ERLERNEN DER COMPUTERBENUTZUNG: DURCH GEZIELT SEQUENZIERTE INSTRUKTION ODER DURCH EXPLORIEREN.

Otto Kühn & Franz Schmalhofer, Freiburg

Zusammenfassung: Lernen durch Explorieren wurde in neuerer Zeit als geeignete Methode zum Erlernen der Computerbenutzung vorgeschlagen. Aufgrund einer Spezifikation der kognitiven Prozesse für das Lernen durch gezielt sequenzierte Instruktion und durch Explorieren werden in der voliegenden Arbeit die Stärken und Schwächen dieser Lernverfahren experimentell untersucht. Die Ergebnisse zeigen, daß bei den beiden Lernverfahren unterschiedliche Fertigkeiten trainiert und daß zum erfolgreichen Explorieren bestimmte Voraussetzungen bezüglich des bereits vorhandenen Wissens erfüllt sein müssen. Insbesondere eignet sich das Explorieren für etwas fortgeschrittenere Computerbenutzer zum Erlernen von zielgerichteten Konzeptionen für Systemeingaben, die für den praktischen Umgang mit dem Computer wichtig sind.

Traditionellerweise obliegt die Strukturierung des Lernmaterials dem Lehrenden, während die Aufgabe des Lernenden darin besteht, das Lernmaterial aufzunehmen und zu verarbeiten. Bei komplexen Systemen ist es oft schwierig, dem unerfahrenen Benutzer die Wirkung der verschiedenen möglichen Eingaben in das System zu erklären. In neuerer Zeit wurde als alternative Methode das Explorieren des Systems durch den Benutzer vorgeschlagen (Carroll et al., 1985). Diese Autoren berichten, daß bei einem bestimmten Textverarbeitungssystem das Lernen durch Explorieren weniger Zeit beanspruchte und zu einem besseren Lernergebnis führte als das Durcharbeiten eines herkömmlichen Manuals zum Selbststudium.

Nach Robert (1986) zeichnet sich Explorieren durch folgende Merkmale aus: vom Lernenden wird ein hohes Maß an Aktivität gefordert, er muß selbst entscheiden, welche Eingaben er macht, die Rückmeldungen des Systems interpretieren, Hypothesen aufstellen und testen, und auftretende Probleme lösen. Seine Motivation wird durch die dabei auftretenden Erfolgs- und Mißerfolgserlebnisse stark beeinflußt. Der Lernende kann sein Wissen genau dort ergänzen, wo Lücken bestehen. Auch dadurch kann die Lernmoti-

vation gefördert werden. Dies gilt natürlich nur für Systeme die eine gute Explorierbarkeit aufweisen. Deshalb ist Explorierbarkeit heute ein wichtiges Kriterium für die Beurteilung der Güte eines Computersystems.

Die Vorteile des Lernens durch Instruktion bestehen vor allem darin, daß der Lehrende aufgrund seines größeren Wissens das am besten geeignete Lernmaterial auswählen und so strukturieren kann, daß dadurch die Aufgabe für den Lernenden möglichst leicht wird. Es stellt sich also die Frage, ob und unter welchen Voraussetzungen die Vorteile des Explorierens die Nachteile aufwiegen.

Aufgrund von Analysen der Künstlichen Intelligenz (KI) können die beim Lernen ablaufenden Prozesse aufgeschlüsselt werden (vgl. Habel & Rollinger, 1984). Unter Berücksichtigung psychologischer Gesetzmäßigkeiten kann dann ein kognitives Modell erstellt werden (vgl. Schmalhofer & Wetter, 1987), und es können präzise Vorhersagen des zu erwartenden Lernerfolgs getroffen werden. Lernen durch gezielte sequenzierte Instruktion kann als induktives Lernen aus einer von einem Lehrer vorgegebenen Sequenz von Beispielen beschrieben werden. Der Lernerfolg hängt dabei entscheidend davon ab, daß eine geeignete Stichprobe von positiven und negativen Beispielen dargeboten wird. Bei Annahme eines limitierten Gedächtnisses ist auch die Sequenz in der die Beispiele dargeboten werden relevant. Solche Lernprozesse können beim Erwerb von Computerfertigkeiten Muster bilden, die Elemente von Computerfertigkeiten darstellen. Diese Muster werden durch Generalisierungs- und Differenzierungsprozesse gebildet und verfeinert, wobei das zur Analyse der Beispiele eingesetzte (Vor-) Wissen bestimmt, was überhaupt gelernt werden kann. Solche Prozesse sind durch das S-T-Struktur-Modell (Schmalhofer, 1986; Schmalhofer & Kühn, 1986) detailliert beschrieben.

Beim Lernen durch Explorieren muß der Lernende zusätzlich die Funktion des Lehrers übernehmen: er muß aufgrund eines sehr globalen Lernziels spezifische Intentionen generieren, aus denen dann aus seinem Vorwissen und mithilfe externer Vorgaben konkrete Eingaben in das System erzeugt werden. Lernen durch Explorieren kann also als induktives Lernen aus einer selbstgenerierten Sequenz von Beispielen beschrieben werden. Diese aufgrund von KI Analysen aufgeschlüsselten Prozesse beim Lernen durch gezielt

sequenzierte Instruktion und durch Explorieren werden in einem psychologischen Experiment untersucht.

Experiment

An der Untersuchung nahmen 40 Computerbenutzer mit geringer Computererfahrung, sowie 40 Neulinge ohne jegliche Vorerfahrung teil. Je 20 dieser Vpn erlernten zusätzliche Computerkenntnisse durch Explorieren bzw. anhand von vorgegebenen Beispielen. Die Beispiele wurden in zwei verschiedenen Sequenzen vorgegeben: in der einen Sequenz waren die positiven und negativen Beispiele so angeordnet, daß sie nach dem S-T-Struktur-Modell zu einem optimalen Lernergebnis führen sollten; in der anderen Sequenz wurden erst alle positiven und dann die negativen Beispiele dargeboten, wie es in den meisten Manualen üblich ist.

Als Lerngegenstand wurden zwei einfache LISP-Funktionen gewählt: die Funktion FIRST, die das erste Element einer Liste zurückgibt und die Funktion SET, die als Nebeneffekt eine Variablenbindung erzeugt. Die Programmiersprache LISP wurde gewählt, zum einen, um das Vorwissen der Probanden gut kontrollieren zu können, zum anderen wegen ihrer hohen Dekomponierbarkeit, welche als eine Voraussetzung für die Explorierbarkeit eines Systems anzusehen ist.

Die Versuche wurden von zwei IBM-PC/ATs und von zwei kompatiblen Personal Computern gesteuert. Alle Eingaben der Vpn wurden samt den zugehörigen Latenzzeiten aufgezeichnet. Für die Eingaben standen den Vpn Hilfen zur Erzeugung syntaktisch korrekter Ausdrücke bereit. Die eingegebenen LISP-Ausdrücke wurden durch einen in Pascal geschriebenen LISP-Interpreter ausgewertet. Dadurch konnten dem Lernenden angemessene Rückmeldungen gegeben werden.

Die Lernphase begann für alle Vpn mit einem kurzen Text, der die zur Analyse der Beispiele erforderlichen LISP-Grundkenntnisse vermitteln sollte. Er enthielt eine kurze Einführung in das LISP-System, wobei die Grundbausteine des Systems erläutert wurden. Die beiden zu lernenden Systemfunktionen wurden durch je einen Satz charakterisiert. Je nach Versuchsbedingung wurden dann entweder Beispiele vorgegeben oder die Vp konnte das System explorieren.

Bei den Beispielen wurde zunächst nur der Eingabeteil dargeboten. Das Ergebnis erschien erst nach einem Tastendruck. Die Lesezeiten für die beiden Teile wurden als Eingabeanalyse- und Gesamtanalysezeit getrennt aufgezeichnet.

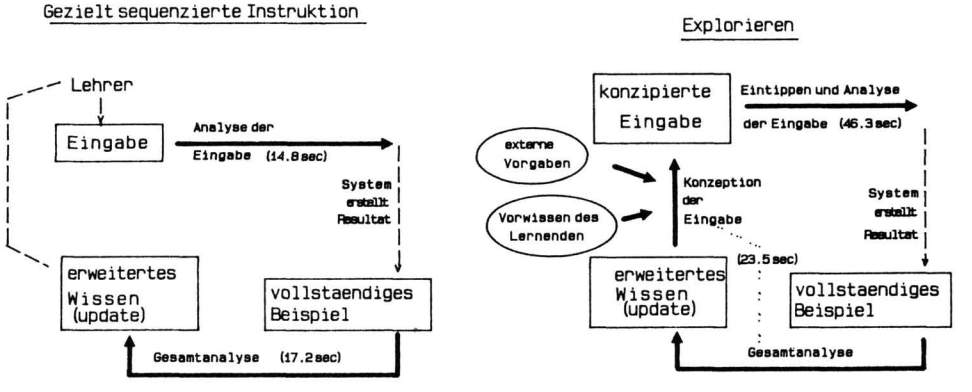
Beim Explorieren wurden durch die Vorgaben "(FIRST '(A B))" , "(FIRST (FIRST '((A B) C)))" , "(SET 'FREUNDE '(HANS KLAUS)) (FIRST FREUNDE)" 3 Blöcke unterschieden. Durch diese Aufteilung sollte sichergestellt werden, daß alle zu erlernenden Funktionen auch tatsächlich exploriert wurden. Für jede solche Vorgabe konnten 8 (Block 1 und Block 2) bzw. 16 Eingaben (Block 3) gemacht werden. Für jede Eingabe der Vp wurden zwei Latenzzeiten registriert: die Zeit, die die Vp nachdachte bevor sie anfing einzutippen (Gesamtanalysezeit der vorangegangenen Ein- und Ausgabe + Konzipieren der neuen Eingabe) und die Zeit vom Tippen des ersten Zeichens bis zur Beendigung der Eingabe (Eintippen + Analyse der Eingabe). Beim Lernen durch Explorieren standen dem Lernenden also insgesamt 32 Lernbeispiele zur Verfügung, ebensoviele wie beim Lernen durch gezielt sequenzierte Instruktion. Ein Lernender, der die vorgegebenen Beispiele in das System eingab, konnte also noch 28 Beispiele selbst generieren. In der Beispielbedingung waren dagegen sämtliche 32 Lernbeispiele fest vorgegeben.

Die Testphase war für alle Bedingungen gleich und bestand aus Programmier- Evaluations- und Satzverifikationsaufgaben. Die 10 Programmieraufgaben, bei denen die Vp eine Eingabe in das LISP-System erzeugen sollte, die zu einem vorgegebenen Ergebnis führte, wurden zuerst dargeboten. Die erzeugte Eingabe wurde dann durch das LISP-System ausgewertet und das Ergebnis angezeigt. War die Lösung nicht korrekt, so konnte die Vp zwei weitere Lösungsversuche unternehmen. Um auch zu überprüfen, welches allgemeine Wissen über die Arbeitsweise des Systems erworben wurde, wurden 30 Evaluationsaufgaben gestellt, bei denen die Vp die Rolle des LISP-Interpreters übernehmen und vorgegebene Eingaben auswerten sollte, und 14 Satzverifikationsaufgaben, bei denen vorgegebene Sätze über das System bezüglich ihrer Korrektheit beurteilt werden sollten. Der Versuch dauerte insgesamt zwischen 1¼ und 3 Stunden. Nach Beendigung des Versuches beurteiten die Vpn die Schwierigkeit des Erlernens auf einer 5-stufigen Ratingskala.

Ergebnisse

Die Abfolge der in der Einleitung genannten Verarbeitungsprozesse bei der gezielt sequenzierten Instruktion und beim Explorieren ist in Abbildung 1 dargestellt. Bei einer Instruktion wird zuerst die vom Lehrer vorgegebene Systemeingabe analysiert. Nach der Systemantwort erfolgt dann eine Analyse der Ausgabe sowie der Beziehung zwischen Ein- und Ausgabe (Gesamtanalyse). Dann wird der Lehrer um die nächste Instruktion gebeten. Beim Explorieren muß stattdessen aus den vorhandenen Vorgaben und dem bereits erworbenen Wissen mit dem Ziel des Informationsgewinns eine Eingabe konzipiert werden. Diese kann dann eingetippt und während des Eintippens nochmals analysiert werden (Tippzeit + Eingabeanalysezeit). Die mittleren Bearbeitungszeiten bei der gezielten Instruktion und beim Explorieren sind aus Abbildung 1 zu entnehmen. Dabei ist zu beachten, daß beim Explorieren für die Gesamtanalyse und die darauffolgende Eingabe nur eine und nicht zwei separate Zeitmessungen vorliegen. Eine statistische Analyse dieser Bearbeitungszeiten ergab keine Unterschiede zwischen Computerbenutzern und Neulingen.

Abb. 1: Kognitive Prozesse bei gezielt sequenzierter Instruktion Instruktion und beim Lernen durch Explorieren



Obwohl sich die einzelnen Teilprozesse von Computerbenutzern und Neulingen in ihrer Dauer nicht unterschieden, können von den beiden Personengruppen dennoch verschiedenartige Beispiele

bearbeitet worden sein. Um einen Überblick über die studierten Beispiele zu erhalten, wurden 4 Klassen von Beispielen definiert: 1) übernommene Beispiele sind solche, die in der Explorationsbedingung direkt vorgegeben waren (z.B. "(FIRST '(A B))") oder der Struktur eines vorgegebenen Beispiels völlig entsprechen (z.B. "(FIRST '(X Y))"); 2) generierte Beispiele sind korrekte Eingaben, die von der Struktur des vorgegebenen Beispiels abweichen (z.B. "(FIRST '((A B) (C D)))"); 3) "near misses" sind solche Beispiele, die bis auf wenige Abweichungen der Struktur einer korrekten Eingabe entsprechen; 4) alle anderen Eingaben wurden als "far misses" bezeichnet.

Tabelle 1 zeigt, wie häufig die 4 Beispielsarten von Neulingen, Computerbenutzern und vom Lehrer erzeugt wurden. Aus Tabelle 1 erkennt man, daß beim Explorieren nicht so viele negative Beispiele erzeugt wurden, wie der Lehrer erzeugt hatte. Dagegen haben sowohl Neulinge als auch Computerbenutzer unnötig häufig die Struktur des vorgegebenen Beispiels repliziert. Solche Eingaben sind uninformativ. Möglicherweise wird jedoch dabei durch die positive Rückmeldung des Systems die Motivationslage des Lernenden stabilisiert. Auch erzeugten die Explorierer einen beachtlichen Anteil an "far misses" (Winston, p. 387), die ebenfalls keine weitere Information liefern.

Tab. 1: Relative Häufigkeiten der von Neulingen, Computerbenutzern und von einem Lehrer erzeugten Beispiele für die vier verschiedenen Beispielsarten

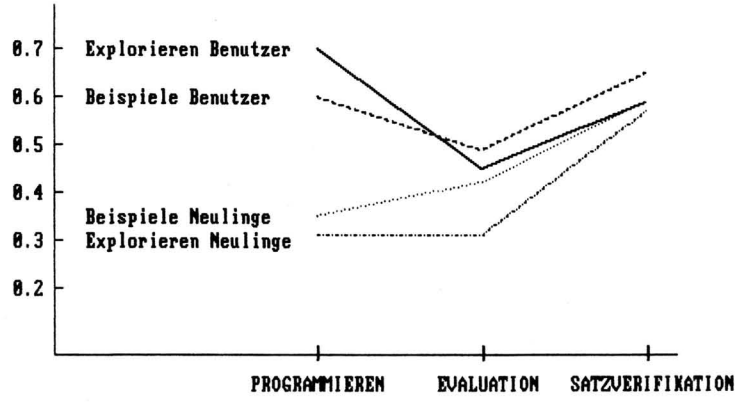
Beispiele:	Neulinge	Computer-benutzer	Lehrer
Übernommen	0.33	0.35	0.16
generiert	0.17	0.25	0.34
"near misses"	0.36	0.31	0.50
"far misses"	0.14	0.09	0.00

Durch die Programmieraufgaben wurde überprüft, wie gut das in der Lernphase erworbene Wissen praktisch eingesetzt werden konnte. Eine Auszählung der Anteile richtig gelöster Aufgaben zeigte, daß die Computerbenutzer, die explorierten, in sämtlichen 3 Durchgängen besser abschnitten (Anteile richtiger Lösungen bei den 3 Durchgängen waren: 0.54, 0.66, 0.70) als die Computerbenutzer, die instruiert wurden (0.41, 0.53, 0.60). Bei den Neulingen erwies sich dagegen gezielt sequenzierte Instruktion etwas besser (0.23, 0.31 bzw. 0.35) als das Explorieren (0.19,

0.28, 0.31). Eine 2x2x3-faktorielle Varianzananlyse der Meßwerte $y_i := 2*\arcsin(\sqrt{p_i})$ (wobei p_i= Anteil richtiger Lösungen; vgl. Winer, 1970, S.400) mit den Faktoren Vorkenntnisse, Lernbedingung und Lösungsdurchgang ergab signifikante Unterschiede bzgl. der Vorkenntnisse ($F(1,76)= 24.4$, $p<.001$) und bzgl. des Lösungsdurchgangs ($F(2,152)=6.2$, $p<.001$). Die Interaktion Lernbedingung x Vorkenntnisse war jedoch nicht signifikant ($F(1,76)= 1.49$, $p>.28$).

Abbildung 2 zeigt die Beziehung der Anteile richtiger Lösungen bei den Programmieraufgaben (3. Versuch) zu den Anteilen richtiger Lösungen bei den Evaluations- und Satzverifikationsaufgaben. Daraus erkennt man, daß im Gegensatz zu den Programmieraufgaben die Computerbenutzer, die explorierten, bei den Evaluations- und Satzverifikationsaufgaben schlechter abschnitten als die Computerbenutzer, die instruiert wurden. Dagegen war für Neulinge die Instruktion durchweg besser geeignet als das Explorieren. Das Explorieren bietet für die Personen mit Vorkenntnissen bzgl. der Programmieraufgaben den Vorteil, daß dabei das Konzipieren von Eingaben geübt wird. Bei den Evaluations- und Satzverifikationsaufgaben sind die Explorationsgruppen dagegen durchwegs im Nachteil, da die selbstkonzipierten Lernbeispiele ihnen weniger umfassende Informationen über das System lieferten als die vom Lehrer vorgegebenen Beispiele.

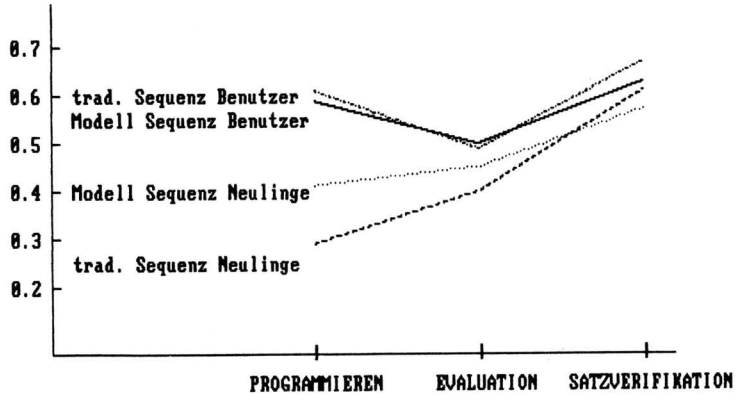
Abb. 2: Anteil richtig gelöster Testaufgaben



Eine 2x2x3-faktorielle Varianzanalyse mit den Faktoren Lernbedingung, Vorkenntnisse und Aufgabentyp der wiederum transformierten Anteile richtiger Lösungen ergab (neben den uninteressanten Unterschieden bzgl. des Aufgabentyps) Unterschiede in den Vorkenntnissen ($F(1,74)= 22.7$, $p<.001$) sowie eine Interaktion zwischen Vorkenntnissen und Aufgabentyp ($F(2,148)=18.9$, $p<.001$). Lernbedingung ($F(1,74)$, $p>.29$) und die Lernbedingung x Aufgabentyp Interaktion ($F(2,148)=1.75$, $p>.17$) waren nicht signifikant. Dagegen zeigte eine 2x2 Varianzanalyse der Daten der Evaluationsaufgaben Unterschiede sowohl zwischen den Lernbedingungen ($F(1,77)=8.9$, $p<.01$) als auch zwischen Computerbenutzern und Neulingen ($F(1,77)=12.5$, $p<.01$). Bei den Satzverifikationsaufgaben waren diese Unterschiede nicht signifikant ($F(1,74)= 2.16$, $p<.15$ für Lernbedingung und $F(1,74)=2.38$, $p<.15$ für Vorkenntnisse).

Vergleich der beiden Beispielsequenzen: Abbildung 3 zeigt die Anteile richtig gelöster Aufgaben von Computerbenutzern und Neulingen für die nach dem S-T-Struktur-Modell bzw. traditionell sequenzierten Beispiele. Bei Neulingen führt die aufgrund des Modells optimierte Beispielsequenz zu besseren Lösungen der Programmieraufgaben. Bei Computerbenutzern tritt dieser Unterschied dagegen nicht auf. Ein ähnliches Ergebnis zeigt sich für die Evaluationsaufgaben, nicht jedoch für die Satzverifikationsaufgaben.

Abb.3: Anteil richtiger Lösungen von Neulingen und Computerbenutzern für die zwei Darbietungssequenzen der Beispiele



Die Ergebnisse werden durch die Nachbefragung gestützt. So beurteilen die Personen, bei denen die Beispiele nach dem Mo-

dell sequenziert waren, die Reihenfolge der Beispiele als günstiger (Mittelwert von M=3.1 auf einer 5-stufigen Rating-Skala mit 1=ungünstig, 5=günstig) im Vergeich zu M=2.6 für die traditionelle Reihenfolge. Darüberhinaus wurde der Versuch als weniger anstrengend empfunden (M=3.5 vs. M=4.5, wobei 1=nicht anstrengend, 5=sehr anstrengend).

Schlußfolgerungen:

Die durchgeführten Analysen und die empirischen Befunde zeigen, daß das Explorieren deutlich mehr Zeit in Anspruch nimmt (69 sec pro Beispiel) als das Lernen aus Beispielen (32 sec), während sich die erzielten Lernresultate nicht so stark unterscheiden. Bei Vorliegen gewisser Vorkenntnisse kann das Explorieren jedoch wegen der unter einer Anwendungszielsetzung geübten Generierung von Systemeingaben zum Erwerb von praktischen Programmierfertigkeiten besser geeignet sein. Andererseits kann durch das Generieren schlechter Beispiele der Wissenserwerb stark behindert werden. Lernen durch Explorieren bietet den großen Vorteil, daß es die Lernmotivation fördern kann. Dieser Umstand sollte bei der Konzeption eines Lernverfahrens nach Möglichkeit ausgenutzt werden, wobei die möglichen Nachteile, die aus schlecht gewählten Lernbeispielen entstehen, vermieden werden müssen.

Der Vorteil des Lernens durch Instruktion ist dagegen darin zu sehen, daß der Kenntniszuwachs nicht so stark vom bereits vorhandenen Wissen des Lernenden abhängt. Wir schlagen deshalb vor, Lernen durch Explorieren und Lernen durch gezielt sequenzierte Instruktion wie folgt zu kombinieren: nach Vermittlung einiger Grundkenntnisse sollte der Lernende möglichst bald Explorieren. Das explorierte System sollte jedoch einen Monitor haben, der auf ein Lernermodell zurückgreift und so Indikatoren für den Lernfortschritt hat. Ein soches Lernermodell könnte auf der Grundlage des S-T-Struktur-Modells aufgebaut werden. Stellt der Monitor fest, daß der Lernende wiederholt alte Eingaben übernimmt oder nur "far misses" produziert, die ihm keine neuen Informationen über das System vermitteln, so wird durch gezielt sequenzierte Instruktion der Kreislauf uninformativer Beispiele unterbrochen. Nach Beseitigung des kritischen Informationsdefi-

zits kann nun der Lernende durch Explorieren weiterhin unter motivierenden Bedingungen Computerkenntnisse erwerben.

Durch eine solche Lernumgebung würde es dem Lernenden ermöglicht, die Erfolgserlebnisse des freien Explorierens durch entdeckendes Lernen zu genießen ohne gleichzeitig die mit dem Explorieren verbundenen Frustrationserlebnisse in Kauf nehmen zu müssen. Während Carroll & Carrithers (1984) vorgeschagen haben, durch "training-wheels" einem neuen Benutzer anfangs nur ein eingeschränkt funktionales System zur Verfügung zu stellen, sollte nach unserem Vorschlag von Anfang an die volle Funktionalität des Systems verfügbar sein. Dadurch wird der Erfahrungsraum für schnelle Lerner nicht eingeschränkt: ein Lernender würde erst dann auf einen bestimmten Ausschnitt des Systems gelenkt, wenn aufgrund des kognitiven Lernermodells eine Lernbarriere festgestellt würde.

Literaturangaben:

Carroll, J.M. & Carrithers, C. (1984): Training wheels in a user interface. Communications of the ACM, 27, 800-806.

Carroll, J.M., Mack, R.L. & Lewis, C.H. (1985): Exploring exploring a word processor. Human-Computer-Interaction, 1985, 1, 283-307.

Habel, Ch. & Rolllinger, C.R. (1984): Lernen und Wissensakquisition. In: Ch. Habel (Hrsg.) Künstliche Intelligenz: Repräsentation von Wissen und natürlichsprachliche Systeme. Frühjahrsschule 1984. Informatikfachberichte, Springer Verlag, Heidelberg.

Robert, J. M. (1986): Some highlights of learning by exploration. In: Proceedings of the International Scientific Conference, Work with Display Units. Stockholm 1986, S. 348-353.

Schmalhofer, F. (1986): The construction of programming knowledge from system explorations and explanatory text: a cognitive model. In C.R. Rollinger & W. Horn (Eds.), GWAI-86 und 2. Österreichische Artificial-Intelligence-Tagung . Heidelberg: Springer-Verlag.

Schmalhofer, F. & Kühn, O. (1986): Die erste Stunde beim Erwerb von Programmierkenntnissen: Eine Computer-Modellierung im Ansatz. In: M. Amelang, Bericht über den 35. Kongress der DGFP in Heidelberg 1986. Verlag für Psychologie, Hogrefe, Göttingen. S. 199.

Schmalhofer, F. & Wetter, Th. (1987): Kognittive Modellierung: Menschliche Wissensrepräsentationen und Verarbeitungsstrategien. In: M.M. Richter & Th. Christaller (Hrsg) Künstliche Intelligenz: Frühjahrsschule Dassel 1986, Informatikfachberichte, Springer-Verlag, Heidelberg.

Winer, B.J. (1971): Statistical principles in experimental design. Second Edition. McGraw-Hill, New York.

Winston, P.H. (1984): Artificial Intelligence. Second Edition. Addison-Wesley, Reading, Massachsetts.

Diese Arbeit wurde durch die Deutsche Forschungsgemeinschaft (Schm 648/1) unterstützt.

Otto Kühn, Franz Schmalhofer
Psychologisches Institut
Universität Freiburg
Niemensstr. 10
D-7800 Freiburg i.Br.