ED 318 764 TM 014 817

AUTHOR van Merrienboer, Jeroen J. G.
TITLE What Cognitive Science May Learn from Instructional Design: A Case Study in Introductory Computer Programming.
PUB DATE Apr 90
NOTE 12p.; Paper presented at the Annual Meeting of the American Educational Research Association (Boston, MA, April 16-20, 1990).
PUB TYPE Viewpoints (120) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Case Studies; *Cognitive Measurement; Educational Strategies; *Instructional Design; Introductory Courses; *Programing; *Research Methodology; Theory Practice Relationship
IDENTIFIERS *Cognitive Sciences

ABSTRACT

The contributions of instructional design to cognitive science are discussed. It is argued that both sciences have their own object of study, but share a common interest in human cognition and performance as part of instructional systems. From a case study based on experience in teaching introductory computer programming, it is concluded that both sciences may reciprocally influence each other. Cognitive science often conducts research on cognitive processes in "silent" instructional strategies, so that guidelines for improving instructional systems are limited to those strategies. Instructional design conducts research on "spoken" instructional strategies, and observations of students working according to those strategies may provide evidence for the importance of particular cognitive processes that are neglected in current cognitive theories. However, these implications are also limited because they cannot lead to detailed descriptions of those cognitive processes. With regard to the tangent plane between cognitive science and instructional design (such as the field of intelligent tutoring systems), it is argued that these sciences must work together to reach their common goals. The ACT theory of skill acquisition is included in the discussion. (Author/SLD)

Paper Presented at the 1990 AERA Annual Meeting, April 16-20, Boston USA

What Cognitive Science may Learn from Instructional Design: A Case Study in Introductory Computer Programming

Jeroen J. G. van Merriënboer

University of Twente
Dept. of Education, Div. of Instructional Technology
P.O. Box 217, 7500 AE Enschede, The Netherlands

Abstract

This paper concerns the contributions of instructional design to cognitive science. It is argued that both sciences have their own object of study, but a common interest in human cognition and performance as part of instructional systems. From a case study in introductory computer programming, it is concluded that both sciences may reciprocally influence each other, although their mutual contributions have limitations. Cognitive science often conducts research on cognitive processes in "silent" instructional strategies, so that the guidelines to improve instructional systems are limited to those strategies. Instructional design conducts research on "spoken" instructional strategies, and observations of students working according to those strategies may provide evidence for the importance of particular cognitive processes which are neglected in current cognitive theories. However, these implications are also limited because they cannot lead to detailed descriptions of those cognitive processes. With regard to the tangent plane between cognitive science and instructional design, such as, for example, the field of Intelligent Tutoring Systems, it is finally argued that both sciences must contribute to each other to reach their common goals.

1. Introduction

This point-of-view paper primarily pertains to the question what cognitive science may learn from instructional design. Whereas most researchers agree that research in cognitive science contributes to instructional design, the opposite question is rarely asked and, moreover, when it is asked it is not unequivocally answered. An obvious ground for this lack of agreement is the fact that no clear evidence for contributions from instructional design to cognitive science exists. However, the absence of evidence does not necessarily indicate the impossibility of such contributions. For instance, it may also point to a deficiency of high-quality research in the field of instructional design (which is perhaps too practical with insufficient attention for research) or to an inadequate schooling of instructional designers who have incomplete knowledge of cognitive science (and may thus use erroneous cognitive theories). Whereas such observations may certainly be to-the-point, they do not answer the question if contributions from instructional design to cognitive science are *fundamentally* possible, and if so, what the nature of those contributions should be and under which conditions they will occur. In the present paper, some preliminary answers to those questions will be provided.

As a first step, the objects of study of the two sciences and the points of contact between them should be identified. The underlying idea is that potential contributions from instructional design to cognitive science, and vice versa, will prominently manifest themselves at the tangent plane between the two sciences. The objects of study clearly differ (see also, Warries, 1987). The object of study of cognitive science may loosely be described as *cognitive processes* (e.g., learning) and related performance of humans. The collected information is organized in cognitive theories. The object of study of instructional design may be described as *instructional processes*[1] (e.g., the execution of instructional plans, such as particular instructional strategies and tactics, by a teacher and/or other delivery systems) and related performance of instructional systems. Here, the collected information is organized in instructional theories.

Whereas both sciences have their unique object of study, they nevertheless clearly touch each other because all instructional systems contain one or more learners. The object of study at this "tangent plane" is human cognition and performance *as component of* an instructional system. Here, instructional processes and cognitive learning processes are heavily interwoven. For this reason, the one-worded term "onderwijsleerprocessen" [instruction-learning processes] is often used in The Netherlands to depict the object of study of this overlapping area. In my opinion, the study of instruction-learning processes may equally justifiable be called applied cognitive science or fundamental research in instructional design (in fact, it is closely related to the RDD-approach in instructional design). But more importantly, both sciences meet at this tangent plane and if contributions of instructional design to cognitive science exist, and vice versa, they will first show up at this plane.

[1]The object of study of instructional design may be defined in a much broader sense (e.g., including aspects that directly concern the development of instructional systems); however, this minimum description is satisfactory for the present goal.

As a second step, the nature of the mutual contributions must be identified. With regard to contributions of cognitive science to instructional design, most researchers agree that studies in cognitive science (and in particular, studies on human cognition and performance in instructional systems) do not only lead to cognitive theories about human cognitive processes and performance (which is their primary goal), but also yield guidelines how to optimize the instructional system under study and may thus contribute to instructional design. Whereas those contributions will certainly not be questioned, an important claim of the present article is that the reach of those contributions is limited. In particular, it will be argued that they are limited to the strategy under study. Guidelines are only provided on a tactical level (i.e., they affect only *one* particular instructional strategy) and whereas they may lead to improvements of the strategy under study, they will never lead to the development of renewing instructional strategies.

With regard to contributions of instructional design to cognitive science, researchers show little agreement on the nature, or even the existence, of such contributions. Research on instructional design aims at the development of theories that explain the instructional processes in different instructional systems, and the theories should yield clear guidelines for the design of those systems. Often, instructional systems will be compared (e.g., in terms of learning outcomes) that pursue the same instructional goals but--in contrast to research in cognitive science--apply totally *different* instructional strategies. In the present article, it will be argued that such theories on instructional systems may indeed contribute to cognitive science: In particular, renewing instructional strategies may be identified which reach higher learning outcomes than traditional strategies, and these strategies may throw a new light on the cognitive processes involved in learning a particular task. For the field of introductory computer programming, the claims that (a) contributions of cognitive science to instructional design are limited to particular instructional strategies, and that (b) contributions of instructional design to cognitive science exist, at least as far as new strategies are identified that provide evidence for the importance of particular cognitive processes, will be further elaborated in the next sections.

2. A Case Study

The case study in teaching introductory computer programming will be presented in such a way that the differences between the approaches from cognitive science and instructional design are accentuated. The cognitive science approach is characterized by its emphasis on a description of cognitive processes. According to the presented viewpoint, research is usually conducted in instructional strategies that are not basically questioned. For this reason, they will be referred to as *silent strategies*. It will be argued that the guidelines for instructional design as offered by cognitive science are limited to those silent strategies. In contrast, the approach from instructional design is characterized by its emphasis on a description of instructional processes, including the explication of instructional strategies which will be referred to as *spoken strategies*.

2.1. Silent Strategies in Cognitive Science

A well-known example of research in introductory computer programming from a cognitive science viewpoint is offered by the work of John R. Anderson and his co-workers at Carnegie Mellon University. By this group, computer programming is seen as a complex cognitive skill and the process of learning the skill is described by the ACT* theory of skill acquisition (Anderson, 1982, 1983; Anderson, Farrell & Sauers, 1984). According to the theory, learning the skill proceeds through three phases.

In the first or declarative stage, the learner receives information about the skill by formal instruction (e.g., by reading books or listening to lectures). New facts are stored in declarative memory and to generate behavior on the basis of this newly acquired knowledge, students must use existing domain-independent procedures (or, productions) to interpret those facts. Although this interpretive use of knowledge has the advantage of flexibility, it also has serious costs in terms of time and errors because of the high working memory load involved. In the second phase, a process called *knowledge compilation* creates task-specific procedures through practice. Hence, during performance of the skill (i.e., in a process of "learning by doing") the declarative knowledge is gradually converted into a fundamentally different, procedural form in which it may directly control behavior. As a result, the skill is performed faster and with increasingly less errors because working memory load heavily decreases. In the third and final stage, a further strengthening of the procedures takes place to make them more selective in their range of applications.

Research on learning computer programming within this framework may have clear implications for instruction and instructional design. However, the research is typically applied in a *silent* instructional strategy, so that the instructional implications are limited in their range. For learning computer programming as well as most other problem solving skills, the silent instructional strategy may be labeled "learning problem solving by solving problems". According to this strategy, students are alternately offered formal instruction and practice. For teaching computer programming, the formal instruction usually includes a small amount of new programming language features along with their syntactical details, and a number of illustrative problems together with their solutions in the form of concrete computer programs to illustrate the use of the new material. Practice includes a relatively large number of (conventional) programming problems for which students have to generate new computer programs. Thus, during formal instruction students are presented with information that is relevant to the performance of the skill, and during practice students solve problems that are--apart from their difficulty--essentially identical to the problems they must learn to solve.

From the ACT* theory, guidelines for instruction may be deduced but these guidelines are, at least theoretically, limited to application within the "learning problem solving by solving problems" strategy that is silently chosen. Some important guidelines for instructional design that are offered by the ACT* theory of skill acquisition (Anderson, 1987a, 1987b) pertain to (a) the balance between formal instruction and practice, (b) the nature of the formal instruction, and (c) the immediacy and form of provided feedback on student errors during practice. With regard to the first guideline, it should be clear that one straightforward implication is that in a system in which one can only learn skills by doing them, the importance of formal instructional diminishes and the importance of practice increases.

The second guideline is directly related to the first. As a direct consequence of the balance between formal instruction and practice, there is reason to doubt the value of elaborate formal--either verbal or textual--instruction. According to the cognitive theory, one should only provide the information that is required for performing the skill and which should be based on a production system model of the skill; one should present small amounts of information at a time in order to prevent processing overload, and one should focus on telling the students *how* to solve problems instead of explaining why particular solutions work.

A third important guideline concerns the importance of immediate feedback on errors made during practice. This is a consequence of the interaction between knowledge compilation and working-memory limitations. Knowledge compilation requires that the information about a decision is preserved until information about the correctness of the decision is available; only then, a production can be compiled by attaching the correct action to the information attached to the decision. Working memory failures will cause the loss of information and thus impair compilation; obviously, these failures are more likely to occur if the delay in feedback increases (Anderson, 1987a; Anderson & Jeffries, 1985).

Together, and as argued by Dijkstra (1990), these guidelines for instruction suggest that skill development and knowledge construction should take place jointly. In the initial stages of learning, the instruction should not overwhelm learners with a large amount of information that seems all relevant to later performance of the skill. Instead, the necessary instruction should be compacted into its bare essentials and gradually elaborated during the skill acquisition process. Furthermore, the instruction should encourage consistent practice (i.e., solving conventional problems) and provide immediate feedback on errors during problem solving. This approach to teaching introductory computer programming is indeed present in the LISP-tutor (Anderson & Reiser, 1985; Anderson & Skwarecki, 1986) and several other ITS's that are build on the basis of the ACT* theory.

As a first conclusion, it should be obvious that cognitive science may contribute to instructional design; at least, it yields guidelines to improve a particular instructional strategy ("learning problem solving by solving problems"). But as a second, more important conclusion, it should be noticed that cognitive science is not able to offer a *renewing* instructional strategy because the silent strategy is never questioned. For instance, it is interesting to notice that ACT* does not provide any guidelines for the design of practice: Given the silent strategy, practice is almost by definition "solving the kind of problems that students must learn to solve". This certainly should not be seen as a point of critique towards cognitive science. It is exactly what cognitive science is expected to do because it studies human cognition and performance in familiar task situations, such as instructional systems. Cognitive theories are primarily developed to explain the cognitive processes involved in the task situation and as an extra benefit, the theories might yield guidelines to optimize the particular task situations. The development of alternative task situations, or, more in particular, new instructional strategies goes beyond the system of interest.

2.2. Spoken Strategies in Instructional Design

As discussed in the previous section, part of cognitive science is interested in the cognition and performance of the human component in instructional systems. Whereas this research is primarily aimed at the development of cognitive theories, it may also yield information for instructional theories as far as the (silent) instructional strategies are concerned in which the research is conducted. In contrast, research in instructional design is primarily aimed at the explication of instructional processes, leading to instructional theories that organize the information about *spoken* strategies and related performance of instructional systems (effectiveness in reaching instructional goals, cost-effectiveness etc.). The question is now: May this research also contribute to the development of cognitive theories?

An example of research in introductory computer programming from an instructional design viewpoint is offered by my own work at the University of Twente. Computer programming is seen as a problem solving activity, and the instructional theory that is under development should explicate the instructional strategies and tactics that make up an effective instructional system for teaching introductory computer programming. Notice that this research does *not* primarily aim at a study of the learning processes involved in a silent strategy for teaching programming (such as the work of Anderson), but instead at an explication of *spoken* strategies and their effects.

Several instructional strategies have been identified and described (van Merriënboer & Krammer, 1987, 1990) on the basis of a study of instructional systems in the educational field. Information was gathered by studying actual courses, textbooks, articles written by practitioners and teachers, etcetera. The three main groups of instructional strategies identified were labeled the spiral strategy, the expert strategy and the completion strategy. Both the spiral and the expert strategy emphasize that students should immediately start off with designing and/or coding programs, that is, program generation. The *spiral strategy* is the most common approach to teaching introductory computer programming. An important feature of this strategy is its emphasis on incremental learning: Each step contains both syntactic and semantic elements, presents a minimal extension of previous (declarative) knowledge, is explained in relation to already acquired knowledge, and is extensively trained in exercises. The exercises are conventional programming problems with increasing difficulty during the course. In fact, the spiral strategy may best be seen as the silent strategy in which most research on computer programming in cognitive science is conducted.

The *expert strategy* is heavily inspired by the discipline of structured programming and it is based on a model of expert behavior. An important feature of this strategy is its emphasis on both algorithm and program design in a systematic top-down fashion. In formal instruction, the focus lies on the presentation of a top-down design model that should enable the learners to concentrate on the semantic content of the algorithm because less attention is required to plan and execute actions on lower program code levels. With regard to practice, students receive problems for which algorithms have to be developed from the outset of the course. Often, intermediate products such as flow-charts, structured diagrams or pseudo-programming languages are used to represent the algorithms.

Finally, the *completion strategy* is the only strategy that does not primarily emphasize the design or coding of new programs. Instead, it emphasizes the reading, extension and modification of existing, well-designed programs. Students are confronted with non-trivial design problems from the beginning of the course, but these problems are always presented in combination with their complete or partial solutions in the form of well-designed and well-documented (partial) computer programs. The student' tasks gradually become more complex during the course, changing from using, reading and tracing programs, through modifying and completing increasingly larger parts of incomplete programs, to independently designing and coding new programs or subprograms.

After their identification and explication, the strategies were evaluated by carefully comparing the quality of their applied instructional processes. This was achieved both by theoretical studies and experimental research. In theoretical studies (e.g., Van Merriënboer

& Krammer, 1987, 1990), the measure to which known instructional principles or tactics are well-applied (or can be applied) in the particular strategies was evaluated to formulate hypotheses about their effectiveness. These instructional principles may be derived both from existing instructional theories and particular cognitive learning theories such as, for instance, ACT*. In several studies, the completion strategy was predicted to be superior to other strategies because it supports most instructional principles that are known from the literature. Those include the ACT*-principles such as, for instance, the emphasis on learning by doing (students immediately start completing programs and training the basic skills involved in programming) and a reduction of processing load (which may be expected to be lower for completing programs than generating programs). In addition, several other principles that are known from other cognitive theories and, last but not least, instructional theories are applied such as, for instance, the availability of useful worked examples during practice (the incomplete programs may serve as such examples) and the provocation of mindful abstraction from concrete examples during practice (the incomplete programs have to be carefully studied before they can be correctly completed).

The predictions concerning the superiority of the completion strategy over other strategies were supported in several experiments (van Merriënboer, 1988, in press a, in press b, van Merriënboer & de Croock, 1989). The completion strategy was found to be more effective than other strategies for a range of learning outcomes, including the generation of new computer programs. The data indicated that students used parts of the programs that they had to complete or modify as blue-prints to map their partial solutions and they generalized from them, by a process which may be called *mindful abstraction* (Salomon & Perkins, 1987), to learn new programming principles, design techniques, and programming language templates. In addition to the automation of skills (as specificized by the mechanism of knowledge compilation in ACT*), the acquisition of schematic, generalized declarative knowledge proved to be an essential process in learning elementary computer programming. This has direct implications for cognitive theories, and in particular theories of skill acquisition that describe the cognitive processes involved in learning complex cognitive tasks like computer programming. In particular, these theories seem to be incomplete with regard to the acquisition of structured declarative knowledge (or schemata); they focus on learning by doing and "rule automation" or knowledge compilation as the most essential processes in the acquisition of computer programming skill, whereas--in more effective instructional strategies--learning by mindful abstraction and "schema acquisition" proved to be at least equally important (for an elaborate discussion, see van Merriënboer & Paas, in press).

Concluding, instructional design may contribute to cognitive science, because the study of new instructional strategies may provide evidence for the importance of particular cognitive processes that are neglected or underestimated by theories in cognitive science. Eventually, this may lead to the revision or extension of those cognitive theories. On the other hand, the contributions of instructional design are limited. Just as cognitive science will probably not yield renewing instructional strategies, instructional design will not lead to the "discovery" of new cognitive processes or a highly detailed description of those processes. This is subject to the nature of the field, which main goal is not a to describe cognitive processes but to prescribe effective instructional strategies. However, the claim stands that studies in instructional design may provide evidence for the importance of particular cognitive processes, and thus lead to the integration of different perspectives in cognitive science. Such an integration of cognitive viewpoints is necessary to explain human cognition and performance in newly-developed instructional systems.

3. Discussion

This paper primarily concerned the contributions of instructional design to cognitive science. It was argued that both sciences have their own object of study. Instructional design studies instructional processes that occur in instructional systems and cognitive science studies cognitive processes that occur in the human mind. At the tangent plane between the two sciences, there is however a common interest in instruction-learning processes. From a case study in introductory computer programming, it was concluded that both sciences may reciprocally influence each other; however, there are limitations to those contributions. The cognitive processes as described by cognitive science may yield guidelines to improve instructional systems, but these guidelines are limited to the *silent* strategies in which the research has been conducted; the instructional processes as described by instructional design (i.e., *spoken* strategies) may provide evidence for the importance of particular learning processes that are neglected in cognitive theories, but these implications are also limited because they cannot directly lead to highly detailed descriptions of those learning processes.

The implications of the presented point-of-view may be illustrated by the developments in Intelligent Tutoring Systems (ITS; Wenger, 1987). ITS's may be considered to be on the tangent plane between cognitive science and instructional design, because they represent both cognitive processes of experts (expert model) and actual students (student model), and instructional processes such as instructional strategies and tactics (instructional model). Whereas cognitive science is primarily interested in expert and student models, instructional design is (or, should be) primarily interested in instructional models. As argued in the previous section, ITS's for teaching introductory computer programming which are build by cognitive scientists will often use a silent instructional strategy.

Indeed, an ITS like the LISP-tutor has an advanced expert model and student model, but the instructional processes "...have not been directly incorporated into the tutoring system as explicit tutorial strategies..." (Wenger, 1987, p. 292). In short, there is no explicit instructional model so that the term silent strategy may be taken literally.

With regard to ITS's, it is the job of instructional design to build instructional models, and thus explicate instructional processes. Obviously, the development of explicit instructional models might also affect other parts of ITS's. For instance, the completion strategy may prove to be important to the design of ITS's for teaching introductory computer programming. Up to the present, most ITS's apply a silent strategy, in which novice students have to write new computer programs. A major problem in those systems is the generation of informative feedback on semantical program errors, because of the combinatoric explosion in the tree of possible solutions for a given programming problem. The completion strategy might offer an effective basis to attack this problem, because the solution tree is heavily constrained if programs have to be completed instead of newly generated. Then, the generation of informative feedback on semantical program errors may be substantially simplified (see, van Merriënboer, van den Berg, & Maaswinkel, 1989). This leads us to the final conclusion. Cognitive science and instructional design have their own object of study, but they may reciprocally contribute to each other; furthermore, both sciences *must* contribute to each other on the tangent plane between both sciences, such as the field of ITS's, to obtain their common goals.

References

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89*, 369-406.

Anderson, J. R. (1983). *The Architecture of Cognition.* Cambridge: The Harvard University Press.

Anderson, J. R. (1987a). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*, 192-210.

Anderson, J. R. (1987b). Producti systems, learning, and tutoring. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production System Models of Learning and Development.* Cambridge, MA: The MIT press.

Anderson, J. R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science, 8*, 87-129.

Anderson, J. R., & Jeffries, R. (1985). Novice LISP-errors: Undetected losses of information from working memory. *Human-Computer Interaction, 1*, 107-131.

Anderson, J. R., & Reiser, B. J. (1985). The LISP-tutor. *Byte, 10*(4), 159-175.

Anderson, J. R., & Skwarecki, E. (1986). The automated tutoring of introductory computer programming. *Communications of the ACM, 29*, 842-849.

Dijkstra, S. (1990). The description of knowledge and skills for the purpose of instruction. In S. Dijkstra, B. H. M. van Hout-Wolters, & P. C. van der Sijde (Eds.), *Research on Instruction.* Englewood Cliffs, NJ: Educational Technology Publications.

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

Van Merriënboer, J. J. G. (1988). Relationship between cognitive learning style and achievement in an introductory computer programming course. *Journal of Research on Computing in Education, 21*, 181-186.

Van Merriënboer, J. J. G. (in press a). Strategies for programming instruction in high school: Program completion vs. program generation. *Journal of Educational Computing Research.*

Van Merriënboer, J. J. G. (in press b). Instructional strategies for teaching computer programming: Interactions with the cognitive style reflection-impulsivity. *Journal of Research on Computing in Education.*

Van Merriënboer, J. J. G., van den Berg, K. G., & Maaswinkel, D. M. (1989). Some experiences with two intelligent tutoring systems for teaching computer programming: PROUST and the LISP-tutor. In J. M. Pieters (Ed.), *Intelligent Tutorial Systems and Instruction.* Enschede: OTG Onderwijsleerprocessen.

Van Merriënboer, J. J. G., & de Croock, M. B. M. (1989, september). *Strategies for computer-based programming instruction: Program completion vs. program generation.* Paper presented on the Third European Conference for Research on Learning and Instruction (EARLI), Madrid, Spain.

Van Merriënboer, J. J. G., & Krammer, H. P. M. (1987). Instructional strategies and tactics for the design of introductory computer programming courses in high school. *Instructional Science, 16*, 251-285.

Van Merriënboer, J. J. G., & Krammer, H. P. M. (1990). The 'completion strategy' in programming instruction: Theoretical and empirical support. In S. Dijkstra, B. H. M. van Hout-Wolters, & P. C. van der Sijde (Eds.), *Research on Instruction.* Englewood Cliffs, NJ: Educational Technology Publications.

Van Merriënboer, J. J. G., & Paas, F. G. W. C. (in press). Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. *Computers in Human Behavior.*

Warries, E. (1987). The knowledge base for instructional design. *Instructional Science, 16*, 105-108.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems.* Los Altos, CA: Morgan Kaufmann Publishers.