Empirische Pilotstudie zur Konstruktion eines problemlösezentrierten Hilfesystems für einen Problemlösemonitor

Gabriele Janke, Claus Möbus, Heinz-Jürgen Thole

Projekt ABSYNT*, Fachbereich Informatik,
Universität Oldenburg, Postfach 2503, D-2900 Oldenburg

Übersicht

Die hier vorgestellte Studie liefert einen Beitrag zum Thema Tutor- und Hilfesysteme beim Erwerb von Programmierwissen. Anhand einer Untersuchung, in der Versuchspersonen Programmieraufgaben in der grafischen rein funktionalen Programmiersprache ABSYNT lösten, wurden Anforderungen an ein individualisiertes Hilfesystem formuliert und anschließend eine erste Version für das Hilfesystem erarbeitet. Dieses basiert auf einem zeitgleich zu der Untersuchung geschriebenen Diagnostikprogramm, das vom Schüler formulierte Hypothesen überprüft und damit Fehler erkennen und fehlende Teile von ABSYNT-Programmen ergänzen kann.

Eingliederung in das Projekt ABSYNT

Das grundlagenorientierte Ziel unseres Projekts ist die Erforschung kognitiver Prozesse beim Erwerb von Programmierwissen, speziell die Analyse von Problemlöseprozessen beim Programmieren. Unter Problemlösen

* gefördert durch die Deutsche Forschungsgemeinschaft, Schwerpunktprogramm Wissenspsychologie, Förderungsnummer Mo 292/3

verstehen wir das Planen von Programmen, die Fehlersuche und -korrektur.

Das anwendungsbezogene Ziel unseres Projekts - Thema dieses Beitrags - ist die Entwicklung eines adaptiven on-line Hilfesystems, bestehend aus einer kombinierten Diagnose- und Hilfekomponente, die Problemsituationen analysiert und individuelle Hilfen anbietet.

Die Problemlöseumgebung ABSYNT

ABSYNT realisiert eine visuelle, rein funktionale Programmiersprache. Sie entstand aus Ideen von Bauer und Goos (1982), die fertige Programme in Form von Baum-Diagrammen anschaulich machten. In unserem Projekt wurde aus diesen illustrierenden Diagrammen eine lauffähige Programmiersprache mit direkt manipulierbaren Objekten entwickelt (Janke und Kohnert 1989).

ABSYNT-Programme sind Rahmen. Ein Rahmen entspricht einer Funktion in einer funktionalen Programmiersprache. Jeder Rahmen gliedert sich in einen Kopf und einen Körper. Im Kopf wird der Name des Rahmens, der ihn repräsentierende Operatorknoten (selbstdefinierter höherer Operatorknoten) und die Anzahl und Reihenfolge der Parameter des Rahmens festgelegt. Im Körper wird die Rechenvorschrift definiert. Dies geschieht in Form von ABSYNT-Bäumen. Sie sind aus Operatorknoten (repräsentieren Operatoren), Konstanten-Knoten (repräsentieren Konstanten), Parameterknoten (repräsentieren Parameter) und Verbindungslinien zwischen den Knoten

Das Startfenster von ABSYNT entspricht dem Top Level von Lisp. Es kann ebenfalls ABSYNT-Bäume, allerdings ohne Parameterknoten, enthalten. Damit können Programmaufrufe repräsentiert werden.

Der Programmierer baut die Bäume in den Rahmen und im Startfenster mit den in der Knotenleiste zur Verfügung gestellten Knoten auf. Diese werden untereinander verbunden und - falls nötig - beschriftet. Veränderungen können auch

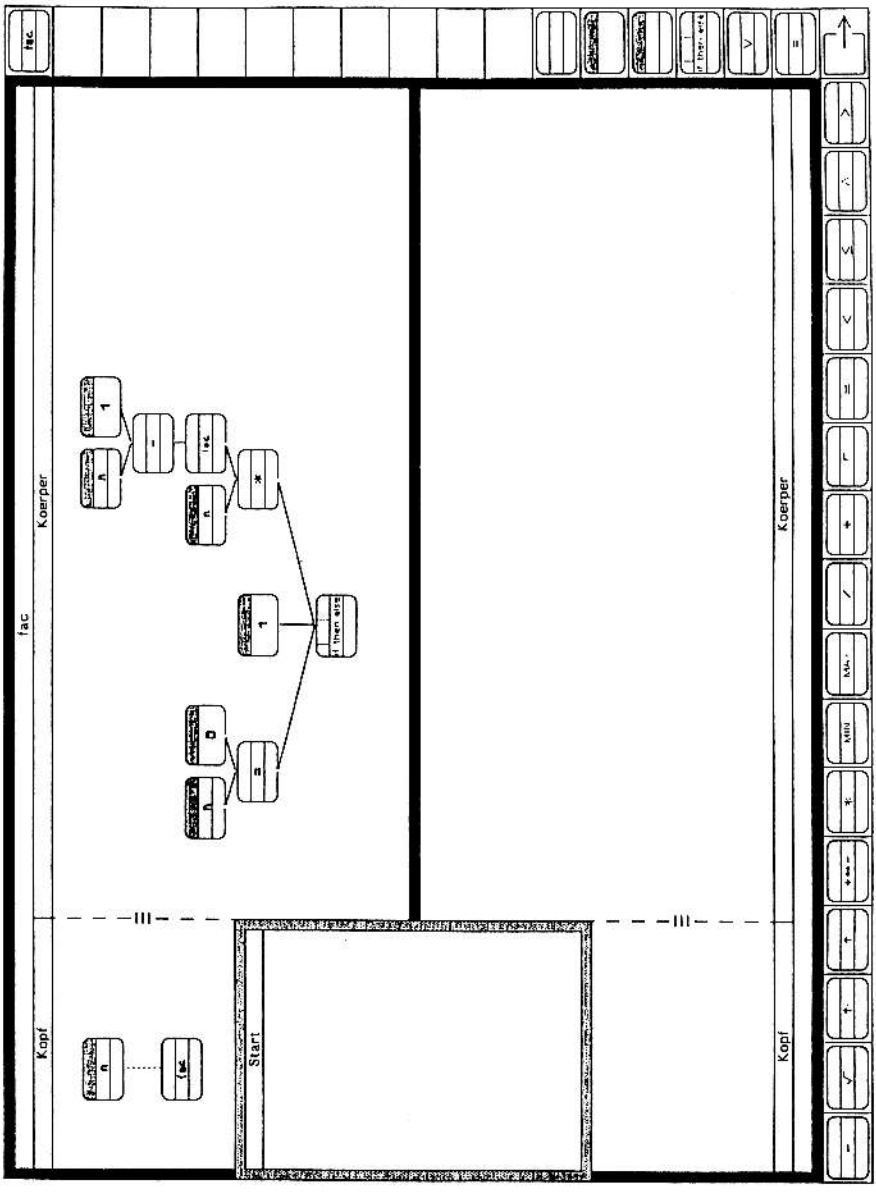

Abb.1

durch Verschieben und Löschen der Knoten und Linien herbeigeführt werden. Abb.1 zeigt einen Zustand des Editors, in dem ein korrektes Programm (Fakultät, rekursiv) im oberen Rahmen erstellt wurde.

Die Berechnung des Programms durch den Interpreter kann schrittweise mit einem Trace demonstriert werden. Mit Hilfe dieses Trace gelang es den Versuchspersonen, alle syntaktischen und viele semantische Fehler zu lokalisieren.

ABSYNT ist in Interlisp/Loops auf einer Siemens 5822 implementiert.

Die Aufgabensequenz

Die Unterrichtsreihe, die gleichzeitig dem Erwerb von Programmierwissen für ABSYNT und der Bearbeitung von Problemen diente, wurde nach Ideen von VanLehn (1987) und Zhu und Simon (1987) entwickelt: Das Wissen, das in den aufeinanderfolgenden Teilen der Unterrichtsreihe erworben wird, baut im Sinne von Gagné (1973) aufeinander auf (Janke 1989/1). Die Untersuchungsergebnisse zeigen, daß bis auf weiteres bei der Analyse von Problemsituationen von der Arbeitshypothese ausgegangen werden kann, daß Fehler und Probleme durch die gerade neu zu erwerbenden Lernziele verursacht werden.

Die Unterrichtsreihe wurde so konzipiert, daß die Versuchsleiter so wenig wie möglich in den Wissenserwerbs- und Problemlöseprozeß eingreifen mußten. Dies garantierte einheitliche Bedingungen für alle Versuchspersonen und eine Vielfalt individueller Lösungen.

Analyse der beobachteten Problemsituationen

Als Problemsituationen haben wir die Situationen definiert, in denen die Versuchspersonen

- den Trace zur Überprüfung eines unvollständigen, fehlerhaften oder suboptimalen Programms verwendeten,

- im Material der Unterrichtsreihe nach passenden Operatoren oder früheren Programmen suchten,

- die Versuchsleiter intervenierten.

Diese haben wir nach den folgenden Gesichtspunkten analysiert (Janke, 1989/2):

- Syntax (Fehlerhaftigkeit, Unvollständigkeit)

- Inhalt des bisherigen Entwurfs (Fehlerhaftigkeit, Unvollständigkeit, Optimalität)

- fehlende Programmteile

- Art und Wirkung der gegebenen Hilfen

Da die Syntaxfehler leicht zu behandeln sind (Janke, 1989/2), sollen im folgenden nur noch die inhaltlichen Fehler und Programmteile berücksichtigt werden.

Drei Gruppen von Problemsituationen mit inhaltlichen Fehlern bzw. Lücken konnten nach ihren beobachtbaren Ursachen unterschieden werden: Die Versuchsperson

- hatte die Aufgabe oder Teile der Aufgabestellung falsch verstanden

- fand keinen geeigneten Operator für die Programmierung eines Teilziels

- interpretierte die operationale Semantik eines Operators falsch.

Die übrigen Problemsituationen lassen sich nicht so leicht nach ihren Ursachen kategorisieren, weil diese nicht direkt beobachtbar sind. Es ist auch nicht allgemein möglich, die Ursachen für eine Problemsituationen allein aus den Situationen am Bildschirm abzuleiten. Häufig läßt sich jedoch ein Zusammenhang zwischen einer Problemsituation und einem

gerade neu zu erwerbenden Lernziel feststellen (Janke, 1989/2).

Die von Derry (1989) vorgeschlagenen Fehlerkategorien sind für unsere Aufgabensequenz als Grundlage für Hilfesstellungen zu grob, was auch in der wesentlich größeren Verschiedenartigkeit unserer Aufgaben begründet ist (von einfachen Rahmen über Abstützung bis zur Rekursion).

Die Analyse der seitens der Versuchsleiter gegebenen Hilfen ergab, daß der Hilfesuchende häufig schon durch sehr unspezifische Hinweise in die Lage versetzt wurde, Fehler zu beheben bzw. das Programm zu ergänzen. Der benötigte Genauigkeitsgrad für eine Hilfe scheint nicht zuletzt individuenabhängig zu sein.

Anforderungen an das Hilfesystem

Aus der Analyse aller in der Untersuchung vorgekommenen Problemsituationen (Janke 1989/2) ergaben sich folgende Anforderungen an ein individualisiertes Hilfesystem, die in den Punkten 4 und 5 mit den dort genannten Autoren im Einklang stehen:

1. Auf eine große Vielfalt von verschiedenen Lösungen muß eingegangen werden können.

2. Alle Typen von Problemsituationen müssen behandelt werden können: 1)syntaktisch unvollständige a)schon fehlerhafte, b)bisher korrekte, 2)syntaktisch vollständige fehlerhafte und 3)korrekte aber umständlich Lösungsansätze.

3. In allen Problemsituationen müssen die Programme bzw. Programmteile diagnostiziert werden können, d.h. von Teilbäumen muß auf angestrebte Teillösungen (Teilziele) inferiert werden können.

4. In allen Problemsituationen müssen prinzipiell Hilfen generiert werden können. Für die Entscheidung über Zeitpunkt und Art der Hilfestellung müssen Schülerdaten berücksichtigt werden, z.B.: aktueller Problembereich, schon erreichte Lernziele, frühere Fehler, frühere

Strategien,... (Anderson,1989, Clancey,1986, Corbett,1989, Moll,1989).

5. Die Hilfen sollten in abgestufter Form gegeben werden können (vom allgemeinen Hinweis zur konkreten Information) und diese Abstufung sollte indiviuell einstellbar sein (Moll, 1989).

Vorversion des Hilfesystems

Das Hilfesystem wird auf dem parallel zu der Untersuchung entwickelten Diagnostikprogramm basieren. Dieses bildet eine Ziel-Mittel-Relation, die das Parsen und Generieren von Lösungen realisiert. Die Ziel-Mittel-Relation kann auch als UND/ODER-Graph im Sinne von Nilsson (1980) interpretiert werden (Möbus und Schröder, 1989).

Abb.2 zeigt einen UND/ODER-Graph, der das Prinzip des Diagnostikprogramms verdeutlicht (im folgenden "UOG" genannt). Er veranschaulicht einen Ausschnitt der Relation, die Lösungen für Körper der "gerade"-Aufgabe generieren und parsen kann. Die "gerade"-Aufgabe besteht darin, ein Programm zu schreiben, das prüft, ob eine natürliche Zahl gerade ist oder nicht (Bauer und Wössner, 1984).

Zunächst die Erklärung der Symbole:

Die Rechtecke repräsentieren Ziele, die als Knoten im ABSYNT-Baum dargestellt sind (terminale Symbole im Sinne einer Grammatik).

Die Ovale repräsentieren Ziele, die noch nicht direkt als Knoten des ABSYNT-Baums dargestellt sind (nicht terminale Symbole).

Verzweigungen, die durch einen Bogen verbunden sind, sind UND-Verzweigungen. Der linke Ast jeder UND-Verzweigung stellt die (Teilbaum-)Wurzel einer konkreten ABSYNT-Lösung dar, die restlichen die Ziele der Eingänge dieser Wurzel.

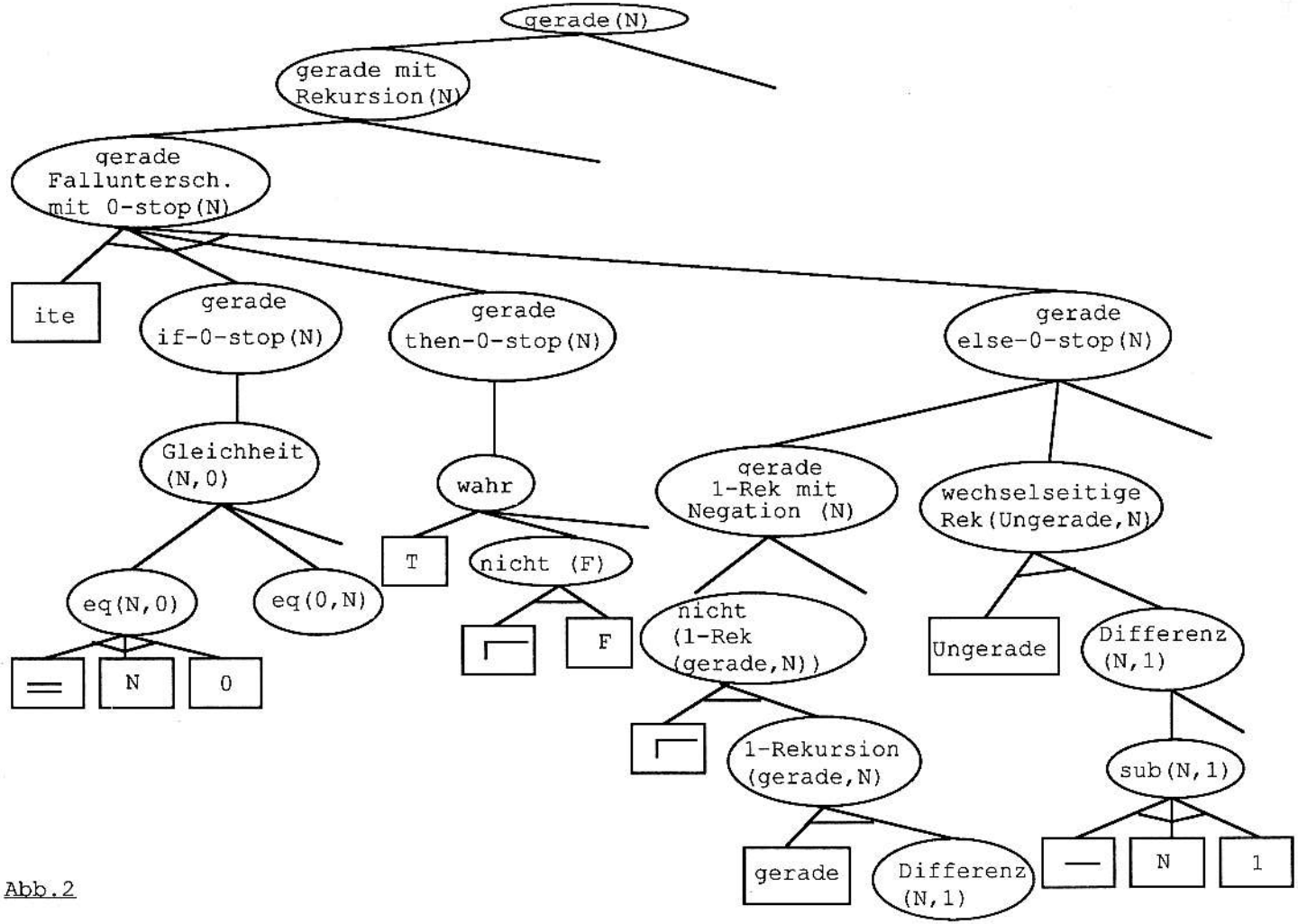
gerade(N)
gerade mit Rekursion(N)
gerade Falluntersch. mit 0-stop(N)
ite
gerade if-0-stop(N)
gerade then-0-stop(N)
gerade else-0-stop(N)
Gleichheit (N,0)
wahr
gerade 1-Rek mit Negation (N)
wechselseitige Rek(Ungerade,N)
eq(N,0)
eq(0,N)
T
nicht (F)
F
nicht (1-Rek (gerade,N))
Ungerade
Differenz (N,1)
=
N
0
1-Rekursion (gerade,N)
sub(N,1)
gerade
Differenz (N,1)
—
N
1

Abb.2

Verzweigungen, die nicht durch einen Bogen verbunden sind, sind ODER-Verzweigungen.

Eine konkrete Lösung, d.h. ein ABSYNT-Baum, wird generiert, indem von oben kommend entlang jeweils einer ODER-Verbindung und aller UND-Verbindungen die konkreten ABSYNT-Knoten verbunden werden unter Verwendung des jeweils linken Zweiges einer UND-Verzweigung als (Teilbaum-)Wurzel. Endet dabei ein Zweig des UOG bei einem nicht konkretisierten Teilziel in einem Oval, dann muß man zu der Stelle im UOG springen, wo dieses Teilziel weiter differenziert wird.

Umgekehrt können ABSYNT-(Teil-)Bäume (Teil-)Zielen des UOG zugeordnet werden.

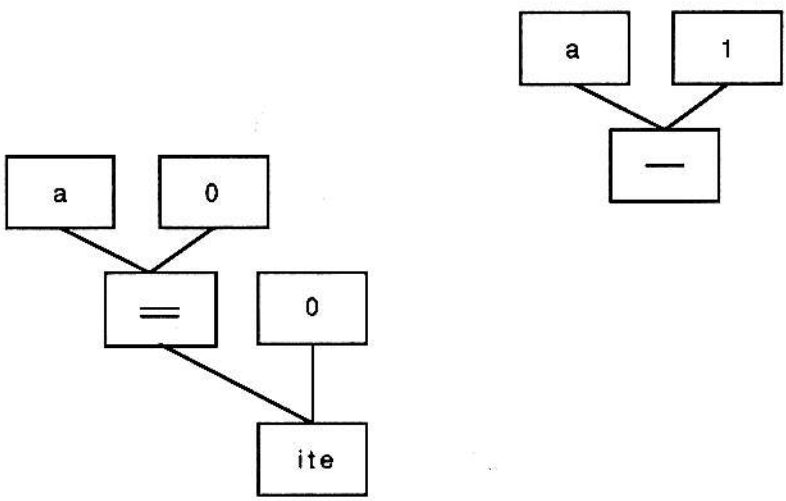

Abb.3 Skizze des Körpers eines syntaktisch unvollständigen, fehlerhaften Programmentwurfs zur "gerade"-Aufgabe

In der in Abb.3 gezeigten Problemsituation kann z.B. der if-Zweig der Fallunterscheidung (ite) dem Teilziel "eq(N,0)" des UOG bzw. den übergeordneten unspezifischeren Teilzielen "Gleichheit(N,0)" und "gerade if-0-stop(N)" zugeordnet werden. Für den bisher fehlerhaften then-Zweig können richtige Ergänzungen (z.B."T") geliefert werden. Der rechte Teilbaum "a

- 1" kann dem Teilziel "sub(N,1)" und damit dem übergeordneten unspezifischeren Teilziel "Differenz(N,1)" zugeordnet werden. Dieses wiederum ist in zwei verschiedenen höheren Teilzielen enthalten ("gerade 1-Rek mit Negation(N)" und "wechselseitige Rek(Ungerade,N)"). Auf diese Weise sind zwei Gruppen von Ergänzungen generierbar. Insgesamt kann durch Parsen und Generieren der fehlerhafte und unvollständige Programmentwurf zu einer richtigen Lösung ergänzt werden. Bei der Ergänzung sollte mit Hilfe eines Schülermodells der für jeden Schüler individuell am besten geeignete Vorschlag ausgewählt werden.

Da die automatische Präsentation einer korrekten Ergänzung mit dem Charakter eines Problemlösemonitors nicht vereinbar ist, soll mit Hilfe der im UOG enthaltenen Teilziele ein abgestuftes Hilfesystem entwickelt werden, das fehlende Programmteile mit Teilzielen in Beziehung setzt (Moll, 1989).

Weitere Forschung

Bei der Generierung von Hilfen entstehen zwei Schwierigkeiten:

1. die Auswahl der für den Schüler geeigneten generierten Ergänzung aus der Menge aller möglichen Ergänzungen

2. die Verbalisierung von Hinweisen, die nicht einfach die Lösung zeigen, sondern die als Erklärung akzeptiert werden

Für die Auswahl der optimalen Schülerhilfe müssen Schülerdaten (Clancey, 1986) einbezogen werden, die als Indikatoren für seinen Wissensstand gelten können.

Hinweise für mögliche Ergänzungen wollen wir aus den Teilzielen des UOG ableiten. Dazu ist es notwendig, die bisher nur aus der Aufgabenanalyse entstandene Zielstruktur empirisch zu validieren, so daß man von einer psychologisch fundierten Intentionsdiagnostik im Sinne von Johnson (1986) sprechen kann.

Literatur

Anderson, J.R., "Psychology and Intelligent Tutoring", in: Bierman, Breuker, Sandberg (Hrsg.) Proceedings of the 4th International Conference on AI in Education, in Amsterdam, 24-26 März 1989, Amsterdam: IOS, 1989, 1

Bauer, F.L. und Goos, G., Informatik, 1. Teil, Berlin, Springer 1982 (3. Auflage)

Bauer,F.L und Wössner,H. "Algorithmische Sprache und Programmentwicklung",Berlin 1984

Clancey, W.J. "Qualitative Student Models", Annual Review of Computer Science, 1986, Vol.I, 381 - 450

Corbett, CA.T. und Anderson, J.R, "Feedback Timing and Student Control in the Lisp Intelligent Tutoring System", in: Bierman, Breuker, Sandberg (Hrsg.) Proceedings of the 4th International Conference on AI in Education in Amsterdam, 24-26 März 1989, Amsterdam: IOS, 1989, 64 - 72

Derry, Sh.J., "Characterizing the Problem Solver: A System for On-line Error Detection", in: Bierman, Breuker, Sandberg (Hrsg.) Proceedings of the 4th International Conference on AI in Education in Amsterdam, 24-26 März 1989, Amsterdam: IOS, 1989, 86-91

Gagné, R.M., "Der Erwerb von Wissen" in: Hofer, Weinert (Hrsg.) Pädagogische Psychologie 2, Fischer Taschenbuchverlag, Frankfurt a.M., 1973, 106- 123

Janke,G. "Voruntersuchung zum Programmieren mit ABSYNT, Teil 1: Aufbau der im Versuch verwendeten Aufgabensequenz", ABSYNT Memo 1/89, Universität Oldenburg, 1989

Janke,G. "Voruntersuchung zum Programmieren mit ABSYNT, Teil 2: Versuchsbeschreibung und Analyse aller Problemsituationen", ABSYNT Memo 2/89, Universität Oldenburg, 1989

Janke, G. und Kohnert, K."Interface design of a visual programming language: evaluating runnable specifications", in: Klix, Streitz, Waren, Wandtke (Hrsg.), MACINTER II Man-Computer Interaction Research, Proceedings of the Second Network Seminar of MACINTER, Berlin/DDR, 21. - 25. März 1988, Amsterdam: North Holland, 1989, 567 - 581

Johnson, W.L., "Intention-Based Diagnosis of Novice Programming Errors", Los Altos, CA.: Morgan Kaufmann Publishers, 1986

Möbus, C. und Schröder, O. "Knowledge specification and instruction for a visual computer language", in: Klix, Streitz, Waren, Wandtke (Hrsg.), MACINTER II Man-Computer Interaction Research, Proceedings of the Second Network Seminar of MACINTER, Berlin/DDR, 21.-25.März 1988, Amsterdam: North Holland, 1989, 535 - 565

Möbus, C. und Schröder, O. "Zur Entwicklung des Problemlösemonitors ABSYNT: Wissenserwerbsmodellierung und Hilfekomponente", in: Kreowski, Krieg-Brückner (Hrsg.) Berichte aus dem Fachbereich Informatik der Universtät Bremen, 2. Tagung zur Küsteninformatik, 18.-20. Mai 1989 in Bederkesa (im Druck)

Moll, Th."Über die Verbesserung der Benutzerunterstützung durch ein Online-Tutorial", in Maaß, Oberquelle (Hrsg.) Software-Ergonomie '89, Stuttgart:Teubner 1989, 223 - 232

Nilsson, N. "Principles of Artificial Intelligence", Palo Alto, Ca.: Tioga Press, 1980

VanLehn, K.:"Learning one Subprocedure per Lesson" Artificial Intelligence 31 (1987) 1 - 40

Zhu,X. und Simon,H. A.:"Learning mathematics from examples and by doing", interner Forschungsbericht der Chinese Academy of Sience und der Carnegie-Mellon University, 5.März 1987

F. Stetter W. Brauer (Hrsg.)

Informatik und Schule 1989: Zukunftsperspektiven der Informatik für Schule und Ausbildung

GI-Fachtagung
München, 15.-17. November 1989

Proceedings

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo Hong Kong

Herausgeber

F. Stetter
Fakultät für Mathematik und Informatik, Universität Mannheim
A 5, D-6800 Mannheim 1

W. Brauer
Institut für Informatik, Technische Universität München
Postfach 202420, D-8000 München 2

3. Fachtagung „Informatik und Schule", veranstaltet vom Fachbereich 7 „Ausbildung und Beruf" der GI und dem Institut für Film und Bild in Wissenschaft und Unterricht (FWU)

CR Subject Classification (1987): K.3

ISBN 3-540-51801-0 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-51801-0 Springer-Verlag New York Berlin Heidelberg

Druck- und Bindearbeiten: Weihert-Druck GmbH, Darmstadt
2145/3140-543210 - Gedruckt auf säurefreiem Papier