Regelbasierte Simulation und Agentensysteme

Jens Müller
Seminar ViCCC
11. Dezember 2007
Überarbeitete Version, Februar 2008

***Zusammenfassung* — Diese Ausarbeitung bietet einen Überblick über Agenten und ihr Einsatz in der regelbasierten Simulation, insbesondere im Hinblick auf den Einsatz bei der Simulation von Bauunternehmen im Studienprojekt ViCCC.**

I. Einleitung

AGENTEN kombinieren die Flexibilität der sozialen Interaktion mit den Vorteilen verteilter und konkurrierender Problemlösung. Damit sind sie als Modell zur Lösung von Problemen, bei denen mehrere verschiedene Verfahren, Einheiten und Perspektiven involviert sind, geeignet.

Dies wird beispielsweise in Modellen von Wirtschaftsmärkten sichtbar, wie an der Börse, wo der Zusammenfall vieler einzelner Entscheidungen Effekte hat, die von einer einzelnen Einheit nicht vorhergesagt werden kann. Betrachtet man die Aufgabe ganz Stuttgart täglich mit frischem Brot zu versorgen, sieht man, dass diese für ein zentrales Planungsmodul nicht leicht ist (wie die Erfahrung mit der Planwirtschaft es zeigt). Das System aus Bäckergeschäften (Agenten), mit jeweils begrenztem Wissen, die völlig unabhängig zielorientiert handeln und nach eigener Optimierung streben, funktioniert hingegen täglich. Die Konkurrenz reinigt das System von den Bäckern, welche die Wünsche der Kunden weniger gut erfüllen. Die Kombination aus verteilter agentenbasierter Architektur mit Anpassungszwängen ist also sehr mächtig und kann benutzt werden, um tatsächliche Probleme zu lösen oder die Wirklichkeit zu modellieren [6]. Mengen von Agenten können weiterhin konkret eingesetzt werden bei der Simulation von Massenverhalten, bei der Planung von Fluchtwegen in Gebäuden oder bei der Verkehrsplanung.

II. Agenten

Ein Agent (von *Agere*, lat: tun, machen, handeln) ist alles, was seine Umgebung über Sensoren wahrnimmt und diese über Aktuatoren beeinflusst [4]. Der Mensch als Agent hat als Sensoren beispielsweise seine Sinnesorgane und als Aktuatoren seine Stimme, Hände, Füße, etc. Ein Software-Agent kann beispielsweise Netzwerkpakete als sensorische Eingabe haben und kann als Aktuator eine Bildschirmausgabe haben.

Ein Agent, der sich so verhält, dass er das beste zu erwartende Ergebnis erstrebt, wird *rationaler Agent* genannt. Dafür müssen wir ein Erfolgskriterium definieren, also Eigenschaften, die die Umgebung am Ende haben soll. Meist geschieht dies über eine Leistungsbewertung, die maximiert werden soll. Bei einem Agenten, der ein Taxifahrer darstellt, kann diese aus der Dauer der Fahrt, dem Verbrauch, der Gefährdung anderer Verkehrsteilnehmer und dem Wert des Taxameters bestehen. Rationalität heißt hier aber nicht Allwissenheit und damit Perfektion - was in der Realität kaum zu erreichen ist - sondern es muss berücksichtigt werden, welches Vorwissen der Agent überhaupt hat und welches neue Wissen er sich durch Aktionen aneignen könnte und wie er dieses Wissen zu nutzen weiß. Es scheint äußerst rational wenn der Taxifahrer bremst, wenn das Auto auf eine Wand zufährt; eventuell ist die Fahrbahn aber glatt und es wäre besser gewesen auszuweichen statt durch das Bremsen manövrierunfähig gegen die Wand zu rutschen. Die Intelligenz der Menschen beginnt da, wo die Rationalität der Agenten aufhört [4], nämlich wenn der Agent auf Probleme trifft, für die keine Regeln oder passende Modelle existieren.

Die Spezifikation der Umgebung eines Agenten geschieht also durch die Festlegung der Sensoren, Aktuatoren, externen Umgebung und des Erfolgskriteriums (*PEAS: Performance Measure, Environment, Actuators and Sensors*).

Dazu zwei Beispiele (nach [7]):

TABLE I
Beispiele für Agenten

Agententyp:	**Taxifahrer**
Leistungsbewertung:	Vom Fahrgast gewünschtes Ziel sicher und schnell erreichen, dabei der Straßenverkehrsordnung gehorchen, eine angenehme Fahrweise und maximale Gewinne einfahren.
Umgebung:	Straßen, Verkehr, Fußgänger, Fahrgäste
Aktuatoren:	Lenkrad, Bremse, Blinker, Hupe, Pedale, Sprachkommunikation mit Fahrgast
Sensoren:	Kameras, Abstandsmesser, Tachometer, GPS Kilometerzähler, Motorsensoren, Tastatur
Agententyp:	**Interaktiver Englischlehrer**
Leistungsbewertung:	Maximale Punktzahl des Schülers beim Abschlusstest
Umgebung:	Mehrere Schüler. Lehrer, der Prüfung erstellt.
Aktuatoren:	Anzeigen von Übungen und Korrekturen, Aussprachhilfe
Sensoren:	Tastatureingabe, Spracherkennung.

Präzise beschrieben wird das Verhalten eines Agenten durch seine Agentenfunktion, welche die Wahrnehmungsfolge (alle bisherigen sensorischen Eingaben) auf eine Aktion abbildet. Aufgabe der KI ist es, das Agentenprogramm zu entwerfen, welches die Agentenfunktion implementiert. Die möglichen Aktionen hängen von der Systemarchitektur ab - um als Aktionen Töne zu erzeugen, muss natürlich eine Soundkarte und ein Lautsprecher ansprechbar sein.

A. Software-Agent

Ein Software-Agent, auch als Softbot oder Software-Roboter bezeichnet, ist ein „Computerprogramm mit Problemlösern, die in interaktive Umgebungen eingebettet und zu flexiblen, autonomen und sogar sozial organisierten Aktionen fähig sind, die auf vordefinierte Ziele ausgerichtet sind.“ [6]

Im Gegensatz zu Objekten im objektorientierten Paradigma haben Agenten keine Methoden, die von anderen Objekten aufgerufen werden, sondern haben einen Auftrag und agieren von selbst.

Software-Agenten haben bestimmte Eigenschaften, die sie von normaler Software abgrenzen.
Sie sind

- autonom: Sie übernehmen Verantwortlichkeiten und interagieren ohne direkten Eingriff anderer Einheiten mit der Umgebung. Sie haben einen eigenen „Charakter“.
- flexibel: Sie sind sowohl zugänglich (auf Stimulationen aus der Umgebung direkt reagieren) und proaktiv (sind Zielorientiert und kennen alternative Maßnahmen, statt wie bei konventioneller Software Aufgabenorientiert).
- sozial: Sie kommunizieren und arbeiten mit anderen Agenten zusammen um gemeinsame Ziele zu erreichen.
- mobil: Sie können ihren Ort innerhalb von Kommunikationsnetzen ändern.

Man sieht, dass hier neben der künstlichen Intelligenz weitere Wissenschaftsgebiete Einfluss haben:

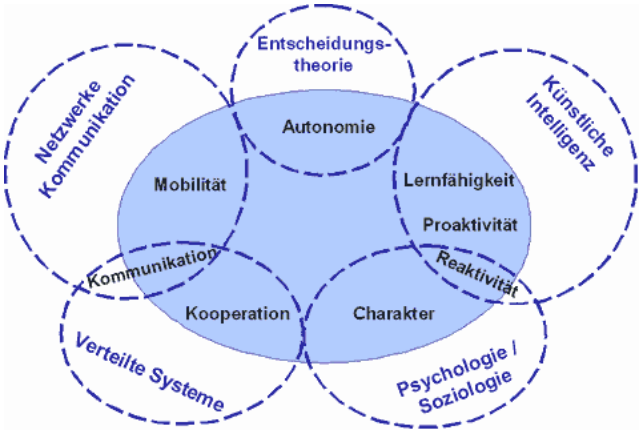

Fig. 1. Einflussgebiete der Wissenschaften auf Agenten, aus [8]

Beispiele für Software-Agenten sind die elektronische Fahrplanauskunft, Spam-Filter, Programme zur Benutzermodellierung, Programme zur automatischen Beantwortung von Emails oder Computergegner in Spielen. Gegenüber herkömmlicher Software erreichen Software-Agenten mehr Flexibilität.

B. Multiagentensysteme

Kleine, einfache Probleme (z.B. alleine Kreuzworträtsel lösen) können durch einzelne Agenten gelöst werden. In komplexen Systemen hat er meistens entweder zu wenig Informationen, einen eingeschränkten Blickwinkel oder zu wenig Fertigkeiten um das Problem alleine zu lösen. Ein Multiagentensystem stellt ein lose verknüpftes Netzwerk von Problemlösern dar, die sich gemeinsam der Lösung eines Problems widmen, das möglicherweise von keinem der Agenten alleine gelöst werden kann.

In Multiagentumgebungen sind verschiedene Möglichkeiten des Umgangs der Agenten miteinander und des Austausches zwischen den Agenten möglich. Die Agenten sind kooperativ, wenn sie gemeinsame Teilziele haben und sich durch die Zusammenarbeit mehr Vorteile erhoffen. Sie können Aufgaben an andere (eventuell spezialisierte) Agenten delegieren. So werden in der industriellen Fertigung einzelne Arbeitsbereiche (Fräsen, Zusammenbauen) von Agenten erledigt.

Wenn Agenten im eigenen Interesse handeln und die Maximierung der Leistungsbewertung eines Agenten gleichzeitig die Leistungsbewertung anderer Agenten negativ beeinflusst, handelt es sich um konkurrierende Agenten. Oft wird man eine Mischung vorfinden: Beim Taxibeispiel haben wir eine partiell kooperative Umgebung wenn es um die Unfallvermeidung geht, allerdings gibt es auch Konkurrenzsituationen (um Parkplätze zum Beispiel). In diesen Fällen muss „verhandelt“ und es müssen Kompromisse eingegangen werden.

Multi-Agentensysteme bieten sowohl die Vorteile verteilter und konkurrierender Problemlösung als auch Flexibilität der sozialen Interaktion.

Hürden bei der Umsetzung von Multiagentensystemen sind einmal die Zerlegung umfangreicher Probleme auf mehrere parallel zu bearbeitende Teilprobleme und die Verteilung beschränkter Ressourcen. Gleichzeitig muss ohne global steuernde Instanz verhindert werden, dass das Übergeordnete System kein schädliches oder oszillierendes Verhalten zeigt, wenn alle Agenten Aktionen ausführen, die lokal sinnvoll erscheinen.

Um zu kommunizieren muss ein Protokoll und eine gemeinsame Agentenkommunikationssprache (ACL) definiert werden, die festlegt, auf welche Ontologie man sich bezieht, ob man auf frühere Kommunikation antwortet und ob man Antworten erwartet. Die Kommunikation kann über direkt ausgetauschte Nachrichten ablaufen oder indirekt über zentrale Blackboards funktionieren [9].

C. Arbeitsumgebung eines Agenten

Die Arbeitsumgebung eines Agenten lässt sich nach folgenden Merkmalen unterscheiden:

- Grad der Beobachtbarkeit
  Die Umgebung ist *vollständig beobachtbar*, wenn sie jederzeit vollständig und präzise für die Sensoren zugänglich ist. Dadurch wird ein interner Zustand als „Erinnerung“ unnötig, da jederzeit alles Relevante beobachtbar ist. Oft sind jedoch Teile der Umgebung für die Sensoren unzugänglich oder die Sensoren ungenau, dann spricht man von einer *teilweise beobachtbaren* Umgebung.
- Deterministisch / Stochastisch
  Hängt der zukünftige Zustand der Umgebung allein vom bisherigen Zustand und den vom Agenten ausgeführten Aktionen ab, so ist die Umgebung deterministisch. Enthält sie dagegen Zufallselemente, die eine Vorhersagbarkeit unmöglich machen (Taxifahrer: Verkehrsflussänderung, Reifen platzen) nennt man sie stochastisch. Eine nicht vollständig beobachtbare deterministische Umgebung *erscheint* uns Stochastisch, da die Ausgangsdaten

für eventuelle Zustandsänderungen uns nicht bekannt sind.

- Episodisch / sequenziell
  In einer episodischen Umgebung ist die Zeit in atomare Episoden oder Runden unterteilt (Wahrnehmung in dieser Episode → Ausführung). Die nächste Episode ist nicht abhängig von in früheren Episoden ausgeführten Aktionen und Erfahrungen. So sollte z.B. eine faire mündliche Prüfung episodisch sein, und die Bewertung des letzten Prüflings nicht vom Abschneiden der vorigen abhängen. Bei sequenziellen Umgebungen können jegliche Kurzzeitaktionen Einfluss weit in die Zukunft haben. Sequenzielle Umgebungen ermöglichen und erfordern also ein Vorausdenken, um rational entscheiden zu können.
- Statisch / Dynamisch
  Ändert sich die Umgebung, während der Agent eine Entscheidung trifft nicht, ist sie statisch (Beispiel: Brettspiele ohne Beschränkung der Bedenkzeit). Dynamische Umgebungen erfordern stets neue Reaktionen und Beobachtungen (Auch wenn der Taxifahrer keine Aktionen ausführt bewegen sich die Autos um das Taxi herum weiter).
- Diskret / Stetig
  Dieses Merkmal bezieht sich auf die Zeit, die Zustände, die Wahrnehmungen und die Aktionen. Umgebungen mit diskreten Zuständen haben eine endliche Zustandsmenge (z.B. Schach). Beim Taxi ist die Wahrnehmung der Geschwindigkeit stetig: Sie ist beliebig im Kontinuum und Änderungen können nicht sprunghaft stattfinden. Auch die Zeit ist in der Realität stetig, sie läuft kontinuierlich ab. In vielen Modellierungen wird sie jedoch in kleinste beobachtbare Zeiteinheiten oder Runden diskretisiert, genauso wie Wahrnehmungen über Sensoren in allen technischen Geräten auf eine diskrete Skala abgebildet werden.

D. Arten von Agentenprogrammen

Die Agentenfunktion kann grundsätzlich als Tabelle über alle möglichen Wahrnehmungsfolgen aufgebaut werden und ist demnach sehr groß: Gibt es W mögliche Wahrnehmungen und hat der Agent eine Lebensdauer (Zahl der Wahrnehmungen, die er entgegennehmen kann) T, so hätte die Tabelle $\sum_{t=1}^{T} W^t$ Einträge. Ist die Lebensdauer des Agenten nicht beschränkt, wäre diese Tabelle unendlich groß. Praktisch ist dies i.A. nicht machbar, da man für jeden Tabelleneintrag eine für diese Situation sinnvolle Aktion bestimmen und eintragen müsste.

Deswegen besteht das Ziel darin, diese Tabelleneinträge durch Regeln zu komprimieren, die durch deutlich weniger Programmcode dargestellt werden können. Ein Programm implementiert die Agentenfunktion, bekommt also eine neue Wahrnehmung und *kann* alte Wahrnehmungen berücksichtigen, indem es diese als interne Zustände speichert. Hier unterscheiden wir folgende Arten von Agentenprogrammen:

1) Einfacher Reflex-Agent: Beim einfachen Reflex-Agenten basieren die Aktionen auf Regeln, die nur von den aktuellen Wahrnehmungen abhängen, unabhängig von allen vorigen.

Solche Agenten haben den Vorteil der Einfachheit (und damit schneller Berechenbarkeit) der Agentenfunktion; in

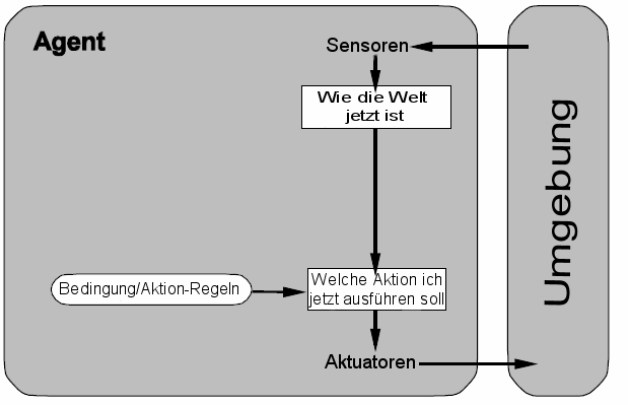

Fig. 2. Aufbau eines Reflex-Agenten nach [7]

den meisten Situationen handeln sie jedoch aus Mangel an Gedächtnis nicht sehr intelligent: Gleiche Wahrnehmungen führen immer zu gleichen Aktionen. Dies gilt besonders bei einer nur teilweise beobachtbaren Umgebung. Hier wäre es gut, wenn sich der Agent daran „erinnern“ könnte, was zuvor wahrgenommen wurde.

Auch bei Menschen können solche Reaktionsmuster in bestimmten Situationen andere überdecken, nämlich bei Reflexen. Wenn man unbewusst einen heißen Gegenstand anfasst, wird man ihn automatisch loslassen.

2) Modellbasierter Reflex-Agent: Der modellbasierte Reflex-Agent erweitert den einfachen Reflex-Agenten um einen internen Zustand, in dem Auswirkungen der früheren Wahrnehmungen und Aktionen gespeichert werden. Dafür benötigt er ein Modell der Welt, das ihm sagt, wie sich früher wahrgenommene Tatsachen außerhalb seines Beobachtungsbereichs weiterentwickeln und wie sich Aktionen des Agenten auf den Zustand der Umgebung auswirken. So ist es möglich auch mit einer teilweise beobachtbaren Umgebung zurechtzukommen - sofern sich die Umgebung an das Modell hält.

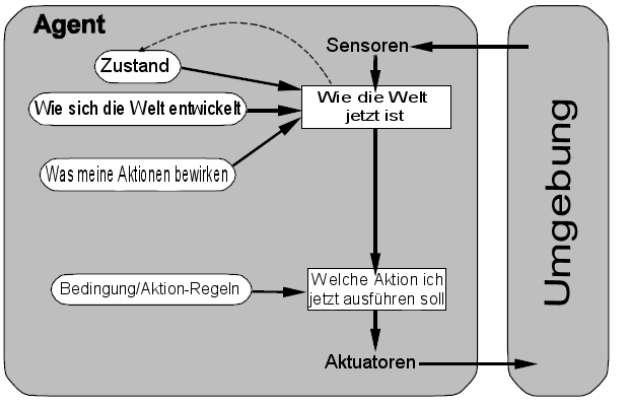

Fig. 3. Aufbau eines modellbasierten Agenten nach [7]

3) Zielbasierter Agent: Das Wissen über den aktuellen Zustand und die Auswirkung der möglichen Aktionen nach festen Regeln reicht unter Umständen nicht aus, um die richtige Aktion durchzuführen. Der Agent braucht dann die Definition eines Ziels, dass er erreichen soll. So kann der

Agent die Aktionen auswählen, die ihn zu diesem Ziel führen. Das ist viel flexibler als die starre Bindung des Ziels an die Regeln. Wenn ein einzelner Schritt nicht ans Ziel führt, muss der Agent verschiedene Folgen von Aktionen im Voraus in Betracht ziehen (Suchen und Planen).

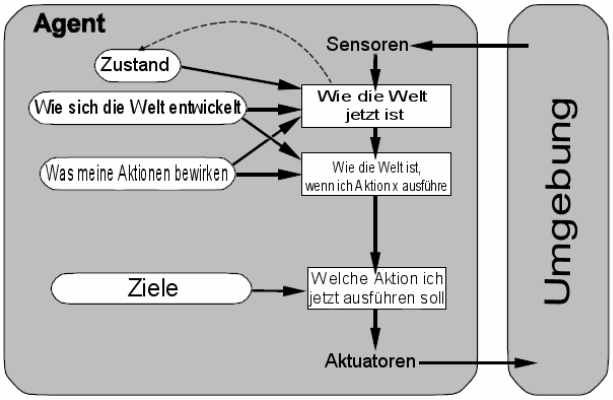

Fig. 4. Aufbau eines zielbasierten Agenten nach [7]

4) Nutzenbasierter Agent: Für einen rationalen Agenten kann nicht nur das Ziel alleine ausschlaggebend sein, sondern auch der Weg (und die Länge des Weges) dahin. Gleichzeitig ist das Ziel eine rein binäre Definition zwischen gutem und schlechtem Zustand. Nutzenbasierte Agenten haben kein eigenes Ziel, sondern eine Nutzenfunktion, die jedem Zustand eine Zahl als Nutzwert zuordnet und so die Güte des Zustandes beschreibt. Ziel ist die Maximierung der Nutzenfunktion. So kann eine passende Abwägung zwischen zueinander in Konflikt stehenden Zielen getroffen werden. Das Problem der Nutzenfunktion ist jedoch, dass zum Berechnen viele Informationen nötig sind, die der Agent in seiner Planung unter Umständen noch nicht kennt.

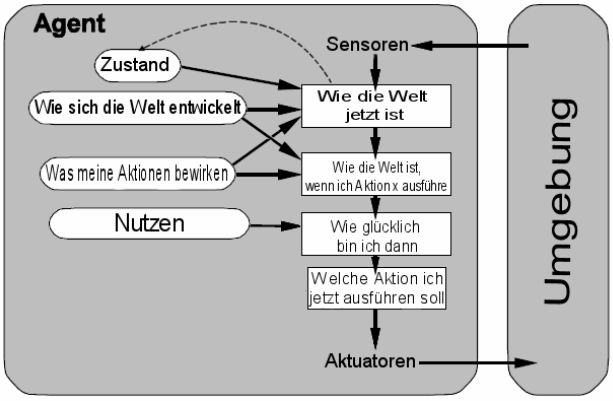

Fig. 5. Aufbau eines nutzenbasierten Agenten nach [7]

5) Lernender Agent: Die bisher untersuchten Agententypen hatten alle gewissermaßen fest verdrahtete Regeln, Modelle, Ziele oder Nutzenfunktionen. Ein lernender Agent kann diese Komponenten verändern, wodurch der Agent kompetenter wird und auch in ihm bisher unbekannten Situationen erfolgreich sein kann.

Ein lernender Agent besteht zunächst aus einem der bisher vorgestellten Agententypen, die Aktionen gemäß der Wahrnehmungsfolge auswählen. Beim lernenden Agenten ist dies das *Leistungselement*. Dieses Leistungselement kann vom *Lernelement* verändert werden um in Zukunft bessere Ergebnisse zu erhalten. Die *Kritik* gibt dem Lernelement Rückmeldung darüber, ob das aktuelle Verhalten des Leistungselements gemäß dem externen Leistungsstandard gut war. Eine weitere Komponente des lernenden Agenten ist der *Problemgenerator*. Nehmen wir an, das Lernelement hätte das Leistungselement so geändert, dass sich die Leistungen des Agenten verbessert hätten. Nun werden in Zukunft immer diese besseren neu gelernten Aktionen ausgeführt werden. Es kann jedoch sein, dass sich durch Experimente mit bisher als schlechter eingeschätzten Aktionen auf längere Sicht neue, bisher unbekannte Möglichkeiten ergeben, die schließlich zu noch besserer Leistung führen. Der Problemgenerator sorgt dafür, dass solche Experimente durchgeführt werden und der Agent neue Erfahrungen sammelt, wobei er durch Belohnung und Bestrafung (identifiziert durch den Leistungsstandard), lernt, wie in potenziell auftretenden Situationen zu handeln oder nicht zu handeln ist.

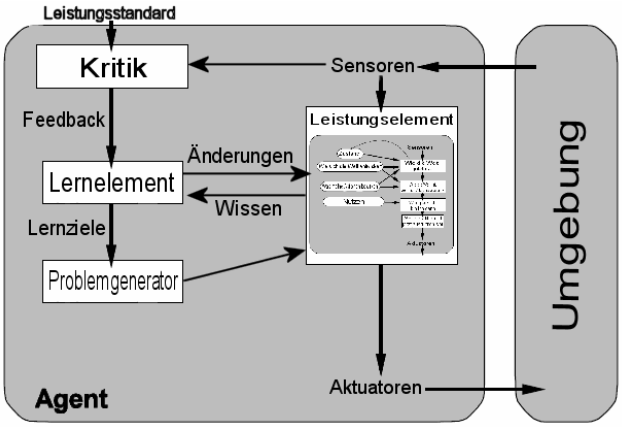

Fig. 6. Aufbau eines lernenden Agenten nach [7]

Welche der hier vorgestellten Architekturen verwendet werden sollten, hängt von der Komplexität des Problems ab. Oft wird man auch hybride Architekturen verwenden. Ein Beispiel hierfür sind *Anytime-Algorithmen*, die bessere Lösungen bieten, je mehr Zeit sie zur Verfügung haben, wobei der Agent aber auch schon sinnvolle Entscheidungen treffen muss, wenn diese in Echtzeit erforderlich sind.

III. Regelbasierte Simulation

Die Regeln, auf denen die Entscheidung des Agenten basieren, bzw. das Wissen, dass sich der lernende Agent zusätzlich aneignet, muss in brauchbarer Form repräsentiert werden. Anhand dieses allgemeinen Wissens können Schlussfolgerungen gezogen werden. Wissen und Schließen ermöglichen so in nur teilweise beobachtbaren Umgebungen kombiniert durch Wahrnehmungen Informationen über verborgene Bereiche der Umgebung zu erhalten.

A. Expertensysteme

Ein Expertensystem ist ein Computerprogramm, welches Spezialwissen und Schlussfolgerungsfähigkeiten von menschlichen Experten in einem begrenzten Aufgabengebiet nachbildet und somit Problemstellungen mit einer zu Experten vergleichbaren Leistung löst [1].

Es verwendet Methoden der künstlichen Intelligenz um Wissen auf einem eng begrenzten Gebiet problemangepasst darzustellen und daraus algorithmisch und heuristisch Schlüsse zu ziehen. Im Dialog mit dem Benutzer kann es diese Schlüsse darstellen und erklären.

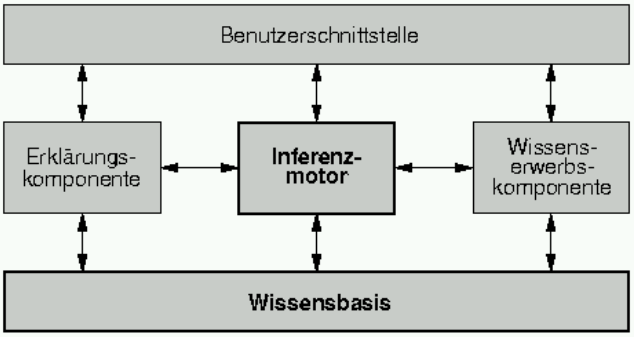

Fig. 7. Komponenten eines Expertensystems aus [3]

Abbildung 7 zeigt die Komponenten eines Expertensystems. Es besteht einmal aus der Wissensbasis (knowledge base), welche das Fachwissen in einer beliebigen Repräsentationsform enthält, sowie aus einer Inferenzmaschine, die daraus Schlüsse ziehen kann. Diese verknüpft die Fakten des Problems mit den Regeln der Wissensbasis und erhält daraus neue Fakten. Die Erklärungskomponente liefert Informationen darüber, wie die Lösung erreicht wurde oder warum keine gefunden werden konnte. Die Wissenserwerbskomponente hingegen erweitert die Wissensbasis und prüft diese aus Konsistenz. Die Benutzerschnittstelle realisiert schließlich die Interaktionen mit dem anfragenden Anwender. Dafür müssen die Fakten in natürliche Sprache oder menschenlesbare Form gebracht werden.

Ein Beispiel für ein Expertensystem sind Programme zur Unterstützung medizinischer Diagnosen oder zur Analyse statistischer Daten.

B. Wissensrepräsentation durch logische Formeln

Wissen kann in Form von Regeln repräsentiert werden. Diese können beispielsweise als aussagenlogische oder prädikatenlogische Formeln bzw. Sätze formuliert sein. Wissensrepräsentation kann erfolgen durch Fakten, die einfach gelten (Naturgesetze oder ähnliches), Zusammenhänge und Auswirkungen der Fakten oder auch die Erfahrung welche Schlussfolgerungsmechanismen und Strategien zum Ziel führen.

Effekt-Axiome beschreiben Veränderungen durch Aktionen, z.B. kann man beim Taxi die Regel aufstellen, dass wenn man im Zeitpunkt t bremst, die Geschwindigkeit im nächsten Zeitpunkt geringer oder gleich sein wird:

$\forall v \forall t(BremseGedrueckt(t) \wedge Geschwindigkeit(v,t) \rightarrow \exists w \leq v : Geschwindigkeit(w, t+1))$.

Frame-Axtiome beschreiben Eigenschaften und Zustände, die Aktionen unverändert lassen. So ändert sich der Status, ob sich ein Fahrgast im Taxi befindet, durch das Bremsen nicht (wenn wir Vollbremsungen außer Acht lassen).

Das Wissen von Agenten kann man in mentale Eigenschaften aufteilen, die zu jedem Beobachtungszeitpunkt existieren:

- Belief (Glauben): Enthält die Menge der Aussagen, von denen der Agent momentan glaubt, dass sie wahr sind.
- Uncertanity (Unwissenheit): Die Menge der Aussagen, die der Agent (als Formel) kennt, aber deren Wahrheitsgehalt er nicht kennt.
- Desire (Wunsch): Die Aussagen, die der Agent gerne wahr haben möchte.
- Intention (Intention): Aussagen, die der Agent momentan für falsch hält, aber die er wahr haben möchte. Dies stellt die Basis für das Handeln dar.

C. Schließen

Ein Satz β ist die logische Konsequenz eines Satzes α ($\alpha \rightarrow \beta$), wenn β in allen Welten wahr ist, in denen auch α wahr ist.

Aus Regeln in der Wissensbasis (Prämissen) und durch Wahrnehmungen erkannte Aussagen, kann auf neue Aussagen durch logische Konsequenz geschlossen werden (Deduktion, Modus Ponens). Solches automatisches Schließen wird auch *Inferenz* genannt. Der Inferenzalgorithmus sollte korrekt (nur logische Konsequenzen ableiten) und vollständig (alle logischen Konsequenzen ableitbar) sein.

Es ist dabei sowohl möglich aus einer Ursache eine Wirkung/Prognose zu schließen (Vorwärtsverkettung) als auch von einer sichtbaren Wirkung auf die deswegen notwendige Ursache/Bedingung zu schließen (Rückwärtsverkettung).

Um auf das Taxibeispiel zurückzukehren gäbe es die Formel `PersonHebtDieHand` $\vee$ `PersonRuftTaxi` $\rightarrow$ `PersonMöchteMitfahren`

und

`PersonMöchteMitfahren` $\wedge \neg$`FahrgastAnBord` $\rightarrow$ `HalteAn`.

Würde durch die Wahrnehmung nun `PersonHebtDieHand` oder `PersonRuftTaxi` wahr werden, kann mittels Modus Ponens auf `PersonMöchteMitfahren` geschlossen werden. Andersherum kann, wenn es nur diese Regeln gibt, aus der Tatsache, dass ein Taxi steht, geschlossen werden, dass es vorher eine Person sah, die mitfahren wollte.

D. Andere Arten der Wissensrepräsentation

In der KI gibt es weitere Möglichkeiten zur Wissensrepräsentation als nur durch logische Formeln:

Ontologien und semantische Netze speichern die Beziehungen von Begriffen untereinander und stellen Hierarchien und Synonyme dar.

Künstliche neuronale Netze bilden das biologische Nervensystem mit „Nervenzellen“ nach, die mehrere binäre Eingänge und einen Ausgang haben. Das Netz bekommt dann ausgesuchte Eingaben auf dessen Basis es selbständig lernt, indem es Verbindungen zwischen Nervenzellen herstellt oder

löscht und deren Eingangsgewicht anpasst. Benutzt werden sie vor allem bei Problemen ohne explizites Wissen wie bei der Mustererkennung.

IV. Fallbeispiel: Agentenbasierter Markt

In allgemeinen agentenbasierten Marktplätzen führen Software-Agenten im Namen menschlicher Anwender autonom wirtschaftliche Transaktionen durch. Ein elektronischer Markt kann aus mehreren Marktplätzen z.B. für Rohstoffe, Dienstleistungen oder Personal bestehen.

Der Vorteil der Software-Agenten im Vergleich zu menschlichen Marktteilnehmern besteht darin, dass sie mit vielen potentiellen Handelspartnern gleichzeitig verhandeln können und so einen viel besseren Preis erreichen können, wobei durch die Schnelligkeit und Automatisierung auch deutlich weniger Transaktionskosten entstehen.

Es gibt dabei auch Nachteile, die z.B. deutlich wurden am Schwarzen Montag 1987, als leichte Kursverluste aufgrund von schlechten Wirtschaftsdaten durch schnell und ähnlich agierende computergestützte Handelssysteme überproportional verstärkt und rückgekoppelt wurden. Die Folge war der stärkste Kursverlust der Wall Street von 22% an einem Tag.

Agenten müssen miteinander verhandeln um Transaktionen abzuschließen. Drei Szenarien der Verhandlungsmöglichkeiten lassen sich hier unterscheiden:

1) Agenten, die für jeweils unterschiedliche Unternehmen arbeiten, führen im Auftrag Transaktionen durch und einigen sich. Käufer- und Verkäuferagenten können automatisiert verhandeln. Beide Agenten haben Ziele vorgegeben bekommen (Ware x möglichst teuer verkaufen, Ware y in brauchbarer Qualität erstehen)
2) Auf der einen Seite verhandelt ein Agent und auf der anderen ein Mensch. Dies erweist sich aufgrund der unterschiedlichen Verständigung von Agent und Mensch als schwierig [2].
3) Agenten eines Unternehmens arbeiten kooperativ zusammen um z.B. den Gewinn zu verbessern (auch Distributed Problem Solving genannt)

Wir betrachten im Folgenden nur Fall 1, der bei virtuellen Marktplätzen vorkommt. Die Software-Agenten gehören zu unterschiedlichen Unternehmen mit jeweils unterschiedlichen Zielen und Strategien, die durchaus inkompatibel sein können [2]. Die Ziele und Strategien eines Agenten sind den anderen Agenten im Allgemeinen nicht bekannt.

Agenten können den Markt permanent beobachten und auch Transaktionen initiieren, wenn bei allen beteiligten Agenten dadurch der Nutzen erhöht wird.

Jeder Agent hat eine Startverteilung an Ressourcen vor der Transaktion sowie eine nur ihm bekannte Zielverteilung (vorgegeben als Nutzenfunktion oder konkret in Menge der Ressourcen), die er durch die Teilnahme am Markt erreichen möchte. Um von der Startverteilung zur Zielverteilung zu gelangen, kooperiert er mit einem anderen Agenten, wodurch Geld und Güter ausgetauscht werden (Transaktion). Dieser andere Agent kann auch als Intermediär oder Händler auftreten und zeitgleich schon wieder mit anderen Agenten in Verhandlung stehen.

Nach [5] wird im folgenden Fallbeispiel ein möglicher Koordinationsmechanismus im Personalmarkt dargestellt, und zwar durch direkte Verhandlung von Agenten der personalsuchenden Unternehmen und Agenten der Bewerber. Äußere Bedingungen sind Zeitschranken bei der Verhandlung und unvollständiges Wissen.

Denkbar wäre alternativ auch ein zentraler Koordinationsagent (Facilitator, Mediator), über den alle Agenten nur indirekt kommunizieren können und der selber vertrauenswürdig ist und keine eigenen Interessen hat.

A. Koordinationsmechanismus

Die Agenten beobachten den Markt und schauen nach brauchbaren Bewerbern bzw. Unternehmen. Durch Ihre Lernfähigkeit können sie auch die Situation am Personalmarkt analysieren und ihre Strategie anpassen. Schließlich werden sie mit einem Agenten der Gegenseite in Kontakt treten und eine Verhandlung über die Konditionen der Anstellung beginnen.

1) Verhandlungsbasis: Zu verhandeln seien hier drei Elemente: Stundenlohn, Arbeitsstunden pro Woche und die Dauer der Mindestanstellungszeit. Dies sind die Verhandlungsattribute. Dem gegenüber stehen die Produktattribute, die bei jedem Produkt (hier Bewerber bzw. Unternehmen) fest sind, wie Alter, Ausbildung und Sprachkenntnisse bzw. Betriebsklima, Renommée. Das Anfangsgebot hängt von den Produktattributen der jeweils anderen Seite ab.

Ein Angebot, dass der Bewerber zum Zeitpunkt t_i empfängt sei ein Vektor $x^i = (x^i_s, x^i_h, x^i_d)$, wobei $i = 0, 1, ..., n$ der Verhandlungsschritt von insgesamt n Verhandlungsschritten ist. x^i_s ist ein Wert für den Stundenlohn, x^i_h die Zahl der Arbeitsstunden pro Woche und x^i_d die Mindestanstellungszeit. Analog dazu ist ein Angebot, dass der Agent des Arbeitgebers erhält ein Vektor $y^i = (y^i_s, y^i_h, y^i_d)$. Sowohl der Arbeitgeber als auch der Bewerber haben minimale und maximale Vorstellungen bezüglich der Konditionen (können auch unendlich sein):

$$s^n_{\mathrm{Min}} \leq x^i_s \leq s^n_{\mathrm{Max}}, h^n_{\mathrm{Min}} \leq x^i_h \leq h^n_{\mathrm{Max}}, d^n_{\mathrm{Min}} \leq x^i_d \leq d^n_{\mathrm{Max}}$$

$$s^g_{\mathrm{Min}} \leq y^i_s \leq s^g_{\mathrm{Max}}, h^g_{\mathrm{Min}} \leq y^i_h \leq h^g_{\mathrm{Max}}, d^g_{\mathrm{Min}} \leq y^i_d \leq d^g_{\mathrm{Max}}$$

In diesen Bereichen, in denen eine Einigung also grundsätzlich möglich ist, gibt es einen bestimmten Nutzen für die jeweilige Partei. Dieser wird durch die Nutzenfunktionen N für den Bewerber und G für den Arbeitgeber beschrieben:

$$N_s : [s^n_{\mathrm{Min}}, s^n_{\mathrm{Max}}] \rightarrow [0,1],\ N_h : [h^n_{\mathrm{Min}}, h^n_{\mathrm{Max}}] \rightarrow [0,1],$$

$$N_d : [d^n_{\mathrm{Min}}, d^n_{\mathrm{Max}}] \rightarrow [0,1],\ G_s : [s^g_{\mathrm{Min}}, s^g_{\mathrm{Max}}] \rightarrow [0,1],$$

$$G_h : [h^g_{\mathrm{Min}}, h^g_{\mathrm{Max}}] \rightarrow [0,1],\ G_d : [d^g_{\mathrm{Min}}, d^g_{\mathrm{Max}}] \rightarrow [0,1]$$

Je höher der Wert ist, desto besser ist das Angebot hinsichtlich dieses Parameters. Ein Wert von 1 entspricht 100% Zustimmung. Eine mögliche Modellierung zeigt Abbildung 8.

Der niedrigste akzeptable Stundensatz ist s^n_{Min} und der erhoffte Stundensatz ist s^n_{Opt}. Wenn er mehr bekommt, steigt der Nutzen natürlich weiter.

Für den Arbeitgeber ist der höchste Nutzen wenn er nichts bezahlen muss. Der Nutzen fällt dann bis zum Lohn s^g_{Max}, den er maximal bereit ist zu zahlen.

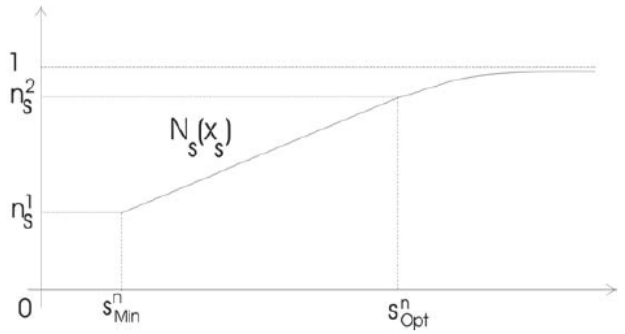

Fig. 8. Nutzenfunktion des Stundenlohns für den Arbeitnehmer für eine Anstellungsverhandlung

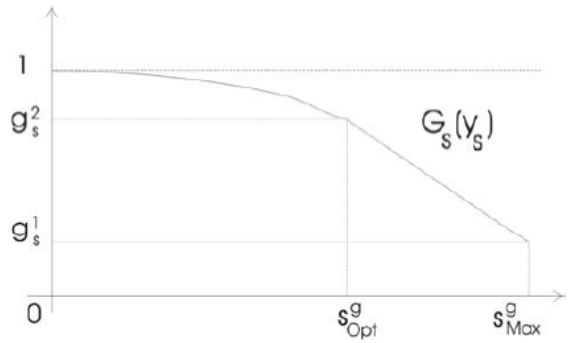

Fig. 9. Nutzenfunktion des Stundenlohns für den Arbeitgeber für eine Anstellungsverhandlung

Die geleisteten Arbeitsstunden und die Mindestanstellungszeit haben in diesem Modell sowohl für den Arbeitgeber als auch für den Arbeitnehmer Nutzenfunktionen folgender Art:

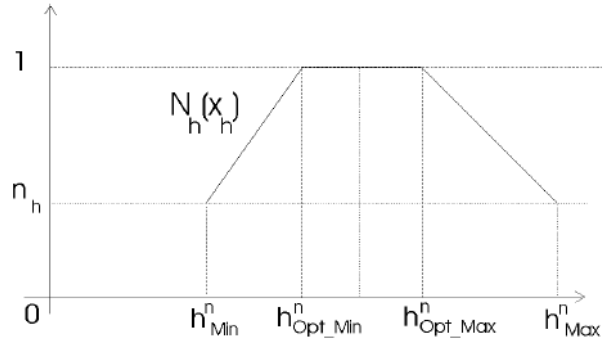

Fig. 10. Nutzenfunktion der Arbeitsstunden für den Arbeitnehmer für eine Anstellungsverhandlung

Die Werte für die minimalen, maximalen und optimalen Mengen, sowie die Steigungen der linearen Teile können natürlich bei allen Fällen verschieden sein.

In einer Verhandlung sind den jeweiligen Partnern die Verhandlungsattribute unterschiedlich wichtig. Darum führen wir relative Gewichtungen $\omega_s^n, \omega_h^n, \omega_d^n$ für den Bewerber und $\omega_s^g, \omega_h^g, \omega_d^g$ für den Arbeitgeber ein, mit
$\omega_s^n + \omega_h^n + \omega_d^n = 1 = \omega_s^g + \omega_h^g + \omega_d^g$.

Der Gesamtnutzen eines Angebotes kann nun nach folgender Funktion für den Bewerber berechnet werden:

$$N(x^i) = \omega_s^n N_s(x_s^i) + \omega_h^n N_h(x_h^i) + \omega_d^n N_d(x_d^i)$$

und für den Arbeitgeber entsprechend

$$G(y^i) = \omega_s^g G_s(y_s^i) + \omega_h^g G_h(y_h^i) + \omega_d^g G_d(y_d^i).$$

2) Bilaterales Verhandlungsprotokoll: Zunächst gehen wir davon aus, dass nur zwei Agenten miteinander verhandeln. Eine Verhandlung ist eine Folge von Angeboten und Gegenangeboten $x^0 y^1 x^2 y^3 ...$, wobei also das erste Angebot vom Arbeitgeber kommt.

Die Parteien berechnen ihr erstes Angebot nach dem höchsten Wert. Aufgrund unserer Nutzenfunktionen ist das $x^0 = (s^g_{\mathrm{Opt}}, h^g_{\mathrm{Opt}}, d^g_{\mathrm{Opt}})$ respektive $y^0 = (s^n_{\mathrm{Opt}}, h^n_{\mathrm{Opt}}, d^n_{\mathrm{Opt}})$. Der Bewerber sendet zwar sein erstes Angebot nicht, berechnet es aber um zu vergleichen.

Nach dem Empfang eines Angebotes hat der Agent des Bewerbers folgende Handlungsalternativen:

1) Angebot annehmen, wenn $N(x^i) \geq N(y^i)$, wobei y^i das momentane Angebot des Bewerbers ist
2) Verhandlung abbrechen, wenn $t_i > t^n_{\mathrm{Max}}$
3) ansonsten ein Gegenangebot y^{i+1} machen.

Genau analog sehen die Handlungsalternativen des Arbeitgebers aus.

Welcher dieser Alternativen gewählt wird, wird hier durch Regeln im Agenten bestimmt. Dies ist nur sinnvoll, wenn sich die Umwelt des Agenten nicht ändert. Adaptive Ansätze gehen gleich von unvollständiger Information über die Umwelt und das Handeln des Verhandlungspartners aus und nutzen heuristische Regeln, die zum Beispiel von einer gewissen konstanten Spanne zwischen erstem Angebotspreis und tatsächlich erzieltem Kaufpreis ausgehen. Zudem gibt es noch Spieltheoretische Entscheidungsansätze und Ansätze mit evolutionären Algorithmen (mehr dazu in [5]).

Beschrieben werden muss jetzt noch, nach welcher Strategie Gegenangebote erstellt werden. Eine Verhandlungsstrategie ist eine Vektorfunktion F, die ein neues Angebot erzeugt:

$$F(t, S_m, S_n) = (f_s(t, S_m, S_n), f_h(t, S_m, S_n), f_d(t, S_m, S_n)),$$

wobei die Teilfunktionen bestimmen, wie die Werte von Stundenlohn, Arbeitsstunden und Mindestanstellungszeit geändert werden. Sie hängen von der Zeit t ab, sowie vom Zustand S_m des Marktplatzes und S_n der Verhandlung. Die letzten beiden sind Vektoren und beinhalten Informationen wie die Zahl der Gesamtagenten im Markt, die Zahl der verhandelnden Agenten und andere Angebote, die der Agent erhielt.

Eine mögliche Verhandlungsstrategie, die nur von der Zeit t abhängt ist folgende:

$$f : [t_{Init}, t_{Max}] \rightarrow [f_{\mathrm{Opt}}, f_{\mathrm{Max}}]$$

$$f(t) = f_{\mathrm{Opt}} + \left(\frac{t - t_{\mathrm{Init}}}{t_{\mathrm{Max}} - t_{\mathrm{Init}}} \right)^p (f_{\mathrm{Max}} - f_{\mathrm{Opt}})$$

falls die Nutzenfunktion N_x bzw. G_x des Attributes im Intervall $[f_{\mathrm{Opt}}, f_{\mathrm{Max}}]$ fällt und

$$f : [t_{\mathrm{Init}}, t_{\mathrm{Max}}] \rightarrow [f_{\mathrm{Min}}, f_{\mathrm{Opt}}]$$

$$f(t) = f_{\mathrm{Opt}} - \left(\frac{t - t_{\mathrm{Init}}}{t_{\mathrm{Max}} - t_{\mathrm{Init}}}\right)^{p} (f_{\mathrm{Opt}} - f_{\mathrm{Min}})$$

falls die Nutzenfunktion im Intervall $[f_{\mathrm{Min}}, f_{\mathrm{Opt}}]$ steigt.

t_{Init} ist der Zeitpunkt, an dem der Agent erstellt wurde, und t_{Max} ist der Zeitpunkt, bis zu dem die Verhandlung abgeschlossen sein muss (im Allgemeinen bei beiden Agenten unterschiedlich). $p > 0$ ist ein Parameter, der die Kompromissfähigkeit beschreibt. Dieser kann auch für die verschiedenen Verhandlungsattribute unterschiedlich gewählt werden. $f_{\mathrm{Min}}, f_{\mathrm{Opt}}, f_{\mathrm{Max}}$ sind die jeweiligen Punkte in den Nutzenfunktionen der Attribute.

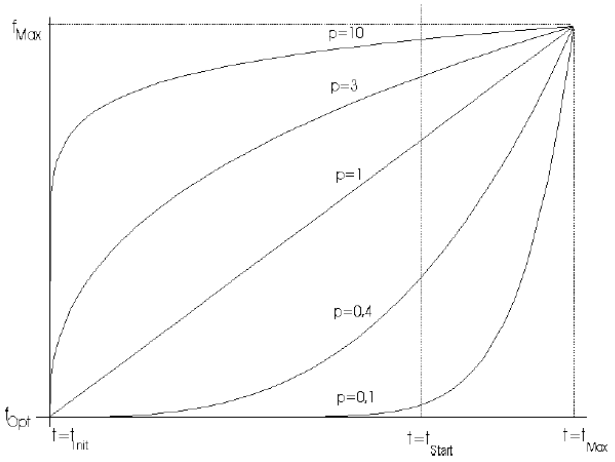

Fig. 11. Verhandlungsstrategie für fallende Nutzenfunktion in $[f_{\mathrm{Opt}}, f_{\mathrm{Max}}]$ mit verschiedenen Werten für p

Man sieht, dass zum Zeitpunkt $t = t_{\mathrm{Init}}$ immer f_{Opt} als erstes Angebot erstellt wird, und dass, wenn die Zeit nah an t_{Max} herankommt, das Ergebnis näher an die Randbereiche geht.

Andere Verhandlungsstrategien können auch Ressourcenabhängig oder Verhaltensabhängig sein. Im ersten Fall wird die Zahl und Höhe der Angebote anderer Agenten mit berücksichtigt. Bei den verhaltensabhängigen Verhandlungsstrategien wird bei der Angebotserstellung die Strategie anderer Agenten berücksichtigt oder die eigene Absicht verschleiert. Die bekannteste verhaltensabhängige Strategie ist sicherlich die Imitation oder „Tit-for-tat“ (Wie du mir, so ich dir).

Die Taktik eines Agenten kann auch aus sich dynamisch ändernden Kombinationen verschiedener Verhandlungsstrategien bestehen.

3) Multilaterales Verhandlungsprotokoll: Bisher waren immer nur Verhandlungen zwischen zwei Partnern betrachtet worden. In vielen Märkten, insbesondere im Personalmarkt, sind mehrere Bewerber an einem Unternehmen interessiert und auch mehrere Unternehmen können an einen Bewerber interessiert sein.

Ein Arbeitgeber wird zunächst eine Vorauswahl aus den zur Verfügung stehenden Bewerbern treffen, die überhaupt in Frage kommen. Diese werden dann basierend auf ihren Produktattributen (Alter, Fähigkeiten, ...), welche wieder eigene Nutzenfunktionen und Gewichte haben, in eine Ordnung $\prec$ gebracht. Der Arbeitgeberagent führt nun das vorher vorgestellte bilaterale Verhandlungsprotokoll nacheinander mit den Agenten in dieser Ordnung durch, $n_1 \prec n_2 \prec n_3 \prec ...$, jeweils ein Zeitschritt mit n_1, dann mit n_2 usw. und wiederholt dies so lange, bis ein Angebot angenommen wurde, die Zeit vorbei ist, oder keine Bewerber mehr vorhanden sind. Dies ist sinnvoll auch dann, wenn im ersten Verhandlungsschritt der als am schlechtesten eingestufte Bewerber das Angebot annimmt, denn das Anfangsangebot des Arbeitgebers ist ja bei jedem Bewerber separat auf dessen Nutzen ausgerichtet. Wenn Bewerber sich mit anderen Arbeitgebern einig werden, oder deren Zeit abgelaufen ist, teilen sie dies mit und werden aus der Ordnung entfernt. Das Vorgehen der Bewerber ist dazu analog [5].

V. Zusammenfassung

Das Agentenparadigma stellt eine Möglichkeit der Simulation dar, die besonders im Falle mehrerer verteilter Einheiten mit jeweils unterschiedlichen Intentionen und Perspektiven von Vorteil ist. Gegenüber zentralistischen Simulationsansätzen modellieren agentenbasierte Ansätze die Wirklichkeit unmittelbarer und ermöglichen dynamische Rückkopplungen und Anpassungen der Simulation durch Austausch oder Hinzufügen von Agenten. Durch die Vielzahl der Anwendungsbereiche und mit der steigenden Verbreitung verteilter Systeme, lässt sich in Zukunft eine größere Bedeutung des Agentenparadigmas erwarten.

Im Studienprojekt ViCCC ist der Einsatz von Agenten an verschiedenen Stellen bei der Berechnung des neuen Simulationszustandes während des Rundenwechsels denkbar: Bei Transaktionen in spielerübergreifenden Märkten (Personal- und Gerätemarkt), bei der Entscheidung über die Vergabe von Ausschreibungen und bei der Simulation des Baufortschritts. Nicht zuletzt für die Ergänzung des Spiels um Computergegner bieten sich Agenten an.

Literatur

[1] Abecker, Andreas: *Themen und Techniken der KI.* – http://www.kde.cs.uni-kassel.de/lehre/ws2007-08/KI/folien/4_01_Methoden_und_Techniken.pdf

[2] Eymann, Torsten: *AVALANCHE - Ein agentenbasierter dezentraler Koordinationsmechanismus für elektronische Märkte.* 2000. – http://www.freidok.uni-freiburg.de/volltexte/147/pdf/eymann.pdf

[3] Gottlob, Georg ; Frühwirt, Thomas ; Horn, Werner: *Expertensysteme.* Springer Verlag Wien, 1990

[4] Hoogervorst, Coen et a.: *Onderzoek naar de toepassing van een Agent voor de verrijzeniskapel, S. 9-12.* – http://sts.bwk.tue.nl/kanaalstraat6/studenten-rapporten/Agent kanaalstraat maart 06.pdf

[5] Kurbel, Karl ; Loutchko, Iouri: *A Model for Multi-Lateral Negotiations on an Agent-based Marketplace for Personnel Acquisition.* – http://www.vg-u.de/euv-new-site/doc/softcomp/Anlage 13.pdf

[6] Luger, George F.: *Künstliche Intelligenz - Strategien zur Lösung komplexer Probleme.* Pearson Education, 2002

[7] Russell, Stuart J. ; Norvig, Peter: *Künstliche Intelligenz - ein moderner Ansatz.* Pearson Education, 2004

[8] Theiss, Mirko ; Greb, Steffen: *Agenten im Bauwesen.* – http://www.inf.bi.rub.de/release/aib/index.html

[9] Vetter, Michael: *Ein Multiagentensystem zur Verhandlungsautomatisierung in elektronischen Märkten.* 2006