden eine Reihe von Architekturen aus dem Grenzgebiet zwischen Künstlicher Intelligenz, den Kognitionswissenschaften, Handlungstheorien und der Emotionspsychologie vorgestellt.

11.2.1 Deliberative Architekturen

Deliberative Agentenarchitekturen entspringen klassischen Problemlösungs- und Planungsansätzen der künstlichen Intelligenz, sowie Modellen der menschlichen Kognition, wie etwa *SOAR* (Laird, Newell & Rosenbloom, 1987) oder *ACT-R* (Anderson, 2003). In der Regel beschäftigen sich deliberative Agenten mit der Lösung ganz bestimmter Probleme oder spezifischen Planungsaufgaben. Sie sind zu diesem Zweck mit einer mentalen Repräsentation ausgestattet, die zumindest für den Planungsvorgang relevante Sachverhalte, wesentliche Ziele, sowie die möglichen Aktionen des Agenten einschließlich ihrer Vorbedingungen und Konsequenzen enthält. Im Rahmen der Verhaltenssteuerung führt ein deliberativer Agent zunächst einmal einen Planungsvorgang durch, der zu einer Lösung der vorliegenden Problemsituation führen soll. Der Planungsprozess basiert dabei auf sukzessiven Modifikationen der mentalen Repräsentation des Agenten. Erst dann, wenn ein fertiger Plan vorliegt, beginnt ein deliberativer Agent damit, diesen Plan in der Realität umzusetzen.

Da auch menschliches Verhalten bis zu einem gewissen Grad durch Planungsaktivitäten und Deliberation geprägt ist, erscheinen deliberative Architekturansätze zunächst als geeigneter Startpunkt, um menschliche Akteure in Simulationsmodellen abzubilden. Andererseits zeigen sowohl die Alltagserfahrung als auch die psychologische Literatur, dass sehr oft (wenn nicht sogar zum überwiegenden Teil) Automatismen und Reaktionen das Verhalten von Menschen bestimmen. Dieser Sachverhalt ist insbesondere in häufig wiederkehrenden Situationen oder auch in Situationen mit hoher Dynamik zu beobachten, in denen entweder bereits vorgefertigte Verhaltensweisen existieren, die nur noch mechanisch ausgeführt werden müssen, oder Planungsaktivitäten aufgrund von schnellen Veränderungen der vorliegenden Situation nicht zum Erfolg führen würden. In dieser Hinsicht erscheinen rein deliberativ ausgerichtete Agentenarchitekturen nicht reichhaltig genug, um menschliches Verhalten in allgemeiner und flexibler Weise abzubilden.

11.2.2 Reaktive Architekturen

Reaktive Agentenarchitekturen sind ab der Mitte der achtziger Jahre als Reaktion auf klassische Planungsparadigmen der künstlichen Intelligenz entstanden. Sie basieren auf einer behavioristischen Sichtweise der Verhaltenssteuerung und abstrahieren demzufolge von jeglichen internen Zuständen, die das Verhalten und Handeln des modellierten Systems beeinflussen könnten. Klassische reaktive Architekturen koppeln sensorische Inputs eines Agenten direkt an seine Verhaltensweisen. Ein reaktiver Agent besteht in der Regel aus mehreren Modulen, die jeweils eine direkte Verbindung zwischen Input und Output des Agenten herstellen und für die Generierung spezifischer Verhaltensweisen zuständig sind. Typische Beispiele für derartige reaktive Agentenarchitekturen sind etwa die *Subsumption Architecture* von Brooks (1991), die aus einer Menge interagierender, verhaltenserzeugender Module besteht und vollständig auf symbolische Repräsentationen verzichtet, sowie die Architektur *Pengi* nach Agre und Chapman (1987).

Reaktive Architekturen werden in der Regel für die Konstruktion technischer Systeme oder Agenten eingesetzt, die robustes Verhalten und schnelle Reaktionen in dynamischen Umgebungen zeigen müssen. Der Verzicht auf interne Zustände, die zu Modifikationen des Systemverhaltens führen können, kollidiert allerdings massiv mit modernen Handlungstheorien, die von der grundlegenden Annahme ausgehen, dass das Verhalten höherer Organismen und insbesondere Menschen durch innere Kräfte und Prozesse geprägt ist. Ebenso verhindert ein Ausschluss interner Zustände auch die Fähigkeit eines Systems, aus Erfahrungen zu lernen. Lernen stellt allerdings eine grundlegende Fähigkeit des Menschen dar, auf die auch in Simulationsmodellen sehr häufig Bezug genommen wird. Nicht zuletzt dienen symbolische Repräsentationen als Teile des internen Zustands eines Agenten auch als Grundlage für Denk- und Planungsprozesse. Zusammenfassend lässt sich also sagen, dass reaktive Architekturen, ebenso wie deliberative, eine zu sehr eingeschränkte Sichtweise auf die Verhaltenssteuerung vermitteln, als dass sie in adäquater Weise für die Modellierung menschlichen Handelns, Entscheidens und Verhaltens eingesetzt werden könnten.

11.2.3 Hybrid-Architekturen

Hybrid-Architekturen kombinieren reaktive und deliberative Mechanismen zur Verhaltenssteuerung und versuchen auf diese Weise, die jeweiligen Vorteile beider Architekturklassen miteinander zu verbinden. Reaktives Verhalten wird genutzt, um routinemäßige Vorgänge oder robustes Verhalten in dynamischen Umgebungen zu modellieren. Deliberatives Verhalten wird hingegen gewählt, wenn keine automatischen Verhaltensweisen existieren oder die vorliegende Situation es erfordert bzw. zulässt, dass eine Planung durchgeführt wird. Das Hauptproblem in Hybridarchitekturen besteht in der Koordination der reaktiven und deliberativen Verhaltensweisen. Deliberation muss etwa zu jeder Zeit unterbrechbar sein

und entsprechend zu einem späteren Zeitpunkt an der ursprünglichen Stelle fortgesetzt werden können, wenn eine Veränderung der Umgebung oder des Zielzustandes des Agenten einen reaktiven Eingriff erfordern. Ebenso besteht die Notwendigkeit, eine Reihe von Zielen, die miteinander konkurrieren oder sich auch ergänzen, in Einklang zu bringen. Die eben geschilderten Probleme sind im Rahmen der Künstlichen Intelligenz bis heute nicht vollständig und allgemeingültig gelöst. Es existieren jedoch einige Ansätze für hybride Architekturen, die spezifische Vorschläge für die Integration von reaktiven und deliberativen Verhaltensweisen machen. Hierzu zählen etwa der *RAP*-Ansatz von Firby (1989), der auf der Idee einer reaktiv angestoßenen Planung und konkurrierenden Zielen basiert, oder auch die *Touring Machine* von Ferguson (1992), die unterschiedliche Verhaltensmodule nach einem Schichtenprinzip anordnet und mit Hilfe eines Kontrollalgorithmus koordiniert.

Hybridarchitekturen haben sich in technischen Anwendungsfeldern als durchaus leistungsfähig erwiesen. Sie fokussieren im wesentlichen reaktive Steuerungsmechanismen und deliberative Planungsansätze, so dass der Schwerpunkt dieser Architekturen sehr deutlich auf Aspekten liegt, die dem kognitiven Bereich zuzuordnen sind. Eine Betrachtung weiterer, interner Kräfte und Eigenschaften, wie etwa Emotionen, oder auch sozialer Einflüsse, die für die Handlungsregulation beim Menschen als wesentlich erachtet werden, unterbleibt bei diesen Ansätzen aufgrund ihrer technischen Ausrichtung. Hybridarchitekturen erscheinen daher für die Modellierung menschlichen Handelns, Entscheidens und Verhaltens bereits wesentlich besser geeignet als die beiden zugrunde liegenden Urformen, lassen aber dennoch wesentliche Einflussbereich außer acht.

11.2.4 BDI-Architekturen

BDI-Architekturen (Rao & Georgeff, 1991) gehören in technischen Einsatzfeldern seit Mitte der 90er Jahre zu den am weitesten verbreiteten Ansätzen zur Strukturierung autonomer Agenten. Aufgrund ihres enormen Einflusses in diesem Bereich sowie der wohldefinierten theoretischen Grundlagen, werden sie gegenwärtig in verstärktem Maße auch zur agentenbasierten Modellierung realer Systeme und insbesondere auch von menschlichem Verhalten eingesetzt.

Bei der Ausgestaltung der Architektur von BDI-Agenten stehen rationale und kognitive Aspekte im Vordergrund. BDI-Agenten sind mit einem internen, mentalen Zustand ausgestattet, der mit Hilfe von sog. *Beliefs*, *Goals*, *Plans* und *Intentions* beschrieben wird. *Beliefs* repräsentieren die Annahmen des Agenten über seine Umwelt und sich selbst. *Goals* bezeichnen Zielzustände, die ein BDI-Agent durch Ausführung geplanter Aktionen zu erreichen versucht. Sie werden in der Regel so lange aufrecht erhalten, bis das Ziel erreicht werden kann oder im Laufe der Zeit festgestellt wird, dass das gegebene Ziel unerreichbar ist. Pläne (*Plans*) dienen der Strukturierung des Handelns und sind den Zielen des Agenten untergeordnet. Zur Menge der *Intentions* gehören schließlich alle diejenigen Pläne, die in

der betrachteten Situation zur Ausführung freigegeben wurden und entsprechend das Verhalten eines Agenten steuern.

Der Nutzen von BDI-Architekturen als Referenzmodell ist unstrittig. Aufgrund ihrer klaren Strukturierung und theoretischen Fundierung eignen sich BDI-Agenten ausgezeichnet als Referenzmodelle für die Konstruktion autonomer, technischer Agenten. Das hohe Abstraktionsniveau, auf dem die ausgewählten Grundkonzepte beschrieben werden, lässt ebenso ohne größere Schwierigkeiten eine Übertragung der BDI-Konzepte auf vielfältige Anwendungsdomänen zu.

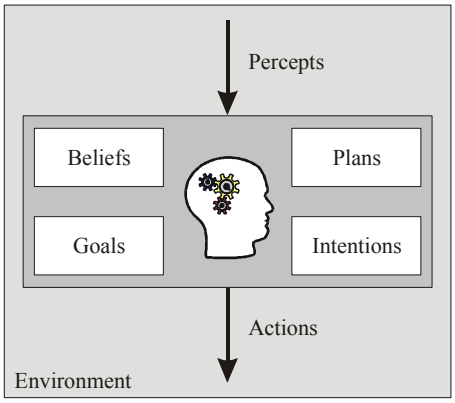

Abbildung 11.2: Die Grundstruktur von BDI-Agenten

Für eine umfassende Modellierung menschlicher Eigenschaften und menschlichen Verhaltens stellt sich dieser Ansatz mit seiner Beschränkung auf rein kognitive Aspekte jedoch als zu restriktiv heraus. Eine integrative Betrachtung interner physischer, emotionaler und sozialer Eigenschaften im Zusammenspiel mit kognitiven Fähigkeiten und Prozessen ist im Rahmen von BDI-Architekturen nicht vorgesehen. Damit bleiben auch in dieser Klasse von Architekturen elementare Einflüsse auf das menschliche Verhalten unberücksichtigt.

11.2.5 Die Architektur Will

Die Architektur *Will* (Moffat & Frjida, 1995) wurde Mitte der neunziger Jahre von Dave Moffat und Nico H. Frijda an der Universität Amsterdam entwickelt. Sie stellt einen sehr einfachen und allgemein gehaltenen Ansatz für eine Agentenarchitektur dar, die klassische KI-Technologien wie Planung und Reaktivität vereint, aber auch neue Konzepte einführt.

Will besteht aus einer Menge parallel agierender Module (s. Abbildung 11.3). Das Modul *Perceiver* modelliert die Wahrnehmungsfähigkeiten eines Agenten, während die *Executor*-Komponente für die Ausführung von Aktionen zuständig ist. *Will* geht von der Annahme aus, dass ein Agent über eine Menge von *Concerns* verfügt, die vergleichbar mit elementaren Belangen eines Organismus oder Motiven sind. *Concerns* werden mit internen und externen Stimuli in Verbindung

gebracht und führen zur Generierung von Zielen. Zusätzlich findet innerhalb des *Predictor*-Moduls eine Bewertung bzw. Einschätzung der Ereignisse und Stimuli statt, die später das Verhalten des Agenten beeinflussen wird. Das *Planner*-Modul strukturiert das Verhalten des Agenten auf Grundlage der vorliegenden Situation, sowie der Informationen, die im *Memory*-Modul abgelegt sind. In einer späteren Arbeit (s. Moffat, 1997) erweitert Moffat die in Abbildung 11.3 dargestellte Architektur noch um ein *Emotor*-Modul, das im wesentlichen mit Emotionen verbundene Handlungstendenzen kapselt.

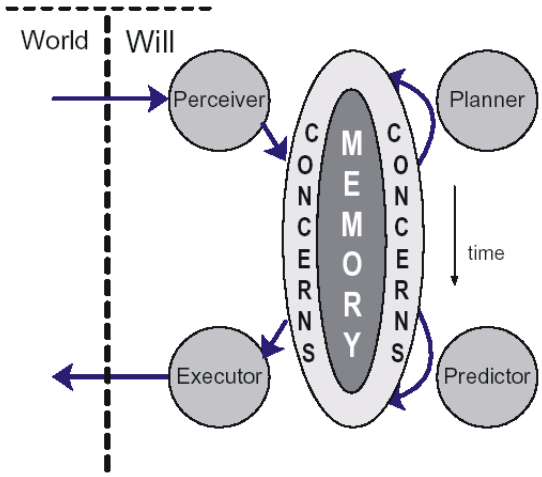

Abbildung11.3: Die Grundstruktur der Architektur Will

Die Kommunikation der Komponenten findet zentral über die *Memory*-Komponente statt. Das Verhalten des Agenten wird an demjenigen *Concern* ausgerichtet, dass zum betrachteten Zeitpunkt die größte Bedeutung für den Agenten hat. Die Einbettung der *Memory*-Komponente in einen Ring von *Concerns* deutet an, dass jegliche Information, die im *Memory* gespeichert wird, auf ihre Relevanz hinsichtlich der *Concerns* des Agenten geprüft wird.

Die Architektur von *Will* zeigt einen klaren und strukturierten Aufbau auf der Grundlage einer Menge von interagierenden Modulen. Dennoch basiert die Ablaufsteuerung innerhalb der Architektur und damit die Verhaltenssteuerung von Agenten, die auf der Grundlage von *Will* realisiert werden, auf einem spezifischen, motiv-ähnlichen Mechanismus. Dieser Mechanismus hat zwar allgemeinen Charakter, schränkt aber dennoch die interne Ausstattung des Agenten deutlich ein. Darüber hinaus vernachlässigt die Architektur die Betrachtung sozialer Verhaltensweisen und interner sozialer Zustände von Agenten.

11.2.6 Die Architektur des Oz-Projekts

Das *Oz*-Projekt an der Carnegie Mellon University beschäftigte sich in den neunziger Jahren mit der Entwicklung interaktiver, simulierter Welten mit künstleri-

scher Ausrichtung. Derartige Welten enthalten in der Regel eine Menge von Agenten, die in der Lage sein müssen, breit angelegte Verhaltensweisen auszuführen. Das Hauptziel dabei ist, die Agenten möglichst glaubwürdig zu gestalten, so dass ein Benutzer des Systems den Eindruck einer möglichst realitätsnahen Welt erhält. Im Rahmen dieses Projektes wurde die Agentenarchitektur *Tok* (Bates, Loyall & Reilly, 1992) entwickelt, die Funktionalitäten für Wahrnehmung, reaktives Verhalten, zielorientiertes Verhalten, Emotionen, Sozialverhalten und Sprachgenerierung aggregiert.

Tok besteht aus einer Menge interagierender Komponenten (s. Abbildung 11.4). Die Wahrnehmungsfunktionalitäten werden innerhalb der Komponente *Sensory Routines* abgebildet. Das Verhalten des Agenten wird innerhalb der Komponente *Hap* gesteuert. Emotionen und soziale Beziehungen werden in der Komponente *Em* abgebildet. Die Module zur Sprachanalyse und –erzeugung sind in Abbildung 11.4 nicht enthalten.

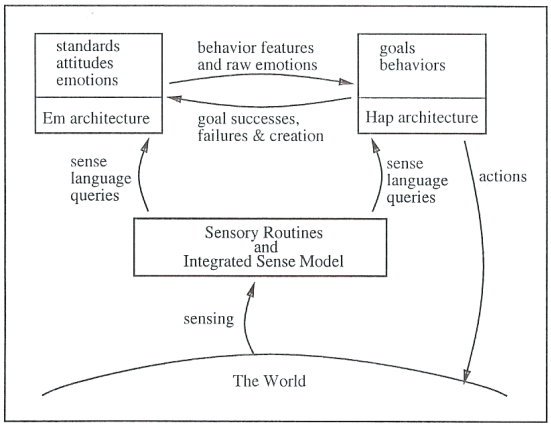

Abbildung11.4: Die Grundstruktur der Architektur Tok

Der grundsätzliche Ablauf in den Agenten ist zyklusorientiert angelegt, wobei jeder Zyklus aus den Einzelschritten Wahrnehmen, Denken und Handeln besteht. Tok verfügt über ausgeprägte Mechanismen zur Analyse sensorischer Informationen und zum Aufbau eines Weltmodells. Das Verhalten orientiert sich an der vorliegenden, externen Situation, dem aktuellen Ziel, dem emotionalen Zustand des Agenten, sowie ggfs. weiteren internen Zuständen. *Tok* verfügt über keine Planungsmechanismen, sondern wählt im Rahmen eines Ziels vordefinierte Pläne zur Ausführung aus. Emotionen entstehen in *Tok* als Folge kognitiver Prozesse und sind beispielsweise an den Erfolg ausgeführter Aktionen gekoppelt.

Die Konzepte, die im Rahmen des *Oz*-Projekts entwickelt wurden, werden mittlerweile auch erfolgreich in kommerziellen Anwendungen genutzt. *Tok* besticht durch eine klare Strukturierung auf der Grundlage eines komponentenorientierten Ansatzes. Allerdings beruht die Funktionalität der Komponenten auf spezifischen theoretischen Ansätzen, so dass eine Übertragung der Architektur auf

Anwendungsgebiete mit anderen Anforderungen nur sehr schwer möglich sein dürfte. Aus diesem Grund erscheint die Architektur *Tok* trotz ihrer wohldefinierten Struktur als zu spezifisch und damit als domänenunabhängiges Referenzmodell nicht geeignet.

11.2.7 Die Architektur Cathexis

Die Architektur *Cathexis* (Velásquez, 1997; Velásquez, 1998) wurde von Juan D. Velásquez am MIT Artificial Intelligence Laboratory entwickelt. Sie integriert Ansätze des emotional beeinflussten Entscheidens mit Wahrnehmungsfunktionen, Motivation, Verhaltenssteuerung und motorischen Kontrollfunktionen. *Cathexis* wurde im wesentlichen eingesetzt, um Software-Agenten zu implementieren und Roboter zu steuern.

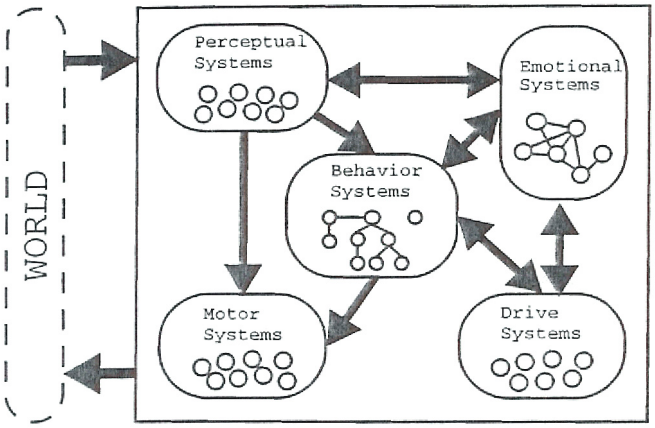

Abbildung 11.5: Die Grundstruktur der Architektur Cathexis

Die Architektur *Cathexis* besteht aus einer Menge von Komponenten, denen verschiedene Funktionen zugeordnet sind. Die *Perceptual Systems* nehmen Informationen aus der Umwelt auf und leiten diese als Stimuli an die *Behavior Systems* und *Emotional Systems* weiter. Die *Emotional Systems* und *Behavior Systems* erhalten zudem *Error* Signale aus den *Drive Systems*, die anzeigen, dass Motive existieren, nach denen das Handeln ausgerichtet werden muss. Innerhalb der *Emotional Systems* werden die Stimuli auf ihre emotionale Signifikanz hin überprüft. Es entsteht dabei ggfs. ein Einfluss auf die Verhaltenssteuerung sowie die Wahrnehmung eines Agenten. Im Rahmen der Verhaltenssteuerung werden entsprechende Verhaltensweisen generiert, die über die *Motor Systems* ausgeführt werden.

Der Aufbau von *Cathexis* gestaltet sich aufgrund der Aufgliederung in Module bzw. Komponenten sehr übersichtlich und wohlstrukturiert. Ebenso verfügt *Cathexis* bereits über wesentliche Elemente, die menschliches Verhalten beeinflus-

sen, wie etwa Motive, Emotionen, reaktive Verhaltensweise, sowie Wahrnehmungsfunktionen und aktuatorische Fähigkeiten. Vernachlässigt werden allerdings elementare kognitive Fähigkeiten wie etwa die Ablage von Informationen in einem Gedächtnis oder Deliberation. Ebenso fehlen soziale Zustände und Fähigkeiten. Diese Restriktionen lassen *Cathexis* als Grundlage für eine flexible Modellierung menschlichen Verhaltens als ungeeignet erscheinen.

11.2.8 Die Architektur Joshua

Die Architektur *Joshua* (Alvarado, Adams, Burbeck & Latta, 2001) ist ein Produkt aus dem *Joshua Blue* Projekt am Watson Research Center der Firma IBM. *Joshua Blue* zielt darauf ab, den Entwurf intelligenter Systeme zu verbessern, indem Konzepte, die an menschliche Emotionen angelehnt sind, berücksichtigt werden. Die Architektur soll in die Lage versetzt werden, ebenso wie der Mensch, Fähigkeiten des logischen Schließens, das Verstehen natürlicher Sprache oder auch emotionale Intelligenz im Laufe der Zeit durch Lernprozesse in einer reichhaltig ausgestatteten Umgebung zu erwerben.

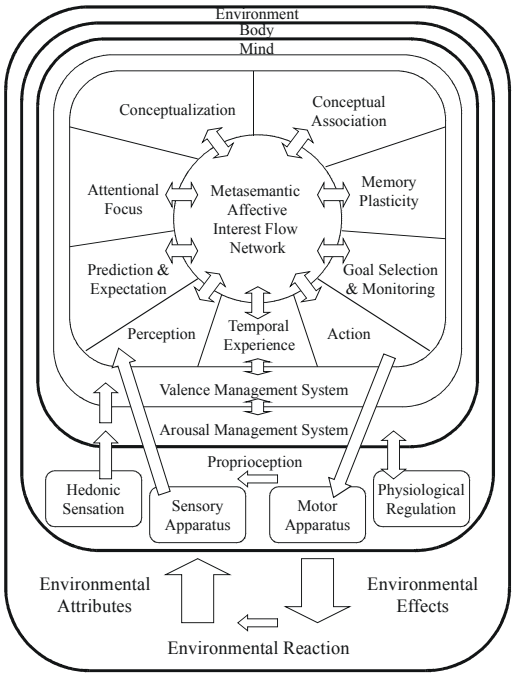

Abbildung 11.6: Die Grundstruktur der Architektur Joshua

Im Zentrum der Architektur *Joshua* steht das sog. *Metasemantic Affective Interest Flow Network*, ein semantisches Netz, das verschiedene (nicht näher spezifizierte) neue Formen mentaler Repräsentationen enthält. Weiterhin verfügt die Architektur über sensorische und motorische Komponenten, die die Verbindung des Agenten mit der Außenwelt herstellen, sowie über Wahrnehmungsfunktionen

zur Verarbeitung sensorischer Daten. *Joshua* verfügt über die Möglichkeit, Assoziationen zwischen wahrgenommenen Informationen herzustellen, Muster in derartigen Assoziationen zu erkennen und diese Muster anschließend auf der Grundlage von Erfahrungen mit einer entsprechenden Semantik zu versehen. Die Erfahrungen werden auf der Grundlage von Zielen hinsichtlich ihrer Relevanz bewertet. Ziele und deren zugeordnete Dinglichkeit sind dabei Bestandteil eines motivationalen Systems. Zudem integriert *Joshua* einen Informationspool, ein Aufmerksamkeits- und Bewusstseinskonzept, sowie eine größere Anzahl an weiteren kognitiven Fähigkeiten, allerdings keinerlei Form von Logik, da sich entsprechend der Erkenntnisse im Bereich der Entwicklungspsychologie logisches Schließen beim Menschen erst im Laufe der Zeit entwickelt und nicht angeboren ist. Verknüpfungen im semantischen Netz von *Joshua* werden verstärkt, wenn sie wiederholt durchlaufen werden oder eine entsprechende Relevanz im Rahmen der emotionalen Bewertung erhalten. Jeder Knoten des semantischen Netzes ist mit einer bestimmten emotionalen Signifikanz versehen, so dass aus der Summe der Konten der emotionale Zustand von *Joshua* emergiert. Zusätzlich besteht zwischen der Aktivierung und dem emotionalen Zustand von *Joshua* eine wechselseitige Beeinflussung. Die Aktivierung wirkt zudem auf die Aufmerksamkeit des Agenten ein.

Gegenwärtig sind in *Joshua* zwei Motive implementiert: ein Motiv zur Vermeidung von Schmerz, sowie ein Motiv für das Streben nach Wohlbehagen. Affekte, die mit Aktionen einhergehen, werden bei den Zielen im Informationspool des Agenten gespeichert. Ebenso werden die ausgeführten Verhaltensweisen im Zusammenhang mit den Zielen abgelegt, so dass auf diese Weise ein operantes Lernen möglich wird.

Eine aussagekräftige Evaluation der Architektur *Joshua* ist zum derzeitigen Stand nicht möglich, da weder die Beschreibung der Architektur genauere Einblicke in ihre konkrete Funktionsweise vermittelt, noch eine nachvollziehbare Dokumentation der realisierten Testumgebung existiert. Angesichts der Komplexität, die offensichtlich in dieser Architektur steckt, ist eine exakte Spezifikation der Komponenten sowie ihrer Wechselwirkungen allerdings unerlässlich. *Joshua* als Referenzarchitektur in Erwägung zu ziehen, erscheint deshalb (zumindest derzeit noch) in Anbetracht der angeführten Unzulänglichkeiten als wenig sinnvoll.

11.2.9 Die Architektur H-CogAff

Eine sehr komplexe Architektur, die darauf abzielt, die wesentlichen Aspekte der Informationsverarbeitung eines normalen, erwachsenen Menschen nachzubilden, wird gegenwärtig an der School of Computer Science der University of Birmingham unter der Leitung von Aaron Sloman entwickelt. Diese Architektur trägt den Namen *H-CogAff* (Sloman & Scheutz, 2002; Sloman, Chrisley & Scheutz, 2003) und stellt eine Erweiterung der im Rahmen des *Cognition and Affect*-Projekts entwickelten Architektur *CogAff* dar.

CogAff basiert auf Annahmen über verschiedene menschliche Fähigkeiten sowie auf Überlegungen zur Evolution der menschlichen Intelligenz. Sie bezieht ingenieursmäßige Entwurfsüberlegungen mit ein, die darauf abzielen, Unzulänglichkeiten bestehender KI-Systeme zu beheben. Zudem versucht diese Architektur Anforderungen zu erfüllen, die an ein intelligentes System gestellt werden, wenn es in der Lage sein soll, vielfältige Motive in einer komplexen und dynamischen Umwelt zu verwalten. Die Architektur *CogAff* integriert zu diesem Zweck reaktive, deliberative wie auch reflektive Formen der Verhaltenssteuerung und ordnet diese in drei Schichten an. Auf der untersten Schicht sind reaktive Mechanismen angesiedelt, die automatische Verhaltensweisen modellieren. Auf der mittleren Ebene befinden sich deliberative Mechanismen, die der Erzeugung und Ausführung von Handlungsplänen dienen. Die oberste Ebene bildet die Schicht des Meta-Management bzw. die reflektive Schicht. Hier werden Fähigkeiten wie etwa Selbstbeobachtung, Selbstbewertung und Selbstkontrolle modelliert, die zu einer Verbesserung der deliberativen Fähigkeiten beitragen können.

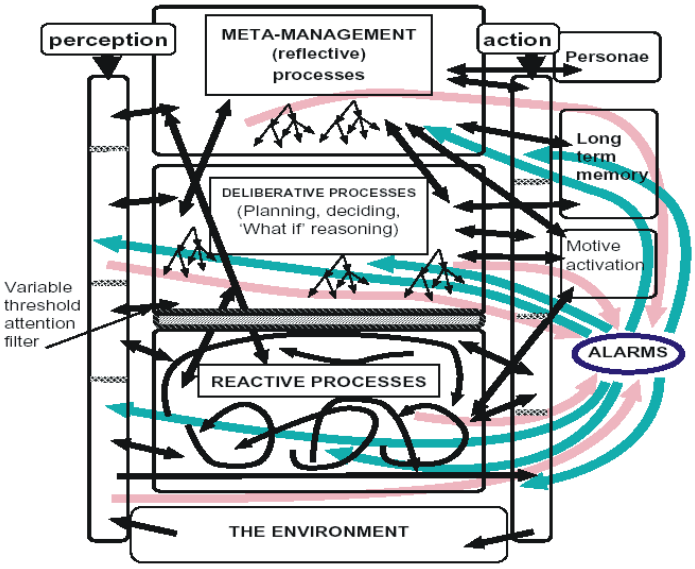

Abbildung 11.7: Die Grundstruktur der Architektur H-CogAff

H-CogAff bringt einige neue Gedanken in das klassische *CogAff*-Schema ein. Hierzu zählt ein erweitertes *Alarm*-Konzept, das mit allen wesentlichen Komponenten der Architektur in Verbindung steht und bewirkt, dass in kritischen oder unerwarteten Situationen das Verhalten eines Agenten neu ausgerichtet werden kann. Der *Attention Filter* kontrolliert die Aufmerksamkeit eines Agenten. Er soll dazu beitragen, dass das Handeln eines Agenten auf die Erfordernisse der vorliegenden Situation sowie seine Ziele fokussiert wird und auf diese Weise Kontinuität und Stabilität erhält. Das *Personae*-Modul gibt einem Agenten die Möglichkeit, in Abhängigkeit seiner Persönlichkeit und der vorliegenden Situation unterschiedliches Verhalten zu zeigen. Ebenso wird die Architektur *H-CogAff* im Ver-

gleich zu ursprünglichen Architektur um einen Langzeit-Informationsspeicher angereichert, um einem Agenten die Möglichkeit zu geben, im Laufe der Zeit Erfahrungen zu sammeln und im Rahmen seiner Verhaltenssteuerung auf diese zurückzugreifen.

Aufgrund des sehr abstrakt und allgemein gehaltenen Ansatzes ist diese Architektur nicht auf Anwendungen einer speziellen Domäne beschränkt und damit sehr breit einsetzbar. Allerdings fehlt es dieser Architektur an Transparenz und Klarheit in Bezug auf die einzelnen Subsysteme sowie deren Interaktion. Zudem bleibt unklar, in welcher Weise miteinander wechselwirkende, interne Zustände, die das menschliche Verhalten maßgeblich beeinflussen, im Rahmen dieser Architektur Berücksichtigung finden können. Besonders zu erwähnen sind in diesem Zusammenhang soziale Eigenschaften und Fähigkeiten, die einen Menschen in grundlegender Weise auszeichnen, in der *H-CogAff*-Architektur allerdings keine Berücksichtigung finden. Ebenso wird im Rahmen der Architekturbeschreibung (s. Sloman, Chrisley & Scheutz, 2003) angedeutet, dass *H-CogAff* die Abbildung einer Vielfalt menschlicher Emotionen ermöglicht. Allerdings bleibt bislang noch ungeklärt, in welcher Weise die Abbildung von Emotionen konkret erfolgen soll. Dasselbe gilt für den größten Teil der in *H-CogAff* neu eingeführten Konzepte und Komponenten. Fairer Weise muss an dieser Stelle jedoch angeführt werden, dass es erklärtes Ziel der Entwickler von *H-CogAff* ist (s. Sloman, Chrisley & Scheutz, 2003), die Bestandteile der Architektur in der Zukunft weiterführend zu spezifizieren und der Architektur auf diese Weise mehr Praxisrelevanz zu verleihen.

11.2.10 Die Ψ-Theorie

Die Ψ-Theorie von Dietrich Dörner (Dörner, 1999) unternimmt den Versuch, menschliches Verhalten in umfassender und integrativer Weise zu modellieren. Sie geht von der Grundüberzeugung aus, dass für eine adäquate Untersuchung des menschlichen Verhaltens eine isolierte Betrachtung von Kognition, Emotion und Volition nicht ausreichend ist. Vielmehr bedarf es einer psychologischen Gesamttheorie, die auf einem Zusammenspiel verschiedenartiger interner Prozesse beruht.

Die Ψ-Theorie basiert auf einer Reihe von einfachen Grundkonzepten. Sämtliche Informationen werden innerhalb der Architektur mit Hilfe von Neuronen gespeichert. Die Neuronen können zu komplexeren Strukturen zusammengeschlossen werden, um beispielsweise Verhaltensprogramme oder sensorische Informationen zu repräsentieren und zu speichern. Die Verhaltensweisen, die Ψ, so wird eine Instanz der Ψ-Architektur verkürzt genannt, hervorbringen kann, resultieren aus den gespeicherten Gedächtnisinhalten sowie aus einer Kombination und Modulation einer geringen Anzahl an psychischen Prozessen.

Den Ausgangspunkt für die in der Ψ-Theorie realisierte Verhaltenssteuerung bildet eine Menge von über die Zeit veränderlichen, internen Zuständen. Aus diesen Zuständen resultieren Bedürfnisse und schließlich Motive, die miteinander

ringen, um das Handeln von Ψ zu leiten. Zu jedem Zeitpunkt existiert dabei genau ein ausgewähltes, handlungsleitendes Motiv.

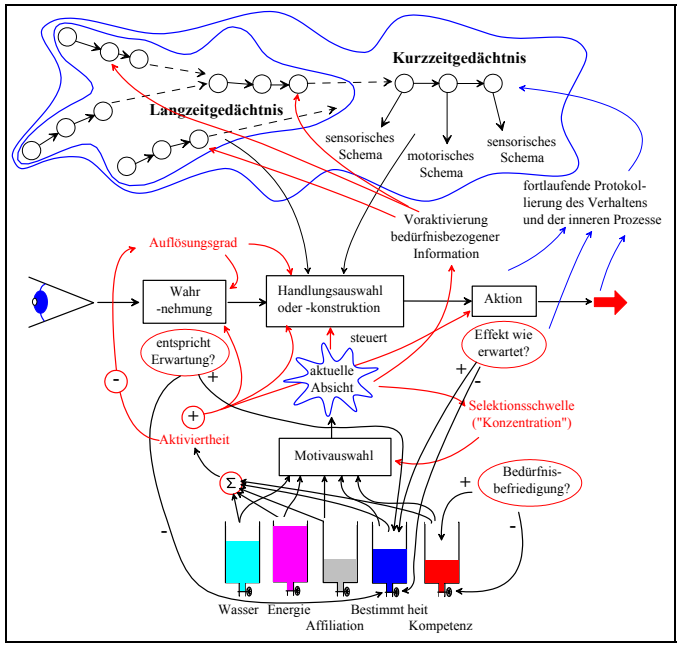

Abbildung 11.8: Die Architektur der Ψ-Theorie

Die Aufbau und die Funktionsweise von Ψ basieren auf einer spezifischen, sehr umfassenden und integrativen, psychologischen Theorie des menschlichen Handelns. Ψ umfasst eine Reihe von wesentlichen Merkmalen und Prozessen, die auch für eine Agentenarchitektur zur Modellierung menschlichen Verhaltens erforderlich sind. Besonders hervorzuheben sind hierbei die dynamischen, internen Zustände, die physische, emotionale und kognitive Eigenschaften eines Menschen nachbilden können, sowie die darauf aufsetzende und wohlorganisierte Verhaltenssteuerung von Ψ, die reaktive und auch deliberative Verhaltensweisen auf der Grundlage von Motiven einschließt. All diese Argumente würden dafür sprechen, Ψ auch im Sinne der vorliegenden Arbeit als Referenzmodell zu betrachten. Bei genauerer Analyse ergeben sich allerdings einige Aspekte, die dieser Sichtweise entgegenstehen.

Zum einen verhindert die fehlende Integration von sozialen Merkmalen, dass Ψ auch für die Modellierung von sozialen Individuen eingesetzt werden kann. Der Ψ-Ansatz wäre also in der vorliegenden Form als Referenzmodell nur für solche Szenarien geeignet, die sich auf die Betrachtung isolierter Agenten zurückziehen. Eine Modellierung künstlicher Sozialsysteme, die Agenten mit unterschiedlichen sozialen Eigenschaften berücksichtigen, ist auf der Grundlage von Ψ derzeit noch nicht möglich. Allerdings gehört es im Rahmen des deutschen Sozionik-Projektes zu den aktuellen Aktivitäten der Gruppe um Dietrich Dörner, Ψ zu „sozialisieren“ und in entsprechender Weise mit sozialen Eigenschaften und Fähigkeiten auszustatten. Es handelt sich in diesem Fall also um kein kategorisches Hemmnis.

Ein zweiter Aspekt, der den Einsatz von Ψ als breit einsetzbares Referenzmodell in wesentlich schwerwiegenderer Art und Weise beeinträchtigt, ist Ψ's Verhaltenssteuerung, die (natürlich) eine Implementierung der psychologischen Handlungstheorie von Dörner darstellt. Der Einsatzbereich der Architektur würde sich damit auf solche Anwendungsszenarien beschränken, für die ein motivbasierter Ansatz zur Verhaltenssteuerung von Agenten geeignet und sinnvoll erscheint. Ein theorieunabhängiger Einsatz der Ψ-Architektur wäre aufgrund der zentralen Rolle dieser Art von Verhaltenssteuerung nur mit wesentlichen Anpassungen an der Architektur möglich.

Ein drittes Argument bezieht sich auf die Strukturierung von Ψ. Ψ stellt sich nach außen hin als monolithisches Gesamtsystem dar, das nur dann funktionsfähig ist, wenn alle Komponenten vorhanden und aktiv sind. Im Zusammenhang mit einem Referenzmodell stellt eine derartige Konstruktion eine fundamentale Schwierigkeit dar, da flexible Anpassungen an Gegebenheiten und Anforderungen spezifischer Anwendungsszenarien in derartig aufgebauten Systemen nur sehr schwierig machbar sind und in der Regel einen sehr hohen Aufwand implizieren.

Als Fazit lässt sich festhalten, dass die Ψ-Theorie wohl einen der umfassendsten und integrativsten handlungstheoretischen Ansätze darstellt und damit sicherlich wertvolle Konstruktionsprinzipien für den Entwurf von Agenten liefert. Für einen unmittelbaren Einsatz als Referenzmodell erscheint Ψ hingegen zumindest zum gegenwärtigen Stand noch ungeeignet.

11.3 Fazit der Architekturevaluation

In den vorangegangenen Abschnitten wurden eine Reihe von Architekturen und Modellierungsansätzen diskutiert, die im Rahmen der Künstlichen Intelligenz und angrenzenden Gebieten ein hohes Maß an Bedeutung erlangt haben oder hinsichtlich der Zielsetzung der vorliegenden Arbeit als besonders relevant erachtet werden. Als Fazit können die beiden folgenden, grundlegenden Erkenntnisse aus der Architekturevaluation gezogen werden:

Klassische Architekturansätze aus dem Bereich der Künstlichen Intelligenz, wie deliberative, reaktive, hybride und BDI-Architekturen, sind in der Regel theoretisch fundiert, wohlstrukturiert und bereits sehr intensiv erforscht. Sie stellen damit mittlerweile eine solide Plattform für die Entwicklung technischer Systeme dar. Um menschliche Akteure in adäquater Weise in Simulationsmodellen abzubilden, fehlen diesen Architekturen jedoch grundlegende Konzepte wie etwa die Betrachtung emotionaler Zustände und deren Auswirkungen auf die Verhaltenssteuerung von Agenten.

Architekturen und Modellierungsansätze, die eher dem Bereich der Kognitionswissenschaften, der Handlungstheorien oder der Emotionspsychologie entstammen, sind in der Regel sehr reichhaltig ausgestattet. Sei umfassen häufig sowohl kognitive als auch emotionale Aspekte im Zusammenhang mit der Verhal-

tenssteuerung von Agenten. Die grundlegende Zielsetzung, die hinter der Entwicklung derartiger Modelle steht, bezieht sich jedoch zumeist auf eine Operationalisierung spezifischer sozialwissenschaftlicher Theorien. Die für derartige Modelle gewählten Konzepte haben daher keinen allgemeinen Charakter, sondern sind elementar an den Inhalten der abzubildenden Theorie ausgerichtet. Dieser Sachverhalt widerspricht ganz grundlegend der konzeptionellen Ausrichtung eines Referenzmodells, wie es in der vorliegenden Arbeit vorgestellt wird. Aus diesem Grund liefern sozialwissenschaftliche Modellierungsansätze zwar wertvolle Hinweise für den Aufbau eines allgemein ausgerichteten und theorieunabhängigen Referenzmodells, sind aber selber nicht in der Lage, damit verbundene Anforderungen zu erfüllen.

Der Ansatz, der der Konzeption und inhaltlichen Ausrichtung des Referenzmodells *PECS* am nächsten kommt, ist die in Abschnitt 11.2.9 vorgestellte Architektur *H-CogAff* von Sloman und Scheutz (2003). *H-CogAff* teilt die Auffassung, dass ein Referenzmodell bzw. eine Referenzarchitektur allgemeinen Charakter haben muss, und bietet ein breites Spektrum an funktionalen Komponenten an. Als nachteilig erweist sich in diesem Ansatz allerdings die fehlende Strukturierung. Insbesondere bleiben die komplexen Interaktionen der einzelnen Architekturbestandteile noch weitgehend unklar. Weiterhin stellt sich die Frage, wie *H-CogAff* konkret an die Anforderungen spezifischer Anwendungsszenarien angepasst werden kann. Das gewählte Schichtenprinzip lässt vermuten, dass die Grundarchitektur sehr eng mit den angedachten Verhaltenssteuerungsmechanismen verwoben ist und sich eine Einpassung ergänzender Konzepte sehr schwierig gestaltet.

12 RESÜMEE UND PERSPEKTIVEN

Das folgende Kapitel bildet den Abschluss der vorliegenden Arbeit und besteht aus zwei Teilen. Im ersten Teil werden ausgehend von der grundlegenden Problemstellung die im Rahmen dieser Arbeit entwickelten Lösungsansätze noch einmal in der Übersicht zusammengestellt und diskutiert. Der zweite Teil dieses Kapitels widmet sich den Perspektiven, die sich auf der Grundlage der Ergebnisse dieser Arbeit eröffnen und entweder bereits Gegenstand weiterführender Forschungsaktivitäten sind oder zumindest in der Zukunft dazu werden können.

12.1 Resümee

12.1.1 Motivation und Zielsetzung dieser Arbeit

Die Agententechnologie hat in den letzten Jahren eine rasante Entwicklung erfahren und mittlerweile großen Einfluss auf verschiedene Bereiche in Wissenschaft und Technik genommen. Insbesondere wurde die agentenbasierte Modellbildung und Simulation als wirkungsvolles Mittel für die Untersuchung realer oder hypothetischer Systeme erkannt. Im Vordergrund stehen hierbei ganz besonders Systeme, die durch menschliches Handeln, Entscheiden und Verhalten beeinflusst werden, oder der Mensch selbst, um seine Eigenschaften und Fähigkeiten aus den Blickwinkeln unterschiedlicher wissenschaftlicher Disziplinen heraus weiterführend zu erforschen.

Eine wesentliche Aufgabe der Informatik besteht in diesem Zusammenhang darin, methodologische Aspekte der agentenbasierten Modellbildung weiter voranzubringen. Durch eine steigende Komplexität der Anwendungsszenarien wachsen auch die Anforderungen an die zugrunde liegende Modellierungsmethodik. Insbesondere macht sich hier die Tendenz bemerkbar, dass der Mensch in vielen Anwendungsbereichen nicht mehr nur als rationaler Entscheider betrachtet wird, sondern eine Vielzahl weiterer Einflüsse mit einbezogen werden, um den Menschen in den jeweiligen Anwendungsfeldern adäquat in Simulationsmodellen abzubilden. Die Konstruktion autonomer Agenten sowie gesamter Modelle, in denen menschliches Verhalten eine Rolle spielt, stellt zunehmend höhere Ansprüche an

den Modellentwickler und entpuppt sich, wie auch Troitzsch und Gilbert es unterstreichen (Troitzsch & Gilbert, 1999), als grundlegende und in hohem Maße komplexe Aufgabe.

Einen Ansatzpunkt, um wiederkehrende, komplexe Entwurfsaufgaben konzeptionell zu unterstützen, stellen innerhalb der Informatik die sog. *Patterns* und innerhalb der Simulationstechnik die sog. *Referenzmodelle* dar. Referenzmodelle umfassen systematische und allgemeingültige Beschreibungen definierter Erkenntnisgegenstände mit ihren relevanten und charakteristischen Eigenschaften und bieten adäquate Modellierungskonzepte und Konstruktionsschemata für die Entwicklung von Simulationsmodellen an. Im Rahmen von Simulationsstudien beeinflussen Referenzmodelle sowohl die Betrachtung des zu untersuchenden Systems als auch die Entwicklung des abstrakten Modells. Sie vermitteln ein gewisses Vorverständnis, das sich positiv auf die Analyse der wesentlichen Eigenschaften und Zusammenhänge des zu untersuchenden Systems auswirkt. Ebenso erleichtern und beschleunigen sie den Modellbildungsprozess durch die Vorgabe entsprechender Strukturierungsprinzipien und Ansätze für die Beschreibung der Modelldynamik.

Das primäre Ziel der vorliegenden Arbeit besteht darin, den Entwurf agentenbasierter Modelle, in denen menschliches Handeln, Entscheiden und Verhalten von ausschlaggebender Bedeutung sind, auf konzeptioneller Ebene zu unterstützen. Wie im vorhergehenden Absatz dargelegt wurde, bieten Referenzmodelle einen viel versprechenden Ansatzpunkt, um dieses Ziel zu erreichen. Diese Arbeit geht dabei von der Grundüberzeugung aus, dass durch die Bereitstellung eines geeigneten und breit einsetzbaren Referenzmodells die Komplexität der Entwurfsaufgabe, die mit der Entwicklung agentenbasierter Simulationsmodelle insbesondere unter Einbeziehung menschlicher Akteure verbunden ist, deutlich reduziert werden kann.

12.1.2 Grundlegende Anforderungen

Der Entwurf eines Referenzmodells, wie es in der vorliegenden Arbeit vorgeschlagen wird, muss sich an einer Reihe von Anforderungen orientieren, die zum einen unmittelbar aus der Definition von Referenzmodellen und zum anderen aus den wesentlichen Merkmalen agentenbasierter Simulationsmodelle, in denen Menschen auf der Grundlage autonomer Agenten modelliert werden, resultieren.

Referenzmodelle müssen allgemein ausgerichtet sein, d. h. weitgehend von individuellen Anforderungen spezifischer Anwendungsszenarien abstrahieren, um ein möglichst breites Einsatzfeld zu adressieren. Zudem müssen Referenzmodelle eine klare Struktur aufweisen und methodologisch mit den Gegebenheiten des beabsichtigten Einsatzgebietes konform gehen. Ein klare Strukturierung trägt dabei ganz wesentlich zur Verständlichkeit des Referenzmodells bei und stellt die Basis für eine gute Adaptierfähigkeit des Referenzmodells an spezifische Anforderungen konkreter Anwendungsfälle dar. Die methodologische Konformität bewirkt,

dass bei der Konzeption gängiges Wissen aus dem beabsichtigten Einsatzgebiet des Referenzmodells mit einfließt.

Aus funktionaler Sicht entstehen bei der Entwicklung von Simulationsmodellen, die menschliches Handeln, Entscheiden und Verhalten mit einbeziehen, drei übergeordnete Aufgabenstellungen, die im Rahmen eines allgemeinen Modellierungsansatzes integriert werden müssen.

Die erste und komplexeste Aufgabenstellung bezieht sich auf die Konzeption und den Entwurf einer Agentenarchitektur, die eine Modellierung der wesentlichen Einflüsse, die bei der Abbildung von Menschen in Simulationsmodellen von Bedeutung sind, in flexibler Weise zulässt. Zu diesen Einflüssen gehören zunächst einmal sensorische und aktuatorische Fähigkeiten, die einen Menschen ganz allgemein in die Lage versetzen, seine Umgebung wahrzunehmen und diese auch zu verändern. Weiterhin müssen komplexe und miteinander wechselwirkende, interne Zustände, Fähigkeiten und Prozesse berücksichtigt werden, die gemäß den Erkenntnissen moderner Handlungstheorien in entscheidender Weise das menschliche Verhalten beeinflussen. Hierzu zählen insbesondere kognitive Eigenschaften und Fähigkeiten, emotionale Zustände und Dispositionen, physiologische Bedingungen und nicht zuletzt soziale Charakteristika und Einflüsse. Darüber hinaus verfügt der Mensch über unterschiedliche Mechanismen der Verhaltenssteuerung, die er entsprechend den Erfordernissen der jeweiligen Situation flexibel einsetzen kann. Man unterscheidet dabei zwischen reaktivem, deliberativem und reflektivem Verhalten. Reaktives Verhalten basiert auf Automatismen und Reaktionen, die ohne umfangreichere Denkprozesse ablaufen können. Deliberatives Verhalten schließt komplexere, zielgerichtete Verhaltensweisen mit ein, die etwa auf Denk-, Erinnerungs- und Planungsprozessen beruhen. Reflektives Verhalten ermöglicht es dem Menschen schließlich, sich selbst, seine Ziele und sein Verhalten zu beobachten, Schlüsse daraus zu ziehen und auch Veränderungen daran vorzunehmen. All diese Kategorien unterschiedlicher Anforderungen müssen in allgemeiner und integrativer Weise im Rahmen einer geeigneten Agentenarchitektur Berücksichtigung finden.

Die zweite, übergeordnete Aufgabenstellung bezieht sich auf die Interaktion von Menschen und deren Abbildung in agentenbasierten Simulationsmodellen. In dieser Hinsicht müssen Agenten zumindest in die Lage versetzt werden, Informationen untereinander auszutauschen. Ein allgemeines Referenzmodell muss in dieser Hinsicht also eine Infrastruktur sowie ein Minimalmaß an Mechanismen vorsehen, die die Interaktion bzw. den Informationsaustausch zwischen Agenten in grundlegender Weise unterstützen.

Ein dritte, elementare Aufgabe ergibt sich aus der wechselseitigen Interaktion des Menschen mit seiner Umwelt. Die Umwelt stellt das physikalische Bezugssystem des Menschen zur Verfügung. Sie generiert sensorische Reize und kann auch durch den Menschen verändert werden. Die dritte Anforderung von ganz grundlegender Bedeutung bei der Konzeption eines agentenbasierten Referenzmodells betrifft also die Bereitstellung von Funktionalitäten im Zusammenhang mit der Modellierung der Umgebung von Agenten.

12.1.3 Der Lösungsansatz

Unter Berücksichtigung der vorstehend genannten Anforderungen wird im Rahmen dieser Arbeit das Referenzmodell *PECS* entwickelt. *PECS* stellt ein domänen- und theorieunabhängiges Konstruktionsschema für agentenbasierte Simulationsmodelle zur Verfügung, in denen menschliche Akteure abgebildet werden. Es legt ein elementares Strukturierungsprinzip fest und gibt einen generischen Modellierungsrahmen vor, der anwendungsspezifisch instanziiert und mit entsprechenden Modellelementen, Verhaltens- und Dynamikbeschreibungen sowie Modelldaten befüllt werden kann.

12.1.3.1 Grundlagen

Um eine übersichtliche und adaptierbare Struktur zu erreichen und eine adäquate Beschreibung interner Zustände, Verhaltensweisen und Zustandsübergänge zu gewährleisten, werden der Ausgestaltung des Referenzmodells *PECS* zwei im Bereich der Simulationstechnik bewährte Prinzipien zugrunde gelegt. Es handelt sich dabei zum einen um das Prinzip des komponentenorientierten, hierarchischen Modellaufbaus sowie um die zustandsorientierte Modellierung auf der Grundlage eines klassischen, systemtheoretischen Ansatzes.

Der komponentenorientierte und hierarchische Modellaufbau ermöglicht eine funktionale Dekomposition komplexer Modelle in eine Menge überschaubarer, interagierender Komponenten, die hierarchisch aggregiert werden können. Er führt zu einer übersichtlichen Strukturierung komplexer Modelle und bildet die Grundlage für eine strukturähnliche Abbildung realer Sachverhalte in Simulationsmodellen. Mit einer guten Zerlegung im Sinne eines komponentenorientierten Ansatzes kann eine sehr flexible Adaptierbarkeit und Erweiterbarkeit eines Modells erreicht werden. Gerade für die Konzeption eines Referenzmodells, wie es im Rahmen dieser Arbeit entwickelt wird, stellt sich der komponentenorientierte und hierarchische Modellaufbau als sehr nützliches Strukturierungsprinzip heraus. Er ermöglicht es, bei Bedarf die Grundarchitektur des Referenzmodells in einfacher Weise an spezifische Anforderungen individueller Anwendungsfälle anzupassen, ohne die Grundkonzeption des Modells aufgeben zu müssen.

Die vorliegende Arbeit geht von der Grundüberzeugung aus, dass der Mensch, ebenso wie viele andere, reale Systeme, im Rahmen von Simulationsmodellen als dynamisches System betrachtet werden kann. Diese Sichtweise eröffnet die Möglichkeit, klassische, systemtheoretische Konzepte für die Modellierung menschlicher Eigenschaften und Fähigkeiten zu nutzen. Menschen werden also im Rahmen des Referenzmodells *PECS* als dynamische Systeme modelliert, die Inputs aus der Umgebung aufnehmen können, über ausgeprägte, interne Zustände verfügen und in der Lage sind, entsprechende Outputs, beispielsweise in Form von spezifischen Verhaltensweisen und extern wirksamen Aktionen, zu generieren. Der interne Zustand des dynamischen Systems korreliert dabei mit den als re-

levant erachteten, internen Zuständen des Menschen und bezieht insbesondere physische, emotionale, kognitive und soziale Aspekte mit ein. Ein systemtheoretisches Modell eines Menschen mit den eben genannten Eigenschaften wird im Rahmen dieser Arbeit als *Agent* bezeichnet.

12.1.3.2 Das Referenzmodell *PECS*

Der Entwurf des Referenzmodells *PECS* basiert auf der Zielsetzung, die grundlegenden Funktionalitäten, die für eine domänen- und theorieunabhängige Modellierung menschlichen Handelns, Entscheidens und Verhaltens erforderlich sind, im Rahmen einer integrativen Modellarchitektur bereitzustellen.

Auf der Grundlage des komponentenorientierten Modellaufbaus setzt sich die Architektur des Referenzmodells *PECS* aus einer Menge zunächst voneinander unabhängiger und nebenläufig arbeitender Komponenten zusammen, die auf der Grundlage von kausalen Abhängigkeiten und diskreten Informationsflüssen miteinander interagieren können. Zudem erfolgt eine hierarchische Integration der Komponenten zu Teilmodellen, um die Modellstruktur übersichtlich und flexibel zu gestalten.

PECS sieht für den Aufbau agentenbasierter Simulationsmodelle grundsätzlich drei Klassen von Komponenten vor. Die Komponente *Agent* ist darauf ausgerichtet, menschliche Akteure mit ihren als relevant erachteten Eigenschaften und Fähigkeiten im Modell abzubilden. Die Komponente *Environment* dient der Modellierung von Gegebenheiten in der Umwelt der Agenten. Die Komponente *Connector* realisiert eine zentralisierte Kommunikationsinfrastruktur, die den Agenten je nach Bedarf zum Austausch von Informationen zur Verfügung steht.

Neben den eigentlichen Komponenten werden im Rahmen von *PECS* auch elementare Konzepte für deren Interaktion vorgeschlagen. Das Bindeglied zwischen den Agenten und der Umwelt stellen zum einen sensorische Informationen dar, die in der Umwelt erzeugt und für die Agenten als Eingansginformationen bereitgestellt werden, und zum anderen Aktionen, mit deren Hilfe Agenten Einfluss auf den Zustand der Umwelt nehmen. Der Informationsaustausch basiert auf Nachrichten, die durch die Komponente *Connector* entweder allgemein zugänglich gemacht oder direkt zugestellt werden.

Den Kern des Modellkonzeptes bilden die Agenten. *PECS*-Agenten sind auf der Grundlage einer funktionalen Dekomposition wesentlicher menschlicher Eigenschaften und Fähigkeiten in eine Menge von Komponenten zerlegt, die unterschiedliche Aufgaben erfüllen und gedanklich drei vertikal angeordneten Schichten zugeordnet werden können. In der obersten Schicht befinden sich die Komponente *Sensor* und *Perception*. Diese beiden Komponenten widmen sich der Verarbeitung sensorischer Informationen sowohl aus der Umgebung als auch aus dem Inneren des Agenten. Auf der mittleren Ebene befinden sich die Komponenten *Physis*, *Emotion*, *Cognition* und *Social Characteristics*. Diese Komponenten nehmen den internen Zustand des Agenten auf und beschreiben damit verbundene

Prozesse und Zustandsübergänge. Wesentlich ist dabei, dass eine grundsätzliche Interaktionsmöglichkeit aller internen Komponenten besteht, um Wechselwirkungen interner Zustände und Prozesse, die beim Menschen vorzufinden sind, auch im Modell in adäquater Weise abbilden zu können. In *PECS* können diese Wechselwirkungen im wesentlichen auf der Grundlage von kausalen Abhängigkeiten modelliert werden. In der untersten Schicht der Agentenarchitektur sind die Komponenten *Behaviour* und *Actor* angesiedelt. Diese beiden Komponenten beschäftigen sich mit der Verhaltenssteuerung des Agenten. Innerhalb der Komponente *Behaviour* werden dabei die Verhaltensweisen des Agenten festgelegt und schließlich in Form von Aktionen durch die Komponente *Actor* zur Ausführung gebracht. Von grundlegender Bedeutung ist im Rahmen der Verhaltenssteuerung die Beeinflussung der Verhaltensauswahl durch den gesamten internen Zustand des Agenten. Die Komponente *Behaviour* steht zu diesem Zweck mit allen internen Komponenten der Agentenarchitektur, d. h. also mit den Komponenten *Physis*, *Emotion*, *Cognition* und *Social Characteristics*, in unmittelbarer Verbindung. Im Zusammenspiel mit den internen Komponenten sind *PECS*-Agenten grundsätzlich in der Lage, sowohl reaktive, als auch deliberative und reflektive Verhaltensweisen zu modellieren.

Das Referenzmodell *PECS* ist aufgrund der allgemeinen Ausrichtung ganz bewusst auf einer hohen Abstraktionsebene angesiedelt. Um *PECS* im Rahmen einer realen Anwendung einzusetzen, sind im wesentlichen die folgenden Aktivitäten erforderlich:

Zunächst ist auf Grundlage der Gegebenheiten des Anwendungsfalles zu entscheiden, welche *PECS*-Komponenten in der vorliegenden Situation tatsächlich benötigt werden. Nicht benötigte Komponenten können in der Regel aufgrund des modularen Aufbaus der Modellarchitektur ohne signifikante Beeinträchtigungen weggelassen werden. Ebenso besteht die Möglichkeit, neue Komponenten, die in der *PECS*-Architektur nicht enthalten sind, in das Modell einzubringen und mit den anderen Komponenten zu verknüpfen. In einem zweiten Schritt wird dann die Ausstattung der einzelnen Komponenten festgelegt. Zu diesem Zweck sind die erforderlichen Modellelemente, Zustandsüberführungsfunktionen und Outputfunktionen in Abhängigkeit der spezifischen Charakteristik des Anwendungsfalles zu definieren.

Auf Grundlage des Referenzmodells *PECS* lassen sich in flexibler Weise Modelle aus verschiedenen Anwendungsbereichen erstellen, die trotz signifikanter Unterschiede in der Modelldynamik doch auf einer sehr ähnlichen Tiefenstruktur aufsetzen können.

Eine Umsetzung der im Rahmen von *PECS* vorgeschlagenen Modellarchitektur kann in allen Systemen bzw. Sprachen erfolgen, die über einen modular-hierarchischen Aufbau verfügen und eine zustandsorientierte Dynamikbeschreibung ermöglichen. Derartige Eigenschaften weisen viele prozedurale oder auch objektorientierte Progammiersprachen wie etwa *C*, *C++* oder *Java* auf. Empfehlenswert ist jedoch der Einsatz moderner Simulationssysteme, wie etwa *Simplex3*, da derartige Systeme wesentlich bessere Unterstützung für die weitreichenden

Anforderungen bieten, die im Rahmen von Simulationsprojekten entstehen. Hierzu zählen insbesondere spezifische Sprachelemente für die Beschreibung von Zustandsübergängen, vordefinierte Laufzeitumgebungen, die die Berechung der Zustandsübergänge nach bewährten Verfahren übernehmen, Experimentierumgebungen, die die Durchführung von Modellexperimenten unterstützen, sowie Analyse-, Auswertungs- und Darstellungsfunktionalitäten, die im Rahmen der Ergebnisauswertung benötigt werden. Ähnliche Funktionalitäten, wenn auch bisher noch in begrenztem Umfang, bieten spezifische, agentenbasierte Sprachen bzw. Modellbibliotheken wie etwa *SDML* oder *Swarm*.

12.1.4 Nachweis der Praktikabilität

Die Qualität von Referenzmodellen zeigt sich in der Regel erst in der praktischen Anwendung. Um die Praxistauglichkeit und breite Einsetzbarkeit des Referenzmodells *PECS* nachzuweisen, wurden im Rahmen dieser Arbeit insgesamt vier Fallstudien realisiert, die unterschiedlichen Anwendungsgebieten entnommen sind.

Die Fallstudie *Evolution von Solidarnetzwerken* entspringt dem Bereich der Sozialforschung und beschäftigt sich auf der Grundlage eines spieltheoretischen Ansatzes mit der Frage, unter welchen Bedingungen stabile Kooperationsbeziehungen zwischen rational und egoistisch handelnden Agenten entstehen können.

Das *Marktmodell* setzt auf einer empirischen Untersuchung aus dem Bereich der experimentellen Ökonomie auf und demonstriert in grundsätzlicher Weise den Einsatz agentenbasierter Simulationsmodelle im Rahmen der Validierung wissenschaftlicher Theorien über das menschliche Verhalten.

Die Fallstudie *Modellierung von Gruppenprozessen* kann dem Bereich der Sozialpsychologie bzw. Kleingruppenforschung zugeordnet werden und beschäftigt sich mit dem Aufbau eines künstlichen Sozialsystems, in dem Agenten von Zeit zu Zeit Gruppen bilden, in Gruppen zusammenarbeiten und diese Gruppen nach einer gewissen Zeit wieder verlassen bzw. auflösen.

In der Fallstudie *Adam* wird das Zusammenspiel verschiedener Regulationsmechanismen des menschlichen Verhaltens auf der Grundlage eines agentenbasierten Ansatzes untersucht. Protagonist dieser Fallstudie ist der Agent *Adam*, der physische, emotionale und kognitive Aspekte, sowie reaktive und deliberative Verhaltensweisen im Rahmen einer *PECS*-Architektur vereint.

Die realisierten Fallstudien können sowohl hinsichtlich ihrer inhaltlichen Ausrichtung als auch hinsichtlich ihrer Komplexität als praxisrelevante Vertreter ihrer jeweiligen Forschungsgebiete angesehen werden. In allen vier Fällen wurden konkrete Instanzen des Referenzmodells *PECS* konzipiert und auf der Grundlage des Simulationssystems *Simplex3* implementiert. Die Erkenntnisse über die Praxistauglichkeit des Referenzmodells *PECS*, die aus der Realisierung der vier Fallstudien gewonnen wurden, sind im folgenden Abschnitt zur Bewertung der Ergebnisse dieser Arbeit enthalten.

12.1.5 Bewertung

Der Entwicklung des Referenzmodells *PECS* liegt die Zielsetzung zugrunde, den Entwurfsprozess agentenbasierter Simulationsmodelle zu unterstützen, in denen menschliche Akteure abgebildet werden. Mit *PECS* wird ein generischer Modellierungsrahmen vorgeschlagen, der anwendungsspezifisch instanziiert und mit Modellelementen sowie Dynamibeschreibungen versehen werden kann.

Im Gegensatz zu den komplexeren Architekturansätzen im Bereich der Künstlichen Intelligenz und angrenzenden Gebieten (s. Kapitel 11) verfügt das Referenzmodell *PECS* über eine klare Aufbaustruktur. *PECS* arbeitet mit einer Menge vordefinierter und abgrenzbarer Komponenten, die bei Bedarf über festgelegte Schnittstellenkonzepte miteinander interagieren. Der wohlstrukturierte Aufbau des Referenzmodells trägt dazu bei, dass die vorgestellten Konzepte für den Anwender gut verständlich sind und schnell operationalisiert werden können. Darüber hinaus wird durch den komponentenorientierten Modellaufbau sowie die für das Referenzmodell gewählte, funktionale Dekomposition eine sehr gute Anpassbarkeit der Modellarchitektur an die Gegebenheiten spezifischer Anwendungsfälle erreicht. Diese Aussage lässt sich insbesondere durch die realisierten Fallstudien belegen, die ganz unterschiedlichen Bereichen angehören und auch unterschiedliche Komplexität insbesondere in Bezug auf die Ausgestaltung der Agenten aufweisen, aber dennoch mit denselben Modellierungskonzepten bearbeitet werden konnten.

Weiterhin haben sich im Rahmen der Fallstudien sowohl die Auswahl der Funktionalitäten der Agenten sowie deren Anordnung im Rahmen der Agentenarchitektur als wirkungsvoll erwiesen. In keiner Fallstudie waren größere Anpassungen der Grundarchitektur von *PECS* erforderlich. Es mussten lediglich spezifische Komponenten etwa zur Aufsammlung statistischer Daten oder zur spezifischen Steuerung des Modellablaufes eingeführt werden. Aufgrund der gewählten Positionierung der internen Komponenten *Physis*, *Emotion*, *Cognition* und *Social Characteristics* im Verhältnis zu den restlichen Komponenten der Agentenarchitektur ist es in einfacher Weise möglich, die Ausstattung von *PECS*-Agenten an die Anforderungen spezifischer Anwendungsszenarien zu adaptieren. Spielen etwa im Rahmen einer Modelluntersuchung Emotionen keine Rolle, kann dieser Funktionsbereich innerhalb der Agentenarchitektur weggelassen werden, ohne die grundsätzliche Funktionsweise des Agenten dadurch zu beeinträchtigen.

Weiterhin konnte durch die Implementierung der Fallstudien nachgewiesen werden, dass der systemtheoretische Ansatz eine sehr komfortable Beschreibung des dynamischen Verhaltens von Agenten zulässt. Zeitkontinuierliche Zustandsübergänge bieten sich etwa in unmittelbarer und intuitiver Weise für die Beschreibung der dynamischen Veränderungen interner Zustände an. Zeitdiskrete Zustandsübergänge ermöglichen eine komfortable Beschreibung von Verhaltensregeln. Da *PECS*-Agenten ganz wesentlich durch interne Zustände, deren Veränderungen in der Zeit und diskrete Verhaltenbeschreibungen geprägt sind, kann der systemtheoretische Ansatz als nahezu ideal für die Spezifikation des dynamischen

Verhaltens von Agenten im Bereich der Modellbildung und Simulation angesehen werden.

Ausgehend von den in den Fallstudien gesammelten Erfahrungen lässt sich zusammenfassend festhalten, dass das Referenzmodell *PECS* sowohl aus Sicht der Modellstrukturierung als auch aus Sicht der Dynamikbeschreibung geeignete Konzepte anbietet, um die Entwicklung agentenbasierter Modelle, in denen menschliches Handeln, Entscheiden und Verhalten abgebildet werden, zu unterstützen. Das Referenzmodell *PECS* trägt also durch die angebotenen Konzepte dazu bei, die Komplexität der Entwurfsaufgabe, die mit der Entwicklung derartiger Simulationsmodelle unumgänglich verbunden ist, maßgeblich zu verringern. Das Ziel der vorliegenden Arbeit kann in dieser Hinsicht als erreicht betrachtet werden.

12.2 Perspektiven

Auf der Grundlage des Referenzmodells *PECS* und der bisher erzielten Ergebnisse eröffnen sich eine Reihe von Möglichkeiten für weiterführende Forschungsaktivitäten. Diese Perspektiven werden in den folgenden Abschnitten motiviert und bilden den Abschluss der vorliegenden Arbeit.

12.2.1 Architekturgetriebene Erforschung menschlicher Eigenschaften und Fähigkeiten

Disziplinen wie etwa die Wahrnehmungspsychologie, die Kognitionswissenschaften, die Emotionspsychologie, die Physiologie, die Sozialwissenschaften, sowie der gesamte Bereich der Handlungs- und Verhaltenstheorien streben danach, den Menschen mit seinen Eigenschaften und Fähigkeiten weiterführend zu erforschen. Sie arbeiten dazu in der Regel mit Theorien, die anhand von Experimenten aufgebaut bzw. überprüft und anschließend theoretisch diskutiert werden. Modellarchitekturen wie das Referenzmodell *PECS* können in diesem Zusammenhang einen wesentlichen Beitrag für eine Ausdehnung interdisziplinärer Forschungsaktivitäten leisten. Der grundlegende Ansatz besteht dabei darin, Theorien über menschliche Eigenschaften und Verhaltensweisen nicht nur auf einer theoretischen Ebene zu belassen, sondern im Rahmen von Agentenarchitekturen und entsprechenden Modellierungsansätzen zu operationalisieren. Durch die Entwicklung entsprechender Modelle können wesentlich präzisere Aussagen über die Grenzen und Leistungsfähigkeit der zugrunde gelegten Theorien getroffen werden. Ebenso lassen sich aus der Implementierung unterschiedlicher Theorien auch weiterführende Erkenntnisse über die Leistungsfähigkeit von Modellierungsansätzen und insbesondere über den Aufbau von Agentenarchitekturen gewinnen. Auf diese Weise

entsteht ein wechselseitiger Erkenntnisgewinn sowohl für die Informatik als auch für die einbezogenen Anwendungsdisziplinen.

Speziell für die Weiterentwicklung des Referenzmodells *PECS* erscheint ein derartig ausgerichteter, interdisziplinärer Forschungsansatz als reizvoll. Durch eine Betrachtung kognitionswissenschaftlicher, emotionspsychologischer, sozialwissenschaftlicher, physiologischer und handlungstheoretischer Ansätze können weiterführende Anforderungen erkundet werden, die in eine Präzisierung und Fortentwicklung des Referenzmodells *PECS* einfließen können. Im Mittelpunkt stehen dabei zunächst einmal die Komponenten der Agentenarchitektur, die in Anlehnung an die hinzugewonnenen Erkenntnisse funktional erweitert werden können. Darüber hinaus ist aber auch die Integration weiterer Komponenten in die Modellarchitektur zu erwarten, die bisher noch nicht vorhanden sind. Zu denken wäre hier etwa an sozialwissenschaftlich motivierte Komponenten, die Organisationen oder soziale Verbände repräsentieren oder auch weitere Komponenten, die die Agentenarchitektur ergänzen.

Eine interdisziplinäre Zusammenarbeit erscheint in diesem Zusammenhang als wertvoller Ansatz, um im Bereich der Informatik die Möglichkeiten, die agentenbasierte Modellierungsansätze bieten, weiter zu erforschen, und im Bereich der Sozialwissenschaften die Erforschung menschlicher Eigenschaften und Fähigkeiten weiter voranzutreiben.

12.2.2 Anwendungsperspektiven

Im Rahmen der vorliegenden Arbeit wurde der praktische Einsatz des Referenzmodells anhand von vier Fallstudien aus unterschiedlichen Anwendungsgebieten demonstriert. Die Erkenntnisse aus den Fallstudien ermutigen dazu, das Referenzmodell *PECS* weiterführend in der Praxis einzusetzen. Als Einsatzgebiete des Referenzmodells *PECS* eignen sich grundsätzlich alle realen und hypothetischen Systeme, in denen menschliches Handeln und Verhalten mit Hilfe von Simulationsmodellen untersucht werden sollen.

Sowohl an der Universität Passau als auch an anderen Instituten wird in der Zwischenzeit an Anwendungen des Referenzmodells *PECS* gearbeitet.

Im Rahmen eines Industrieprojektes untersucht der Lehrstuhl für Operations Research an der Universität Passau auf der Grundlage eines agentenbasierten Modellierungsansatzes die Durchführung von Friedenseinsätzen der Streitkräfte. Diese Thematik gewinnt im Bereich der militärischen Simulation immer mehr an Bedeutung und eignet sich in idealer Weise als Einsatzgebiet für das Referenzmodell *PECS*. Betrachtet man die Akteure in einem derartigen Friedenseinsatz-Szenario, so zeigt sich schnell, dass deren Verhalten durch physische, emotionale, kognitive und soziale Faktoren beeinflusst wird. Aus dem Bereich der Physis kommen etwa der Ernährungs- oder der Gesundheitszustand zum tragen. Das Verhalten der Soldaten sowie der Bewohner des modellierten Gebietes wird ebenso maßgeblich durch Emotionen wie beispielsweise Angst oder Wut bestimmt. Weiterhin werden

gruppendynamische Einflüsse, die dem sozialen Bereich zuzuordnen sind, als hochgradig relevant erachtet. Nicht zuletzt spielen das Wissen aller Beteiligten sowie weitere kognitive Aspekte eine ausschlaggebende Rolle für den Verlauf von Friedenseinsätzen und müssen in entsprechender Weise in Simulationsmodellen repräsentiert werden.

Ein weiteres Einsatzgebiet für das Referenzmodell *PECS* eröffnet sich im Bereich der industriellen Gruppenarbeit. Hier steht oftmals die zentrale Frage im Vordergrund, mit welchen Mitgliedern Projekt- oder Arbeitsgruppen besetzt werden müssen, um möglichst gute Ergebnisse aus der Gruppenarbeit zu erzielen. Ein konkretes Anwendungsbeispiel hierzu schildern Martínez-Miranda, Aldea und Bañares-Alcántara (2002). Ausgangpunkt dieser Studie ist die Team-Zusammensetzung bei Entwurf eines chemischen Prozesses. Agenten repräsentieren dabei die real verfügbaren Mitarbeiter und sind mit unterschiedlichen Fähigkeiten, Emotionen und Persönlichkeitseigenschaften ausgestattet.

Auch aus dem praktischen Einsatz des Referenzmodells *PECS* resultieren zumindest zwei Nutzeffekte. Zum einen fließen die Erkenntnisse, die aus den realen Anwendungen gewonnen werden, in den Entwurf der Modellarchitektur zurück. Zum anderen können die Vorgaben des Referenzmodells genutzt werden, um schneller und einfacher zu konkreten Modellen, die in der Praxis benötigt werden, zu kommen.

Auch im Bereich der Lehre können agentenbasierte Modelle gewinnbringend eingesetzt werden. Beispielsweise im Bereich der Psychologie lassen sich mit Hilfe von Simulationsmodellen Erkenntnisse über menschliche Fähigkeiten eindringlicher und präziser näher bringen als in herkömmlichen theoretischen Ansätzen. Im Bereich des *BMBF* (Bundesministerium für Bildung und Forschung) werden hierzu bereits erste Projekte, die sich mit der Nutzung agentenbasierter Methoden im Bereich der Lehre befassen, mit Drittmitteln ausgestattet (s. http://www.monist.de).

12.2.3 Entwicklung eines *PECS*-Frameworks

Das Referenzmodell *PECS* bietet dem Entwickler agentenbasierter Simulationsmodelle bislang Unterstützung auf konzeptioneller Ebene, indem es Richtlinien für die Strukturierung und die Dynamikbeschreibung von Modellen vorgibt. Zudem existieren Beispielmodelle, die als Vorlage für den Entwurf und die Ausgestaltung weiterer, agentenbasierter Modelle dienen können.

Im Bereich des Software-Engineering existiert der Ansatz der sog. *Software-Frameworks* (Balzert, 2000). Hierbei handelt es sich um anpassbare und erweiterbare Systeme kooperierender Klassen, die eine Implementierung eines wiederverwendbaren Entwurfes für einen bestimmten Einsatzbereich zur Verfügung stellen. Software-Frameworks legen die wesentlichen Entwurfsparameter von Anwendungen fest. Hierzu zählen insbesondere die Software-Architektur mit einer Menge von interagierenden Klassen und entsprechenden Schnittstellen, sowie

elementare Vorgaben für den Kontrollfluss innerhalb der Architektur. Der Vorteil von Frameworks besteht darin, dass der Entwickler von grundlegenden Entwurfsentscheidungen entbunden wird und sich nur um die Ausgestaltung der einzelnen Bestandteile der Architektur kümmern muss. Frameworks stellen also den Aspekt der Wiederverwendung von Architekturen in den Vordergrund.

Dieser Gedanke erscheint auch für den Bereich der agentenbasierten Modellbildung und insbesondere als Entwicklungsperspektive für das Referenzmodell *PECS* als sehr reizvoll. Durch die Bereitstellung eines entsprechenden *PECS*-Frameworks könnte dem Simulationsentwickler neben der Unterstützung auf konzeptioneller Ebene eine weiterführende Unterstützung auch auf der Ebene der Implementierung angeboten werden. Auf diese Weise lässt sich eine weitere Vereinfachung des Entwicklungsprozesses für agentenbasierte Simulationsmodelle erreichen.

Um ein derartiges Framework zu realisieren, stellen sich zwei grundlegende Aufgaben. Zum einen ist die Architektur und Ausgestaltung des Frameworks zu konzipieren. Hierzu ist festzulegen, welche Klassen mit welchen Aufgaben und Schnittstellen im Rahmen des Frameworks angeboten werden. In dieser Hinsicht liefert die Beschreibung des Referenzmodells *PECS* bereits wesentliche Hinweise.

Die zweite Aufgabe von grundlegender Bedeutung besteht in der Auswahl der technologischen Plattform für das *PECS*-Framework. Betrachtet man den aktuellen Stand der Simulationstechnik sowie der objektorientierten Programmierung, existieren in dieser Hinsicht zwei grundsätzliche Ansätze.

Der erste Ansatz besteht darin, die Entwicklung auf einem universell einsetzbaren Simulationssystem mit einer entsprechenden Modellbeschreibungssprache aufzusetzen. Dieser Ansatz bietet den Vorteil, dass im Rahmen eines derartigen Simulationssystems bereits alle grundlegenden Funktionalitäten vorhanden sind, die für die Durchführung von Simulationsstudien benötigt werden. Als technologische Plattform bieten sich in dieser Hinsicht das Simulationssystem *Simplex3* (Schmidt, 2000), das sich bereits im Rahmen der Fallstudien als sehr hilfreich herausgestellt hat, oder etwa auch das System *Swarm* (s. Kapitel 2) an, das bereits über einen Framework-Charakter verfügt. Beide Systeme eignen sich allerdings zum derzeitigen Entwicklungsstand nicht vollständig für die Realisierung eines *PECS*-Frameworks. Das Simulationssystem *Simplex3* kennt derzeit keinen Vererbungsmechanismus und keine dynamische Generierung von Komponenten. Das System *Swarm* ist ebenso wie die anderen agentenbasierten Simulationsansätze nicht in der Lage, zeitkontinuierliche Zustandsübergänge zu beschreiben. Um ein derartiges *PECS*-Framework aufzusetzen, müssten daher zunächst einmal grundlegende Eingriffe an der Konzeption der bestehenden Simulationssysteme und –bibliotheken vorgenommen werden.

Der zweite, grundlegende Ansatz bestünde daran, das *PECS*-Framework auf der Grundlage einer allgemeinen, objektorientierten Programmiersprache wie beispielsweise *Java* (s. http://java.sun.com) umzusetzen. Objektorientierte Sprachen stellen auf Grundlage der Vererbung einen wesentlichen Mechanismus für die Wiederverwendung von Klassen zur Verfügung. Allerdings bieten sie standard-

mäßig keinerlei Unterstützung im Bereich der Simulationstechnik, z. B. hinsichtlich der Ablaufsteuerung von Modellen oder der Berechnung von Zustandsübergängen.

Beide vorgeschlagenen Wege bieten also sowohl Vor- als auch Nachteile, die bei einer Realisierung eines *PECS*-Frameworks sicherlich noch detaillierter abgewogen werden müssen. Beide Wege sind auch mit einem relativ hohen Entwicklungsaufwand verbunden. Allerdings wird ein derartiges Framework deutlich zur Akzeptanz des Referenzmodells beitragen können und eine wesentliche Vereinfachung und Zeitersparnis für den Entwickler agentenbasierter Simulationsmodelle bieten.

12.2.4 Steuerung virtueller Agenten

Avatare bzw. virtuelle Agenten repräsentieren Menschen in Virtual Reality-Szenarien und verfügen über realitätsnahe Bewegungen und Gesichtsausdrücke. Die aktuelle Forschung in diesem Bereich beschäftigt sich etwa mit Methoden zur Erfassung geometrischer und kinematischer Daten, zur Generierung realistischer, menschlicher Erscheinungsformen und Bewegungen, sowie zur Steuerung des Verhaltens autonomer Avatare (Magnenat-Thalmann & Thalmann, 2000).

Zur Verhaltenssteuerung autonomer Avatare können Agenten eingesetzt werden. Sie stellen in diesem Zusammenhang einen Teil der Gesamtarchitektur von Avataren dar und übernehmen die Berechnung des internen Zustands der Avatare sowie die Generierung entsprechender Verhaltensweisen.

Aus Sicht der Agententechnologie erscheinen in diesem Zusammenhang zumindest zwei Aspekte als sehr interessant: Zum einen bieten Avatare die Möglichkeit, simulierte Agenten mit einem realitätsnahen Aussehen auszustatten. Zum anderen stellen Avatare, sofern sie über autonome Verhaltensweisen verfügen, sehr hohe Anforderungen an die Verhaltenssteuerung. Zu den extern sichtbaren Verhaltensweisen eines Avatars gehören neben der Veränderung der Gestik und Mimik auch entsprechende Anpassungen der Körperhaltung und Bewegungen. Es entstehen also parallel auszuführende Aktivitäten, die in Abhängigkeit des internen Zustands des Avatars zu koordinieren sind und parallel gesteuert werden müssen. Die Entwicklung autonomer Avatare eröffnet also ein breites Feld für interdisziplinäre Forschungsaktivitäten.

Ich hoffe, dass die vorliegende Arbeit mit den diskutierten Modellierungsansätzen und Fallstudien dazu beitragen kann, die Simulationstechnik als probates Mittel zur Erforschung menschlicher Eigenschaften und Fähigkeiten weiterführend zu etablieren. Die angedeuteten Perspektiven sollen sowohl Informatiker als auch Sozialwissenschaftler dazu ermutigen, die Grenzen ihrer eigenen Disziplinen zu überschreiten und im Rahmen interdisziplinärer Forschungsprojekte sowohl die Simulationstechnik als auch die Sozialwissenschaften weiter voranzubringen.

LITERATURVERZEICHNIS

Adler, M., Durfee, E., Huhns, M., Punch, W., Simoudis, E. (1992). AAAI Workshop on Cooperation Among Heterogeneous Intelligent Agents. *AI Magazine 13 (2)*. 39-42.

Aebli, H. (1981). *Denken: das Ordnen des Tuns*, Bd. I und II. Stuttgart: Klett-Cotta.

Agre, P. E., Chapman, D. (1987). Pengi: An Implementation of a Theory of Activity. In *Proceedings of AAAI-87*. Los Altos: Morgan-Kaufmann. 268-272.

Albayrak, S., Bussmann, S. (1993). Kommunikation und Verhandlungen in Mehragenten-Systemen. In Müller, J. (Hrsg.). *Verteilte Künstliche Intelligenz: Methoden und Anwendungen*. Mannheim, Leipzig, Wien, Zürich: BI Wissenschaftsverlag. 55-91.

Alexander, J. C. (1987). *The micro-macro link*. Berkeley, London: University of California Press.

Allen, J., Hendler, J, Tate, A. (1990). *Readings in Planning*. San Mateo, CA: Morgan Kaufmann.

Allport, G. W. (1937). *Personality: A psychological interpretation.* New York: Holt, Rinehart & Winston.

Allport, G. W. (1966). Traits revisited. *American Psychologist, 21*, 1-10.

Alvarado, N., Adams, S. S., Burbeck, S., Latta, C. (2001). Integrating Emotion and Motivation into Intelligent Systems. Poster presented at the Workshop *Emotional and Intelligent II: The Tangled Knot of Social Cognition.* AAAI Fall Symposium, December 2001, North Falmouth, USA.

Anderson, J. R. (1996). *Kognitive Psychologie: eine Einführung*, 2. Auflage. Heidelberg: Spektrum Akademischer Verlag.

Anderson, J. R. (2003). *Homepage der ACT-R Research Group*. Carnegie Mellon University. URL: http://act-r.psy.cmu.edu/

Averill, J. R. (1968). Grief: Its nature and significance. *Psychological Bulletin*, 70. 721-748.

Balzert, H. (2000). *Lehrbuch der Software-Technik: Software-Entwicklung*. 2. Auflage. Heidelberg, Berlin: Spektrum Akademischer Verlag.

Bates, J., Loyall, A. B., Reilly, W. S. (1991). Broad Agents. *SIGART BULLETIN 2 (4)*, August 1991. 38-40

Bates, J., Loyall, A. B., Reilly, W. S. (1992). An Architecture for Action, Emotion, and Social Behaviour. In Castelfranchi, C., Werner, E. (Eds.). *Artificial Social Systems*. Berlin: Springer. 55-68.

Bergius, R. (1998). Motivation, Motivationsforschung. In Häcker, H., Stapf, K. H. (Hrsg.). *Dorsch Psychologisches Wörterbuch*, 13. Auflage. Bern, Göttingen, Toronto, Seattle: Verlag Hans Huber. 550-553.

Bergius, R. (1998a). Norm(en), soziale. In Häcker, H., Stapf, K. H. (Hrsg.). *Dorsch Psychologisches Wörterbuch*, 13. Auflage. Bern, Göttingen, Toronto, Seattle: Verlag Hans Huber. 584.

Bierbaumer, N., Jänig, H. (1997). Motivation und Emotion. In Schmidt, R. F., Thews, G. (Hrsg.). *Physiologie des Menschen*, 27. Auflage. Berlin: Springer. 167-183.

Biundo, S., Günter, A., Hertzberg, J., Schneeberger, J., Tank, W. (1995). Planen und Konfigurieren. In Görz, G. (Hrsg.). *Einführung in die Künstliche Intelligenz*. 2. Auflage. Bonn: Addison-Wesley. 754-798.

Brandstätter, V., Gollwitzer, P. M. (1996). Motivation. In Strube, G., Becker, B., Freksa, C., Hahn, U., Opwis, K., Palm, G. (Hrsg.). *Wörterbuch der Kognitionswissenschaft*. Stuttgart: Klett-Cotta. 411-413.

Brauer, W., Malsch, Th., Rammert, W. (1999). *Sozionik: Erforschung und Modellierung künstlicher Sozialität*. http://www.tu-harburg.de/tbg/SPP/spp-antrag.html, (August 1999).

Brenner, W., Zarnekow, R., Wittig, H. (1998). *Intelligente Softwareagenten: Grundlagen und Anwendungen*. Berlin, Heidelberg, New York: Springer-Verlag.

Brooks, R. A. (1991). Integrated systems based on behaviours. *SIGART Bulletin*, 2 (4). 46-50.

Burkhardt, R. (1997). *UML - Unified Modeling Language*. Bonn, Reading: Addison-Wesley.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns*. Chichester, New York: John Wiley & Sons.

Busemann, S. (1995). Sprachverarbeitung. In Görz, G. (Hrsg.). *Einführung in die Künstliche Intelligenz*. 2. Auflage. Bonn: Addison-Wesley. 361-557.

Caldas, J. C., Coelho, H. (1992). Strategic Interaction in Oligopolistic Markets – Experimenting with Real and Artificial Agents. In Castelfranchi, C., Werner, E. (Eds.). *Artificial Social Systems.* Berlin: Springer. 147-163.

Doran, J. E., Palmer, M., Gilbert, N., Mellars, P. (1994). The EOS project: modelling Upper Paleolithic Social Change. In Gilbert, N., Doran, J. E. (eds.). *Simulating Societies: The Computer Simulation of Social Phenomena.* London: UCL Press. 195-222.

Dörner, D., Selg, H. (1996). *Psychologie: Eine Einführung in ihre Grundlagen und Anwendungsfelder*. Stuttgart, Berlin, Köln: Kohlhammer.

Dörner, D. (1999). *Bauplan für eine Seele*. Reinbek bei Hamburg: Rowohlt.

Dreschler-Fischer, L. (1995). Bildverstehen. In Görz, G. (Hrsg.). *Einführung in die Künstliche Intelligenz*. 2. Auflage. Bonn: Addison-Wesley. 559-702.

Drogoul, A., Ferber, J. (1994). Multi-agent simulation as a tool for studying emergent processes in societies. In Doran, J. E., Gilbert, N. (eds.). *Simulating Societies: The Computer Simulation of Social Phenomena.* London: UCL Press. 127-142.

Drogoul, A., Corbara, B., Lalande, S. (1995). Manta: new experimental results on the emergence of (artificial) ant societies. In Gilbert, N., Conte, R. (eds.). *Artificial Societies: The Comuter Simulation of Social Life.* London: UCL Press.

Edwards, W. (1954). The theory of decision making. *Psychological Bulletin*, 51. 380-417.

Ellgring, H. (1990). Mensch-Umwelt-Beziehungen. In Pöppel, E., Bullinger, M., Härtel, U. (Hrsg.). *Medizinische Psychologie und Soziologie*. Weinheim: Chapman & Hall. 219-258.

Epstein, J. M., Axtell, R. (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. Washington, DC: Brookings Institution Press.

Erman, L. D., Hayes-Roth, F., Lesser, V. R., Reddy, D. R. (1980). The Hearsey-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys*, 2 (2). 213-253.

Eschenbacher, P. (1995). *Konzeption einer deklarativen und zustandsorientierten Sprache zur formalen Beschreibung und Simulation von Warteschlangen- und Transportmodellen.* Erlangen, Ghent, Istanbul, San Diego: SCS – Society for Computer Simulation Int.

Eysenck, H. J. (1973). *The inequality of man.* London: Temple Smith.

Eysenck, H. J. (1990). Biological dimensions of personality. In Pervin, L. S. (Ed.). *Handbook of personality theory and research.* New York: Guilford Press. 244-276.

Ferguson, I. A. (1992). *Touring Machines: An architecture for dynamic, rational, mobile agents.* PhD thesis, University of Cambridge Computer Laboratory, University of Cambridge. Technical report No. 273.

Festinger, L. (1957). *A theory of cognitive dissonance*. Stanford: Stanford University Press.

Firby, R. J. (1989). *Adaptive execution in complex dynamic worlds*. PhD thesis, Department of Computer Science, Yale University.

Freksa, C. (1996). Künstliche Intelligenz. In Strube, G., Becker, B., Freksa, C., Hahn, U., Opwis, K., Palm, G. (Hrsg.). *Wörterbuch der Kognitionswissenschaft*. Stuttgart: Klett-Cotta. 344-347.

Friedman, J. W. (1986). *Game theory with applications to economics.* 2nd. Edition. Oxford: Oxford University Press.

Frijda, N. H. (1986). *The emotions*. Cambridge: Cambridge University Press.

Gamma, E., Helm, R., Johnson, R., Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading: Addison-Wesley.

Genesereth, M. R., Nilsson, N. (1987). *Logical Foundations of Artificial Intelligence.* Los Altos: Morgan Kaufmann.

Gilbert, N., Troitzsch, K. (1999). *Simulation for the Social Scientist*. Buckingham, Philadelphia: Open University Press.

Gollwitzer, P. M. (1990). Action phases and mind-sets. In Higgins, E. T., Sorrentino, R. M. (Editors). *Handbook of motivation and cognition: Foundations of social behavior.* Volume 2. New York: Guilford Press. 53-92.

Gollwitzer, P. M. (1991). *Abwägen und Planen.* Göttingen: Hogrefe.

Görz, G., Wachsmuth, I. (1995). Einleitung. In Görz, G. (Hrsg.). *Einführung in die Künstliche Intelligenz*. 2. Auflage. Bonn: Addison-Wesley. 1-13.

Harrer, A. G. (1999). Group Learning in Hybrid Communities in Intelligent Tutoring Systems. In Lajoie, S., Vivet, M. (Editors). *Artificial Intelligence in Education: Open Learning Environments, New Computational Technologies to Support Learning, Exploration and Collaboration*. Amsterdam: IOS Press. 31-38.

Häcker, H., Stapf, K. H. (Hrsg.) (1998). *Dorsch Psychologisches Wörterbuch*, 13. Auflage. Bern, Göttingen, Toronto, Seattle: Verlag Hans Huber.

Heckhausen, H. (1989). *Motivation und Handeln*. Berlin: Springer.

Hertzberg, J. (1989). *Planen. Einführung in die Planerstellungsmethoden der Künstlichen Intelligenz*. Mannheim: BI-Wissenschaftsverlag.

Hegselmann, R., Flache, A. (1997). Rational vs. adaptive egoism in support networks – How different micro foundations shape different macro hypotheses. In: Leinfellner, W., Köhler, E. (Editors). *Game Theory, Experience, Rationality*. Dordrecht: Kluwer. 261-275

Hegselmann, R., Flache, A. (1998). Understanding Complex Social Dynamics. A Plea For Cellular Automata Based Modelling. In: Journal of Artificial Societies and Social Simulation. Vol. 1, No. 3, http://www.soc.surrey.ac.uk/JASSS/1/3/1.html

Herzog, O., Kirn, S., Krallmann, H., Spaniol, O., Zelewski, S. (1999). Intelligente Softwareagenten und betriebswirtschaftliche Anwendungsszenarien. http://www.wirtschaft.tu-ilmenau.de/wi/wi2/SPP-Agenten/SPP.html, (August 1999).

Huhns, M. N. (1994). Foreword. In Singh, M. P. (Ed.). *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications*. Lecture Notes in Artificial Intelligence (799). Berlin: Springer-Verlag. 7-12.

Jäger, R. (1996). Verhalten. In Strube, G., Becker, B., Freksa, C., Hahn, U., Opwis, K., Palm, G. (Hrsg.). *Wörterbuch der Kognitionswissenschaft*. Stuttgart: Klett-Cotta. 762.

Jungermann, H., Pfister, R., Fischer, K. (1998). *Die Psychologie der Entscheidung*. Heidelberg: Spektrum-Verlag.

Keller, B. (1999). Arbeits-, Betriebs- und Organisationspsychologie: Die Arbeitswelt gestalten. In Zimbardo, P. G., Gerrig, R. J. *Psychologie*. 7., neu übersetzte und bearbeitete Auflage. Berlin: Springer.

Klinger, A. (1999). *Referenzmodelle für die Abbildung von Personalsteuerung in der Simulation*. Delft, Erlangen, Ghent, San Diego: SCS – European Publishing House.

Klusch, M. (1998). *Kooperative Informationsagenten im Internet*. Schriftenreihe Forschungsergebnisse zur Informatik (36). Hamburg: Verlag Dr. Kovač.

Kraus, C. (1999). *Agentenbasierte Modellierung von Gruppenprozessen. Eine Anwendung des Referenzmodells PECS.* Diplomarbeit im Fach Informatik. Lehrstuhl für Operations Research und Systemtheorie, Fakultät für Mathematik und Informatik, Universität Passau.

Langton, C., Minar, N., Burkhart, R., Askenazi, M. (1996). *The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations*. http://www.swarm.org, (Januar 2001).

Laird, J. E., Newell, A., Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence* (33). 1-64.

Lazarus, R. S. (1991). Cognition and motivation in emotion. In *American Psychologist*, April 1991.

Madsen, K. B. (1974). *Modern Theories of Motivation*. Kopenhagen: Munksgaard.

Maes, P. (1995). Artificial Life Meets Entertainment: Life Like Autonomous Agents. *Communications of the ACM 38 (11)*. 108-114.

Magnenat-Thalmann, N., Thalmann, D. (2000). Preface. In Magnenat-Thalmann, N., Thalmann, D. (Eds.). *Deformable Avatars*. Boston, Dordrecht, London: Kluwer Academic Publishers. 9-10.

Malsch, Th. (1997). Die Provokation der „Artificial Societies". In *Zeitschrift für Soziologie*, Jg. 26, Heft 1. 3-21.

Martínez-Miranda, J., Aldea A., Bañares-Alcántara, R. (2002). A Social Agent Model to Simulate Human Behaviour in Teamwork. In Urban, C. (Ed.). *3rd Workshop on Agent-Based Simulation*. Proceedings. Delft, Erlangen, Ghent, San Diego: SCS European Publishing House. 18-23.

Maslow, A. H. (1955). Deficiency motivation and growth motivation. In Jones, M. R. (Ed.). *Nebraska symposium on motivation*. Lincoln: University of Nebraska Press.

McCarthy, J., Hayes, P. J. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4, 1969.

McClelland, D. C. (1965). Toward a theory of motive acquisition. *American Psychologist*, 25. 321-333.

Mellars, P. (1985). The ecological basis of social complexity in the Upper Paleolithic of southwestern France. In Douglas-Price, T., Brown, J. A. (eds.). *Prehistoric Hunter-Gatherers: The Emergence of Cultural Complexity*. New York: Academic Press. 271-297.

Meyer, W. U., Schützwohl, A., Reisenzein, R. (1993). *Einführung in die Emotionspsychologie*, Band 1. Bern, Göttingen, Toronto, Seattle: Verlag Hans Huber.

Meyer-Fujara, J., Puppe, F., Wachsmuth, I. (1995). Expertensysteme und Wissensmodellierung. In Görz, G. (Hrsg.). *Einführung in die Künstliche Intelligenz*. 2. Auflage. Bonn: Addison-Wesley. 705-753.

Minsky, M. L. (1986). *The Society of Mind*. New York: Simon & Schuster.

Moffat, D., Frijda, N. H. (1995). Where there's a Will there's an Agent. In Wooldridge, M. J., Jennings, N. R. (Eds.). *Intelligent Agents*. Lecture Notes in Artificial Intelligence 890. Berlin: Springer-Verlag. 245-260.

Moffat, D. (1997). Personality Parameters and Programs. In Trappl, R., Petta, P. (eds.). *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*. Berlin: Springer. 120-165.

Mogel, H. (1996). Ökopsychologie. In Dörner, D., Selg, H. (Hrsg). *Psychologie: Eine Einführung in ihre Grundlagen und Anwendungsfelder*. Stuttgart, Berlin, Köln: Kohlhammer. 264-281.

Mogel, H., Ohler, P., Urban, C. (2000). Psychologie und Informatik – Wie können diese beiden Disziplinen voneinander profitieren? In: Rimane, R. (Hrsg.). *Gedanken zur Zeit: Festschrift anläßlich des 60. Geburtstages von Herrn Prof. Dr. Bernd Schmidt*. Erlangen: SCS - European Publishing House. 159-188.

Mosler, H.-J. (2002). Multi agent simulations of social psychological theories. Method, results, and application. In. Urban, C. (Ed.). *3rd Workshop on Agent-Based Simulation.* Erlangen: SCS-European Publishing House. 24-31.

Mosler, H.-J., Brucks, W. (2001). Social Influence Among Agents. The Simulation of Social Psychological Theories. In Saam, N. J., Schmidt, B. (Eds.). *Cooperative Agents. Applications in the Social Sciences.* Series A: Philosophy and Methodology of the Social Sciences. Dordrecht: Kluwer Academic Publishers. 125-147.

Moss, S., Gaylard, H., Wallis, S., Edmonds, B. (2001). *SDML: A Multi-Agent Language for Organizational Modelling*. http://www.cpm.mmu.ac.uk, (Januar 2001).

Müller, H. J. (1996). Agent. In Strube, G., Becker, B., Freksa, C., Hahn, U., Opwis, K., Palm, G. (Hrsg.). *Wörterbuch der Kognitionswissenschaft*. Stuttgart: Klett-Cotta. 17.

Müller, J. P. (1998). The Right Agent to Do the Right Thing. In: Müller, J. P., Singh, M. P., Rao, A. (Eds.). *Intelligent Agents V: Agent Theories, Architectures and Languages.* 5th International Workshop, ATAL '98, Paris, France, July 4-7, 1998. Proceedings. Berlin: Springer. 211-225.

Newell, A., Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs: Prentice-Hall.

Oatley, K., Johnson-Laird, P. N. (1987). Towards a cognitive theory of emotions. In *Cognition and Emotion*, 1. 29-50.

Opwis, K. (1996). Ziel. In Strube, G., Becker, B., Freksa, C., Hahn, U., Opwis, K., Palm, G. (Hrsg.). *Wörterbuch der Kognitionswissenschaft*. Stuttgart: Klett-Cotta. 830.

Öhman, A. (1987). The psychophysiology of emotion: An evolutionary-cognitive perspective. In Ackles, P. K., Jennings, J. R., Coles, M. G. H. (Eds.): *Advances in psychology*, Vol. 2. Greenwich, CT: JAI Press. 79-127.

Pichler, F. (1975). *Mathematische Systemtheorie, Dynamische Konstruktionen*. Berlin: De Gruyter.

Plötzner, R. (1996). Lernen. In Strube, G., Becker, B., Freksa, C., Hahn, U., Opwis, K., Palm, G. (Hrsg.). *Wörterbuch der Kognitionswissenschaft*. Stuttgart: Klett-Cotta. 361-362.

Plutchik, R. (1980). *Emotion: A psychoevolutionary synthesis*. New York: Harper & Row.

Plutchik, R. (1984). Emotions: A general psychoevolutionary theory. In Scherer, K. R., Ekman, P. (Eds.). *Approaches to emotion*. Hillsdale, NJ: Erlbaum. 197-219.

Plutchik, R. (1993). Emotions and their vicissitudes: Emotions and psychopathology. In Lewis, M., Haviland, J. M. (Eds.). *Handbook of emotions.* New York: Guilford Press. 53-66.

Plutchik, R. (1994). *The psychology and biology of emotion*. New York: Harper Collins College Publishers.

Pschyrembel, W. (1985). *Pschyrembel Klinisches Wörterbuch*. Berlin, New York: De Gruyter.

Puppe, F. (1996). Vorwärtsverkettung. In Strube, G., Becker, B., Freksa, C., Hahn, U., Opwis, K., Palm, G. (Hrsg.). *Wörterbuch der Kognitionswissenschaft*. Stuttgart: Klett-Cotta. 773.

Quatrani, T. (1998). *Visual Modeling With Rational Rose and UML*. Reading, Harlow: Addison-Wesley.

Rammert, W. (1998). Giddens und die Gesellschaft der Heinzelmännchen: Zur Soziologie technischer Agenten und Systeme Verteilter Künstlicher Intelligenz. In Malsch, T. (Hrsg.). *Sozionik: Soziologische Ansichten über künstliche Sozialität*. Berlin: Edition Sigma. 91-128.

Rao, A. S., Georgeff, M. P. (1991). An abstract architecture for rational agents. In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning*. Boston.

Rapaport, A., Chammah, A. M. (1965). *Prisoner's dilemma*. Ann Arbor: University of Michigan Press.

Russel, S. J., Norvig, P. (1995). *Artificial Intelligence. A Modern Approach.* Upper Saddle River, New Yersey: Prentice Hall.

Schachter, S., Singer, J. E. (1962). Cognitive, social and physiological determinants of emotional states. In *Psychological Review*, 69. 379-399.

Schmidt, B. (1994). Simulation und objektorientierte Programmierung. In Biethahn, J., Hummeltenberg, W., Schmidt, B., Witte, T. (Hrsg.). *Simulation als betriebliche Entscheidungshilfe*. Fortschritte in der Simulationstechnik, Band 8. Braunschweig, Wiesbaden: Vieweg.

Schmidt, B. (1996). Referenzmodelle. In *SiP – Simulation in Passau*, Heft 2/1996.

Schmidt, B. (2000). *Einführung in die Simulation mit Simplex3*. Erlangen: SCS – European Publishing House.

Schmidt, B. (2000a). *Die Modellierung menschlichen Verhaltens*. Delft, Erlangen, Ghent, San Diego: SCS – European Publishing House.

Schwarz, G. (2000). *Der autonome Agent Adam. Modellierung unterschiedlicher Kontrollmechanismen menschlichen Verhaltens*. Diplomarbeit im Fach Informatik. Lehrstuhl für Operations Research und Systemtheorie, Fakultät für Mathematik und Informatik, Universität Passau.

Sibbel, R., Urban, C. (2000). Perspectives of Agent-Based Modeling and Simulation for Hospital Management. In Urban, C. (Ed.). *Agent-Based Simulation*. Erlangen: SCS – European Publishing House. 77-82.

Sloman, A. (1996). *What sort of architecture is required for a human-like agent?* Invited talk at Cognitive Modeling Workshop AAAI 96. Portland, Oregon.

Sloman, A., Scheutz, M. (2002): A Framework for Comparing Agent Architectures. In *Proceedings UKCI '02: UK Workshop on Computational Intelligence*. Sept. 2002, Birmingham UK, http://www.cs.bham.ac.uk/~ukci.

Sloman, A., Chrisley, R., Scheutz, M. (2003). The Architectural Basis of Affective States and Processes. In Fellous, Arbib (eds.). *Who needs emotions?: The Brain Meets the Machine*. Oxford: Oxford University Press (im Erscheinen).

Strube, G. (1996). Emergenz. In Strube, G., Becker, B., Freksa, C., Hahn, U., Opwis, K., Palm, G. (Hrsg.). *Wörterbuch der Kognitionswissenschaft*. Stuttgart: Klett-Cotta. 139.

Sundermeyer, K. (1993). Modellierung von Agentensystemen. In Müller, J. (Hrsg.). *Verteilte Künstliche Intelligenz: Methoden und Anwendungen*. Mannheim, Leipzig, Wien, Zürich: BI-Wissenschaftsverlag. 22-44.

Süß, M. (1999). *Rationaler Egoismus in Solidarnetzwerken. Entwurf und Implementierung eines Simulationsmodells auf Basis des Referenzmodells PECS und einer Animation in Java.* Diplomarbeit im Fach Informatik. Lehrstuhl für Operations Research und Systemtheorie, Fakultät für Mathematik und Informatik, Universität Passau.

Tack, W. (1996). Entscheidungstheorie. In Strube, G., Becker, B., Freksa, C., Hahn, U., Opwis, K., Palm, G. (Hrsg.). *Wörterbuch der Kognitionswissenschaft*. Stuttgart: Klett-Cotta. 144-145.

Taylor, M. (1987). *The possibility of cooperation.* London: Wiley & Sons.

Tolman, E. C. (1948). Cognitive maps in rats and men. *Psychological Review*, 55. 189-208

Ulmer, H.-V. (1997). Arbeits- und Sportphysiologie. In Schmidt, R. F., Thews, G. (Hrsg.). *Physiologie des Menschen*, 27. Auflage. Berlin: Springer. 167-183.

Urban, C. (2000). PECS: A Reference Model for the Simulation of Multi-Agent Systems. In Suleiman, R., Troitzsch, K. G., Gilbert, N. (Eds.). *Tools and Techniques for Social Science Simulation.* Heidelberg, New York: Physica-Verlag. 83-114.

VDI (Hrsg.). (1997). *Richtlinie 3633 – Simulation von Logistik-, Materialfluß- und Produktionssystemen – Begriffsdefinitionen. (Entwurf)*. Berlin: Beuth. 14.

Velásquez, J. D. (1997). Modeling Emotions and Other Motivations in Synthetic Agents. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. Providence: MIT / AAAI Press.

Velásquez, J. D. (1998).Modeling Emotion-Based Decision-Making. In Canamero, L. (Chair). *Emotional and Intelligent: The Tangled Knot Of Cognition.* Papers from the 1998 AAAI Fall Symposium, October 23-25, Orlando, Florida. Technical Report FS-98-03. Menlo Park: AAAI Press. 164-169.

Von Neumann, J. & Morgenstern, O. (1944). *Theory of games and economic behaviour*. Princeton: Princeton University Press.

Wallace, S. A., Laird, J. E. (1998). Toward a Methodology for AI Architecture Evaluation: Comparing Soar and CLIPS. In: Jennigs, N. R., Lespérance, Y. (Eds.). *Intelligent Agents VI: Agent Theories, Architectures and Languages.* 6th International Workshop, ATAL '99, Orlando, Florida, USA, July 15-17, 1999. Proceedings. Berlin: Springer. 117-131

Weppner, H. (1998). *Individuenbasierte Simulation eines oligopolistischen Marktes auf Basis des Referenzmodells PECS.* Diplomarbeit im Fach Informatik. Lehrstuhl für Operations Research und Systemtheorie, Fakultät für Mathematik und Informatik, Universität Passau.

Wooldridge, M. J., Jennings, N. R. (1995). Agent Theories, Architectures, and Languages: A Survey. In Wooldridge, M. J., Jennings, N. R. (Eds.). *Intelligent Agents.* Lecture Notes in Artificial Intelligence 890. Berlin: Springer-Verlag. 1-39.

Zapf, D. (1996). Handlung, Handlungstheorie. In Strube, G., Becker, B., Freksa, C., Hahn, U., Opwis, K., Palm, G. (Hrsg.). *Wörterbuch der Kognitionswissenschaft*. Stuttgart: Klett-Cotta. 245-246.

Zimbardo, P. G., Gerrig, R. J. (1999). *Psychologie*. 7., neu übersetzte und bearbeitete Auflage. Berlin: Springer.