De Corte, Erik; Verschaffel, Lieven; Schrooten, Hilde

Kognitive Effekte computergestützten Lernens. Zum Stand der Forschung

Unterrichtswissenschaft 20 (1992) 1, S. 12-33

Unterrichtswissenschaft

Zeitschrift für Lernforschung
20. Jahrgang/Heft 1/1992

Thema:
Kind und Computer

Verantwortliche Herausgeber:
Gunther Eigler und Norbert M. Seel

Allgemeiner Teil

Erik De Corte, Lieven Verschaffel, Hilde Schrooten

Kognitive Effekte computergestützten Lernens: Zum Stand der Forschung

Cognitive Effects of Computer-Oriented Learning

In diesem Beitrag wird der Versuch unternommen, den Stand der Forschung zu kognitiven Effekten des Programmierens in der Sprache Logo zu beschreiben. Im Gegensatz zu den enttäuschenden Befunden früherer Untersuchungen berichten neuere Untersuchungen eher von positiven Ergebnissen in bezug auf den Erwerb und Transfer von Denkfähigkeiten. Ein Schlüsselfaktor für den Erfolg dieser Untersuchungen ist, daß wirksame Lehr-Lern-Umgebungen entwickelt wurden, die die Fehler der früheren Forschung in Rechnung stellten. Wir beginnen mit einem Überblick über eine repräsentative Stichprobe dieser Untersuchungen und identifizieren dann die wesentlichen Merkmale effektiver Lehr-Lern-Umgebungen. Im Anschluß daran formulieren wir einige Annahmen für die zukünftige Forschung. Selbstverständlich wird auch die Fragestellung thematisiert, welchen Beitrag Logo zu den berichteten Befunden geleistet hat.

In this contribution an attempt is made to outline the state of the art of the research concerning the cognitive effects hypothesis with respect to Logo programming. Contrary to the disappointing findings of the earlier studies investigating this hypothesis, some more recent investigations have reported rather positive results with respect to the acquisition and transfer of thinking skills. A key factor in the success of these studies is that powerful teaching-learning environments were developed, taking into account the shortcomings of previous research. Starting from a review of a representative sample of these studies, the crucial characteristics of such teaching-learning environments are identified and suggestions for future research are formulated. The issue of the contribution of Logo to the obtained results is also addressed.

1. Einleitung

Denken und Problemlösen zu lernen, wird heutzutage allgemein als ein bedeutendes Bildungsziel betrachtet (vgl. Resnick, 1987). In der Tat ist es in unserer sich rasch verändernden und zunehmend komplexer werdenden Welt nahezu unmöglich, Schüler mit all dem notwendigen Wissen auszustatten, damit sie mit jedem auftretenden Problem fertig werden können. Deshalb ist es unerläßlich, ihnen solche Fähigkeiten beizubringen, die sie brauchen um Probleme autonom und kreativ zu lösen.

Seit Anfang der 70er Jahre wird häufig argumentiert, daß Computer zu programmieren förderlich sei für die Entwicklung bedeutsamer Denk- und Problemlösefähigkeiten, die dann auch auf andere Inhaltsbereiche übertragen werden könnten.

Bei einer großen Anzahl von Untersuchungen, die im vergangenen Jahrzehnt durchgeführt wurden, um kognitive Effekte von Programmie-

ren, besonders in der Sprache Logo, nachzuweisen, konnten allerdings keine positiven Effekte im Hinblick auf den Transfer von Denkfähigkeiten festgestellt werden (vgl. De Corte & Verschaffel, 1989; Swan & Black, 1987). Gleichwohl wurde die Annahme bezüglich der kognitiven Effektivität des Programmierens bisher nicht aufgegeben. Vielmehr führte man die ernüchternden Befunde der meisten Untersuchungen auf verschiedene methodische Schwächen zurück.
Wir werden gleich drei Hauptmängel dieser Forschung diskutieren: (1) Die Schüler erwarben keine ausreichende Fähigkeit im Programmieren, was einerseits auf die kurze Dauer praktischer Erfahrungen und andererseits auf das Fehlen einer systematischen Instruktion zurückzuführen ist; (2) es fehlte eine explizite und systematische Zielvorgabe, einen Transfer erreichen zu wollen; (3) es wurden nicht die richtigen Transferaufgaben verwendet. Danach beschreiben wir einige neuere Untersuchungen, in denen mit Erfolg versucht wurde, die genannten Mängel zu beheben. Zuletzt werden wir versuchen, die positiven Merkmale dieser Untersuchungen zu identifizieren und als Richtlinien für die zukünftige Forschung herauszustellen.

2. Fehlerquellen der früheren Untersuchungen

Unzureichendes Programmierwissen und ungenügende Programmierfähigkeiten

Das Ausbleiben von Transfereffekten in den frühen Untersuchungen ist größtenteils auf unzureichendes Wissen über Techniken des Programmierens und auf ungenügende Programmierfähigkeiten der kindlichen Probanden zurückzuführen. Heutzutage wird wohl jedermann der Aussage zustimmen, daß ein Minimum an Kompetenz eine conditio sine qua non ist, damit Transfer überhaupt möglich wird. Im Gegensatz dazu versäumten aber einige der frühen Studien, die Programmierfähigkeit der Kinder zu überprüfen. Andere Untersuchungen zeigten, daß Kinder bei ihren Bemühungen, Programmieren zu lernen, im Durchschnitt keine befriedigenden Resultate erzielten (vgl. Carver & Klahr, 1986; Milojkovic, 1983; Pea & Kurland, 1983). Dieses Ergebnis wird durch verschiedene neuere Untersuchungen untermauert, die speziell auf eine Evaluation der kindlichen Programmierfähigkeiten abzielten. Diese Untersuchungen zeigten, daß die meisten Kinder am Ende eines Programmierkurses weder in der Lage waren, die Kernkonzepte der Programmiersprache (wie z.B. „Variable", „Iteration" und „Rekursion") effektiv zu nutzen, noch konnten sie etwas mit den heuristischen Verfahren anfangen, die als charakteristisch für die Konstruktion von Computerprogrammen betrachtet werden (wie zum Beispiel „Planen", „Problem-Zerlegung" und „Debugging") (vgl. für einen detaillierteren Überblick: De Corte & Verschaffel, 1989).

Was ist der Grund dafür, daß Kinder in diesen Untersuchungen nur ein mäßiges bereichsspezifisches Wissen und ebenso mäßige Fähigkeiten erwarben? Eine erste Ursache sehen wir in der kurzen Dauer der Erfahrungen mit Logo. In den meisten Untersuchungen konnten die Kinder durchschnittlich weniger als 30 Stunden mit dem Computer arbeiten. Nun kann aber begründet angenommen werden, daß dies völlig unzureichend ist, um ein angemessenes Niveau an Expertise im Programmieren zu erreichen. Denn nach allgemeinen Schätzungen der kognitionspsychologischen Literatur erfordert die Entwicklung von Expertentum in einem komplexen Bereich hunderte oder gar tausende von Stunden praktischer Erfahrung. Da das Programmieren eines Computers wirklich eine komplexe Fähigkeit ist, sollte man nicht erwarten, daß Kinder innerhalb weniger Monate — oder auch eines Jahres — eine Meisterschaft im Umgang mit Logo entwickeln können (vgl. Leron, 1985; Sleeman, 1985).

Ein zweiter Grund für die nur mäßige Programmierkompetenz von Kindern liegt sicherlich auch in der schlechten Qualität der Lernumgebung. Papert (1980) zufolge können Programmierfähigkeiten auf der Basis einer Strategie des selbständigen Entdeckens erworben werden. Programmieren in Logo sollte demnach in einer spontanen und natürlichen Weise erfolgen — analog zu der Art, wie ein Kind zu sprechen lernt. Er nennt dies „Lernen ohne belehrt zu werden" oder „Lernen ohne Curriculum". Diese Auffassung vom Lehr-Lern-Prozeß geht auf die konstruktivistische Sichtweise von Piaget zurück, daß Kinder aufgrund der Interaktion mit der Umwelt ihre eigenen intellektuellen Strukturen aufbauen und entwickeln; Unterricht kann diese intellektuelle Entwicklung nur begleiten, aber nicht entscheidend beeinflussen. In der Praxis beurteilten viele Lehrer und Forscher Paperts (1980) Aussagen als entmutigend: Was sollte irgendeine pädagogische Intervention bewirken, wenn sie nur auf Initiativen der Kinder zu reagieren hat? Neuerdings wird zwar gelegentlich wieder die Auffassung vertreten, daß einheitliche Merkmale der Logo-Lernumgebung doch irgendwie den Aufbau von Programmierkenntnissen und -fähigkeiten beeinflussen (vgl. Leron, 1985). Doch die Mehrheit der Forscher und Lehrer lehnt diese Sichtweise als unzureichend ab; sie nehmen relativ übereinstimmend an, daß die unzureichende Expertise in der Logo-Programmierung, die Kinder in den früheren Studien erreichten, direkt mit dem Fehlen einer angemessenen Instruktion verbunden sei.

Folgerichtig betonen sie die Bedeutung einer direkteren Instruktion, die nicht mehr allein auf die Beherrschung der relevanten Logo-Konzepte, sondern auch und insbesondere auf den Erwerb wesentlicher Denkfähigkeiten abzielen sollte. Eine bedeutende Vorbedingung dafür ist, daß diese Denkfähigkeiten von Anfang an genau zu identifizieren und zu konzeptualisieren sind, was in vielen der frühen Studien nie der Fall war. Greeno (1976) hat diese Bedingung wie folgt umschrieben: „We can

generally do a better job of accomplishing something and determining how well we have accomplished it when we have a better understanding of what we are trying to accomplish" (p. 123).

Obwohl die Tendenz, im Unterricht mehr Struktur einzuführen, immer deutlicher hervortritt, wird wohl niemand die Notwendigkeit leugnen, den Kindern auch Gelegenheiten zu bieten, Logo-Aufgaben frei zu explorieren. Anders ausgedrückt: Man betrachtet offensichtlich mehr und mehr das „Entdeckenlassende Lehren" als eine komplementäre Strategie des Unterrichts. Leron (1985) spricht in diesem Zusammenhang von einem „quasi-Piagetian learning" und charakterisiert es folgendermaßen: „Though given a more active role to teachers and learning materials, it is still closer to Papert's Piagetian learning than to the sort of teaching normally found in schools" (p. 32).

Das Fehlen einer intentionalen und systematischen Transfer-Orientierung

Ein zweiter Aspekt, der in Betracht zu ziehen ist, um die enttäuschenden Ergebnisse der frühen Untersuchungen zu erklären, liegt in deren Orientierung an einem Lehren, das nicht systematisch und intentional auf Transferleistungen abgestimmt war. In diesen Untersuchungen wurde mehr oder weniger implizit angenommen, daß Programmieren unmittelbar und automatisch einen Transfer bewirken könnte. In Anbetracht der vorliegenden Forschungsbefunde und der Literatur zum Lerntransfer kann man diese Annahme nur mehr als Wunschdenken beurteilen: Es wird immer deutlicher, daß Begriffe und Fähigkeiten, die in einem Inhaltsbereich erworben werden, nicht spontan auf andere Inhaltsbereiche übertragen werden (vgl. Gick & Holyoak, 1983; Pressley, Snyder & Cariglia-Bull, 1987).

Zur Zeit teilen die meisten Forscher den Standpunkt, daß, wenn Lernende kognitive Transferleistungen erbringen sollen, es notwendig ist, das Lehren ausdrücklich und intentional darauf abzustimmen. Das impliziert, daß die Lehrer den Kindern zeigen, wie sie Fähigkeiten, die sie in einem Bereich erwerben (z.B. in einer Logo-Umgebung), effektiv auf andere Problemsituationen anwenden können; das aber schließt ein, daß sie den Kindern Gelegenheiten bieten müssen, um überhaupt Erfahrungen machen zu können, wie bestimmte kognitive Prozeduren in unterschiedlichen Kontexten angewandt werden können.

Unangemessenheit der Transfer-Messungen

Ein drittes Element für die Erklärung der enttäuschenden Befunde der frühen Transfer-Untersuchungen bezieht sich auf die Natur der Aufgaben, die benutzt wurden, um Transferleistungen zu messen. Eine

Hauptforderung muß darin bestehen, daß die Tests auch tatsächlich die Denkprozesse messen, die in der Lernsituation trainiert wurden. Anders ausgedrückt: Es reicht nicht aus, Tests auf der Basis ihrer „Augenschein-Validität" auszuwählen oder zusammenzustellen. In den meisten frühen Untersuchungen wurde diese Bedingung nicht erfüllt. So gelangte Carver (1986) zu dem Urteil: "In addition to neglecting to document learning, most researchers fail to carefully consider the skills they expect students to learn, so there is often a mismatch between the transfer tests and the training" (10).

Eine andere Forderung an die Konstruktion und/oder Auswahl von Transfer-Aufgaben betrifft die altbekannte Unterscheidung zwischen „nahem" (horizontalem) und „fernem" (vertikalem) Transfer (Burton & Magliaro, 1986). „Naher" Transfer meint die Übertragung einer erworbenen Fähigkeit auf eine Situation, die der ursprünglichen Lernsituation mehr oder weniger ähnlich ist; „ferner" Transfer schließt demgegenüber die Ausweitung einer Fähigkeit auf Kontexte ein, die sich wesentlich von der Situation unterscheiden, in der die Fähigkeit erworben wurde. In den meisten frühen Untersuchungen hat man ausschließlich Messungen des „fernen" Transfers vorgenommen. Nun zeigt aber die Geschichte der Problemlöseforschung in überzeugender Weise, daß gerade ein Transfer allgemeiner Denkfertigkeiten nur schwer zu erreichen ist; wenn irgendwelche positiven Befunde erzielt wurden, so sprachen sie eher für einen „nahen" Transfer. Es gibt wirklich keinen Grund zu glauben, daß Logo eine magische Ausnahme bilden könnte. Deshalb wird man in zukünftigen Untersuchungen notwendigerweise eine größere Vielfalt von Transfer-Aufgaben benutzen müssen, die Probleme einschließen, die sich nicht allzu stark von denen unterscheiden, die in der Programmier-Umgebung zu bearbeiten sind.

Schließlich wollen wir noch einen Gesichtspunkt erwähnen, der insbesondere von Papert (1987) und anderen, seiner Argumentation nahestehenden Forschern betont wird: Sie argumentieren, daß die gegenwärtig verfügbaren Evaluationsinstrumente nicht adäquat seien, um die subtilen Veränderungen in den kindlichen Problemlöseprozessen zu messen, die aus Logo-Erfahrungen resultieren. Aus ihrer Sicht müssen die üblichen kollektiven Papier- und Bleistift-Tests, mit denen man nur Endprodukte feststellen kann, durch eher prozeßorientierte Evaluationstechniken ergänzt werden.

Tatsächlich konnten in verschiedenen Untersuchungen, die eine oder mehrere der gerade genannten Schwächen der früheren Forschung berücksichtigten, Transfereffekte erzielt werden. Tabelle 1 faßt die Schlüsselmerkmale einer repräsentativen Stichprobe dieser Studien zusammen. Im Anschluß daran werden wir drei Untersuchungen detaillierter beschreiben.

Tabelle 1:
Überblick über die wesentlichen Merkmale der neuesten erfolgreichen Untersuchungen.

STUDY	GRADE	CONDITIONS	DURATION	SKILLS	RESULTS
CARVER, 1988	3-6	LOGO PR.-SOLV. GRAPH. + LISTS LISTS + GRAPH.	50 HRS	DEBUGGING	+
CLEMENTS, IN PRESS	3	LOGO PR.-SOLV. + TRANSFER INSTR. CONTROL	58 HRS	DECIDING ON NATURE OF PROBLEM CHOOSING PERFORMANCE STRATEGY SELECTING REPRESENTATION MONITORING	+/- - + +
LEHRER, 1988	3	LOGO PR.-SOLV. LOGO GEOMETRY CONTROL	58 HRS	PROBLEM SOLVING (TOH) PLANNING METACOGNITION	- + -
LITTLEFIELD, 1988	5	LOGO PR.-SOLV. LOGO PR.-SOLV. + TRANSFER INSTR. CONTROL	25 HRS	PROBLEM DECOMPOSITION IDENTIFICATION OF ERRORS PLANNING AHEAD	? ? +
SWAN, 1989	4-6	LOGO PR.-SOLV. CUT-PAPER MANIP. CONTROL	45 HRS	SUBGOALS FORMATION FORWARD CHAINING SYSTEMATIC TRIAL AND ERROR ALTERNATIVE REPRESENTATION ANALOGIES	+ + + + +
DE CORTE, 1990	6	LOGO PR.-SOLV. LOGO PR.-SOLV. + TRANSFER INSTR. CONTROL	60 HRS	PLANNING DEBUGGING PROBLEM DECOMPOSITION USING EXTERNAL REPRESENTATION	- + + +

3. Neuere erfolgreiche Untersuchungen

Die Untersuchung von De Corte und Kollegen

Ausgangspunkt für das über ein Jahr dauernde Lehrexperiment von De Corte, Verschaffel & Schrooten (i.Dr.) waren zum einen die Ergebnisse der frühen Logo-Transferuntersuchungen und zum anderen die verfügbare Literatur zum Transfer von Problemlösefähigkeiten im allgemeinen. Wir nahmen an, daß ein Transfer kognitiver Prozesse erzielt werden kann, wenn zumindest zwei der folgenden drei Bedingungen erfüllt werden: (1) Die Schüler haben hinreichendes bereichsspezifisches Wissen erworben, d.h. sie kennen die Logo-Konzepte und -Prozeduren; (2) sie beherrschen in ausreichendem Maße die intendierten Denkprozesse im Kontext der Programmierung in Logo; (3) die Schüler haben gelernt, diese Denkprozesse in wenigstens einem anderen Inhaltsbereich anzuwenden. Im einzelnen wurden folgende Hypothesen formuliert: Wenn die beiden ersten Bedingungen erfüllt sind, gelingt Transfer; die Erfüllung der dritten Bedingung steigert den Transfer-Effekt.

De Corte et al. führten in drei Klassen der 6. Stufe mit jeweils 24 Kindern ein Experiment mit einem Vortest-Nachtest-Design und einer Kontrollgruppe durch. Zwei Klassen dienten als Experimentalgruppen (E1 und E2), die dritte als Kontrollgruppe. In E1 standen die ersten beiden genannten Transfer-Bedingungen im Mittelpunkt, in E2 alle Bedingungen. Die Kontrollklasse wurde nicht in Logo trainiert, während

in beiden Experimentalgruppen jeweils an einem Nachmittag pro Woche über ein ganzes Jahr ein Logo-Kurs durchgeführt wurde (das waren insgesamt etwa 60 Stunden); die Klassen waren jeweils mit neun Computern ausgestattet.

Die zweite von uns unterschiedene Transferbedingung, die auf eine ausreichende Programmierfähigkeit zielt, versuchten wir dadurch zu verwirklichen, daß wir eine metakognitive Strategie in den Mittelpunkt der Logo-Programmierung stellten. Diese Strategie konzentrierte sich auf *Planen* und *„debugging"* (Fehlerbeseitigung), während der gesamte Logo-Kurs zwei heuristische Methoden in das Zentrum der Lehraktivitäten rückte, nämlich die Zerlegung (*„decomposition"*) eines Problems in Teilprobleme sowie die *Konstruktion einer externalen Repräsentation* der Problemstellung. Die beiden Phasen der Programmierstrategie, Planung und „debugging", umfaßten jeweils mehrere Arbeitsschritte. Die *Planung* wurde unabhängig vom Computer durchgeführt, eine integrierte Verschlüsselungs- und Testphase dagegen mit Hilfe des Computers.

Für die *Phase der Planung* wurden drei Arbeitsschritte unterschieden (vgl. Abbildung 1):

— Anfertigung einer Zeichnung des intendierten Bildschirmeffekts,
— Konstruktion einer Art Baumdiagramm, in dem die komplexe Zeichnung in Bausteine unterteilt wurde, die leicht zu programmieren sind; dieses Diagramm umfaßt zugleich die Sequenz, in der die unterschiedlichen Teile auf dem Bildschirm zu zeichnen sind.
— Anfertigung separater Zeichnungen zu den verschiedenen Bausteinen bzw. Teilen, um für jeden Teil ebenso die Längen und Winkel anzugeben wie die Start- und Endposition der „turtle".

Auf die Planungsphase folgt die integrierte *Ausführungs- und Testphase* am Computer. Diese Aktivität wird durch zwei Prinzipien geleitet, nämlich *„Topdown"-Programmieren* und unmittelbares *Testen und Beseitigen der Fehler* (vgl. zur Veranschaulichung dieser Prinzipien: De Corte et al., i.Dr.).

"Top-down"-Programmieren beinhaltet, daß die Kinder angeleitet werden, mit der globalsten Prozedur, der sog. „Mutter-Prozedur", zu beginnen; sie umfaßt zum einen die Namen der nachfolgenden Bausteine (der zweiten Ebene des Diagramms) und zum anderen die Namen der „Verbindungslinien", die die „turtle" von der Endposition eines Bausteins zu der Startposition des nächsten Bausteins bewegen. Nacheinander wird jede Komponente dieser Prozedur so lange spezifiziert, bis die unterste Ebene des Diagramms erreicht ist.

Das zweite Prinzip besteht darin, daß jede neue Prozedur, sobald sie definiert ist, auf Fehler getestet wird, die dann direkt beseitigt werden. Das geschieht durch Aufrufen der „Mutter-Prozedur", die das Ergebnis der Prozedur auf dem Bildschirm anzeigt, so daß eine sofortige Evaluation möglich wird; weiterhin zeigt die Fehlermeldung („Es gibt keine Prozedur mit dem Namen . . .") an, welche Prozedur als nächste zu schreiben ist.

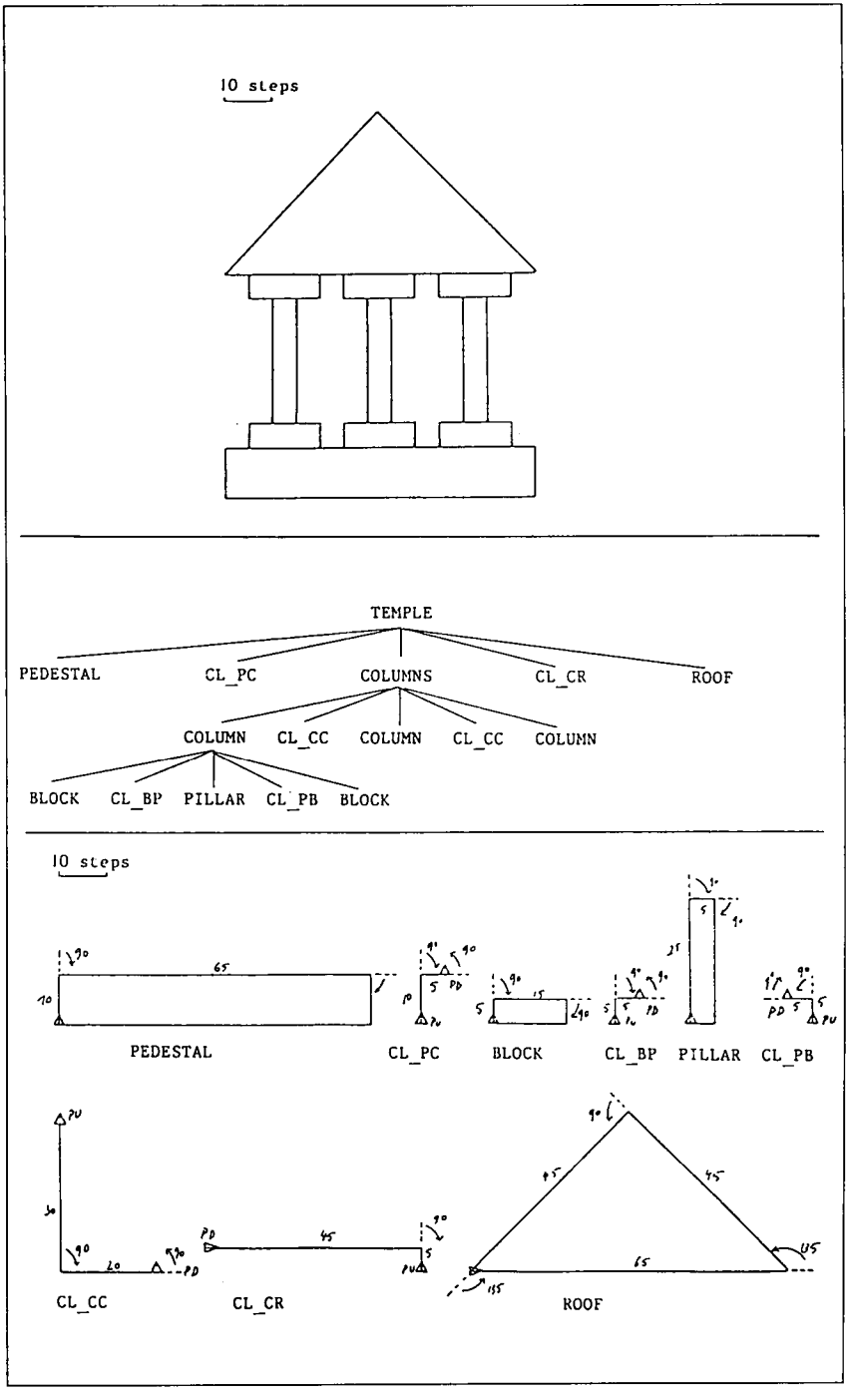

Abbildung 1:
Plan für das Schreiben eines Logo-Programms zum Zeichnen eines Schlosses

In der zweiten Experimentalgruppe wurde eine explizite Unterweisung zum Lerntransfer in Form eines zusätzlichen Mini-Kurses zur Lösung von Wort-Problemen realisiert; hierbei lernten die Kinder, die Prozeduren, die ihnen im Logo-Kontext beigebracht wurden (nämlich Planen, „debugging", Dekomposition und der Gebrauch externaler Repräsentationen), auf arithmethische Wort-Probleme anzuwenden. Auch die Kinder dieser Experimentalgruppe wurden mit einer Strategie vertraut gemacht, die zwei Hauptphasen umfaßt, nämlich eine Planungsphase und eine Ausführungs- und Testphase. Wie bei Logo besteht die Planungsphase in der Zergliederung eines komlexen Problems in Teilprobleme, die dann leicht gelöst werden können. Das Ergebnis dieser Aktivität kann auch in Art eines Baumdiagramms dargestellt werden (vgl. Abbildung 2). In der darauf folgenden integrierten Ausführungs- und Testphase wird dann jedes Teilproblem gelöst und unmittelbar getestet.

The carpet in our living room which measures 5 m by 4,40 m, has to be renewed. The price of the carpet is 350 fr per m^2. The craftsman charges 600 fr per hour and works 3 hours on the job. What will be the total cost for the renewal of the carpet?

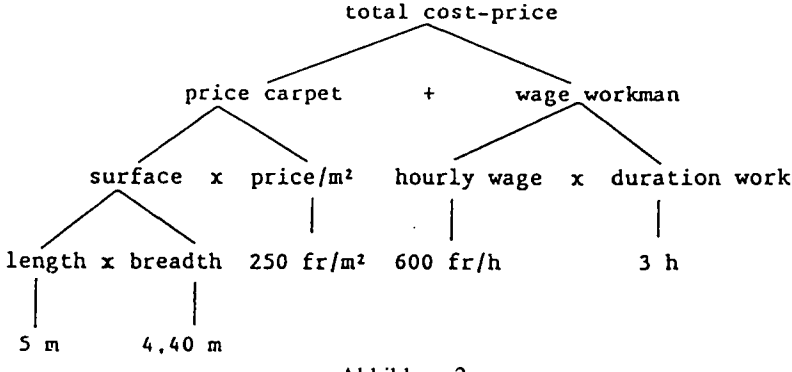

Abbildung 2:
Plan für das Lösen eines komplexen Wort-Problems

Aufgrund einer gemäßigt-konstruktivistischen Konzeption von Lernen und vor dem Hintergrund der Literatur zu Transferbedingungen im allgemeinen und zu Logo im besonderen vertreten wir die Auffassung, daß effektive Lernumgebungen einerseits durch eine gute Balance zwischen entdeckendem Lernen und persönlicher Exploration und andererseits durch eine systematische Unterweisung und Anleitung gekennzeichnet sind, wobei stets auch die individuellen Differenzen der Fähigkeiten, Erwartungen und Motivationen auf seiten der Schüler zu berücksichtigen sind. Die Grundidee für die Elaboration der Logo-Lernumgebung, die in beiden Experimentalklassen realisiert wurde, bestand darin, sowohl auf die Beherrschung der intendierten bereichsspezifischen Konzepte (Transferbedingung 1) als auch auf die

Denkleistungen abzuzielen, die bei Logo eine besondere Rolle spielen (Bedingung 2). Die Hauptaspekte unserer instruktionalen Intervention können wie folgt zusammengefaßt werden:
Bei der Lehrmethode *„Modellbildung"* (modeling) wurde die Programmierstrategie zunächst als Ganzes von einem Mitglied der Forschungsgruppe vorgestellt, wobei auch die verschiedenen Arbeitsschritte der Programmierung erklärt wurden, um den Schülern zu helfen, ein konzeptuelles Modell der Prozesse zu entwickeln, die notwendig sind, um eine konkrete Programmieraufgabe zu lösen. Sodann wurde jede Komponente der Strategie separat behandelt und geübt. Bei der Lehrmethode *"Scaffolding"* (im Sinne der Verabreichung von Hilfestellungen)[1] unterstützte der Lehrer die Schüler durch direkte Hilfen bei der Aufgabenbearbeitung. Eine Möglichkeit des „scaffolding" besteht zum Beispiel darin, den Schülern mit speziellen Hilfsprogrammen bei der Konstruktion des Baumdiagramms beizustehen. Schließlich wurde den Schülern reichlich Gelegenheit gegeben, die gesamte Programmierstrategie in Kleingruppen zu je drei Kindern zu üben, wobei sie zunehmend schwierigere Probleme zu lösen hatten. Anfangs wurden die Gruppen intensiv betreut, indem ihnen Hinweise, Rückmeldungen und Hilfen gegeben wurden. In dem Maße aber, wie die Leistungen im Programmieren besser wurden, wurden auch diese Hilfen graduell zurückgenommen.
Vor der Messung der Transfereffekte kontrollierten wir die Einhaltung der oben erwähnten Transferbedingungen. Ein Wissenstest zeigte für die Kinder beider Experimentalklassen, daß sie ein gutes Verständnis der Logo-Konzepte und -Prozeduren erworben hatten (Transferbedingung 1). Die Ergebnisse dreier Tests zur Programmierstrategie zeigten, daß auch die Transferbedingung 2 erzielt wurde. Die Mehrzahl der Kinder erreichten in beiden Experimentalklassen das festgelegte Leistungskriterium. In der Experimentalklasse, in der die Schüler ausdrücklich unterrichtet wurden, Denkprozesse auf einen anderen Bereich anzuwenden, konnte allerdings die dritte Transferbedingung nicht hinreichend gesichert werden. Demzufolge war eine abschließende Prüfung der Hypothese, daß die Erfüllung der oben genannten dritten Bedingung den Lerntransfer fördere, zumindest aufgrund dieser Studie nicht möglich.
Darüber hinaus waren in beiden Experimentalklassen und in der Kontrollklasse fünf Tests zu bearbeiten, mittels derer ein Transfer der vier Denkfähigkeiten gemessen werden sollte, die im Zentrum des Logo-Kurses standen, nämlich die metakognitiven Aspekte *„Planen"* und *„debugging"* (als Erkennen und Beseitigen von Fehlern) sowie die heuristischen Strategien *„Problemzerlegung"* und *„Konstruktion einer externalen Repräsentation"*. Die Datenanalyse erfolgte in Entsprechung zu dem zugrundeliegenden MANOVA-Design. Die Transferergebnisse für drei der vier Prozeduren, nämlich Fehlerbeseitigung, Problemzerlegung und Konstruktion einer externalen Repräsentation, zeigten Effekte des Arbeitens in einer Logo-Umgebung, insbesondere im Hinblick auf

den Transfer auf Situationen, die sich nicht allzu sehr vom ursprünglichen Lernkontext unterschieden. Wie aufgrund der unzureichenden Absicherung der dritten Transferbedingung nicht anders zu erwarten war, erzielten die Schüler der zweiten Experimentalklasse keine besseren Resultate als die der ersten.
In Übereinstimmung mit den Transferbedingungen, die dieser Untersuchung zugrundelagen, weisen diese Ergebnisse darauf hin, daß die Absicherung der beiden ersten Transferbedingungen (Kenntnis der Logo-Konzepte und -Prozeduren sowie Beherrschung der angestrebten Denkprozesse innerhalb der Logo-Umgebung) hinreichend ist, um einen Transfer zu erzielen.

Die Untersuchung von Clements und Mitarbeitern

Das Ziel dieser Studie (Clements, 1990) bestand darin, die Effekte einer theoretisch begründeten Logo-Umgebung auf die metakognitiven Ausführungsfähigkeiten *achtjähriger* Kinder zu untersuchen. Ausgehend von Sternbergs (1985) Komponenten-Theorie standen folgende Metakomponenten im Blickpunkt: Entscheiden in bezug auf die Natur des Problems, Auswählen und Kombinieren von Verarbeitungskomponenten, selektieren einer geeigneten Repräsentation und Kontrollieren der Lösungsprozesse.
48 Kinder wurden nach dem Zufallsprinzip einer Logo-Experimental- oder einer Kontrollgruppe zugeteilt. Ein Vortest zeigte für beide Gruppen keine Unterschiede bei Leistungen, die für das Programmieren in Logo relevant sind. Im Verlauf der Logo-Sitzungen (insgesamt fast 58 Stunden) wurden die Schüler explizit auf bestimmte Fähigkeiten trainiert, die für das Lösen von Problemen bedeutsam sind. Bei jeder Sitzung arbeiteten sechs Paare von Kindern unter der Anleitung eines oder zweier Erwachsener zusammen. Die Kontrollgruppe arbeitete etwa 19 Stunden mit zwei Software-Paketen, nämlich mit „Milliken's Writing Workshop" (das ist ein integriertes Paket von Schreibprogrammen, einem Word-Prozessor und Editorprogrammen) und mit einem Programm zum Zeichnen.
Die einzelnen Logo-Sitzungen hatten gewöhnlich folgenden Aufbau. Die *Einführungsphase* umfaßte einen Rückblick und eine Diskussion der Arbeit vom Tag zuvor, Fragen und, einmal pro Woche, eine spezielle Einheit („programmers chair"), bei der ein Schülerpaar ein vollständiges Programm darbot, das dann diskutiert wurde. In der *zweiten Phase,* einer Großgruppen-Sitzung, präsentierte eine Lehrperson neue Informationen über Logo-Konzepte und -Prozeduren (z.B. Problemzerlegung, Prozeduralisierung) oder ein strukturiertes Problem. Die *dritte Phase* bestand aus Einzelarbeit der Schüler an Problemen, die vom Lehrer zugeteilt wurden, oder an selbstgewählten Projekten. Dabei konnten die Schüler das TEACH-Programm benutzen; das ist ein Hilfsprogramm, das

Lernenden erlaubt, eine Prozedur zu definieren und unmittelbar zu beobachten, wie sie ausgeführt wird (vgl. Clements, 1985).

Ein zentrales Element dieses Unterrichts in Logo war die Nutzung sog. *Homunculi;* das sind Cartoon-Figuren, die Anthropomorphismen metakognitiver Prozesse konstituieren sollen. Vier solche „Homunculi" wurden eingeführt: Der *„Problem Decider"*, der *„Representer"*, der *„Strategy Planner"* und der *„Debugger"* (vgl. dazu die ausführliche Beschreibung von Clements in diesem Heft).

Diese Homunculi sollten vier Lehrverfahren unterstützen: *Erklärung, „scaffolding", Modellierung und Reflexion.* Um den Kindern beizubringen, eine Vielzahl von Problemen zu lösen, benutzten die Lehrpersonen diese Charaktere, um die bestimmten Problemlöseprozesse zu beschreiben und zu erklären — zum Beispiel: „Diese Zeichnung scheint geeignet zu sein, das Problem zu beschreiben. Dein *Representer* hat eine ausgezeichnete Wahl getroffen." Wenn bei der Einzelarbeit Probleme auftraten, konnte der Lehrer die Homunculi benutzen, um die Kinder daran zu erinnern, daß bestimmte Denkprozesse für die Lösung eines Problems nützlich sein können — zum Beispiel: „Was würde der Strategie-Planer machen?" Wenn nötig, konnte der Lehrer auch die Nutzung der Homunculi beim Lösen tatsächlicher Probleme modellieren. Schließlich wurde insbesondere in der Anfangsphase einer Lektion, während der „programmer's chair"-Sitzung und in der dritten Phase das Verfahren der Reflexion angewandt, indem die Kinder gebeten wurden, unter Rekurs auf die Homunculi über ihre Strategien nachzudenken.

Außerdem wurden die Kinder in eine allgemeine Programmierstrategie aus dem Repertoire des „strategy planner" eingeführt, die folgende Schritte umfaßte:

(1) Fertige eine kreative Zeichnung an.

(2) Fertige eine Skizze für die Planung unter Nutzung eines Planungsbogens an:
   — bewege die „turtle" dorthin, wo die Prozedur beginnt,
   — setze das „turtle"-Ende an die Stelle und Überschrift, wo sie startete,
   — benenne jede Linie, Kurve oder Prozedur,
   — benutze ein Lineal und einen Winkelmesser, um Streckenabschnitte und Winkel zu messen,
   — zeichne die „Bewegungen" zwischen Teilen der Figur als gepunktete Linien,
   — benutze für jede neue Prozedur, die aufgeschrieben werden muß, einen neuen Planungsbogen.

(3) Kommuniziere mit einem Partner, der die Anweisung liest und sie auf die rechte Seite des Planungsbogens einträgt.

(4) Gib die Anweisungen in den Computer ein.

(5) Beseitige zunächst für jede einzelne Prozedur die Fehler, dann die des gesamten Programms.

Zusätzlich zu der formalen anthropomorphen Unterweisung in metakognitive Funktionen wurden Transferleistungen explizit verfolgt. Dazu wurden Probleme in verschiedenen Kontexten gelöst, und es wurde die Effektivität der Anwendung der gelernten Metakomponenten herausgestellt. Es wurden auch Versuche unternommen, die Logo-Programmierung auf andere Problemsituationen des üblichen Schulunterrichts zu beziehen.

Um die metakognitiven Funktionen in Problemlösungssituationen zu messen, wurde ein dynamisches Testinstrument von Clements und Nastasi (1988) verwendet. Die Schüler bearbeiteten eine Serie von Problemen, deren korrekte Lösung von der relativ intensiven Nutzung einzelner Metakomponenten abhängig war. Waren sie nicht in der Lage, das Problem korrekt zu lösen, bot der Exerimentator eine Serie von Hilfen an, die zunehmend spezifischer wurden. Zwei verschiedene Maße wurden erhoben: Die *„Nutzung"* als Anzahl der Hilfen, die notwendig waren, bis die Kinder die Benutzung einer Metakomponente äußerten, und die *„Korrektheit"* als Anzahl von Hilfen, die notwendig waren, ehe die Kinder die korrekte Antwort erhielten.

Unter Zugrundelegung standardisierter Werte für die vier Metakomponenten wurde für beide Maße eine MANOVA gerechnet. Die Ergebnisse zeigten für beide abhängige Variablen eine signifikante Gesamtdifferenz zugunsten der Logo-Gruppe gegenüber der Kontrollgruppe. Für die Meßwerte der „Nutzung" erbrachte eine gestufte Diskriminanzanalyse signifikante Unterschiede für drei der vier Metakomponenten, nämlich für „Entscheiden über die Natur eines Problems", „Repräsentation" und „Kontrollieren der Lösung". Für die Meßwerte der „Korrektheit" erwiesen sich lediglich zwei der Metakomponenten als signifikant, nämlich „Kontrollieren der Lösung" und „Repräsentation". Das Ausbleiben von Transfereffekten für die Komponente „Auswählen/ Kombinieren" konnte auf das Fehlen von Abstraktion im entsprechenden Unterricht zurückgeführt werden; hier wurden spezifische Strategien wie das Anfertigen einer „Planskizze" stärker hervorgehoben als allgemeine Fertigkeiten des Planens.

Die Untersuchung von Swan

Swan (1989) befaßte sich mit folgenden Forschungsfragen: (1) Ist im Hinblick auf den Erwerb und Transfer von Problemlösestrategien eine explizite Unterweisung im Lösen von Logo-Problemen effektiver als entdeckendes Lernen? (2) Ist eine auf Problemlösen ausgerichtete Unterweisung in der Logo-Umgebung effektiver als eine vergleichbare Unterweisung mit konkreten Materialen? Vor dem Hintergrund dieser

Fragen befaßte sich die Untersuchung von Swan mit folgenden fünf allgemeinen Problemlösestrategien, die als nützlich für das Lösen von Programmierproblemen betrachtet werden:

— *Teilziel-Bildung* (Aufbrechen eines komplexen Problems in mehrere einfachere Probleme),

— *Vorwärts-Verkettung* (von dem, was in einer Problemstellung gegeben ist, zum Ziel hin),

— *systematischer Versuch-und-Irrtum* (rekursives Prüfen möglicher Lösungen in systematischer Weise),

— *alternative Repräsentation* (Konzeptualisieren eines Problems aus verschiedenen Perspektiven),

— *Analogie-Bildung* (Entdecken einer partiellen Ähnlichkeit zwischen zwei Dingen, die ansonsten mehr oder weniger unähnlich sind, und Abbildung von Wissen über einen Bereich auf einen anderen).

Eine ausführliche Aufgabenanalyse dieser Prozeduren diente als Grundlage für die Gestaltung der Unterweisung im Problemlösen sowie für die Auswahl von Transferaufgaben. An der Untersuchung nahmen 100 Schüler der vierten bis sechsten Klasse teil, sie alle hatten bereits mindestens ein Jahr (30 Stunden) Erfahrung im Programmieren in der Sprache Logo. Pro Klassenstufe wurden die Vpn nach dem Zufallsprinzip einem von drei Treatments zugewiesen: (1) Logo-Graphik, (2) Papierschneidetechnik, (3) entdeckendes Lernen anhand von Logo-Problemen. In den beiden ersten Treatments erhielten die Schüler dieselbe Basisinstruktion zum Problemlösen, praktizierten aber die gelernten Strategien in verschiedenen Umgebungen: einmal beim Erstellen von Logo-Programmen zum Zeichnen von Figuren, zum anderen beim Ausschneiden und Zusammenkleben von Figuren. In der dritten Gruppe bearbeiteten die Schüler Probleme der Logo-Programmierung, erhielten aber keine direkte Instruktion im Hinblick auf Problemlösungen. Alle Vpn arbeiteten zu zweit jeweils zweimal 45 Minuten pro Woche (insgesamt 15 Stunden) unter Aufsicht einer Lehrperson.

Die Unterweisung im Problemlösen wurde für beide Treatments in fünf Einheiten unterteilt, eine Einheit pro Strategie; in einer darauf folgenden praktischen Phase des Problemlösens konnte die Anwendbarkeit jeder Strategie ausprobiert werden. Um eine Strategie einzuführen und ihre einzelnen Schritte zu erklären, wurde ein Wanddiagramm auf der Grundlage einer Aufgabenanalyse benutzt. Abbildung 3 zeigt ein solches Diagramm für die Vorwärts-Verkettung.

Daraufhin zeigte der Lehrer, wie jede Strategie bei der Lösung eines Problems anzuwenden ist. Dann wurde das Diagramm an die Wand gehängt, und die Schüler lösten selbständig eine vorgegebene Menge von Problemen, die Lösungen erforderten, die eine der unterschiedenen

FORWARD CHAINING

1. What is the problem?
   What is the problem goal?
   What is given in the problem?
2. How can you change what is given so it is more like the goal?
   Try it.
3. Is it really closer to the goal?
   If not, redo step 2.
4. If it is, make your change the new problem given. Go to steps 2 and 3 and redo for the new given. When the given matches the goal, you are done.

Abbildung 3:
Mauer-Schaubild für das Vorwärts-Verketten (Swan, 1989b)

Strategien voraussetzen. Für jedes der zu lösenden Probleme hatten die Vpn einen Arbeitsbogen auszufüllen, der den Ausgangs- und Zielzustand sowie die Schritte umfaßte, die zur Lösung notwendig waren. In Abbildung 4 sind zwei Arbeitsbogen (für die beiden ersten Treatments) wiedergegeben, die sich auf die Bildung von Teilzielen beziehen.

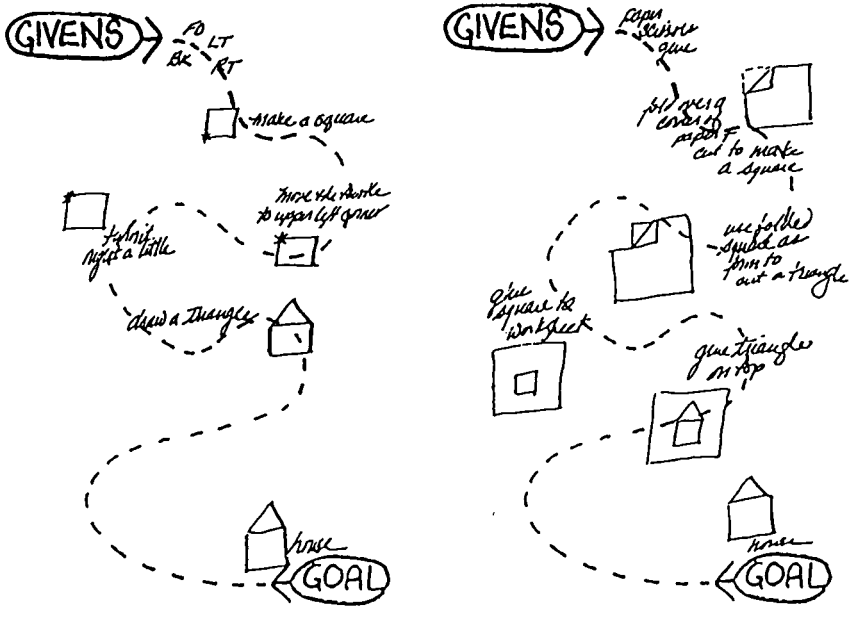

Abbildung 4:
Arbeitsblätter aus (a) der Logo-Graphikbedingung und (b) der „Papierschneide-Bedingung", die sich beide auf die Bildung von Teilzielen beziehen (Swan, 1989b)

In der dritten Versuchsbedingung erfuhren die Vpn keine direkte Instruktion in bezug auf das Problemlösen. Sie bearbeiteten Projekte der Logo-Programmierung, die sie aus Listen ausgewählt hatten und die vier Kernkonzepte des Programmierens (Prozeduren, Variablen, Bedingungen und Rekursion) betrafen.
Die Test-Batterie, die am Anfang und am Ende des gesamten Kurses zu bearbeiten war, umfaßte fünf Probleme aus anderen Kontexten. Jede der unterschiedenen Strategien wurde durch eine Problemstellung abgedeckt. Unterschiede zwischen Vor- und Nachtestwerten wurden unter Anwendung einer Varianzanalyse für Meßwiederholungen geprüft. Für alle Strategien zeigten die Ergebnisse der verschiedenen Analysen in bezug auf Wissenserwerb und Transfer, daß eine explizite Unterweisung im Lösen von Logo-Problemen mit anschließenden praktischen Übungen am effektivsten war. Sie übertraf nicht nur eine ähnlich angelegte Instruktion, die eine Papierschneidetechnik benutzte, sondern auch das entdeckende Lernen bei der Programmierung in Logo. Hinsichtlich der Wirksamkeit alternativer Repräsentationen waren die Untersuchungsbefunde allerdings nicht so eindeutig.
Insgesamt zeigt die Untersuchung von Swan, daß eine angemessene Unterweisung im Problemlösen eine größere Bedeutung für Transferleistungen haben kann als Logo-Umgebungen. Gleichwohl scheinen beide Faktoren notwendig zu sein, denn Schüler in der Gruppe, die eine direkte Instruktion außerhalb einer Logo-Umgebung erhielten (d.i. Versuchsbedingung 2) erzielten keine besseren Leistungen als Kinder, die Logo-Probleme aufgrund entdeckenden Lernens lösen sollten.

4. Diskussion und Zusammenfassung

In jeder der drei beschriebenen Untersuchungen wurden positive Transfereffekte gefunden. Wie bereits erwähnt, geht dies zu einem gewissen Teil darauf zurück, daß in diesen Studien versucht wurde, einige der Unzulänglichkeiten der früheren Forschung zu überwinden. Wir wollen nun etwas systematischer auf die eingangs formulierten Annahmen zurückgreifen und untersuchen, wie die verschiedenen Forscher auf sie eingegangen sind.

Effektive Lehr-Lern-Umgebungen zum Beherrschen von Logo

Ein positives Merkmal der drei Studien ist, daß das Arbeiten mit Logo in einem Zusammenhang mit „effektiven" Lernumgebungen gesehen wird; diese sind durch eine gute Balance zwischen entdeckendem Lernen und systematischer Unterweisung gekennzeichnet und setzen sich zum Ziel, zu einer „Meisterschaft" im Programmieren in Logo beizutragen. Eine wesentliche Komponente der jeweiligen Lehr-Lern-Umgebungen besteht in der direkten Unterweisung in Strategien des Problemlösens im

Kontext von Logo. Ein erster Schritt für die Gestaltung einer solchen Unterweisung umfaßt die Identifikation einer bestimmten Menge an grundlegenden Fähigkeiten, die zu trainieren sind. Einige Autoren wie Carver (1988) oder Swan (1989) gingen noch einen Schritt weiter: Sie identifizierten nicht nur solche Grundfähigkeiten, sondern entwarfen — unter Rückgriff auf Konzepte und Techniken der gegenwärtigen Kognitionspsychologie — detailliertere Modelle von Prozeduren des Problemlösens.

Obwohl die Lehrstrategien, die in den einzelnen Untersuchungen für die systematische Unterweisung in problemlösungsrelevanten Denkprozessen benutzt wurden, differierten, gibt es auf einer allgemeineren Ebene eine Menge von Gemeinsamkeiten. So können die meisten Untersuchungsansätze aus einer Kombination von sechs effektiven Lehrstrategien beschrieben werden, die Collins, Brown und Newman (1989) in bezug auf Problemlösen unterschieden haben (vgl. auch De Corte, 1990):

— *„Modeling"* beinhaltet, daß der Lernende einen Experten dabei beobachtet, wie er eine bestimmte Aufgabe löst. Das erlaubt dem Lernenden, ein geeignetes mentales Modell in bezug auf die Aktivitäten zu konstruieren, die für bestimmte Leistungen erforderlich sind.
— *„Coaching"* bezieht sich darauf, daß der Lehrer aufgrund seiner Beobachtungen dem Lernenden während der Ausführung einer Aufgabe Hilfen und Rückmeldungen gibt, um seine Leistungen zu verbessern.
— *„Scaffolding"* besteht darin, dem Lernenden bei der Bearbeitung einer Aufgabe direkte Strukturierungshilfen zu geben, um zum Aufbau eines problemlösungsrelevanten Handlungsgerüstes beizutragen; dieses Verfahren ist eine Variante des Konzepts der Zone proximaler Entwicklung von Vygotski.
— *„Articulation"* bezieht sich auf eine Technik, den Lernenden zu helfen, ihr Wissen und ihre Problemlöseprozedur zu explizieren und auszudrücken.
— *„Reflection"* soll Lernende anregen, ihre eigenen kognitiven Strategien und Lösungsprozesse mit denen von Experten und anderen Schülern zu vergleichen, und zieht schließlich ein mentales Modell der Expertenleistung nach sich.
— *„Exploration"* zielt darauf ab, die Autonomie des Lernenden in der Fähigkeit des Problemlösens ebenso zu steigern wie die des Entdeckens, Identifizierens und Definierens neuer Probleme.

Ein interessantes Merkmal einiger Untersuchungen ist in den Bemühungen zu sehen, die ursprüngliche Logo-Umgebung so anzupassen oder anzureichern, daß ein Teil der Instruktion mit Bezug auf Problemlösen durch den Computer besorgt wird. Ein gutes Beispiel dafür bietet die Studie von Clements (1990), der eine sehr wirksame Hilfe für die Fehlerbeseitigung bereitstellte, indem er dem Kind erlaubte, eine Prozedur zu definieren und zu beobachten, wie sie ausgeführt wird. Auch

unsere eigene Untersuchung kann als Beispiel herangezogen werden: Wir boten den Kindern eine Logo-Umgebung an, die den Prozeß der Dekomposition eines Problems mit Hilfe gut gewählter Fragen einleitete (De Corte et al., i.Dr.). Das führt uns zu der für die zukünftige Forschung relevanten Frage, welcher Teil der Unterweisung im Problemlösen besser in der Hand des Lehrers bleibt, auch wenn sie — zumindest partiell — in einer Logo-Umgebung erfolgen kann. Anders ausgedrückt: Sollen wir intelligente Computersysteme entwickeln, die detaillierte Analysen der Problemlösefähigkeiten und -strategien von Schülern durchführen können und auf dieser Grundlage dann eine angepaßte Betreuung und Instruktion erlauben? Oder sollen die letztgenannten Aktivitäten eher vom Lehrer übernommen werden, während die Hauptrolle des Computers dann darin bestehen sollte, eine Umgebung zu schaffen, die Lernende anregt, ihr Wissen und ihre Denkfähigkeit praktisch anzuwenden (Scardamalia, Bereiter, McLean, Swallow & Woodruff, 1989)?
Ein weiteres positives Merkmal einiger Studien (z.B. Carver, 1988; De Corte et al., i.Dr.; Littlefield et al., 1988) ist darin zu sehen, daß die Forscher untersuchten, ob die Lernenden am Ende der pädagogischen Intervention tatsächlich das ihnen vermittelte Wissen und die trainierten Denkfähigkeiten beherrschten. Versäumt man nämlich, Lernprozesse vor dem Messen von Transfer zu evaluieren, so ist das Ausbleiben von Transfereffekten nur schwer unter Bezugnahme auf eine bestimmte Denkfähigkeit zu interpretieren (vgl. auch Littlefield et al., 1989).

Explizites Lehren des Transfers eingeübter Fertigkeiten

Ein wesentliches Merkmal der beschriebenen Untersuchungen ist, daß zielgerichtet und explizit auf Transferleistungen hingearbeitet wurde. In der Terminologie von Salomon und Perkins (1987) sind dabei zwei Aspekte zu unterscheiden: *Bedeutungsvolle Abstraktion der Denkfähigkeiten,* die übertragen werden sollen, und *Dekontextualisierung* dieser Fähigkeiten, indem ihre Nützlichkeit in anderen Situationen gezeigt wird (Salomon & Perkins, 1987). Diese Unterscheidung findet sich auch bei Littlefield (1988) wieder, indem zwischen „framing" und „bridging" differenziert wird. *„Framing"* meint den Vorgang, eine spezifische Menge von Verhaltensweisen auf einen breiteren Rahmen des Problemlösens zu beziehen, und *„bridging"* beinhaltet, Prozesse, die in einem Kontext ablaufen, auf ähnliche Prozesse zu beziehen, die in anderen Kontexten ablaufen (Littlefield et al., 1988; vgl. auch Delclos & Burns, 1989).
Im Hinblick auf diese beiden Komponenten von transferwirksamer Instruktion unterscheiden sich die bisher angesprochenen Untersuchungen. In den meisten wurde lediglich der erste Aspekt berücksichtigt, das heißt, die im Logo-Kontext gelernten Fertigkeiten wurden auf eine allgemeinere, abstrakte Ebene übertragen. Drei Studien (Clements, 1990;

De Corte et al., i.Dr.; Littlefield et al., 1988) versuchten auch, beide Komponenten, Abstraktion und Dekontextualisierung, zu berücksichtigen. So wurde beispielsweise in unserer eigenen Untersuchung eine bedeutungsvolle Abstraktion (oder „framing") durch ein ausdrückliches Benennen der Denkfähigkeiten realisiert, die in der Programmierstrategie eingebettet waren. Dekontextualisierung (oder „bridging") wurde insofern verfolgt, als die Kinder gelehrt wurden, wie die erworbenen Fähigkeiten beim Lösen komplexer Wort-Probleme anzuwenden sind. In Anbetracht der vorliegenden Forschungsbefunde scheint es vernünftig zu sein, von solchen Studien, in denen beide transferwirksame Elemente beim Unterrichten berücksichtigt werden, einen besonderen Erkenntnisfortschritt zu erwarten.

Konstruktion und/oder Selektion adäquater Transferaufgaben

Ein positiver Trend in dieser Hinsicht ist, daß zunehmend mehr Forscher Transferaufgaben entwickelten, deren Lösung genau die Fähigkeiten voraussetzten, die während der Unterrichtslektionen gelehrt wurden. Darüber hinaus benutzten sie nicht nur produktorientierte Papier-und-Bleistift-Tests, sondern auch eher prozeßorientierte Messungen. In unserer eigenen Untersuchung zum Beispiel ließen wir Kinder laut denken, während sie eine Aufgabe des „debugging" (Fehlerbeseitigung) bearbeiteten (vgl. auch Carver, 1988); das versetzte uns in die Lage, die jeweils zur Anwendung gelangenden Prozesse des „debugging" nachzuzeichnen. Auch andere Autoren (Lehrer et al., 1987; Littlefield et al., 1988) benutzten dynamische Erhebungstechniken. Aus unserer Sicht sollte dieser Trend zu prozeßorientierten Messungen fortgesetzt werden.
Eine andere Tendenz, die wir unterstützen sollten, ist eine Charakterisierung von Transferleistungen entsprechend der Unterscheidung zwischen „nahem" und „fernem" Transfer. Im Gegensatz zu früheren Untersuchungen, in denen eine Serie allgemeiner Problemlöseaufgaben ausgewählt wurde, um „fernen" Transfer zu messen, sind Forscher mittlerweile weniger ambitioniert; nun konstruieren sie zunehmend Aufgaben, die (ganz im Sinne eines „nahen" Transfers) mehr Ähnlichkeit mit der Logo-Lernumgebung aufweisen. Ein Beispiel aus unserer eigenen Forschung ist der Notizblock-Test, der speziell entwickelt wurde, um einen „nahen" Transfer der Fähigkeit zu erfassen, ein Problem in Teilprobleme zu zerlegen.
Die soeben skizzierte Forschung läuft u.E. auf die Schlußfolgerung hinaus, daß Logo per se zwar kein (universelles) Medium für das Denken darstellt, aber bei einer adäquaten Einbettung in eine effektive Lehr-Lern-Umgebung durchaus ein effektives Hilfsmittel für den Erwerb und Transfer von Denkfähigkeiten sein kann. Dies hat Carver (1988) wie folgt umschrieben: „We must use the programming medium as a scaffold for instruction rather than a substitute for it" (S. 261).

Das aber wirft eine unter Umständen folgenschwere Frage auf: Wenn wir die in verschiedenen Untersuchungen beobachteten kognitiven Effekte nicht der Tätigkeit des Programmierens zusprechen, sondern eher der gesamten Lernumgebung, in die sie eingebettet ist, haben wir dann noch einen vernünftigen Grund, die Programmierung eines Computers als einen herausgehobenen Bereich für das Training von Denkfähigkeit zu betrachten? Können wir dann nicht auch annehmen, daß ein vergleichbarer Aufwand an Zeit, Anstrengungen und Expertise gleichermaßen effektive Lernumgebungen auch im „traditionellen" Unterricht erzeugen kann?

In diesem Zusammenhang argumentiert beispielsweise Clements (1990) eher intuitiv, daß Logo teilweise der impliziten Auslösung von Denkfähigkeiten genüge. Die Studie von Swan (1989), die zum Teil auf die Beantwortung dieser Fragen abzielte, stützt die Hypothese, daß Programmieren wenigstens etwas zu dem Transferprozeß auf konkrete Materialien beiträgt. Sie betrachtet dieses Ergebnis als eine Bestätigung von Papert (1980), der die Auffassung vertrat, daß Programmieren in der Tat einheitliche Qualitäten offenbare, weil es das unterstütze, was er als „transitional objects to think with" bezeichnet. Anders ausgedrückt: Computerrepräsentationen abstrakter Vorstellungen, die aber in ziemlich konkreter Art manipuliert werden können, vermögen konkretes und formales Denken miteinander zu verknüpfen. Die Ergebnisse der Untersuchung von Lehrer und Randle (1987), in der die Effekte einer Unterweisung für das Programmieren in Logo mit einer interaktiven Software-Bedingung verglichen wurden, weisen ebenfalls in diese Richtung.

Aus unserer Sicht sollten zukünftige Forschungs- und Entwicklungsarbeiten nicht auf Lernumgebungen beschränkt bleiben, in denen das Programmieren im Zentrum steht. Das Lehren des Denkens sollte integraler Bestandteil des gesamten Schulcurriculums sein und nicht nur in isolierten Logo-Lektionen betont werden. Der Erwerb allgemein anwendbarer Problemlösefähigkeiten erfordert, daß Kinder angeleitet werden, sie in einer Vielzahl von Bereichen und Problemsituationen anzuwenden. Das kann nur realisiert werden durch (a) eine Instruktionsplanung über Inhaltsbereiche hinweg und (b) eine Konzentration auf die Herausbildung einer einheitlichen Strategie für denkorientierten Unterricht.

Literatur

BURTON, J.K. & MAGLIARO, S.G. (1986): *Computer programming and generalized problem-solving skills: A critical review of the literature* (Internal report). Blacksburgh, VA: Virginia Polytechnic Institute and State University.

CARVER, S.M. (1986): *Transfer of Logo debugging skill: Analysis, instruction and assessment* (Unpublished doctoral dissertation). Pittsburgh, PA. Department of Psychology, Carnegie-Mellon University.

CARVER, S.M. (1988): Learning and transfer of debugging skills: Applying task analysis to curriculum design and assessment. In R.E. Mayer (Ed.), *Teaching and learning computer programming. Multiple research perspectives* (pp. 259-297). Hillsdale, NJ: Erlbaum.

CARVER, S. & KLAHR, D. (1986): Assessing children's Logo debugging skills with a formal model. *Journal for Educational Psychology, 76,* 1051-1058.

CLEMENTS, D.H. (1985, April): *Effects of Logo programming on cognition, metacognitive skills and achievement.* Paper presented at the Annual Meeting of the American Educational Research Association, Chicago.

CLEMENTS, D.H. (1990): Metacomponential development in a Logo programming environment. *Journal of Educational Psychology, 82,* 141-149.

CLEMENTS, D.H. & MERRIMAN, S. (1988): Componential developments in Logo programming environments. In R.E. Mayer (Ed.), *Teaching and learning computer programming. Multiple research perspectives* (pp. 13-53). Hillsdale, NJ: Erlbaum.

CLEMENTS, D.H. & NASTASI, B.K. (1988): Social and cognitive interactions in educational computer environments. *American Educational Research Journal, 25,* 87-106.

COLLINS, A., BROWN, J.S. & NEWMAN, S.E. (1989): Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics. In L.B. Resnick (Ed.), *Knowing, learning, and instruction. Essays in honor of Robert Glaser* (pp. 453-494). Hillsdale, NJ: Erlbaum.

De CORTE, E. (1990): Learning with new information technologies in school: Perspectives from the psychology of learning and instruction. *Journal of Computer Assisted Learning, 6,* 69-87.

De CORTE, E., VERSCHAFFEL, L. & SCHROOTEN, H. (in press): Cognitive effects of learning to program in Logo: A one-year training study with sixth graders. In E. De Corte, M. Linn, H. Mandl & L. Verschaffel (Eds.), *Computer-based learning environments and problem solving* (NATO ASO Series. F: Computers and Systems Sciences). Berlin: Springer.

DELCLOS, V.R. & BURNS, M.S. (1989, March): *Mediational elements in computer programming instruction.* Paper presented at the Annual Meeting of the American Educational Research Association, San Francisco.

GICK, M.L. & HOLYOAK, K.J. (1983): Schema induction and analogical transfer. *Cognitive Psychology, 15,* 1-38.

GREENO. J.G. (1976): Cognitive objectives of instruction: Theory of knowledge for solving problems and answering questions. In D. Klahr (Ed.), *Cognition and instruction* (pp. 123-159). Hillsdale, NJ: Erlbaum.

LEHRER, R. & RANDLE, L. (1987): Problem solving, metacognition and composition: The effects of interactive software for first-grade children. *Journal of Educational Computing Research, 3* (4), 409-427.

LERON, U. (1985): Logo today: Vision and reality. *The Computer Teacher,* 26-32.

LITTLEFIELD, J., DELCLOS, V.R., LEVER, S., CLAYTON, K.N., BRANSFORD, J.D. & FRANKS, J.J. (1988): Learning Logo: Method of teaching, transfer of general skills, and attitudes toward school and computers. In R.E. Mayer (Ed.), *Teaching and learning computer programming. Multiple research perspectives* (pp. 111-135). Hillsdale, NJ: Erlbaum.

LITTLEFIELD, J., DELCLOS, V.R., BRANSFORD, J.D., CLAYTON, K.N. & FRANKS, J.J. (1989): Some prerequisites for teaching thinking: Methodological issues in the study of Logo programming. *Cognition and Instruction, 6* (4), 331-366.

MILOJKOVIC, J.D. (1983): *Children learning computer programming: Cognitive and motivational consequences.* Unpublished doctoral dissertation, Stanford University, Department of Psychology.

PAPERT, S. (1980): *Mindstorms. children, computers, and powerful ideas.* New York: Basic Books.

PAPERT, S. (1987): Information technology and education. Computer criticism vs. technocentric thinking. *Educational Researcher, 16* (1), 22-30.

PEA, R.D. & KURLAND, D.M. (1983): *On the cognitive effects of learning computer programming* (Technical Report No. 9). New York: Bank Street College of Education, Center for Children and Technology.

PRESSLEY, M., SNYDER, B.L. & CARIGLIA-BULL, T. (1987): How can good strategy use be taught to children? Evaluation of six alternative approaches. In S.M. Cormier & J.D. Hagman (Eds.), *Transfer of learning. Contemporary research and applications* (Educational Technology Series, pp. 81-120). San Diego: Academic Press.

RESNICK, L.B. (1987): Learning in school and out. *Educational Researcher, 16* (9), 13-20.

SALOMON, G. & PERKINS, D.N. (1987): Transfer of cognitive skills from programming: When and how? *Journal of Educational Computing Research, 3,* 149-169.

SCARDAMALIA, M., BEREITER, C., McLEAN, R.S., SWALLOW, J. & WOODRUFF, E. (1989): Computer-supported intentional learning environments, *Journal of Educational Computing Research, 5,* 51-68.

SLEEMAN, D. (1985): *AI and education. Two ideological positions.* Unpublished manuscript. Stanford University, School of Education.

STERNBERG, R.J. (1985): *Beyond IQ: A triarchic theory of human intelligence.* Cambridge, MA: Cambridge University Press.

SWAN, K. (1989): Logo programming and the teaching and learning of problem solving. *Journal of Artificial Intelligence in Education, 1* (1), 73-92.

SWAN, K. & BLACK, J.B. (1987): *The cross-contextual transfer of problem-solving skills* (CCT report 87-3). New York: Columbia University, Teachers College, Department of Communication, Computing and Technology.

Anschrift der Autoren:
Erik De Corte, Lieven Verschaffel und Hilde Schrooten, Center for Instructional Psychology and Technology, Department of Education. University of Leuven, Versaliusstraat 2, B-3000 Leuven, Belgium.