Ein contract-net-basiertes Lernverfahren für eine benutzeradaptive Interface-Agentur

Britta Lenzmann & Ipke Wachsmuth
Technische Fakultät, Universität Bielefeld
Postfach 100131, 33501 Bielefeld
{britta,ipke}@techfak.uni-bielefeld.de

Zusammenfassung

In diesem Beitrag wird ein agentenorientierter Ansatz zur Adaption an Benutzerpräferenzen vorgestellt. Eine Interface-Agentur – bestehend aus mehreren kommunizierenden und kooperierenden Unteragenturen – adaptiert an unterschiedliche Benutzerpräferenzen durch die Erstellung eines impliziten, in der Agentur verteilten Benutzermodells. Durch Ausnutzung von Benutzer-Feedback organisieren sich Agenten im contract-net-basierten Verhandlungsaustausch mit anderen Agenten so, daß eine dynamische Anpassung an Benutzerpräferenzen gemäß der aktuellen Situation realisiert werden kann.

1 Lernende Interface-Agenten

Zur Verbesserung der Mensch-Maschine-Interaktion werden in jüngerer Zeit vermehrt Interface-Agenten eingesetzt, die als Vermittler zwischen Benutzer und technischem System agieren [Laurel, 1990]. Einerseits unterstützen solche Computer-Programme den Benutzer, indem sie in seinem/ihrem Interesse und nach seinen/ihren Vorlieben handeln [Norman, 1994]; andererseits erlauben sie intuitivere Kommunikationsformen, indem sie qualitative Benutzeranweisungen in Kommandos übersetzen, die vom Anwendungssystem verstanden und ausgeführt werden [Wachsmuth & Cao, 1995].

Um den Benutzer bei der Aufgabenbearbeitung zu unterstützen, benötigen Interface-Agenten Wissen über ihre Benutzer und die Anwendung. Konventionelle Ansätze greifen hierfür auf die Erhebung und Repräsentation von Informationen in Form von Domänen- und Benutzermodellen zurück. Probleme bei diesem Vorgehen sind unter anderen der hohe Aufwand der Wissensmodellierung und die geringe Dynamik des Systems. Daraus resultieren neuere Ansätze, in denen Interface-Agenten mit maschinellen Lernverfahren ausgestattet werden, durch die Informationen über ihre Benutzer und die Anwendung automatisch akquiriert und entsprechend der aktuellen Anforderungen angepaßt werden können [Maes & Kozierok, 1993].

Auf diese Weise sind lernende Interface-Agenten entwickelt worden, die ihre Benutzer z.B. bei der Terminplanung oder Informationsbeschaffung und -filterung unterstützen [Maes, 1994], [Mitchell *et al.*, 1994]. Hierbei kommen Techniken wie Lernen durch Beobachtung, durch Beispiele oder Lernen von anderen Agenten zum Einsatz. Die Anwendungen haben gezeigt, daß die Probleme der konventionellen Wissensmodellierung auf diesem

Weg weitestgehend gelöst werden können. Trotz der sich ergebenden Vorteile zeichnen solch lernende Interface-Agenten nach wie vor benutzerspezifische Daten auf und legen diese in einem expliziten Benutzermodell ab.

Da die Speicherung persönlicher Daten in Form eines expliziten Benutzermodells aber durchaus kritisch zu betrachten ist [Norman, 1994], verfolgen wir einen Ansatz, in dem sich eine *lernende Interface-Agentur* an Benutzerpräferenzen durch die Erstellung eines impliziten, in der Agentur verteilten Benutzermodells adaptiert. Die Interface-Agentur besteht, im Gegensatz zu den oben beschriebenen Ansätzen lernender Interface-Agenten, aus mehreren kommunizierenden und kooperierenden Unteragenturen, die jeweils Agenten des gleichen Aufgabentyps, aber mit variierten internen Funktionalitäten – korrespondierend zu den unterschiedlichen Benutzerpräferenzen – zusammenfassen. Auf der Basis von direktem und indirektem Benutzer-Feedback organisieren sich Agenten im Verhandlungsaustausch mit anderen Agenten so, daß eine dynamische Anpassung an Benutzerpräferenzen realisiert werden kann. Das Lernen bezieht sich somit auf die Interface-Agentur und wird nicht, wie in obigen Ansätzen, durch Lernen einzelner Agenten erzielt. Die Modellierung von Interface-Agenten durch ein Multiagentensystem, das mit maschinellen Lernverfahren ausgestattet ist, erlaubt dann eine Benutzeradaption ohne explizite Speicherung persönlicher Daten.

Im Folgenden werden die Grundlagen des hierfür entwickelten Verfahrens beschrieben (Kapitel 2), das Lernverfahren durch Verhandlung im contract-net detailliert ausgeführt (Kapitel 3) und anhand der Realisierung im interaktiven 3D-Grafiksystem VIENA illustriert (Kapitel 4). Abschließend werden Erweiterungen und zukünftige Arbeiten diskutiert (Kapitel 5).

2 Grundlagen des Lernverfahrens

Die Grundlage der Benutzeradaption ohne Verwendung eines expliziten Benutzermodells ist ein Multiagentensystem, in dem Agenten ähnlicher Funktionalität zu Unteragenturen zusammengefaßt werden. Jede Unteragentur entspricht einer bestimmten Klasse von Benutzerpräferenzen für einen gegebenen Aufgabentyp, und jeder Agent innerhalb einer Unteragentur realisiert eine bestimmte Benutzerpräferenz dieser Präferenzklasse.

Die Aufgabe des Lernverfahrens besteht nun darin, die Agentur so zu organisieren, daß diejenigen Agenten jeder Unteragentur aktiviert werden können, die den aktuellen Benutzerpräferenzen in der aktuellen Situation entsprechen. Eine dynamische Anpassung muß dann realisiert werden im Hinblick auf

1. Präferenzen unterschiedlicher Benutzer sowie
2. temporär wechselnde Präferenzen eines individuellen Benutzers während eines Systemlaufs.

Das Multiagentensystem lernt diese Aufgabe durch Ausnutzung des Benutzer-Feedbacks. Im einzelnen handelt es sich um implizit positives und explizit negatives Feedback. Implizit positives Feedback ist gegeben, wann immer eine Benutzeranweisung erfolgt, die von der vorausgegangenen unabhängig ist. Korrigiert dagegen der Benutzer die von der Interface-Agentur angebotene Lösung ("wrong"), so liegt ein explizit negatives Feedback vor.

Das Benutzer-Feedback repräsentiert Reinforcement-Signale, die von der Interface-Agentur intern als *credit-Werte* kodiert und interpretiert werden. Credits können als eine Art Qualitätsmaß verstanden werden, die von jedem Agenten der Gesamtagentur lokal gespeichert und verwaltet werden. Die Adaption an Benutzerpräferenzen wird dann durch folgende zwei Schritte erreicht:

1. Einstellung von credit-Werten entsprechend dem Feedback und dem Situationsverlauf

2. Aktivierung von geeigneten Agenten mit maximalen credits

Hierfür müssen eine Reihe von Kommunikations- und Kooperationsprozessen zwischen Agenten und Unteragenturen durchgeführt werden. Diese Schritte sind in einem contract-net-ähnlichen Verhandlungsschema [Davis & Smith, 1983] eingebettet, in dem Agenten die Rolle eines Auftragnehmers und/oder Auftraggebers übernehmen.

3 Contract-net-basiertes Lernverfahren

In einem contract-net-basierten Verfahren wird die Bearbeitung von Aufträgen und Unteraufträgen durch ein Ausschreibungsverfahren realisiert, in dem Aufträge von Agenten als Auftraggeber ausgeschrieben, Angebote von den adressierten Auftragnehmeragenten erstellt und Aufträge an geeignete Anbieter vergeben werden. Um die Adaption an Benutzer- und Situationsanforderungen zu realisieren, haben wir in dieses Verfahren die obigen beiden Schritte integriert. Die Einstellung von credit-Werten wird dann von allen Agenten durchgeführt, die sich momentan in der Auftragnehmerrolle befinden, während die Aktivierung von geeigneten Agenten mit maximalen credits von allen momentan aktiven Auftraggeberagenten umgesetzt wird.

Die Einstellung von credit-Vektoren gemäß der aktuellen Situation wird realisiert, indem Auftragnehmer relevante Informationen aus allen bisher eingegangenen Verhandlungsmeldungen extrahieren, speichern und zur Aktualisierung der neuen credit-Werte heranziehen. Aus den eingehenden Meldungen aller sich an der Verhandlung beteiligenden Auftragnehmern ermittelt ein Auftraggeber den bisher erfolgreichsten Auftragnehmer und verpflichtet diesen zur Auftragsausführung. Sind mehrere Agenten gleich erfolgreich, wird einer aus dieser Menge ausgewählt.

Das Adaptionsverfahren kann somit durch eine Menge von Funktionen beschrieben werden, die auf der Basis von Meldungsdaten credit-Werte ermitteln und aus diesen beste Auftragnehmer bestimmen. Somit ergibt sich die Gesamtfunktionalität durch Verknüpfung der Auftraggeber- und Auftragnehmerfunktionen. Abbildung 1 zeigt dieses Verfahren im Ausschnitt für einen beliebigen Auftraggeber k und m Auftragnehmer.

Eine erste prototypische Implementierung des generellen Adaptionsverfahrens basierend auf einfachen Heuristiken zur Einstellung von credits und zur Aktivierung von geeigneten Auftragnehmern ist in [Lenzmann & Wachsmuth, 1996] beschrieben. In dieser Version adaptierte das System zwar bereits an wechselnde Präferenzen von Benutzern, jedoch erlaubte die einfache Repräsentation von credits durch skalare Werte eine nur eher kurzzeitige Adaption. Zudem war das Verfahren auf eine bestimmte Anzahl von Auftraggebern und Auftragnehmern beschränkt. Um ein flexibleres Adaptionsverhalten zu ermöglichen,

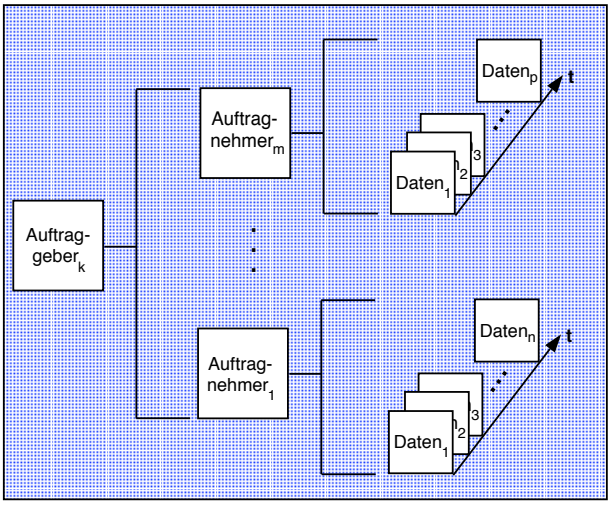

Abbildung 1: Ausschnitt des contract-net-basierten Lernverfahrens

wurden die Auftraggeber- und Auftragnehmerfunktionen, wie im weiteren beschrieben, verfeinert.

3.1 Agenten als Auftragnehmer

Die verfeinerte Idee zur Einstellung von credit-Werten und zur Bestimmung geeigneter Auftragnehmer ist durch typische Auftragsabwicklungen in Unternehmen motiviert: Ein Auftragnehmer gilt als erfolgreich, falls er viele Aufträge in der Vergangenheit (1) angeboten bekommen, (2) bearbeitet und (3) erfolgreich ausgeführt hat. Mit diesem Hintergrund werden credit-Werte als zeitabhängige Vektoren modelliert. Zu jedem Verhandlungsschritt t berechnet jeder Auftragnehmer *an* einer Unteragentur *ua* folgenden credit-Vektor:

$$credit_{ua}^{an}(t) \;=\; (zuversicht_{ua}^{an}(t), erf\_bearb\_auf\ {}_{ua}^{an}(t), bearb\_auf\ {}_{ua}^{an}(t), auf\ {}_{ua}^{an}(t))$$

wobei *auf* die Anzahl von Aufträgen bezeichnet, die an die Unteragentur *ua* gesendet wurden, *bearb_auf* die, die vom Auftragnehmer *an* bearbeitet wurden und *erf_bearb_auf* die, die von *an* erfolgreich bearbeitet wurden. *zuversicht* repräsentiert den augenblicklichen Wert, mit dem *an* annimmt, die aktuelle Aufgabe erfolgreich bearbeiten zu können. Da Auftragnehmer unterschiedlich erfolgreich mit unterschiedlichen Auftraggebern zusammenarbeiten können, legt jeder Auftragnehmer den obigen credit-Vektor für jeden seiner Auftraggeber an und manipuliert diesen in Abhängigkeit von dem aktuellen Auftraggeber.

Die Werte von *auf*, *bearb_auf* und *erf_bearb_auf* werden auf Basis der übertragenden Nachrichtendaten zu jedem Verhandlungsschritt t aktualisiert. Im einzelnen extrahiert jeder Auftragnehmer *an* einer Unteragentur *ua* folgenden Datenvektor:

$$daten_{ua}^{an}(t) \;=\; (kennung\ {}_{ua}^{an}(t), absender\ {}_{ua}^{an}(t), empfänger\ {}_{ua}^{an}(t), typ\ {}_{ua}^{an}(t), inhalt\ {}_{ua}^{an}(t))$$

wobei *typ* die Verhandlungsstrategie bezeichnet (Auftragsausschreibung oder direkte Auftragsvergabe) und *inhalt* die Art des Benutzer-Feedbacks (positiv oder negativ) repräsen-

tiert. Zur Berechnung der ersten drei Einträge des credit-Vektors werden diese Informationen nach bestimmten Regeln verwendet; so wird z.B. *auf* wie folgt berechnet:

$$auf\,{}^{an}_{ua}(t) := \begin{cases} 0, & \text{falls} \quad t = 0 \\ auf\,{}^{an}_{ua}(t-1)+1, & \text{falls} \quad (typ\,{}^{an}_{ua}(t) = \text{Ausschreibung} \wedge \\ & \qquad\quad empfänger\,{}^{an}_{ua}(t) = ua) \vee \\ & \qquad\quad (typ\,{}^{an}_{ua}(t) = \text{Auftrag} \wedge \\ & \qquad\quad empfänger\,{}^{an}_{ua}(t) = an) \\ auf\,{}^{an}_{ua}(t-1), & \text{sonst} \end{cases}$$

Der Wert von *auf* wird also um 1 inkrementiert, falls eine Ausschreibung an die Agentur *ua* oder ein Auftrag an den Auftragnehmer *an* geschickt wird. In ähnlicher Weise werden *bearb_auf* und *erf_bearb_auf* um 1 inkrementiert, wann immer ein Auftrag vom *an* bearbeitet wird bzw. ein Auftrag von *an* erfolgreich bearbeitet wird (d.h. positives Benutzer-Feedback). Der Wert von *zuversicht* wird unter Ausnutzung der bereits ermittelten credit-Vektoreinträge wie folgt aktualisiert:

$$zuversicht^{an}_{ua}(t) := \begin{cases} 1, & \text{falls} \quad (erf\_bearb\_auf\,{}^{an}_{ua}(t) - erf\_bearb\_auf\,{}^{an}_{ua}(t-1) = 1) \vee \\ & \qquad\quad [(bearb\_auf\,{}^{an}_{ua}(t) - bearb\_auf\,{}^{an}_{ua}(t-1) = 0) \wedge \\ & \qquad\quad (erf\_bearb\_auf\,{}^{an}_{ua}(t) \div bearb\_auf\,{}^{an}_{ua}(t) > \alpha) \wedge \\ & \qquad\quad (bearb\_auf\,{}^{an}_{ua}(t) \div auf\,{}^{an}_{ua}(t) > \beta)] \\ -1, & \text{falls} \quad (erf\_bearb\_auf\,{}^{an}_{ua}(t) - erf\_bearb\_auf\,{}^{an}_{ua}(t-1) = 0) \wedge \\ & \qquad\quad (bearb\_auf\,{}^{an}_{ua}(t) - bearb\_auf\,{}^{an}_{ua}(t-1) = 1) \\ 0, & \text{sonst} \end{cases}$$

Der Zuversichtswert ist demnach hoch, wenn der Auftragnehmer den Auftrag erfolgreich bearbeitet hat oder zwar den Auftrag nicht bearbeitet hat, aber sich in den vorangegangenen Interaktionen als dominant herausgestellt hat. Für den letzteren Fall werden zwei Schwellwerte α und β definiert, die bisher einen festen Wert von 70 Prozent annehmen (eine automatische Änderung dieser Werte, wie z.B. deren Abnahme mit zunehmender Dominanz eines Auftragnehmers, wird aber bereits in Experimenten untersucht). Ein Auftragnehmer hat eine niedrige Zuversicht, falls die zuletzt von ihm generierte Lösung negatives Feedback erhalten hat. In allen anderen Fällen ergibt sich eine mittlere Zuversicht. Die Verwendung eines dreigeteilten Zuversichtswertebereichs ist für die momentane Realisierung ausreichend, so daß eine Verfeinerung der Werteskala vorläufig nicht betrachtet wird.

Diese Art der credit-Werteinstellung liefert die Vorbereitung zur Realisierung der Anpassung an unterschiedliche Präferenzen wechselnder Benutzer sowie an temporär wechselnde Präferenzen des einzelnen Benutzers.

3.2 Agenten als Auftraggeber

Verschickt ein Auftraggeber eine Ausschreibung eines Auftrags, so senden alle verfügbaren Auftragnehmer Angebote zurück, in denen der aktuelle credit-Vektor enthalten ist. Zur Bestimmung des bisher erfolgreichsten Auftragnehmers evaluiert ein Auftraggeber jedes hereinkommende Angebot und vergleicht dieses mit den bereits vorliegenden, und zwar nach folgenden Heuristiken:

Regel 1:
Wähle *an* mit $\mathit{zuversicht}_{ua}^{an}$ maximal;
`If` nicht eindeutig `do` *Regel 2*;

Regel 2:
Wähle *an* mit $\mathit{erf\_bearb\_auf}_{ua}^{an} \div \mathit{bearb\_auf}_{ua}^{an} > \gamma$ und $\mathit{bearb\_auf}_{ua}^{an} \div \mathit{auf}_{ua}^{an} > \delta$;
`If` mehr als ein *an* genügt dem Test `do` *Regel 3*;
`else if` kein *an* genügt dem Test `do` *Regel 4*;

Regel 3:
Wähle *an* mit $\mathit{erf\_bearb\_auf}_{ua}^{an} \div \mathit{bearb\_auf}_{ua}^{an}$ maximal;
`If` nicht eindeutig `do` wähle ein *an* mit $\mathit{bearb\_auf}_{ua}^{an} \div \mathit{auf}_{ua}^{an}$ maximal;

Regel 4:
Wähle *an* mit $\mathit{erf\_bearb\_auf}_{ua}^{an} - \mathit{bearb\_auf}_{ua}^{an}$ minimal;
`If` mehr als ein *an* genügt dem Test `do` *Regel 3*;

Die priorisierte Betrachtung des Zuversichtswerts erlaubt, daß im Falle von negativem Feedback andere Auftragnehmer aktiviert werden können. Regel 2 ermöglicht die Modellierung kurzzeitiger Präferenzwechsel durch Ausnutzung von Dominanzen einzelner Auftragnehmer, so daß die dominante Präferenz eines Benutzers nicht durch eine temporäre, eventuell stark situationsbedingte Präferenzänderung vergessen wird. Wiederum sind die notwendigen Dominanzschwellwerte γ und δ zunächst auf einen festen Wert von 70 Prozent festgesetzt, wobei aber eine automatische Abnahme mit zunehmender Dominanz untersucht wird. Regel 3 beschreibt ein letztes Konfliktauflösungskriterium für die qualitative Unterscheidung von Auftragnehmern, und Regel 4 wählt Auftragnehmer, die eine minimale Anzahl an negativem Feedback verursacht haben.

Durch die Betrachtung des Zuversichtswertes im ersten Schritt und den Test auf Dominanz in den vorausgegangenen Interaktionen kann – zusammen mit den credit-Werteinstellungen – die durch die Auftragnehmerfunktionalität der Agenten (siehe 3.1) vorbereitete Adaption an Benutzerpräferenzen vollständig realisiert werden.

Um die Anzahl von Kommunikations- und Kooperationsschritten zu minimieren, vergibt ein Auftraggeber bei eindeutigen Voraussetzungen Aufträge direkt an ausgezeichnete Auftragnehmer und verzichtet somit auf die Ausschreibung und Auswertung von Angeboten. Hierfür verwaltet und evaluiert jeder Auftraggeber eine Auftragshistorie. Unter Ausnutzung der darin verfügbaren Informationen erhalten Auftragnehmer, die über eine Periode von Verhandlungen erfolgreich gearbeitet haben, Aufträge direkt. Ein ähnlicher Ansatz ist in [Dowell, 1995] gegeben, wo ein lernender contract-net-Algorithmus zur Reduzierung von Kommunikations-Overhead entwickelt wurde.

Jeder Agent der Interface-Agentur kann die Rolle eines Auftraggebers sowie eines Auftragnehmers annehmen und muß daher jeden der oben beschriebenen Vektoren bearbeiten können. Die Handhabung und Berechnung dieser Vektoren ist Bestandteil eines Kommunikations- und Kooperationsrahmens, mit dem jeder Agent ausgestattet ist. Hiermit können interne Funktionalitäten unabhängig entwickelt werden, und eine übergeordnete Kontrollinstanz ist nicht erforderlich.

4 Beispielanwendung

Das vorgestellte Lernverfahren ist innerhalb des VIENA-Systems realisiert, in dem eine 3D-Grafikszene mittels verbaler und gestischer Benutzereingaben interaktiv manipuliert werden kann [Wachsmuth & Cao, 1995]. Eine Interface-Agentur übersetzt dabei qualitative Anweisungen in interne, vom Grafiksystem interpretierbare Kommandos. Dazu ist eine Vielzahl verschiedener Aufgaben zu lösen, die an einzelne Agenten oder auch an Systeme von Agenten (Agenturen) verteilt werden.

Die Gesamtagentur besteht aus unterschiedlichen kommunizierenden und kooperierenden Unteragenturen, die über unterschiedliche Funktionalitäten verfügen. So berechnet z.B. eine Raumagentur räumliche Transformationen und eine Farbagentur ändert Farbe und Aussehen von Objekten. Neben Szenenobjekten, die referenziert und transformiert werden können, wurde zur Verbesserung der Navigations- und Interaktionsmöglichkeiten eine anthropomorphe Figur namens Hamilton in die Szene eingebracht, die von einer eigenen Agentur gesteuert wird.

Die Implementation des Lernverfahrens wurde anhand mehrerer Beispiele getestet, primär für Benutzerpräferenzen für unterschiedliche räumliche Bezugsysteme. In der Situation in Abbildung 2 können z.B. zwei mögliche Lösungen der Instruktion "Hamilton, gehe nach links" generiert werden. Auf Basis des hamilton-deiktischen Bezugsystems bewegt sich die Figur in Richtung 'H', während auf Basis des benutzer-deiktischen Bezugsystems eine Translation in Richtung 'U' erfolgt. Experimente mit dem VIENA-System haben gezeigt, daß beide Bezugsysteme relevant sind und in Abhängigkeit von den individuellen Benutzerpräferenzen ein Bezugsystem dem anderen vorgezogen wird [Jörding & Wachsmuth, 1996].

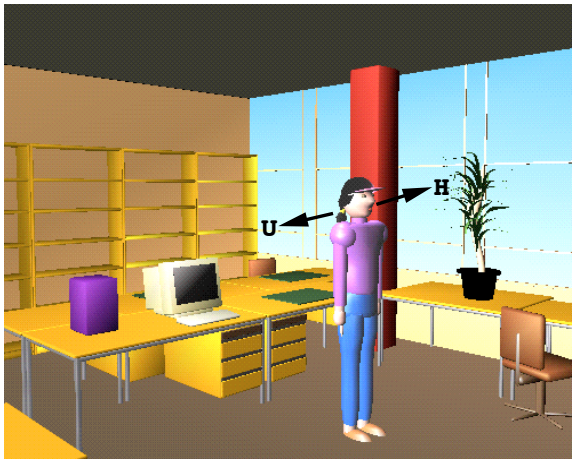

Abbildung 2: Mögliche Lösungen der Instruktion "Hamilton, gehe nach links".

In ähnlicher Weise wurden für die Raumagentur zwei Raumagenten realisiert, die räumliche Transformationen auf Basis des benutzer-deiktischen bzw. des objekt-intrinsischen Bezugsystems berechnen. Ferner wurde das Verfahren anhand von Benutzerpräferenzen für unterschiedliche Farbwahrnehmungen getestet.

Die durchgeführten Experimente haben gezeigt, daß das Lernverfahren eine Adaption an Benutzerpräferenzen effektiv und zufriedenstellend im Hinblick auf die gestellten Anforderungen umsetzt. Als wesentlicher Nutzen für die Domäne wird damit die intuitive Instruierbarkeit des Grafiksystems stark verbessert. Das vorgestellte Lernverfahren führt dazu, daß sich das System kurzfristig auf die Lösungserwartungen des Benutzers einstellt; zudem wird die explizite Speicherung persönlicher Daten vermieden.

5 Diskussion

In diesem Beitrag wurde ein Ansatz einer lernenden Interface-Agentur zur Adaption an Benutzerpräferenzen vorgestellt. Die Interface-Agentur stellt eine komplexe intelligente Schnittstelle zu einem technischen System dar, in diesem Fall einem interaktiven Grafiksystem. Die wesentliche Grundlage des hier beschriebenen Lernverfahrens ist, daß die Semantik einer Instruktion (die ja jeweils zu einer konkreten Systemantwort, in unserer Beispielanwendung zu einer Aktualisierung der visualisierten Szene führen muß) nicht absolut berechnet, sondern mit dem System *in der Interaktion verhandelt* wird [Wachsmuth & Cao, 1995]. Innerhalb der Interaktion kann der Benutzer vom System visualisierte Vorschläge durch weitere Anweisungen korrigieren. Dies beruht darauf, daß das System überhaupt – vermöge der alternativen Lösungsstrategien einzelner seiner Agenten – unterschiedliche Lösungsvorschläge liefern kann, die der Benutzer, entsprechend seiner Präferenz, entweder akzeptiert oder korrigiert. Das System versucht dann, die zunächst vorgeschlagene Lösung entsprechend zu modifizieren, wobei Erfolge bzw. Mißerfolge den an einer Lösung effektiv beteiligten Agenten zugeschrieben werden und zu einer in dem Multiagentensystem verteilten Lernleistung führen.

Agenten, organisiert in Unteragenturen, nutzen dabei ein contract-net-basiertes Lernverfahren, um sich entsprechend den Benutzeranforderungen selbst zu organisieren. Durch Ausnutzung von direktem und indirektem Benutzer-Feedback werden interne, zeitabhängige credit-Vektoren von Agenten als Auftragnehmer verwaltet, auf deren Basis Agenten als Auftraggeber erfolgreichste Auftragnehmer ermitteln. Wissen über Benutzer liegt somit implizit und in der Agentur verteilt vor, womit die Erstellung eines expliziten Benutzermodells vermieden werden kann.

Das Verfahren wurde innerhalb eines interaktiven 3D-Grafiksystems realisiert und getestet. Die bei informellen Versuchen beobachteten Benutzerpräferenzen zeigten, daß diese nicht starr und eindeutig sein müssen, sondern zusätzlich von aktuellen Situationsparametern beeinflußt werden. Dies unterstreicht noch die Rolle eines flexiblen Adaptionsverfahrens, das sich durch Abspeicherung von Benutzerprofilen schwerlich erreichen lassen dürfte. Daher sollen sich nächste Arbeiten auf die Untersuchung solcher Situationsparameter, deren Einfluß auf Benutzerpräferenzen und deren Integration in den Lernprozeß beziehen. Zudem sollen umfassendere Evaluationen des Lernverfahrens durchgeführt werden.

Literatur

[Davis & Smith, 1983] Davis, R., Smith, G. Negotiation as a Metaphor for Distributed Problem Solving. In Bond, A.H. and Gasser, L. (Eds.): *Readings in Distributed Artificial Intelligence* (pp. 333-356). Morgan Kaufmann, 1983.

[Dowell, 1995] Dowell, M.L. Learning in Multiagent Systems. Ph.D. Thesis at the Department of Electrical and Computer Engineering, University of South Carolina, 1995.

[Jörding & Wachsmuth, 1996] Jörding, T., Wachmsuth, I. An Anthropomorphic Agent for the Use of Spatial Language. Angenommen für ECAI-96 Workshop on Representation and Processing of Spatial Expressions.

[Laurel, 1990] Laurel, B. Interface agents: Metaphors with character. In Laurel, B. (Ed.): *The art of human-computer interface design* (pp. 355-365). Reading: Addison-Wesley, 1990.

[Lenzmann & Wachsmuth, 1996] Lenzmann, B., Wachsmuth, I. A User-Adaptive Interface Agency for Interaction with a Virtual Environment. In Weiss, G. & Sen, S. (Eds.): *Adaption and Learning in Multi-Agent Systems* (pp. 140-151). Berlin: Springer, 1996.

[Maes & Kozierok, 1993] Maes, P., Kozierok, R. Learning interface agents. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)* (pp. 459-465). AAAI Press/The MIT Press, 1993.

[Maes, 1994] Maes, P. Agents that Reduce Work and Information Overload. *Communications of the ACM 37*(7), 1994, 31-40.

[Mitchell *et al.*, 1994] Mitchell, T., Caruana, R., Freitag, D., McDermott, J., Zabowski, D. Experiences with a learning personal assistent. *Communications of the ACM 37*(7), 1994, 80-91.

[Norman, 1994] Norman, D.A. How Might People Interact with Agents. *Communications of the ACM 37*(7), 1994, 68-71.

[Wachsmuth & Cao, 1995] Wachsmuth, I., Cao, Y. Interactive Graphics Design with Situated Agents. In W. Strasser & F. Wahl (Eds.): *Graphics and Robotics* (pp. 73-85), Springer.