Antizipative Modellierung des Benutzerverhaltens mit Hilfe von Aktionsvorhersage-Algorithmen

ALEXANDER KÜNZER, FRANK OHMANN & LUDGER SCHMIDT

Institut für Arbeitswissenschaft, RWTH Aachen

Schlüsselwörter: Adaptive Benutzungsschnittstellen, Aktionsvorhersage-Algorithmen, Benutzermodellierung

1. Einleitung

In den letzten Jahren ist der Funktionsumfang von Softwareanwendungen immer stärker angestiegen. Solche hochfunktionellen Anwendungen (HFA) sind sehr komplex geworden, weil sie die Bedürfnisse großer und heterogener Gruppen von Benutzern mit unterschiedlichem Kenntnisstand, unterschiedlicher Erfahrung und auch für verschiedene Aufgabenbereiche zufrieden stellen müssen (Fischer 2001). Trotzdem sind leicht benutzbare Benutzungsschnittstellen (UIs) nicht weit verbreitet – trotz der Berücksichtigung eines benutzerzentrierten Entwicklungsprozesses (DIN EN ISO 13407) oder allgemeiner Gestaltungskriterien für interaktive Softwaresysteme (DIN EN ISO 9241-10). Daher besteht weiterhin ein Bedarf an intelligenteren Benutzungsschnittstellen, die Novizen ebenso wie Experten unterstützen, ihre Aufgaben durchzuführen (Lau 1999).

Im Bereich der Benutzermodellierung wurden verschiedene Ansätze entwickelt, z. B. Kommandovorschlagsliste (Greenberg 1993, Davison & Hirsh 1998), intelligente Hilfe- oder Tutorsysteme (IHS/ITS, Encarnação und Stoev 1999) oder sogenannte „glaubwürdige“ Agenten (Maes 1994, Horvitz et al. 1998). Diese Systeme basieren alle auf Benutzermodellen, mit denen eine Vorhersage des Benutzerverhaltens möglich ist. Das bedeutet, dass der Computer die Interaktion des Benutzers mit der Anwendung beobachtet und so zu ermitteln versucht, welche Ziele und Aufgaben der Benutzer verfolgt (z. B. „Was macht er oder sie gerade ?“, „Gibt es irgendwelche Probleme ?“).

In diesem Artikel wird die Entwicklung eines Benutzermodells vorgestellt, welches auf einer Analyse von Aktionssequenzen beruht. Mit Hilfe dieses Ansatzes wird anschließend ein Unterstützungssystem für den Anwender realisiert. Dazu werden sogenannte Aktionsvorhersage-Algorithmen implementiert, die problemlos in – auch bereits bestehende – Anwendungen integriert werden können. Außerdem erlaubt die Verwendung eines solchen standardisierten Ansatzes den einfachen Vergleich verschiedener Aktionsvorhersage-Algorithmen mit unterschiedlicher Vorhersagequalität. Als Anwendungsfall dient eine selbstentwickelte multimodale Benutzungsschnittstelle zur Steuerung des 3D-Laserschweißens in zukünftigen Autonomen Produktionszellen (APZ). Verschiedene aus der Literatur bekannte aber auch eigene Implementierungen von Aktionsvorhersage-Algorithmen wurden anhand eines realen Interaktionsszenarios verglichen, um Unterschiede und den möglichen Nutzen solcher einfachen Unterstützungssysteme zu bewerten.

2. Hintergründe

2.1 Adaptive Benutzungsschnittstellen

Obwohl es in den letzten Jahrzehnten vielfältige Entwicklungen im Bereich der Mensch-Rechner-Interaktion gab, besitzen die meisten Software-Anwendungen immer noch starre Schnittstellen, welche die großen Unterschiede menschlicher Interaktionstypen und -stile nicht berücksichtigen. Anpassbare Benutzungsschnittstellen erlauben es dem Benutzer zwar, die Benutzungsschnittstelle nach seinen eigenen Bedürfnissen zu modifizieren, normalerweise besitzt der Benutzer jedoch nicht genügend Kenntnisse der Anwendung und über seine eigenen Bedürfnisse, so dass diese Art der Individualisierung meist nicht genutzt wird (Ross 2000). Einen besseren Ansatz bieten sogenannte adaptive Benutzerschnittstellen (AUI), welche die Bedürfnisse des Anwenders vorherzusehen versuchen, um die Benutzungsschnittstelle entsprechend anzupassen.

Adaptive User Interfaces (AUI) können als eine Art „intelligente“ Schnittstelle angesehen werden, bei der die Benutzungsschnittstelle mit Techniken der Künstlichen Intelligenz (KI) versucht, die Ziele und Bedürfnisse des Anwenders zu antizipieren und sich diesen anzupassen (Dieterich et al. 1993). Eine andere Definition von AUIs stammt von Langley (1997).

> *“An adaptive user interface is a software artifact that improves its ability to interact with a user by constructing a user model based on partial experience with that user.”*

Ein Benutzermodell kann eingesetzt werden, um Ziele und Präferenzen des Benutzers vorherzusagen oder Verhaltensmuster zu entdecken, z.B. zur spekulativen Befehlsausführung, Eingabevervollständigung, Ausführungsbeschleunigung oder Unterstützung. Beispiele hierzu sind z. B. das Lumière-Projekt (Horvitz et al. 1998), welches später kommerziell in den MS Office Produkten eingesetzt wurde. Ein anderes Beispiel stammt von Davison und Hirsh (1998), die ein System zur Vorhersage von Unix-Kommandos entwickelten, das von Korvemaker und Greiner (2000) auf vollständige Unix-Kommandos inkl. Parameter ausgedehnt wurde.

2.2 Generische Benutzermodellierung durch Beobachtung von Handlungssequenzen

Intelligente Benutzungsschnittstellen wurden bereits in vielen Projekten erforscht. Im jeweiligen Anwendungsumfeld erreichten sie dabei oftmals beeindruckende Ergebnisse. Jedoch führte die Übertragung auf andere Anwendungsfelder zu neuen Herausforderungen (Gorniak & Pool 2000). Viele der manuell modellierten Ansätze (z. B. Aufgaben- und Benutzermodelle) sind zu aufwändig und unflexibel (Carberry 2001), wie z. B. im Lumière Projekt, in dem Experten das Benutzerverhalten mit Bayes'schen Netzen modellierten oder wie in VAMPIRE ("Visual Model-based Pick-And-Place InteRface Editor", Eisenstein and Rich 2002), bei welchem Aufgaben feingranular in Form von Makros beschrieben werden müssen. Es besteht daher ein Bedarf an generischen Ansätzen, die von der Domäne unabhängig sind und allein aus dem bisherigen Verhalten des Benutzers lernen können (Kobsa 2001).

In dieser Veröffentlichung werden verschiedene Ansätze betrachtet, die weitgehend unabhängig von einer speziellen Domäne oder Anwendung sind. Zu diesem Zweck werden Benutzeraktionen mit Domänen- bzw. Aufgabenbezug beobachtet, d. h. die Aktionen sind im Rahmen des semiotischen Modells (Foley et al. 1990) auf der semantischen Ebene anzusiedeln. Diese Verdichtung der Aktionen reduziert die tatsächlich zu berücksichtigenden Aktionen auf diejenigen mit Aufgabenbezug, welche dann im weiteren als syntaktische Interaktionssequenz kontextfrei weiter betrachtet werden (Hilbert & Redmiles 1999). In diesen Ansatz kann auch das Zustandsmodell der Anwendung berücksichtigt werden, in dem die beobachtete Benutzerinteraktion durch entsprechende Ereignisse der Anwendung angereichert werden.

Ursprung dieser Aktionen sind die physikalische Ereignisse, welche über die Eingabegeräte zeitlich verteilt, nacheinander in das Betriebssystem gelangen (vgl. Abbildung 1). Hier führen diese Ereignisse zur entsprechenden Ereignissen des Betriebssystems und werden dann über eine abstrakte Interaktionsebene bis möglicherweise zur Ebene mit Domänen- bzw. Aufgabenbezug weitergeleitet. Allerdings kann dabei auf jeder Ebene ein Filterprozess stattfinden, falls das Ereignis nicht auf höheren Ebenen benötigt wird (bspw. müssen die meisten Anwendungen keine Informationen über Mauszeigerbewegungen erhalten). Der linke Aktionsstrang in Abbildung 1 zeigt eine Aktion welche keine Auswirkungen auf die eigentliche Aufgabe besitzt (z. B. kann es sich um das Zoomen einer Abbildung handeln) und daher vor der obersten Ebene herausgefiltert wird.

Auch die Benutzungsschnittstelle kann wie im rechten Aktionsstrang Ereignisse auslösen, um etwa eine Nachricht anzuzeigen. Neben dem Filtern besteht auch die Möglichkeit der Aggregation von Aktionen. Dadurch können auch komplexere Aktionsmuster oder -sequenzen auf die für die Beobachtung wesentlichen Aktionen reduziert bzw. umgewandelt werden. Im Beispiel in Abbildung 1 ist das Drucken über einen Druckdialog, welcher dann mit OK bestätigt wird gezeigt – alternativ könnte die resultierende Aktion `Document.Print` auch über eine Werkzeugleiste direkt ausgelöst werden.

Basierend auf diesen Aktionssequenzen ist es dann möglich, mit adäquaten Benutzermodellen die nächsten möglichen Aktionen des Benutzers vorherzusagen. Einerseits können diese Vorhersagen das Interaktionsverhalten von Experten berücksichtigen, so dass ein Unterstützungssystem für Novizen realisiert werden kann. Andererseits ist es aber auch möglich, die Interessen des Anwenders selbst als Grundlage zu

verwenden, um Shortcuts vorzuschlagen bzw. eine individuelle Anpassung der Oberfläche durchzuführen. Eine Kombination beider Ansätze ist ebenfalls möglich.

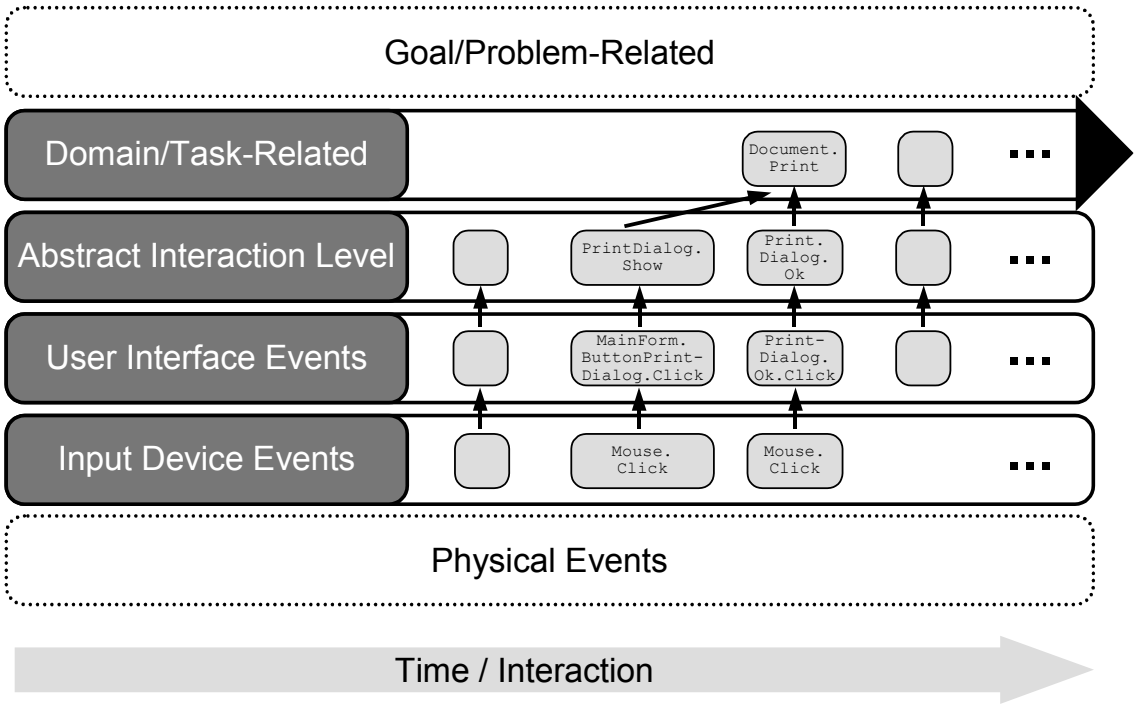

Abbildung 1: Ebenen der Abstraktion in Benutzerschnittstellen nach Hilbert und Redmiles (1999) mit einem Filter- und Sammlungsprozess zur Ermittlung domänen-/aufgabenorientierter Aktionen (kleine Boxen = Aktionen).

2.3 Aktionsvorhersage-Algorithmen (AVAs)

Die eigentliche Vorhersage der Benutzeraktionen wird mit Hilfe eines Aktionsvorhersage-Algorithmus (AVA, engl. Action Prediction Algorithm) durchgeführt. Dabei werden die möglichen nächsten Aktionen samt der Wahrscheinlichkeiten prognostiziert, indem die vorherigen Aktionen als Eingabe dienen. Abbildung 2 zeigt die Funktionsweise eines AVAs schematisch. AVAs können in verschiedener Weise implementiert werden, z. B. durch (Hidden) Markov Modelle (Cook & Wolf. 1998; Zukerman et al. 1999) oder Bayes'sche Netze (Horvitz et al. 1998; Schlick et al. 2002). Nachteilig ist dabei häufig, dass viele Lernfälle benötigt werden, um die Modelle adäquat anlernen zu können (Fischer 2001). Dies kann problematisch sein, falls für eine Anwendung entsprechend viele Trainingsdaten nur schwer zu erhalten sind.

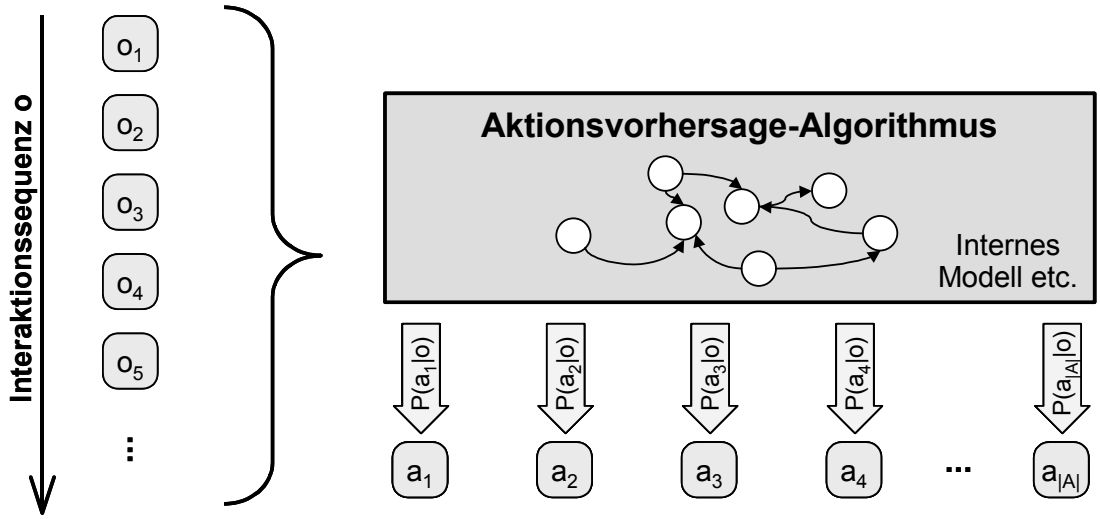

Abbildung 2: Schematische Darstellung eines Aktionsvorhersage-Algorithmus.

Sun (2000) gibt die folgende Formulierung für Sequenz-Vorhersagen: Sei $s_i \in S$ ein Sequenz-Element, dann kann das Problem beschrieben werden als eine Wahrscheinlichkeitsverteilung für die Elemente der Sequenz mit $p(s_{j+1} \mid s_i, s_{i+1}, \ldots, s_j)$, wobei $1 \le i \le j < \infty$. Wenn $i = 1$ ist, basiert die Vorhersage auf allen vorhergehenden Elementen. Gemäß der Idee eines idealen Online-Lern-Algorithmus (IOLA) von Davison und Hirsh (1998) und eigenen Erfahrungen auf diesem Gebiet kann ein AVA wie in Formel 1 angegeben definiert werden, wobei A* die Menge aller endlichen Aktionssequenzen über dem endlichen Alphabet A, also der Menge aller möglichen Aktionen der Anwendung, ist.

$$\phi : A^* \to R^{|A|} : o_{1:m} \mapsto (p_1, p_2, \ldots, p_{|A|}) \qquad (1)$$
$$mit\ p_i = P(a_i \mid o_{1:m})\ und\ a_{1 \le i \le |A|}, o_{1 \le j \le m} \in A$$

3. AVA-basierte Benutzer-Unterstützungssysteme

3.1 Vorgehen zur AVA-basierten Unterstützung

Der Ablauf der Unterstützung mithilfe eines AVA ist in Abbildung 3 visualisiert und ergibt sich direkt aus der Funktionsweise eines AVA. Basierend auf den beobachtbaren Aktionen des Benutzers wird zuerst der Filter- und Aggregationsprozess erforderlich, um die abstrakten Aktionen auf der semantischen Ebene zu aggregieren. Diese Interaktionssequenz stellt dann die Eingabe für den verwendeten AVA dar und kann mögliche Folgeaktionen samt Wahrscheinlichkeiten vorhersagen, die dann zur adaptiven Anpassung der Anwendung bzw. der Unterstützungskomponente genutzt werden.

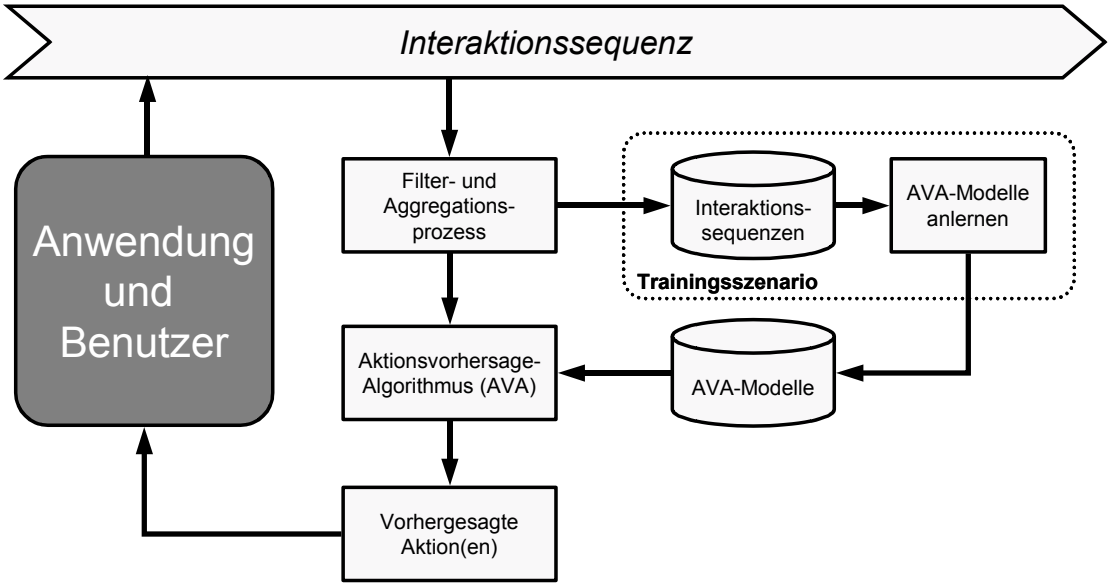

Abbildung 3: Schematisches Ablaufdiagramm zur adaptiven Unterstützung mittels eines AVA

Im Trainingsfall können mit demselben Mechanismus auch Interaktionssequenzen aufgezeichnet und gesammelt werden, die dann zum Anlernen des AVA-Modells genutzt werden.

3.2 Adaptive Hilfe- und adaptive Tutorsysteme

Es wurden zwei Unterstützungssysteme entwickelt, die auf AVAs basieren. Abbildung 4 zeigt eine Bildschirmdarstellung des adaptiven Tutors (links) und der adaptiven Hilfe (rechts). Der adaptive Tutor ermöglicht dem Benutzer eine aktivere Interaktion, während die adaptive Hilfe passiv gestaltet ist. In beiden Unterstützungssystemen lassen sich aber Statusinformationen anzeigen (1), welche dem Benutzer aktuelle Vorgänge erläutern können. Der adaptive Tutor zeigt die vorgeschlagenen Aktionen (2) samt entsprechendem Hilfetext an. Die Hyperlinks erlauben es dem Benutzer, die angezeigte Aktion direkt über den Tutor auszuführen. Das adaptive Hilfesystem (rechts) hat ähnliche Elemente, erlaubt aber keine direkte Interaktion.

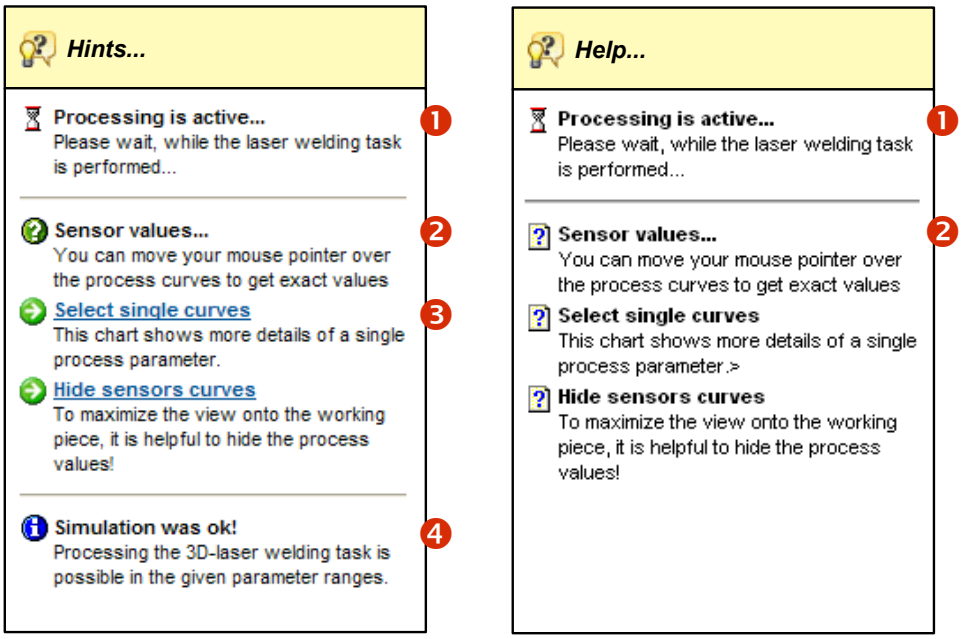

Abbildung 4: Beispiel des adaptiven Tutors und der adaptiven Hilfe.

Sowohl die adaptive Hilfe als auch der adaptive Tutor zeigen die vorgeschlagenen Aktionen und Hinweise basierend auf dem bisherigen Verhalten des Benutzers an. Das initiale Aufgabenmodell der Anwendung wird aus der Beobachtung von erfahrenen Benutzern konstruiert und enthält so deren Wissen bzgl. der Applikationsnutzung. Für den Fall, dass das Interaktionsverhalten eines einzelnen Benutzers zu Grunde gelegt wird, kann die Hilfe bzw. der Tutor lernen, persönliche Interessen und Vorlieben des Benutzers zu unterstützen, so dass Eingabevervollständigung, Shortcuts aber auch Unterstützung realisierbar ist. Eine spekulative Befehlsausführung, d. h. der Computer führt die Aktionen des Benutzers pro-aktiv aus, wird nicht verfolgt, da absolut korrekte Vorhersagen nicht garantiert werden können.

Der generische Ansatz erlaubt eine einfache – auch nachträgliche – Integration in vorhandene Anwendungen. Eine explizite Modellierung des Domänen- bzw. Anwendungswissens ist nicht erforderlich. Die Anpassung basiert auf einem AVA, der die wahrscheinlichsten Aktionen vorhersagt. Somit ist die Hilfe / der Tutor unabhängig von einer speziellen Implementierung und erlaubt es, den verwendeten AVA auszutauschen, z. B. aus Leistungsgründen. Dadurch kann sowohl der adaptive Tutor als auch die adaptive Hilfe in beliebige Anwendungen integriert werden. Ansonsten können die Vorhersagen auch benutzt werden, um andere Unterstützungsarten zu implementieren.

Um den Nutzen des adaptiven Hilfesystems zu erhöhen, werden nicht nur die vorgeschlagenen Aktionen, sondern auch einige zusätzliche Elementtypen verwendet:

Konzepte. Die Ansammlung verschiedener (auch identischer) Aktionen (z.B. Bewegen und Positionieren eines 3D-Objektes) kann durch ein Konzept wie in Abb. 3 (2) zusammengefasst werden. Konzepte werden ebenfalls bei der Prädiktion von Aktionen berücksichtigt.

Status. Statusmitteilungen (z.B. „Aufgabe wird erledigt“) werden zusätzlich im Tutor angezeigt (siehe Abbildung 4 (1)). Im Falle der intensiven Nutzung des Tutors ist

dadurch sichergestellt, dass Mitteilungen der Anwendung vom Benutzer nicht übersehen werden.

Information. Informationselemente werden benutzt, um den Benutzer über wichtige Situationen innerhalb der Anwendung zu informieren (ähnlich wie bei den Status-Elementen), wie in Abb. 4 (4). Informationselemente werden aber im Gegensatz zu Statusmitteilungen bei der Prognose weiterer Aktionen berücksichtigt.

3.3 Integrationsanforderungen für aktionsbasierte Unterstützungssysteme

Eine Integration dieses generischen Ansatzes für die adaptive Hilfe bzw. den adaptiven Tutor ist einfach und erfordert das Einfügen des Vorhersagemoduls sowie einer Modifikation der vorhandenen Applikation in folgendender Hinsicht:

Aktionen. Alle Aktionen, welche relevante Aspekte der Anwendung abdecken, müssen definiert werden. Dazu müssen Bezeichnungen und Hinweise angegeben werden, welche in der Hilfe bzw. dem Tutor angezeigt werden. Heutzutage unterstützen Betriebs- und Programmiersysteme Ereignis- (z. B. MS Windows) und Aktionsbehandlungen (z.B. Borland's Delphi), auf die zurückgegriffen werden kann.

Logging. Die Anwendung muss fähig sein, alle definierten Aktionen aufzuzeichnen. Dadurch werden Interaktionssequenzen gewonnen, welche die Basis zur Vorhersage weiterer Aktionen bilden.

Remoting. Um dem Benutzer die Navigation über den Tutor zu ermöglichen, ist es nötig, dass die Funktionen der Anwendung durch den Tutor aufgerufen werden können. Eine Makrofähigkeit bzw. ein Nachrichtensystem (z. B. wie in Windows) können dazu verwendet werden.

3.4 Einsatzbereiche für AVA-basierte Unterstützungsfunktionen

Ein AVA-basierter Unterstützungsansatz bietet sich vor allem für Mensch-Maschine-Systeme an , welche einen hohen Anteil an prozessorientierter Navigation aufweisen. Die im folgenden vorgestellte Anwendung ACTIVE-UI erfüllt diese Anforderungen, genauso wie bspw. die Interaktion auf Webseiten. Mensch-Maschine-Systeme mit hohem inhaltlichen Ausrichtung und geringerer Frequenz der Interaktionsereignisse sind hingegen für den AVA-Ansatz weniger geeignet (z. B. Konstruktion via CAD-System, Operator-Schnittstellen). Allerdings kann generell der Zustandsraum eines Mensch-Maschine-Systems durch Berücksichtigung entsprechender Systemereignisse auch bei AVA-Ansätzen berücksichtigt werden.

4. Implementierung der Aktionsvorhersage-Algorithmen

4.1 Aus der Literatur bekannte Aktionsvorhersage-Algorithmen

Um einen geeigneten AVA für das adaptive Unterstützungssystem auszuwählen, wurden verschiedene Modelle mit dem Ziel implementiert, Unterschiede und Nutzen vergleichen zu können. Markov-basierte Algorithmen haben eine geringe Vorhersagegüte und benötigen umfangreiche Trainingsdatenbestände (Luczak et al. 2001, Künzer & Luczak 2003).

4.1.1 Der IPAM-Algorithmus

Der IPAM-Algorithmus (Incremental Probabilistic Action Modeling) wurde von Davison und Hirsh (1998) vorgestellt. Es ist ein einfacher AVA, der aber gute Ergebnisse liefert und dazu ein einfaches Markov-Modell verwendet, das durch ein spezielles Online-Lernverfahren aktualisiert wird. Der Parameter alpha des IPAM-Algorithmus beschreibt die Gewichtung der zuletzt aufgetretenen Aktionen in den Vorhersagen und wird von Davison und Hirsh zwischen 0,6 und 0,8 empfohlen. Während der ursprüngliche IPAM-Algorithmus auf einem Markov-Modell erster Ordnung beruht, haben die Autoren IPAM für höhere Ordnungen erweitert (genannt IPAM n –Algorithmus).

4.1.2 Die PPM-Algorithmen

Der PPM-Algorithmus (Prediction by Partial Matching) entstand ursprünglich aus einem Kompressionsalgorithmus (Ziv & Lempel 1977), der finite Kontexte benutzt und auch zur Vorhersage von Benutzeraktionen verwendet werden kann (Cleary & Witten 1984). Ein PPM-Algorithmus der Ordnung n besteht aus n+1 Markov-Vorhersagern. PPM+ ist eine Erweiterung von PPM, die zusätzlich einen Startvektor für das erste Ereignis einer Sequenz verwendet, anstatt dort die absoluten Häufigkeiten zu berücksichtigen. PPMU+ benutzt eine spezielle Anlerntechnik, bei der nur die längste gefundene Sequenz beim Anlernen aktualisiert wird (Howard und Vitter 1994). PPM* besitzt keine maximal vorgegebene Markov-Ordnung und wurde von Cleary et al. (1995) vorgestellt. PPMC+ berücksichtigt bei der Prognose auch kürzere Sequenzen als die maximal gefundene und wurde von Moffat (1990) präsentiert.

4.2 Neuentwickelte Aktionsvorhersage-Algorithmen

4.2.1 Der Aktionsvorhersage-Algorithmus LEV

Es wurde ein eigener AVA entwickelt, der auf der Ähnlichkeit von Sequenzen basiert. Als Ähnlichkeitsmaß wird die Levenshtein-Distanz (Levenshtein 1966) herangezogen. Da der vorliegende AVA dieses verbreitete Ähnlichkeitsmaß verwendet, wurde er „LEV-Algorithmus“ genannt. Dabei ist die Grundidee, auch dann gültige Vorhersagen zu treffen, wenn der Benutzer Fehler oder unnötige Schritte während der Interaktion im existierenden System macht. Dazu versucht der Algorithmus zunächst eine Vorhersage aufgrund von exakten Übereinstimmungen mit Sequenzen aus den Trainingssequenzen zu treffen. Wenn keine exakten Übereinstimmungen existieren, versucht der Algorithmus Übereinstimmungen mit der Levenshtein-Distanz 1 zu finden, dann mit Distanz 2 usw. Für den Fall, dass exakte Übereinstimmungen vorhanden sind, ist das vom Algorithmus gelieferte Ergebnis identisch mit dem eines Markov-Modells der vorgegebenen Ordnung. Ansonsten wird die Suche ausgeweitet auf die Suche nach ähnlichen Treffern. Abbildung 5 zeigt den Pseudo-Quellcode des LEV-Algorithmus.

```
function SeqCount(o: Sequence; LevDist: Integer): Integer;
begin
  Result := 0;
  for each o'_{Subseqs of len |o|} in O do
    if CalcLevenshteinDistance(o',o) = LevDist then
      inc(Result);
end;

function CalcPrediction(o: Sequence): Vector;
begin
  Result := ZeroVector;
  PredLen := Min(MarkovOrder, |o|);
  for i := 0 to PredLen + 1 do
  begin
    for each a in A do
      Result[a] := SeqCount(o_{|o|-PredLen+1;|o|}a, i);
    if Result <> ZeroVector then break;
  end;
  Normalize(Result);
end;
```

Abbildung 5: Pseudo-Quellcode für den Levenshtein-Algorithmus.

4.2.2 Der Aktionsvorhersage- Algorithmus KO

Nach der Handlungsregulationstheorie (z. B. Hacker 1978) finden Benutzerinteraktionen in hierarchisch-sequentiell strukturierten Handlungssequenzen statt (Abb. 5). Daher wurde ein weiterer AVA entwickelt, welcher als „KO-Algorithmus“ bezeichnet wurde (da er andere AVAs in Bezug auf die Vorhersagequalität AVAs „ausschaltete“). Der KO-Algorithmus berücksichtigt nicht nur die längste gefundene Sequenz zur Prognose (wie bspw. PPM), sondern zieht dazu auch kürzere Sequenzen heran. Besteht eine beobachtete Sequenz wie im Beispiel in Abbildung 6 aus zwei Untersequenzen H_2H_1, so gibt es drei mögliche Vorhersagen in der Trainingssequenz: H_2H_1 aber auch zwei Mal die Untersequenzen H_1. Dadurch kann der KO-Algorithmus die Aktion Y zusätzlich zur Aktion X vorhersagen.

Der KO-Algorithmus kombiniert den Ansatz der mindestens vorkommenden Untersequenzen (z. B. Mannila et al. 1997) mit einer Nearest-Neighborhood-Technik, wobei alle geeigneten Teilsequenzen genutzt werden, um Vorhersagen zu treffen. Dieser selbstentwickelte AVA benutzt zusätzlich Gewichte w(i), abhängig von der Länge der zu vergleichenden Teilsequenz. Gute Ergebnisse wurden dabei mit der Gewichtungsfunktion $w(i) = i^{19}$ erzielt. Abbildung 7 zeigt den Pseudocode für den KO-Algorithmus.

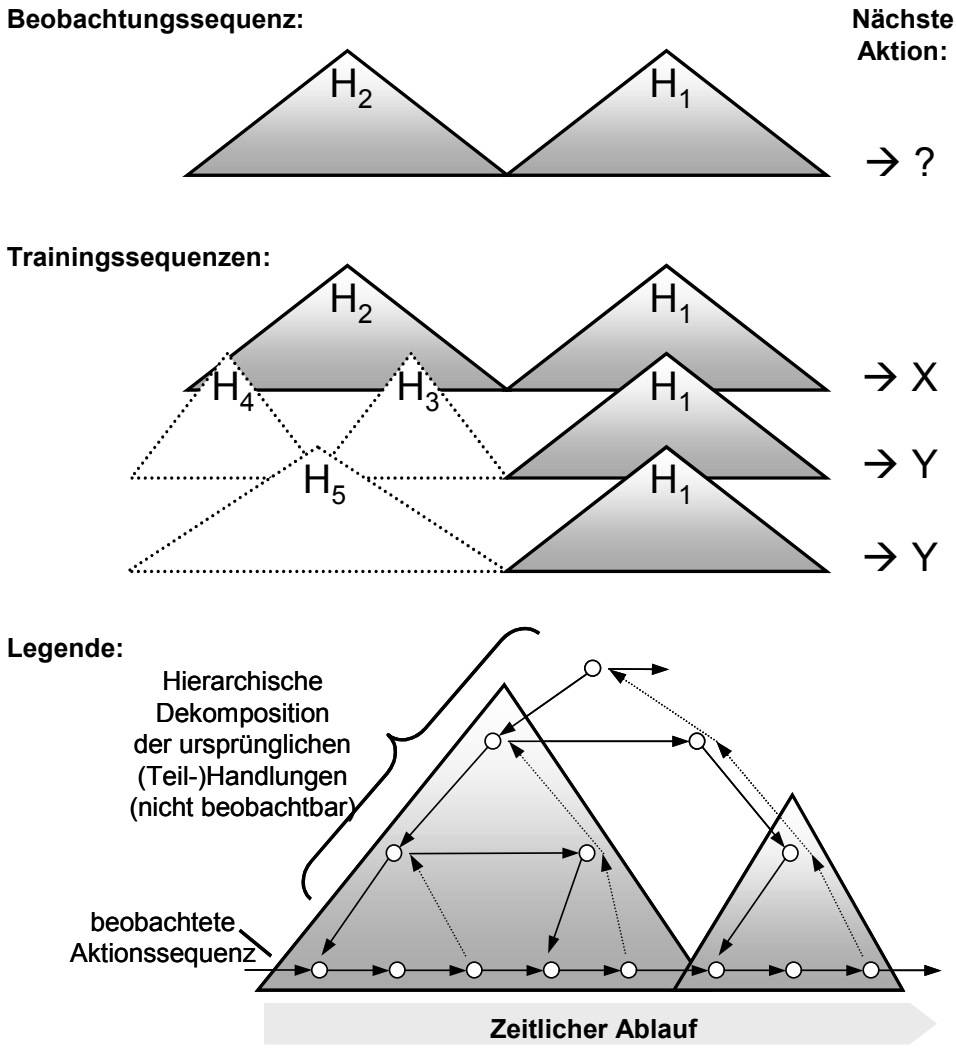

Abbildung 6: Beispiel für die Vorhersage einer Aktion basierend auf hierarchisch-sequentiellen Handlungssequenzen nach Hacker (1978)[1].

Eine weiterführende Betrachtung des KO-Algorithmus wurde von Künzer (2004) durchgeführt, wobei unterschiedliche Gewichtungsfunktionen und – durch eine verbesserte Implementierung – auch höhere Exponenten im Rahmen einer Grenzwertbetrachtung untersucht wurden. Dabei erwies sich der KO-Algorithmus ein als universeller AVA-Ansatz, wobei allerdings die ursprüngliche Hypothese in Bezug auf die Berücksichtigung unterschiedlicher Sequenzlängen aber verworfen werden musste. Vielmehr konvergiert der KO-Ansatz gegen eine neue Variante des PPM-Ansatzes, bei welchem die höheren Vorhersagewahrscheinlichkeiten auf den initialen Filterprozess der Mindestfrequenzen zurückzuführen sind. Die über den KO-Algorithmus hergeleitete neue Variante des PPM*-Algorithmus mit Pruning wird daher als PPMP* bezeichnet. Es zeigte sich aber, dass die Wahl der Gewichtungsfunktion $w(i) = i^{19}$ Vorhersagewahrscheinlichkeiten erzielt, welche annähernd den maximalen, vom PPMP*-Algorithmus zu erzielenden Werten, entspricht.

[1] Die hierarchisch-sequentiellen Handlungsstrukturen sind im Beispiel durch die Dreiecke nur angedeutet. Während im zeitlichen Ablauf lediglich eine Aktionssequenz zu beobachten ist (Legende, unten), stecken dahinter eigentlich die hierarchisch-verfeinerten Pläne. Auch die Handlungen (Dreiecken) können sich ihrerseits auch aus Teil-Handlungen zusammensetzen.

```
function SubseqCount(o: Sequence): integer;
begin
  Result := 0;
  for each o' in O do
    Result := Result + CountSubseqsInSeq(o, o');
  if Result < MinimumFrequence then Result := 0;
end;

function CalcPrediction(o: Sequence): Vector;
begin
  Result := ZeroVector;
  for each a in A do
  begin
    Result[a] := SubseqCount(a) * w(0);
    for i := 1 to min(|o|, SearchDepth - 1) do
      Result[a] := Result[a] + SubseqCount(o_{|o|-i+1:|o|}a) * w(i);
  end;
  Normalize(Result);
end;
```

Abbildung 7: Pseudocode zur Vorhersage mit dem KO-Algorithmus.

5. Voruntersuchung: Früher Prototyp von ACTIVE-UI

5.1 Fallstudie

Als Fallstudie diente eine selbstentwickelte multimodale Benutzungsschnittstelle namens ACTIVE-UI ("Autonomous Production Cells' Multimodal and Adaptive – User Interface", Schlick 2000) zur Überwachung von 3D-Laserschweißvorgängen in künftigen Autonomen Produktionszellen. Damit ACTIVE-UI sowohl für Novizen als auch für Experten besser benutzbar ist, soll ACTIVE-UI durch ein adaptives Hilfesystem auf AVA-Basis ergänzt werden. Um hierzu von realen Benutzern typische Interaktionssequenzen zu sammeln, wurde ein Experiment konzipiert, bei dem 30 Benutzer mit Erfahrung im Umgang mit ACTIVE-UI ohne Tutor ein typisches Interaktionsszenario zu bearbeiten hatten (Künzer et al. 2001). Als Interaktionsfall sollte ein Schweißauftrag durchgeführt werden (Konfigurieren der Sensoren, Auswahl eines numerischen Kontrollprogramms, Simulation und Prozessanalyse, sowie Überprüfen von wichtigen Prozesswerten).

5.2 Anlernen der Aktionsvorhersage-Algorithmen

Die Algorithmen benötigen verschiedene Parameter, u. a. die Ordnung der Markoketten. Die Auswahl dieser Werte kann durch Kreuzvalidierung und einer Anzahl gegebener Trainingsfälle systematisch getestet werden. Kommen neue Anwendungsfälle hinzu, können die Parameter später erneut geschätzt werden, so dass ständig eine Anpassung an das aktuelle Benutzerverhalten erfolgt. In der hier vorgestellten Untersuchung wurden immer die optimalen Parameter für jeden Algorithmus verwendet.

5.3 Verfahren

Die aufgezeichneten Interaktionssequenzen von 30 Benutzern wurden verwendet, um verschiedene AVAs als unabhängige Variable zu bewerten. Die Vorhersagewahrscheinlichkeit als abhängige Variable wurde durch die mittlere Vorhersagewahrscheinlichkeit (Mean Prediction Probability, MPP) operationalisiert, wie sie in Formel 2 und 3 angegeben sind (Künzer et al. 2001).

Zusätzlich wurde die mittlere Prognosegenauigkeit (Mean Prediction Accuracy, MPA) benutzt, die ähnlich berechnet wird, nur dass korrekte Vorhersagen mit 1 bewertet werden und falsche Vorhersagen mit 0. Obwohl die MPA weniger aussagekräftig ist (Albrecht et al. 1998), wird sie doch in anderen Publikationen zu diesem Thema oft herangezogen.

$$PP(o,\phi) = \frac{1}{|o|}\sum_{i=1}^{|o|} P_\phi\left(o_i \middle| o_{1:i-1}\right) \tag{2}$$

$$MPP(O,\phi) = \frac{1}{|O|}\sum_{o\in O} PP\left(o,\phi\right) \tag{3}$$

Sowohl MPP (und bei den anderen Untersuchungen auch die MPA) werden mit einer sechsfachen Kreuzvalidierung berechnet, wobei alle Interaktionssequenzen auf sechs Gruppen verteilt werden. Fünf dieser Gruppen wurden als Trainingsbasis benutzt, um den AVA anzulernen, die sechste Gruppe dient dann zur Evaluation des AVAs. Um den Effekt einer ungünstigen Verteilung der einzelnen Sequenzen auf die sechs Gruppen zu vermeiden, werden die Berechnungen mit insgesamt fünf Replikationen wiederholt. Jede dieser fünf Replikationen enthält dabei eine zufällige Permutation der Trainingsbasis.

5.4 Ergebnisse

Die durchschnittliche Dauer der Aufgabenbearbeitung betrug 346 Sekunden (zwischen 208 und 598 Sekunden). Während dieser Zeit benötigten die Benutzer 14 bis 29 Schritte (im Mittel 23) und nutzten 14 bis 26 verschiedene Aktionen (durchschnittlich 22) von insgesamt 34 vorhandenen.

Anschließend wird eine einfaktorielle ANOVA für die Variable *MPP* durchgeführt. Die Nullhypothese ist dabei, dass die *MPP* der stochastischen Modelle gleich ist. Die alternative Hypothese postuliert einen signifikanten Unterschied von wenigstens zwei AVAs ($\alpha_{\text{ANOVA}} = 0.05$). Da die Nullhypothese verworfen werden musste, wurden mit dem Post-hoc Student-Newman-Keul Test die signifikanten Unterschiede zwischen den AVAs überprüft ($\alpha_{\text{Posthoc}} = 0.05$).

Die Ergebnisse der Untersuchung sind in Abbildung 8 dargestellt. Die 95%-Konfidenzintervalle zeigen eine überlegene *MPP* von KO 3/19 (*MPP*=0,5464), vor PPM+ 4 (*MPP*=0,5305) und PPMU+ 4 (*MPP*=0,5233). Der PPMC+ 4 Algorithmus und alle IPAM-Implementierungen hatten eine niedrigere *MPP*.

Die einfaktorielle ANOVA ergab einen F-Wert von F_{ANOVA}=65,093 (df=9). Da der Wahrscheinlichkeitswert des F-Tests die kritische Marke von p_{ANOVA}<0,001 unterschreitet, wurde die Nullhypothese verworfen. Der Post-hoc Test ergab, dass die Unterschiede zwischen KO 3/19, PPM+ 4 und PPMU+ 4 nicht signifikant sind. Alle homogenen Untergruppen sind in Abbildung 8 eingezeichnet. Die *MPP* des neuen LEV-Algorithmus ist nicht unter den höchsten Werten, hatte aber dennoch einen höheren Wert als die IPAM- und PPMC+ Algorithmen.

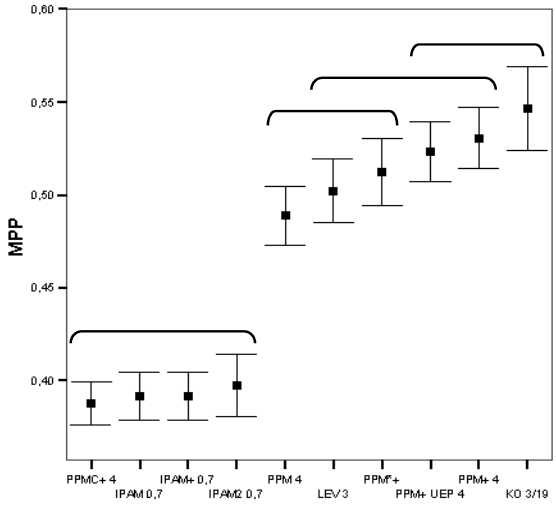

Abbildung 8: 95% Konfidenzintervalle von MPP *mit signifikanten homogenen Untergruppen* ($\alpha_{\text{Posthoc}} = 0.05$).

6. Voruntersuchung: Greenberg Referenzdateien

Um diese Ergebnisse besser mit anderen Publikationen vergleichen zu können, wurde die Vorhersagewahrscheinlichkeit und -genauigkeit der einzelnen Algorithmen ebenfalls anhand der Greenberg-Dateien getestet (Greenberg 1988). Die acht von Greenberg zur Verfügung gestellten Dateien enthalten jeweils zwei Dateien aus vier unterschiedlichen Gruppen. Diese Dateien wurden in einzelne Sessions unterteilt, die Kommandoparameter entfernt und durch einheitliche Aktions-IDs innerhalb jeder Datei umkodiert.

Die Auswertung erfolgte wieder über die bereits beschriebene sechsfache Kreuzvalidierung. Tabelle 1 zeigt die *MPP* und Tabelle 2 die *MPA* dieser Interaktionssequenzen. Zusätzlich wurde ebenfalls für jede Datei eine einfaktorielle ANOVA gerechnet. In den Tabellen wurde die signifikant beste homogene Untergruppe gemäß des Student-Newman-Keul Tests mit einem Sternchen (*) versehen, während der höchste Wert jeder Zeile fett gedruckt ist.

Der KO3/18-Algorithmus war immer in der Untergruppe mit den besten *MPP*s zu finden, und in sieben von neun Fällen erreichte er sogar die höchste *MPP* von allen Algorithmen. Verglichen mit den Ergebnissen von Abbildung 8 kamen PPM+4 und PPMU+4 nur in vier bzw. drei von neun Fällen in die Spitzengruppe. Das zweite interessante Ergebnis ist, dass die MPA, die hauptsächlich in anderen Publikationen verwendet wurde, zu ähnlichen Ergebnissen führt. Wieder kam KO3/19 in acht von neun Fällen in die Spitzengruppe.

Tabelle 1: Vergleich der Vorhersagewahrscheinlichkeiten (MPP) für unterschiedliche AVAs bei den Greenberg-Referenzdateien.

	$\|o\|$	$\|A\|$	IPAM/0,7	IPAM2/0,7	KO3/18	LEV3	PPM4	PPM+4	PPMU+4	PPMC4	PPM*	PPM*+
novice-programmers-15	71	14	0,3283	0,3724	***0,4653**	0,4262	0,4247	*0,4381	0,4309	0,3854	0,4277	*0,4411
novice-programmers-20	54	18	*0,2596	*0,2602	*0,2972	*0,2906	***0,2984**	*0,2953	*0,2863	*0,2682	*0,2944	*0,2913
experienced-programmers-14	101	51	0,1428	0,1423	***0,2075**	0,1808	0,168	0,1879	0,1794	0,1279	0,1617	0,1815
experienced-programmers-28	438	70	0,2032	0,2123	***0,2485**	0,2211	0,1778	0,2272	0,2235	0,1445	0,1730	0,2225
computer-scientists-11	37	24	*0,2236	*0,1913	*0,2139	***0,2263**	0,1489	*0,2238	*0,2234	0,1151	0,1372	*0,2121
computer-scientists-26	123	55	0,1461	0,1350	***0,1889**	0,1676	0,1247	0,1755	0,1709	0,1008	0,1216	0,1724
non-programmers-16	46	32	0,2817	0,2613	***0,3395**	0,2918	0,2581	0,3085	0,2988	0,2093	0,2451	0,2955
non-programmers-24	34	25	*0,3517	0,3394	***0,3945**	*0,3754	0,3233	*0,3686	*0,3574	0,2748	0,3239	*0,3693
combined-references-file	904	132	0,1357	0,1549	***0,1983**	0,1732	0,1647	0,1791	0,1723	0,1338	0,1598	0,1742
rank 1-5					1	4		2	5			3

Tabelle 2: Vergleich der Vorhersagegenauigkeiten (MPA) für unterschiedliche AVAs bei den Greenberg-Referenzdateien.

	$\|o\|$	$\|A\|$	IPAM/0,7	IPAM2/0,7	KO3/18	LEV3	PPM4	PPM+4	PPMU+4	PPMC4	PPM*	PPM*+
novice-programmers-15	71	14	0,4299	*0,5572	0,5069	0,4972	*0,6177	0,5045	0,5517	*0,6189	***0,6213**	0,5080
novice-programmers-20	54	18	0,3045	0,3834	***0,4807**	0,3918	*0,4262	0,3881	0,3846	*0,4292	*0,4342	0,3962
experienced-programmers-14	101	51	0,2106	***0,3339**	*0,3309	0,2812	0,2502	0,2728	0,2768	0,2517	0,2621	0,2848
experienced-programmers-28	438	70	0,2702	*0,3239	***0,3377**	*0,3198	0,2445	*0,3217	*0,3326	0,2449	0,2511	*0,3283
computer-scientists-11	37	24	0,2986	0,3948	***0,5179**	0,4112	0,1855	0,4118	0,4295	0,1859	0,1827	0,409
computer-scientists-26	123	55	0,1947	0,3223	***0,3623**	0,2988	0,1608	0,3099	0,3118	0,1631	0,1651	0,3142
non-programmers-16	46	32	0,4200	*0,4767	***0,4839**	*0,4274	0,3695	*0,4364	*0,4443	0,3688	0,3623	*0,4291
non-programmers-24	34	25	*0,4608	*0,4949	*0,5258	*0,5064	*0,5032	*0,5032	***0,5298**	*0,4946	*0,5184	*0,5184
combined-references-file	904	132	0,1719	0,2456	***0,2767**	*0,2630	0,2405	*0,2628	*0,2736	0,2408	0,2460	*0,2683
rank 1-5				3	1			5	2			4

7. Untersuchung: Unterstützungsfunktion in ACTIVE-UI

7.1 Versuchsdurchführung

Abschließend wurde eine Untersuchung mit der aktuellen Version von ACTIVE-UI durchgeführt, um den Nutzen adaptiver Unterstützungssysteme sowie auch die Effekte zwischen den AVAs und der Art des Unterstützungssystems zu betrachten.

Dazu wurden die bereits beschriebene adaptive Hilfe und der adaptive Tutor in ACTIVE-UI integriert. Als dritte Bedingung wurde eine kontextsensitive Hilfe geschaffen, und die vierte Bedingung war ACTIVE-UI ohne jegliche Unterstützung.

Die Testpersonen ohne Erfahrung mit der Benutzung von ACTIVE-UI hatten insgesamt drei verschiedene Schweißaufträge durchzuführen, wobei der Schwierigkeitsgrad dabei sukzessive anstieg.

Abhängige Variablen waren die Art der Unterstützung und ein vorher durchgeführtes Training des Benutzers. Die beiden adaptiven Systeme wurden mit neun Interaktionssequenzen von erfahrenen ACTIVE-UI-Benutzern bei Durchführung dieser Aufgaben angelernt. Als AVA wurde PPM+4 gewählt, weil sein Verhalten gut bekannt ist und in den Vorstudien gute Werte bezüglich der *MPP* lieferte.

Die Testreihen wurden von insgesamt 48 Studenten verschiedenen Alters durchgeführt. Das Durchschnittsalter betrug 24,6 Jahre, und alle Teilnehmer waren erfahrene Computer-Anwender. Vor Versuchsbeginn wurde das räumliche Vorstellungsvermögen mittels eines dreidimensionalen Würfeltests (Three-Dimensional Cube Test, Gittler 1999) ermittelt. Der durchschnittlich erzielte Wert betrug +0,89. Werte zwischen –3 und 0 weisen auf ein geringes räumliches Vorstellungsvermögen hin, wohingegen Werte zwischen 0 und +3 für ein hohes räumliches Vorstellungsvermögen stehen.

Tabelle 3 zeigt die Aufteilung der Versuchspersonen auf die unterschiedlichen Hilfesysteme. Im dritten Versuchsdurchlauf durfte die Hälfte der Testpersonen ein Pre-Training durchführen. Das Vorgehen dieser Untersuchung (Between-Subject-Design) ist detaillierter in Künzer et al. (2003) und Ziefle et al. (2004) dargestellt.

Tabelle 3: Aufteilung der Versuchspersonen zu den unterschiedlichen Hilfesystemen.

unabhängige Variable	adaptiver Tutor n = 12		adaptive Hilfe n = 12		kontextsensitive Hilfe n = 12		ohne Hilfe n = 12	
Pre-Training	Ja n = 6	Nein n = 6	Ja n = 6	Nein n = 6	Ja n= 6	Nein n = 6	Ja n = 6	Nein n = 6

7.2 Personenbezogene Ergebnisse

Abbildung 9 zeigt die Auswertung aller 48 Versuchspersonen im Hinblick auf den Trainingseffekt (vom ersten bis zum dritten Versuch), wobei es signifikante Unterschiede unter den Versuchspersonen gibt: Für die Lösung einer Aufgabe werden je nach angebotener Hilfe unterschiedliche Bearbeitungszeiten benötigt. Der allgemeine Lerneffekt ist hoch signifikant ($F(2,94) = 197,58$; $p < 0,01$) und zeigt, dass der dritte Versuch deutlich schneller absolviert wurde als der erste Versuch. Weiterhin gibt es einen deutlichen Hinweis zwischen Lerneffekt und angebotenem Hilfssystem ($F(2,6)$

= 2,77; $p < 0,05$). Die höchste Verbesserung der Bearbeitungszeit (64%) zeigten die Testpersonen unter Verwendung des adaptiven Tutors. Die Unterstützungssysteme *kontextbasierte Hilfe* (52%) und sowohl die *adaptive Hilfe,* als auch *ohne Hilfe* erreichten eine zeitliche Verbesserung um 54 %. Interessanterweise hing der Lernerfolg und der Nutzen des Hilfesystems davon ab, ob die Testpersonen einen anfänglichen Probeversuch durchgeführt hatten ($F(1,3) = 3,22$; $p < 0,05$) und so in einer Eingewöhnungsphase die Programmstruktur erlernen konnten. Einen anderen starken Einfluss auf die Effektivität hatte das räumliche Vorstellungsvermögen der Testpersonen (Abbildung 10). Je geringer dieses war, desto geringer war die Anzahl gelöster Aufgaben bei gleichzeitig höherer Bearbeitungszeit.

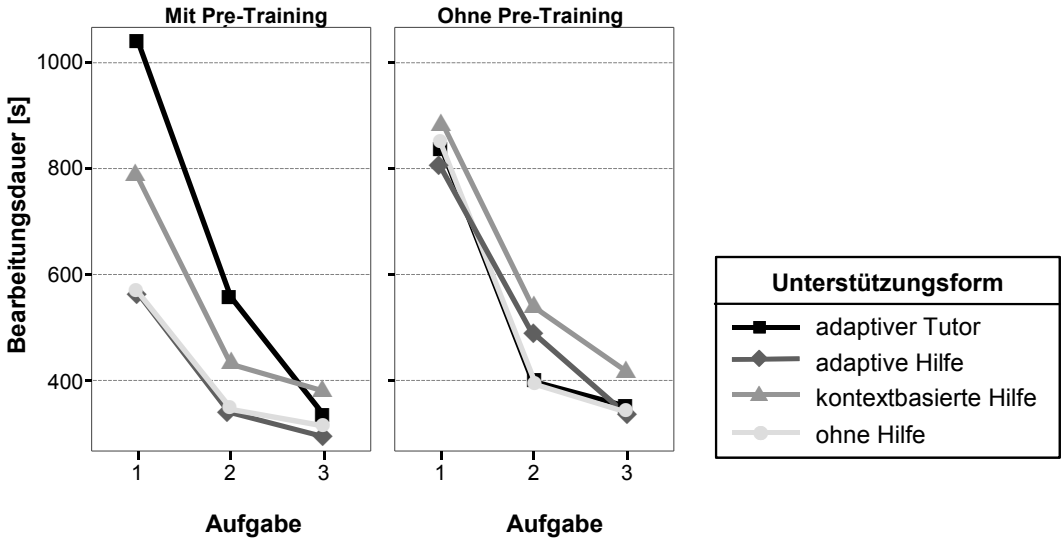

Abbildung 9: Einfluss des Trainingseffekts auf die benötigte Gesamtzeit in Abhängigkeit von der Versuchsanzahl und des verwendeten Hilfesystems.

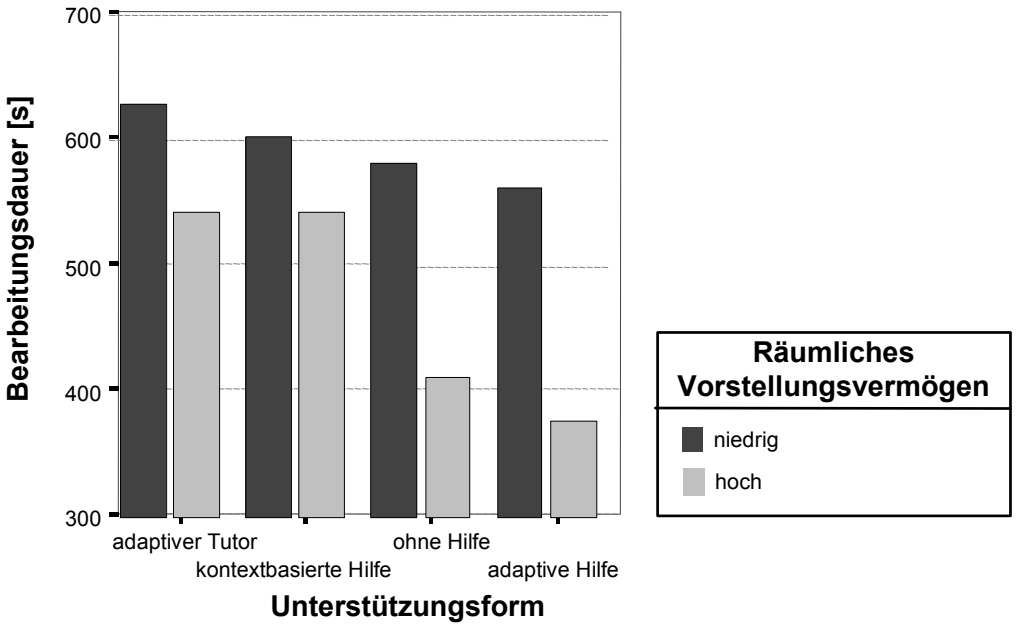

Abbildung 10: Einfluss des räumlichen Vorstellungsvermögens und der Unterstützungsform auf die benötigte Gesamtzeit.

In den Abbildungen 11 und 12 sind die Anzahl der gelösten Aufgaben und die Anzahl von Umwegschritten in Bezug auf den optimalen Pfad für die erste und dritte Aufgabe aller Tutorengruppen in Abhängigkeit vom räumlichen Vorstellungsvermögen dargestellt. Versuchspersonen ohne Hilfesystem lösten die geringste Anzahl an Aufgaben, wohingegen bei Unterstützung durch den adaptiven Tutor die besten Resultate erzielt wurden. Dies konnte auch bei Messungen der benötigten Umwegschritte signifikant bestätigt werden. Auch hier zeigt die Verwendung eines adaptiven Tutors die stärkste Abnahme an eingeschlagenen Umwegschritten.

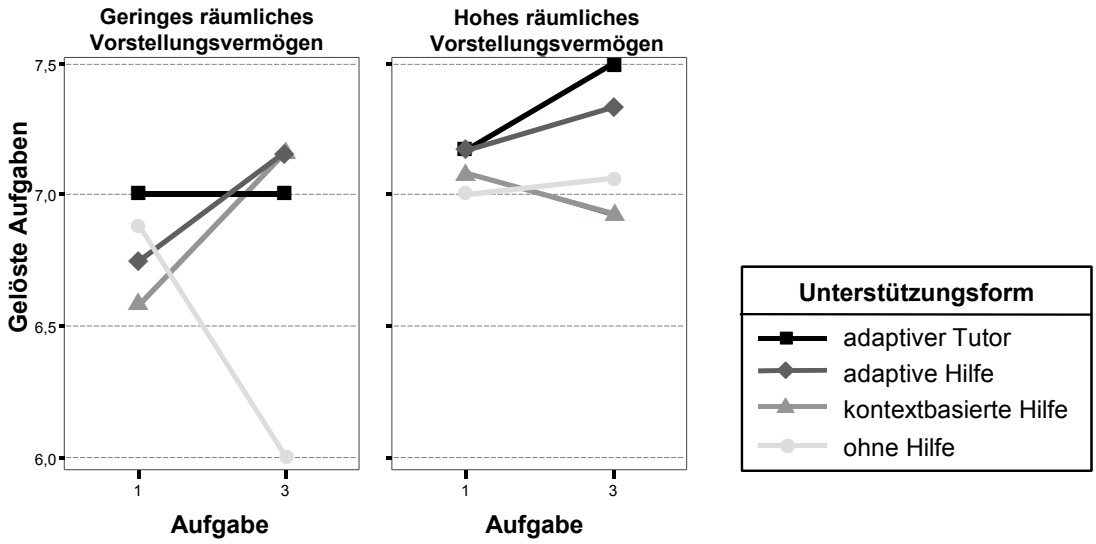

Abbildung 11: Auswertung der Anzahl gelöster Aufgaben in Abhängigkeit vom räumlichen Vorstellungsvermögen.

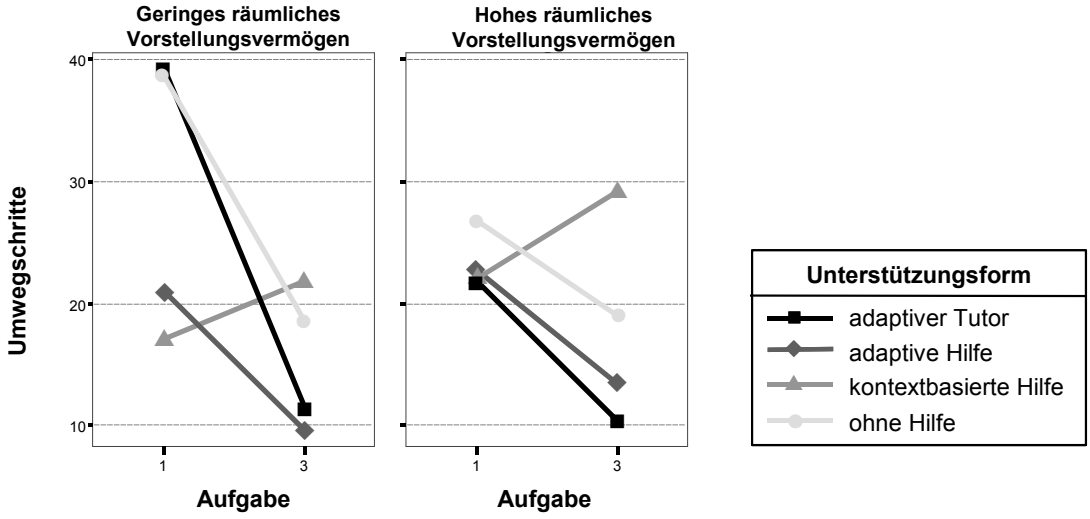

Abbildung 12: Auswertung der Anzahl eingeschlagener Umwegschritte in Abhängigkeit vom räumlichen Vorstellungsvermögen.

7.3 AVA-bezogene Ergebnisse

Obwohl PPM+ 4 als AVA für die adaptiven Unterstützungsfunktionen benutzt wurde, ist es natürlich möglich, die *MPP* aller Algorithmen ohne vorheriges Training zu

berechnen. Das Ergebnis ist in Abbildung 13 grafisch dargestellt. Einerseits zeigte sich, dass die Wahl von PPM+4 sinnvoll war, da die übrigen Algorithmen keine signifikant besseren *MPPs* erzielen konnten. Trotzdem lieferte KO3/18 erneut die höchste MPP.

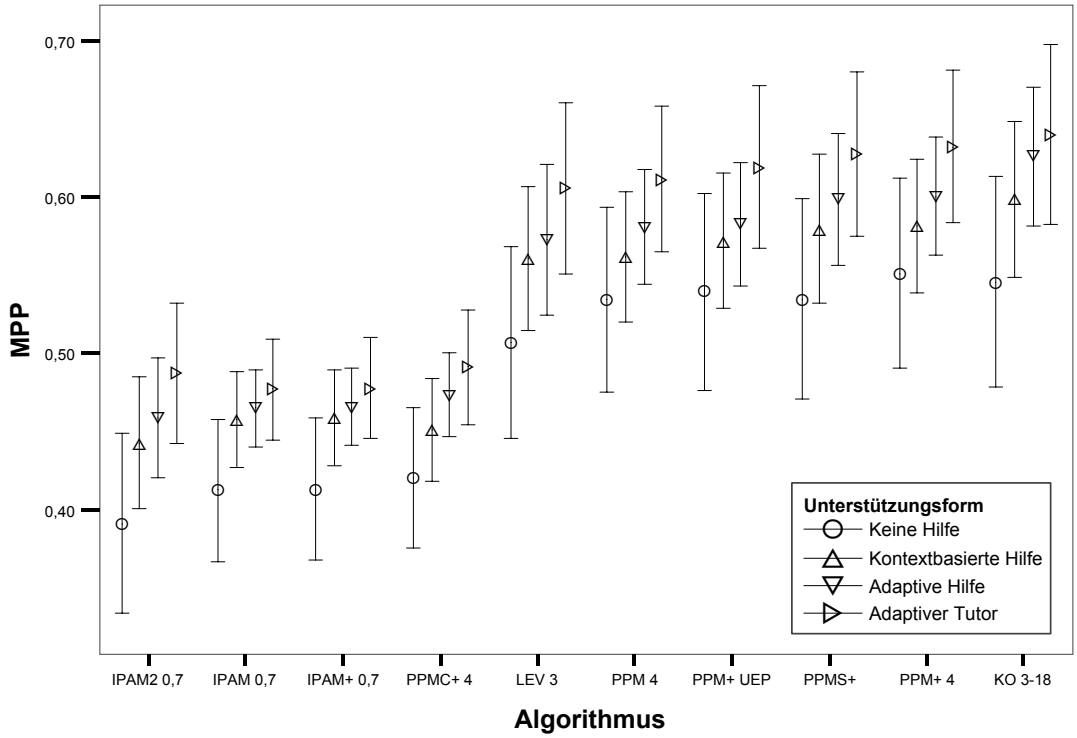

Abbildung 13: Vorhersagewahrscheinlichkeit für die unterschiedlichen Unterstützungssysteme in ACTIVE-UI.

Die *MPPs* können darüber hinaus auch als Maß dafür angesehen werden, wie ähnlich die Interaktionssequenzen der Testpersonen zu denen der Experten sind. Bedingt durch die vom Umfang her kleine Datenbasis gab es aber keine signifikanten Effekte, allerdings lieferte für alle AVAs die Bedingung „keine Hilfe" bezüglich der *MPP* die niedrigsten Ergebnisse, gefolgt von „kontextsensitiver Hilfe", „adaptiver Hilfe" und „adaptiver Tutor", der am besten abschnitt. Daher wurde eine Normalisierung der *MPPs* durchgeführt, mit dem Ziel, die Unterschiede der Level- und Tendenzunterschiede der Werte für jeden AVA anzupassen, wobei die Formel $x_N = (x - x_{\min})/(x_{\max} - x_{\min})$ zu Grunde gelegt wurde. Eine einfaktorielle ANOVA lieferte für die normalisierten Ergebnisse einen F-Wert von F_{ANOVA}=13,378 (df=3; p_{ANOVA}<0,001) und der Post-hoc Student-Newman-Keul Test zeigte, dass die homogene Untergruppe „adaptive Hilfe" (*MPP*=0,6680) und „adaptiver Tutor" (*MPP*=0,6405) signifikant besser ist, als „keine Hilfe" (*MPP*=0,5701). Dies kann durch den Einfluss des Hilfe- oder Tutorsystems selbst erklärt werden, die dem Benutzer Vorschläge machen, die dieser tatsächlich befolgt. Besonders den Anwendern ohne Unterstützungsfunktion gelang es nicht, dem optimalen Interaktionspfad zur Lösung der Aufgaben zu folgen. Auch wenn die Bearbeitungszeiten am Ende niedrig waren, zeugen der hohe Fehleranteil und der niedrige Anteil an der Aufgabenbearbeitung davon, dass diese Benutzer aufgrund der fehlenden Unterstützung dazu gezwungen waren ein mentales Modell der Anwendung zu entwickeln, wozu sie eine Trial-And-Error-Strategie anwendeten.

8. Zusammenfassung und Ausblick

Dieser Artikel beschreibt die Verwendung von generischen Aktionsvorhersage-Algorithmen (AVAs) zur Entwicklung eines adaptiven Unterstützungssystems. Dazu wurden mit dem LEV- und dem KO-Algorithmus zwei neue AVAs vorgestellt und in drei verschiedenen Szenarios mit anderen AVAs verglichen. Die mittlere Vorhersagewahrscheinlichkeit (Mean Prediction Probability, *MPP*) des KO-Algorithmus lieferte dabei bessere Werte als bekannte Algorithmen wie IPAM oder PPM.

Basierend auf den AVAs wurden eine adaptive Hilfe und ein adaptiver Tutor entwickelt, die durch die Beobachtung von Expertenverhalten angelernt werden können. Die adaptiven Unterstützungssysteme können dadurch Anfängern helfen, ihre Aufgaben effektiver durchzuführen, aber auch Experten erhalten aufgaben- und adaptive Hilfe- und Funktionsunterstützung im Rahmen ihrer Interaktionen.

Eine Auswertung dieser Ansätze mit 48 Probanden zeigte, dass die adaptiven Unterstützungsansätze zu einer Steigerung der Effektivität bei den Benutzern führen. Die Verwendung des PPM+-Algorithmus war gut geeignet, da keiner der anderen AVAs eine signifikant höhere *MPP* aufwies. Die Nutzung der Unterstützungssysteme innerhalb von ACTIVE-UI führte darüber hinaus zu Interaktionssequenzen mit einer höheren *MPP* als ohne Unterstützung, was zeigt, dass die Benutzer auch von der Vorschlägen beeinflusst werden, da die *MPP* auch als Ähnlichkeitsmaß interpretiert werden kann.

Trotzdem differiert die *MPP* der verschiedenen AVAs und ist stark von der konkreten Interaktionsaufgabe abhängig. Deshalb sollte zukünftig eine dynamische Auswahl des bestgeeigneten AVA und seiner Parameter dazu führen, die Vorhersagequalität der adaptiven Unterstützungssysteme weiter zu erhöhen.

9. Danksagungen

Die Autoren danken Brian Davison und Ian Witten für ihr Interesse und die hilfreichen Kommentare. Teile dieser Untersuchungen wurden von der Deutschen Forschungsgemeinschaft im Rahmen des Sonderforschungsbereiches 368 „Autonome Produktionszellen“ gefördert.

Literatur

Albrecht, D.W.; Zukerman, I. & Nicholson, A.E. (1998). Bayesian Models for Keyhole Plan Recognition in an Adventure Game. In: *User Modeling and User-Adapted Interaction,* 8, S. 5-47.

Carberry, S. (2001). Techniques for Plan Recognition. In: Kobsa, Alfred (Hrsg.). *UMUAI - User Modeling and User-Adapted Interaction,* Bd. 11 (1-2). New York: Springer Verlag, 2001, S. 31-48. Tenth Anniversary Issue of UMUAI.

Cleary, J.G. & Witten, I.H. (1984). Data compression using adaptive coding and partial string matching. In: *IEEE Transactions on Communications 32*.

Cleary, J.G.; Teahan, W.J. & Witten, I.H. (1995). Unbounded length contexts for PPM. In: *Proceedings of Data Compression Conference*. Los Alamitos, CA : IEEE Computer Socienty Press, S. 52-61.

Cook, J. E. & Wolf, A. (1998). Discovering Models of Software Processes from Event-Based Data. In: *ACM Transactions on Software Engineering and Methodology*. Vol. 7.

Davison, B. & Hirsch, H. (1998). Predicting sequences of user actions. In: *Notes of the AAAI/ICML 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Analysis*.

Dieterich, H.; Malinowski, U.; Kühme, T. & Schneider-Hufschmidt, M. (1993). State of the Art in Adaptive User Interfaces. In: Schneider-Hufschmidt, M. (Hrsg.) ; Kühme, T. (Hrsg.) ; Malinowski, U. (Hrsg.). *Adaptive User Interfaces: Principles and Practice*. Amsterdam : North-Holland, S. 13-48.

Eisenstein, J. & Rich, C. (2002). Agents and GUIs from Task Models. 2002. In: *Conference on Intelligent User Interfaces (IUI 2002)*.

Encarnação, L.M. & Stoev, S.L. (1999). An Application-Independent Intelligent User Support System Exploiting Action-Sequence Based User Modelling. In: *Proceedings of 7th International Conference on User Modeling*. Wien, New York : Springer-Verlag.

Fischer, G. (2001). User Modeling in Human-Computer Interaction. In: *User Modeling and User-Adapted Interaction*, Bd. 11. Dordrecht.

Foley, J.D.; Dam, A.V.; Feiner, S.K. & Hughes, J.F. (1990). *Computer Graphics - Principles and Practice*. Addison-Wesley.

Gittler, G. (1999). *Three-dimensional Cube Test.* Stuttgart : Beltz.

Gorniak, P.J. & Poole, D. (2000). Predicting future user actions by observing unmodified applications, *17th National Conference on AI*, AAAI-2000.

Greenberg, S. (1988). *Using Unix: Collected Traces of 168 users.* In: Research Report 88/333/45. Canada : Department of Computer Science, University of Calgary.

Greenberg, S. (1993). *The Computer User as Toolsmith: The Use, Reuse, and Organization of Computer-based Tools*.

Hacker, W. (1978). *Allgemeine Arbeits- und Ingenieurspsychologie: Psychische Struktur und Regulation von Arbeitstätigkeiten*; Schriften zur Arbeitspsychologie. Bd. 20. Bern : Hans Huber.

Hilbert, D.M. & Redmiles, D.F. (1999). *Extracting Usability Information from User Interface Events*. TR UCI-ICS-99-40, Department of Information and Computer Science, University of California, Irvine.

Horvitz, E.; Breese, J.; Heckermann, D.; Hovel, D. & Rommelse, K. (1998). The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In: *Fourteenth Conf. on Uncertainty of Artificial Intelligence*, San Francisco, CA.

Howard, P.G. & Vitter, J.S. (1994). *Design and Analysis of Fast Text Compression Based on Quasi-Arithmetic Coding*. DUKE-TR-1994-11.

Kobsa, A. (2001). Generic User Modeling Systems. In: Kobsa, Alfred (Hrsg.). *UMUAI User Modeling and User-Adapted Interaction* Bd. 11 (1-2). New York: Springer Verlag, S. 49-63.

Korvemaker, B. & Greiner, R. (2000). Predicting UNIX command lines: Adjusting to user patterns. In Adaptive User Interfaces: Papers from the 2000 AAAI Spring Symposium, S. 59-64.

Künzer, A. & Luczak, H. (2003). Design of an Adaptive User Interface Based on Action Prediction. In The Ergonomics Society of Korea (Hrsg). *Ergonomic in the Digital Age. Proceedings of the 15th Triennial Congress of the IEA and the 7th Joint Conference of Ergonomics Society of Korea.* Seoul 2003, S. 1-4.

Künzer, A. (2004). *Handlungsprädiktion zur Gestaltung einer adaptiven Benutzungsunterstützung in autonomen Produktionszellen*. Shaker-Verlag: Aachen. In Vorbereitung.

Künzer, A. ; Ziefle, M.; Bodendieck, A. & Luczak, H. (2003). Effects of Different User-Adaptive Help Systems on Task Performance. In Luczak, H.; Zink, K.J. (Hrsg.). *Human Factors in Organizational Design and Management - VII. Proceedings of the Seventh International Symposium on Human Factors in Organizational Design and Management held in Aachen, October 1-2*. IEA Press: Santa Monica, CA, USA 2003, S. 461-467.

Künzer, A.; Schlick, C.; Ohmann, F.; Schmidt, L. & Luczak, H. (2001). Eine empirische Untersuchung zur Modellierung von Handlungsvorhersagen mit Hilfe dynamischer Bayes-Netze. In: *Human Factors bei der Entwicklung von Fahrzeugen* (DGLR), Bonn.

Langley, P. (1997). Machine Learning for Adaptive User Interfaces. In: *Proceedings of the 21st German Annual Conference on AI*. Springer: Germany.

Lau, T. (1999). *A comparison of sequence-learning approaches: implications for intelligent user interfaces*. University of Washington.

Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions and reversals. In: *Soviet Physics Doklady* Bd. 10(8), S. 707-710.

Luczak, H.; Schlick, C.; Künzer, A. & Ohmann, F. (2001). Syntactic user modeling with stochastic processes. *Theoretical Issues in Ergonomics Science*. London.

Maes, P. (1994). Agents that reduce work and information overload. In: *Communications of the ACM 37* (1994), Nr. 7, S. 30-40.

Mannila, H.; Toivonen, H. & Verkamo, A. I. (1997). Discovery of frequent episodes in event sequences. In: *Data Mining and Knowledge Discovery*, 1(3), 1997.

Moffat, A. (1990). Implementing the PPM data compression scheme. In: *IEEE Transactions on Communications*.

Ross, E. (2000). *Intelligent User Interfaces: Survey and Research Directions*. TR CSTR-00-004, Dep. of Computer Sc., University of Bristol.

Schlick, C. (2000). Simulation of Rule-Based Behavior for a Multimodal Interaction Task with Stochastic Petri Nets. In: *Proc. of the XIVth Triennal Congress of the IEA*. Santa Monica: CA.

Schlick, C.; Winkelholz C.; Motz F.; Künzer, A. & Luczak, H. (2002). Stochastic operator models for multiple target search tasks. In: *IEEE International Conference on Systems, Man and Cybernetics*. Hammamet, Tunisia.

Sun, R. (2000). Introduction to sequence learning. In: *Sequence Learning: Paradigms, algorithms, and Applications*. Springer-Verlag, Berlin.

Ziefle, M. ; Künzer, A. & Bodendieck, A. (2004). The impact of user characteristics on the utility of adaptive help systems. In: *Proceedings of the WWCS 2004*, Kuala Lumpur, Maylaysia. In Druck.

Ziv, J. & Lempel, A. (1977). A Universal Algorithm for Sequential Data Compression. In: *IEEE Transactions on Information Theory*, S. 337-343.

Zukerman, I.; Albrecht. W. & Nicholson, A. (1999). Predicting user's request on the WWW. In: *UM99 - Proceedings of the 7th International Conf. on UM*.