November 2008

# The Disruptive Potential of Immediate Feedback

Lael J. Schooler
*Carnegie Mellon University*

John R. Anderson
*Carnegie Mellon University*

# The Disruptive Potential of Immediate Feedback

Lael J. Schooler
John R. Anderson

Carnegie-Mellon University

**Abstract**

Three experiments investigate the influence of feedback timing on skill acquisition in the context of learning LISP. In experiment 1 subjects receiving immediate feedback went through the training material in 40% less time than did those receiving delayed feedback, but learning was not impaired. A second experiment involved the use of an improved editor and less supportive testing conditions. Though subjects in the immediate condition went through the training problems 18% faster than did those in the delay condition, they were slower on the test problems and made twice as many errors. The results of experiment 3, a partial replication of the first two experiments, indicated a general advantage for delayed feedback in terms of errors, time on task, and the percentage of errors that subjects self-corrected. A protocol analysis suggests that immediate feedback competes for working memory resources, forcing out information necessary for operator compilation. In addition, more delayed feedback appears to foster the development of secondary skills such as error detection and self-correction, skills necessary for successful performance once feedback has been withdrawn (Schmidt, Young, Swinnen, & Shapiro, 1989).

## Introduction

The timing of feedback is an important issue in skill acquisition. Schmidt, Young, Swinnen, and Shapiro (1989) conducted a study on the timing of feedback with respect to motor skills. They found that shorter feedback latencies improved acquisition and performance while feedback was there, but that delayed feedback resulted in improved subsequent performance once feedback had been withdrawn. Schmidt et al (1989) explain these results in terms of the Guidance Hypothesis. In this view, during the initial stages of skill acquisition immediate feedback guides the behavior of the learner, leading to superior initial performance. This guidance, however, can lead to a dependence on the feedback by obscuring the need to learn secondary skills necessary to perform the task without feedback. The abilities to detect and self-correct errors exemplify such secondary skills.

The results of Lewis and Anderson (1985) are consistent with the predictions of the guidance hypothesis. Within the context of an adventure game, they assessed the effects of immediate and delayed feedback on subsequent performance. Subjects receiving immediate feedback were more likely to select appropriate operators, but those that received delayed feedback were better able to detect errors. Somewhat different results were obtained by Anderson, Conrad, and Corbett (1989). They assessed the effects of immediate and delayed feedback within the context of the GRAPES LISP Tutor. Subjects in the immediate condition moved through the material more quickly than did those in the delay condition, but there were no statistically significant differences in test performance.

This paper will further consider the effects of immediate versus delayed feedback in the context of learning LISP

## Experiment 1

This experiment investigates the effects that various levels of guidance have on complex skill acquisition. Guidance was indirectly manipulated by crossing feedback timing (immediate or delayed) and its focus (directive or nondirective) Directive feedback focuses on the function, or variable, that the student should have used, guiding the subject along a correct solution path. In contrast, nondirective feedback focuses on the function they used incorrectly, but offers no guidance. Immediate feedback provides more guidance than does delayed feedback, by preventing the subject from exploring incorrect solution paths. In this framework, immediate directive feedback provides the most guidance and delayed nondirective the least. These dimensions were integrated into the feedback given to subjects as they solved LISP problems.

## Method

Subjects Subjects were members of the Carnegie-Mellon University community with no previous experience with LISP

Design: The feedback dimensions of timing (immediate or delayed), and focus (directive or nondirective) were crossed, generating 4 cells, with 7 to 9 subjects in each cell. Subjects within a given condition received a single form of feedback throughout the tutoring session. Subjects were matched for math SAT's between the immediate and delayed conditions.

Procedure: Subjects first filled out a background questionnaire about their previous programming experience and their SAT's. Subjects then read materials that familiarized them with both the tutor and the basics of LISP evaluation. Next they solved 4 practice problems involving easily understood arithmetic functions, such as *plus* and *difference*.

Once they had finished the practice problems they were given a second pamphlet describing lists and symbols in LISP and definitions of functions related to their manipulation. After they finished studying the pamphlet, the experimenter started them on the main problems. The solutions to these involved the construction of LISP expressions composed of functions that combine and manipulate lists and symbols. When they made an error, subjects received feedback according to the specifications of their cell.

The general procedure for the second day was similar to the first. However, subjects did not receive any direct feedback from the tutor, though it continued to follow their solutions and record errors. From the subjects' point of view, the tutor acted largely as task master, editor and interpreter; presenting problems and evaluating completed LISP expressions. During both sessions, the subjects received the standard (cryptic) error messages from the LISP interpreter, along with the results of evaluation.

Materials: The subjects learned a modified subset of the LISP extractor and combiner functions (*car, cdr, last, reverse, cons, append & list*). The modifications included changes to the names of some functions such that they were of approximately equal mnemonic value (e.g, *car* to *head*). Other modifications involved restricting the number of arguments that the functions could take. Whereas in standard LISP certain combiner functions, such as *append* and *list*, take a variable number of arguments, our version required that all combiners take two.

The problems were blocked according to difficulty. Within each block, problems were presented randomly. Each of the first seven problems required the use of one of the seven functions and each function was used once within the block. The second block required the use of two functions. Each of the seven was used twice within the block; paired with a different function each time it was used. The third block of problems involved solutions requiring the use of between 3 and 7 functions. In these problems, some functions were used more often than others.

Two problem sets, composed of unique problems, were generated to these specifications. The relative frequency of any given function was comparable in each set.

## Tutor Design

Interface: The problem description is displayed in the top portion of the screen (see figure 1). It specifies the initial state, which amounts to the initial variable bindings, and the goal state, the result that the completed expression should return. The subject edits her solution in the space directly below the problem description. Functions and variables that can be used to form the solution are displayed continually. A student selects a solution component, a function or variable, by pressing the key corresponding to the first letter of its name. In figure 1, the student has already entered the function *list* into the editor by pressing the [l]-key. The cursor, here signified by bold brackets, indicates where the component selected next will be entered into the solution. The cursor can be moved to the left and right with the [-]-key and the [+]-key respectively. The editor automatically expands the *list* expression, prompting the student to fill in its first argument, *<expr-1>*. The student's solution continues in figure 2.

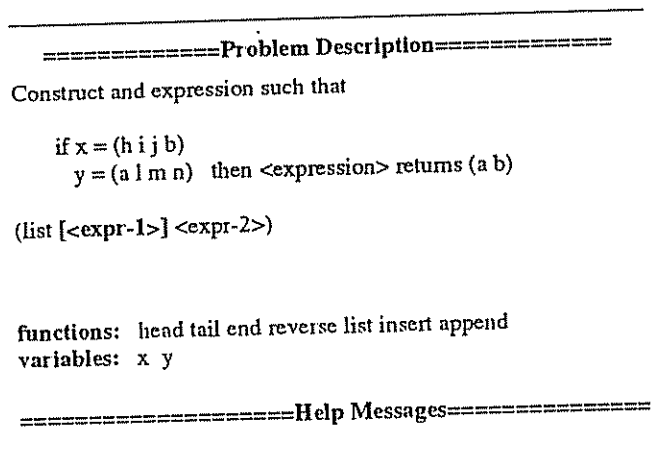


**Figure 1:** A screen from the solution to the lhlr problem.

Suppose the student wants to make a list whose first element is the first element of the variable *y*. She accomplishes this by first entering the function *head*, with the [h]-key (step 2). The system expands the *head* expression, and she enters the variable *y* (step 3). The system then prompts her to fill in the second argument of

the *list* expression. Her decision to use the function *tail* (step 4) results in an error.

At this point, since our subject is in the immediate feedback condition, the tutor automatically places the cursor at the point at which the error was detected, and provides feedback (figure 3). She next deletes the offending function (step 5), before successfully completing the problem (step 8). The system then sends the completed expression to the LISP interpreter, and returns the result directly below the subject's expression.

| *Subject Input* | *Result* |
|---|---|
| - | **[<function>]** |
| l | (list **[<expr-1>]** <expr-2>) |
| h | (list (head **[<expr-1>]**) <expr-2>)) |
| y | (list (head y) **[<expr-2>]**) |
| t | (list (head y) (**[tail]** <expr-1>))<br>{Feedback} |
| del | (list (head y) **[<expr-2>]**) |
| h | (list (head y) (head **[<expr-1>]**)) |
| e | (list (head y) (head (end **[<expr-1>]**)) |
| x | (list (head y) (head (end x)))<br>returns: (a b) |

**Figure 2:** A subjects solution to the lhhr problem, with error.

Directive Feedback

==========**Help Messages**==========
The system expected to find head at the
current cursor position.

If x = (h i j k) then (head x)
returns: h

Nondirective Feedback

==========**Help Messages**==========
The system did not expect to find tail at
the current cursor position.

If x = (b g h) then (tail x)
returns: (g h)

**Figure 3:** Alternate forms of feedback.

Error Detection: The system detects errors by matching the subject's solution against legal solution templates. When it detects a discrepancy between the subject's solution and a legal solution template, it creates a candidate error. Only if all templates fail to match does the system consider an error to be made. When this happens the system must select one of the candidate errors as the basis for generating feedback. It chooses the candidate error associated with the template that most closely matches the student's solution.

## Results and Discussion

Training: Two measures of errors were collected. Detected errors are those that the system gives feedback on, or those that cause errors when the final solution is evaluated in LISP. Total errors are all the errors that a subject made; the difference between the two indicates the number errors that were self-corrected. The entire solution cannot always be fully analyzed, since once a function has been marked as an error, it is difficult, if not impossible, to determine whether or not its arguments are in error. The status of these arguments are marked as unknown. The total error count provides a conservative estimate of the total number of errors in the solution. As subjects worked on problems until they arrived at correct solutions, it was possible for them to make multiple errors on each problem.

A two way ANOVA was performed on the subjects' total errors, detected errors, and time on task during training. The factors were feedback timing (immediate or delay) and focus (nondirective or directive). There was a main effect of feedback focus on errors. Subjects receiving directive feedback made fewer detected errors than those receiving nondirective feedback, with means of 22 and 35 errors respectively, $F(1,22)= 6.37$, $p < .02$[1]. Considering that the repair of an error was stated in the directive feedback, this does not come as a surprise. Though subjects receiving nondirective feedback made more errors than did those receiving directive feedback, there was not a significant difference in the time they spent completing the problems, $F(1,22)= .53$, $p < .47$.

Feedback timing had no discernible effect on the total number of errors, $F(1,22) = .09$, $p < .76$, nor on the number of detected errors, $F(1,22)= .24$, $p < .62$. The onset of feedback did, however, have a statistically significant effect on time on task, with subjects in the immediate condition finishing the problems about 40% faster than those working with delayed feedback, $F(1,21) = 13.78$, $p < .002$. This could be an artifact of the editor. Since the arguments to a function are deleted along with a function, a subject receiving delayed feedback might have to retype part of her solution when correcting an error. For example, in order to replace *list* in the expression *(list (head x) y))*

[1]Though the subjects' math SAT scores were used as covariates in the analysis, the unadjusted means are reported.

with append, the subject would have to delete *list*, yielding *<function>*. Next she would have to add *append* and retype the arguments.

Test: As there were no lasting effects of feedback on the second day, this experiment replicates Anderson et al (1989). Subjects in the immediate feedback condition worked through the material in less time than did subjects receiving delayed feedback, yet did not appear to suffer any ill effects

## Experiment 2

There were a number of problems with Experiment 1 that were rectified in Experiment 2. In the delayed condition of Experiment 1 feedback and the results of evaluation were provided automatically as soon as a LISP expression was complete. Once an expression had been filled, the delay condition degenerated into an immediate feedback condition. To prevent this in Experiment 2, feedback and the results of evaluation were provided only after an expression had been filled and the subject hit return as an explicit indication of completing their solution. Though a seemingly minor difference, this gave subjects some control over when feedback was provided, perhaps facilitating self-correction.

The apparent advantage of immediate feedback could have derived from difficulties subjects in the delay condition had with the editor. Moreover, they were only able to self-correct 7% of their errors, before receiving feedback. If, as the guidance hypothesis suggests, one of the principal advantages of delayed feedback results from learning secondary skills such as error detection, such skills should be encouraged by the interface. In Experiment 1 simplicity of use was emphasized in the design of the interface, so that the subjects could focus on learning LISP. Though conceptually simple, it was somewhat cumbersome to use an interface that simultaneously deletes a function and its arguments. Subjects in the delay condition were sometimes forced to delete potentially correct parts of their solution. The interface used in Experiment 2 allowed functions within an expression to be replaced

### Method

Subjects: The thirty-two subjects were comparable to those in Experiment 1.

Design: The feedback dimensions of timing (immediate or delayed), focus (directive or nondirective) were crossed, generating 4 cells, with 8 subjects each. All cells were matched for math sats.

Procedure: From the subjects' point of view the procedure was identical to that of experiment 1.

### Results and Discussion

Training: Subjects in the delay condition received feedback on fewer detected errors than did those in the immediate condition, respectively making 17.3 and 36.1 errors, $F(1,23)= 16.2$, $p < .0005$. As in Experiment 1, subjects receiving directive feedback made fewer total (and detected) errors than did subjects receiving nondirective feedback, $F(1,23)= 12.13$, $p < .002$. Subjects in the delayed condition self-corrected 37% of their total errors. By definition subjects in the immediate condition did not have an opportunity to self-correct.

Subjects took longer to finish the problems under nondirective as compared to directive feedback, with respective means of 1824 and 1265 seconds, $F(1,23)= 12.13$, $p < .002$. The advantage for immediate over delayed feedback in experiment 1 was attenuated in this experiment, though it did not disappear completely. Subjects in the immediate condition took less time to complete the problems than did those receiving delayed feedback, taking respectively 1390 and 1699 seconds, $F(1,23)= 2.35$, $p < .13$; subjects in the immediate condition still finished the training problems 18% faster than did those in the delay condition.

Test: Overall subjects in the delay condition made fewer total errors then subjects in the immediate condition with means of 22.0 and 45.3 total errors respectively, $F(1,23)= 9.86$, $p < .005$. Analogous effects were found for time on task. Subjects in the delayed condition spent 1156 seconds on the problems, whereas subjects in the immediate condition took an average of 1727 seconds to complete them, $F(1,23) = 10.35$, $p < .004$. Other dependent measures also showed an advantage of delayed feedback. The total number of steps required to solve the problems, the time required to add a correct step, and the number of errors per correct step were all statistically significant at the .05 level or better.

The guidance hypothesis predicts delayed feedback should foster the development of secondary skills such as self-correction. Consistent with this prediction, subjects in the delay and immediate conditions self-corrected 45% and 30% of their errors respectively, though this difference was not statistically significant.

## Experiment 3

The first two experiments yielded somewhat different results. In Experiment 1, the delay and immediate conditions were relatively similar to each other, there were no differences between the delayed (23 total errors & 17% self correction) and immediate (21 & 13%) conditions. In contrast to Experiment 1, where the testing conditions provided little support, the performance of subjects trained in the delay condition of experiment 2 (22 & 45%) was

condition (45 & 30%). These contrasts should be approached with some caution, not only because the comparisons are across experiments, but because of differences in the conditions during testing. We have no indication, for example, of how subjects trained under the delay conditions of Experiment 1 would perform under the testing conditions of Experiment 2.

The primary goal of experiment 3 was to replicate parts of experiments 1 and 2, in particular to determine whether differences in the interface could account for the differing results.

## Method

Subjects: The thirty-two subjects were comparable to those in Experiments 1 and 2.

Design: In comparison to the first tutor, the second editor was easier to use and its feedback was more delayed. Though these were the major differences between the tutors used in the first and second experiments, numerous other, seemingly minor, changes were also made. If we were to compare the original conditions of experiment 1 with those of experiment 2 we could only attribute subsequent differences in performance to differences between the tutors as whole. The two versions of the tutor were re-implemented within the same general architecture, so that we can localize the cause of any differences in subjects' performance by systematically manipulating the relevant dimensions.

Subjects were trained in one of four conditions, corresponding to the delayed and immediate conditions of the previous two experiments. They were then tested under either the test conditions of experiment 1 or those of experiment 2. This design generated a total of 8 cells, with 4 subjects per cell. For example, their were 8 subjects trained under the delay conditions of experiment 2. Half of these subjects were tested under the delay conditions of experiment 1 and the other half under delay conditions of experiment 2. As the immediate feedback conditions of experiment 1 and 2 were practically identical, these conditions were collapsed together for the analysis.

## Results

Training: The first day results are relatively straightforward. Subjects receiving immediate feedback were somewhat faster than those working in the delayed feedback conditions of either experiment 1 or 2, respectively taking 1713, 2239 and 1795 seconds. $F(1,27)= 2.13$, $p < .14$. Since all subjects received directive feedback, there was relatively little floundering in any of the conditions. Subjects in the immediate condition, and the delay conditions of experiments 1 and 2 made respectively, 21, 24 and 25 errors, $F(1,27)=.5$, $p < .61$.

Test: One subject in the delay condition, who was 2.98 standard deviations slower than the mean, was dropped from the analysis. No other subject was over 2 SD's from the mean on this measure.

Overall, subjects trained in delay conditions (1260 sec) completed the test problems more quickly than did those trained with immediate feedback (1666 sec), $F(1,28)= 2.87$, $p < .10$. Further, subjects in the delay conditions (18 total errors) made fewer mistakes than did subjects in the immediate condition (34 total errors), $F(1,28)= 6.05$, $p < .02$. Subjects in the delay conditions self-corrected 34% of their errors, while subjects in the immediate condition self-corrected only 14% of theirs, $F(1,28)= 6.37$, $p < .015$.

| training | testing | time | total errors | self-corrects |
|---|---|---|---|---|
| Imd | Exp1 | 1689 | 33.8 | 10% |
| | Exp2 | 1648 | 34.5 | 18% |
| Exp1 | Exp1 | 1244 | 18 | 27% |
| | Exp2 | 1462 | 12 | 58% |
| Exp2 | Exp1 | 1166 | 20.7 | 30% |
| | Exp2 | 1138 | 22.5 | 18% |

**Table 1. Testing results of experiment 3.**

As for the replications, subjects trained in the delayed condition of experiment 2 and then tested under the test conditions of experiment 2 performed better than did subjects trained with immediate feedback and tested under these same conditions. Because of the small n, these differences were not significant. The results of the experiment 1 replication were not consistent with those of experiment 1. Subjects trained in the delay condition of experiment 1 performed better than subjects trained with immediate feedback and tested under identical conditions. Both replications suggest an advantage for delayed over immediate feedback.

It could be argued that the differences between the immediate and delayed feedback conditions result solely from the difficulty that subjects trained with immediate feedback have with the interface. To control for this, we can look at subjects' performance on the first step of their solutions, where the advantages that the delay subjects have working with the interface should be diminished. In contrast to the overall result, subjects in the delay (425 sec) condition take longer to make their first moves than do

subjects in the immediate (345 sec) condition, $F(1,28)= 4.55$, $p < .04$. Subjects in the delay (4.3 errors) condition make somewhat fewer mistakes then do those in the immediate (6.3) conditon, $F(1,28)= 2.77$, $p < .11$. This result is consistent with the hypothesis that subjects in the delay condition are spending more time planning their solutions than do subjects who had received immediate feedback during training. In experiments with the geometry tutor, good subjects demonstrated similar behavior (Koedinger & Anderson, 1990).

To summarize Experiment 3, though the subjects in the immediate conditions went through the training problems 18% faster than did those in the delay conditions, they made twice as many errors on the test problems. The results are consistent with experiment 2. This experiment suggests that differences in the interfaces alone do not fully account for the differences in the results between the first and second experiment.

## General Discussion

Within the ACT* framework, in order for an appropriate production, or operator, to be compiled, all of the relevant information needs to be in working memory. Whereas Lewis and Anderson (1985) found an advantage for immediate feedback, Anderson et al (1989) did not. Anderson et al (1989) attribute the differences between their results and those of Lewis and Anderson (1985) to differences in the working memory requirements of the two tasks. They suggest that Lewis and Anderson (1985)

> was a situation where the total correct solution was never laid out before subjects and they had to integrate in memory a sequence of moves. In contrast, in the LISP domain students have at the end a working LISP function in front of them to study. (Anderson et 1989)

This interpretation suggests that not only forgetting, but any processing that forces relevant information out of working memory could disrupt production compilation, impairing learning. For example, Sweller (1988) suggests that learning may be hindered by problem solving, since they both rely on the same, limited, cognitive processing capacity. Like problem solving, the processing of feedback could also compete for limited cognitive resources. When a subject is provided with feedback, the feedback necessitates that they set new goals to process it. When they re-emerge from the feedback episode, the previous goals may have been lost, increasing the likelihood that the subject would rely on the feedback. In contrast, if the feedback processing were somewhat less disruptive, then they might return from the feedback episode with their goals intact. A protocol of a subject RM illustrates these costs and some benefits of immediate feedback.

At this point in the protocol, RM's current goal is to get the second element of the list (q a r s), which is bound to the variable y. This can be accomplished by first taking the *tail* (cdr) of the list and then the *head* (car) of the result. Since the LISP interpreter evaluates expressions from the inside out, the arguments to a function are evaluated before the function. This necessitates implementing plans in the reverse order in which they are generated.

In statement 3 he appears to have the components of a successful plan, including the order in which the functions need to be applied. He forgets, however, about the manner in which LISP evaluates its arguments. The system responds with directive interrogative feedback, suggesting the use of head. In statement 4, he proceeds to include *head* in his solution, without understanding why. In statement 5, he wants to manipulate the list, though it would be appropriate to use *tail*, he doesn't think to do this, even though previously it had been a component in his plans; presumably, the plan has been pushed out of working memory. In statement 8 the system has just suggested *tail*, which he does not recognize as part of his original plan. In statement 9, once the components of the solution are largely in place, he reflects on the solution, and learns how to get the second element of a list; he has several more opportunities to do this in the problems, which he does without error.

---

## Rm Protocol

1 **RM:** ..(um)..so I want to manipulate y with some functions. So I'll say ..(ah)..(ah)

2 **E:** What's your reasoning here?

3 **RM:** I'm thinking maybe I would want to use the head function. No the tail function.

**edit:** (list ( **tail** <expr-1>) <expr-2>)
**error:** The system expected to find head at
the current cursor postion.
What does (head (h i j)) return?

{RM answers question}

4 **RM:** And so that right here. The system expected to find head. I'm not sure why but I will go along with it and say list head.

**edit:** (list (head **<expr-1>**) <expr-2>)

5 **RM:** And I would want to head the ...oh well maybe I'll say ..(um). (ah) .I don't want to use reverse no. I don't only want to say y I don't think. (ah) head...

6 **E:** What are you thinking?

7 **RM:** (um) Trying to think how I can rearrange the order . of the y list q a r s without changing a .without reversing it. Let's see list head ...(um). (ah) I would want to say (um)...(hm) I wish I could use something other than head here, but (ah) the system want's it, so its going to get (ah). (ah) .I guess I could ...(sh).(um).(um). I'll try list again, see if ah.

**edit:** (list (head ( **list** <expr-1> <expr-2>)) <expr-2>)
**error:** The system expected to find tail at the current cursor position.
What does (tail (w f)) return?

{RM answers question}

8 **RM:** So since it wants tail, I'll give it tail. List head tail and I don't know where the computer is going with this.

**edit:** (list (head (tail **<expr-1>**)) <expr-2>)

9 **RM:**(um) What are we tailing?. We are tailing y ...So that means. Oh I get it, I feel so stupid now. Now I get it because, if you tail q a r s, you get a r s and you head that and you get a which is all I wanted in the first place, so I'll just say y here

**edit:** (list (head (tail y)) **<expr-2>**)

**Figure 4.** Verbal Protocol of RM working with immediate feedback

From this protocol it can be seen that even relatively primitive feedback can provide the basis for quality self-generated explanation (statement 9). Second, that feedback can disrupt working memory, forcing out relevant information (statement 3). Third, it is not difficult to see that if RM had not paused to reflect on his solution, he could have very easily finished the problem, relying on the feedback. R.C. Anderson, Kulhavy & Andre (1972) found that copying feedback such as this impaired learning.

There are clearly advantages to immediate feedback. First, it increases the probability that relevant information will be in working memory (Anderson et al, 1989). Second, it decreases the time spent floundering, focusing the subjects attention on relevant information and decreasing time on task. Along with these benefits, however, are the potential costs. First, subjects can grow dependent on the feedback, so that secondary skills such as error detection and self correction do not develop (Schmidt et al, 1989). Second, immediate feedback may compete for the resources of working memory, impairing learning. Perhaps, a form of feedback can be found in which these benefits and costs are optimally balanced.

## References


Anderson, R.C.,Kulhavy, R.W., & Andre, T. (1972) Conditions under which feedback facilitates learning from programmed lessons. *Journal of Educational Psychology*, 63, 186-188.

Anderson, J.R., Conrad, C.G, Corbett, A.T. (in press). Skill Acquisition and the LISP Tutor. *Cognitive Science*,13,467-505.

Koedinger, R., & Anderson, J. R. (1990). Knowledge-Based Environments for Learning and Teaching. Working notes of the AAAI Spring Symposium Series.

Lewis, M.W., & Anderson, J.R. (1985). Discrimination of Operator Schemata in Problem Solving: Learning from Examples. *Cognitive Psychology*, **17**, 26-65

Schmidt, R.A., Young, D.E., Swinnen, S., & Shapiro, D.C. (1989). Summary Knowledge of Results for Skill Acquisition: Support for the Guidance Hypothesis. *Journal of Experimental Psychology: Learning, Memory, and Cognition*,**15**(2), 352-359

Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science*,12,257-285.