*In Jacko, J., Sears, A., Beaudouin-Lafon, M. and Jacob, R. (Eds.) Proceedings of ACM CHI'2001 Conference on Human Factors in Computing Systems, 245-252. New York: ACM Press.*

# Locus of Feedback Control in Computer-Based Tutoring: Impact on Learning Rate, Achievement and Attitudes

**Albert T. Corbett**
Human Computer Interaction Institute
Carnegie Mellon University
Pittsburgh PA 15213
412-268-8808
corbett+@cmu.edu

**John R. Anderson**
Psychology & Computer Science Departments
Carnegie Mellon University
Pittsburgh PA 15213
412-268-2788
ja+@cmu.edu

## ABSTRACT
The advent of second-generation intelligent computer tutors raises an important instructional design question: when should tutorial advice be presented in problem solving? This paper examines four feedback conditions in the ACT Programming Tutor. Three versions offer the student different levels of control over error feedback and correction: (a) *immediate feedback* and immediate error correction; (b) immediate *error flagging* and student control of error correction; (c) feedback on *demand* and student control of error correction. A fourth, *No-tutor* condition offers no step-by-step problem solving support. The immediate feedback group with greatest tutor control of problem solving yielded the most efficient learning. These students completed the tutor problems fastest, and the three tutor-supported groups performed equivalently on tests. Questionnaires revealed little student preference among the four conditions. These results suggest that students will need explicit guidance to benefit from learning opportunities that arise when they have greater control over tutorial assistance.

### Keywords
Intelligent Tutoring Systems, Instructional Interface Design, Student Modeling, Feedback in Problem Solving

## INTRODUCTION
In learning and problem solving, when should errors be pointed out and advice offered? While human tutors have always grappled with this issue [10,14], it only emerged as a formal research topic early in the 20th century. A major stimulus for this research was the advent of Pressey's "teaching machine" [16] which for the first time held out the promise of prompt, individualized instruction on a wide scale – a promise fulfilled in the second half of the century by computer-based instruction.

Early automated instruction supported single-answer problem solving, and under the influence of Thorndyke and Skinner, most feedback research focused on timing, contrasting immediate and delayed feedback. This research yielded mixed results. Overall, feedback timing has little consistent impact on learning, although delayed feedback can facilitate retention [5,11,12].

With the development of "second generation" intelligent tutoring systems that can provide feedback and advice on subgoals in complex problem solving tasks, a new issue arises. Immediate feedback on component problem solving actions runs the risk of interfering with overall task performance and learning in several ways. It can disrupt performance of real-time tasks, such as radar monitoring [15] and may disrupt cognitive processes in task execution more generally [17,19]. Immediate feedback prevents students from learning error-detection skills and may discourage metacognitive self-monitoring processes [9] more generally. Finally, immediate feedback on problem solving actions may have adverse affective and motivational consequences [13].

We explored the issue of feedback timing and control in the context of the ACT Programming Tutor (APT). As described in the following section, APT is a computer-based problem solving environment in which students learn to write short programs. It is a *cognitive tutor* constructed around a cognitive model of the programming knowledge students are acquiring. This cognitive model enables the tutor to follow the student's step-by-step solution in problem solving, providing help on each step as needed. In this study we compare four versions of the tutor. Three of these versions can offer symbol-by-symbol assistance in problem solving, but vary in the degree of control students have over the use of this assistance. An *immediate feedback* version provides feedback on each symbol and requires the student to fix errors immediately, so the student always remains on a recognized solution path. An error flagging version also signals errors immediately, but the student has full control over code editing. *A feedback-on-demand* version offers no assistance until the student requests it. The fourth *no tutor* comparison version does not provide any symbol-by-symbol level support and only notifies the student whether or not each program works correctly.

## THE ACT PROGRAMMING TUTOR AND FEEDBACK VARIATIONS
APT is a cognitive tutor constructed around a cognitive model of the programming knowledge the student is

acquiring. This cognitive model enables the tutor to trace the student's solution path through a complex problem solving space in a process we call *model tracing*. The tutor can provide feedback on each problem solving action (each symbol the student adds to the program) and can provide advice on steps that achieve problem solving goals. APT has previously served as a useful research tool in extensive studies of learning and problem solving [1], student modeling [7] and feedback content [8]. See [1] for related work on feedback control.

Figure 1 depicts the screen of the APT Lisp Module near the beginning of a tutor problem. The screen is divided into four windows. The problem description is displayed in the top window and remains on the screen except when the tutor is providing a feedback message. The student's code appears in the second window. In this figure the student has typed a left parenthesis and the operator *defun*, which is used to define a new function in Lisp. The tutor completes the template for a call to *defun* with three symbols in angle brackets <NAME> <PARAMETERS> and <BODY>. These symbols are placeholders representing arguments of *defun* and the student will replace each one with additional Lisp code. In the third window, the student has access to a Lisp interpreter. Students can execute Lisp code in this window at any time. In particular, they can test their solutions in the Lisp window by trying them out on sample values. The bottom window contains a summary of the editing and help commands available to the student. Figure 1 displays the minimal set of commands employed in the immediate feedback version of the tutor.

APT reflects the ACT-R theory of skill knowledge [4], which assumes that goal-oriented problem solving knowledge can be represented as a set of independent production rules that associate problem states and goals with problem solving actions and consequent state changes. The tutor conventionally provides immediate feedback on each production firing (each program symbol the student codes) as described in the following section.

### Immediate Feedback and Error Correction

The standard immediate feedback version of the programming tutor advances the cursor on a top-down, left-to-right, depth-first sequence through the placeholder symbols. If the student types an incorrect symbol, the tutor immediately provides feedback, deletes the symbol and requires the student to try again. If the student makes repeated errors, the tutor ultimately provides a correct symbol along with an explanation. The student may also request help at any editor node - either a description of the current goal or an explanation of how to accomplish it. In this mode, students always remain on a problem solving path recognized by the tutor and always reach a successful conclusion to problem solving.

This immediate feedback mode has been employed successfully in cognitive programming and mathematics tutors for 15 years. Both niversity programming students and high school mathematics students score substantially higher on tests (roughly a letter grade) than comparable students who complete equivalent problems on their own [2] and Schofield [18] has documented that high school students find cognitive mathematics tutors highly motivating.

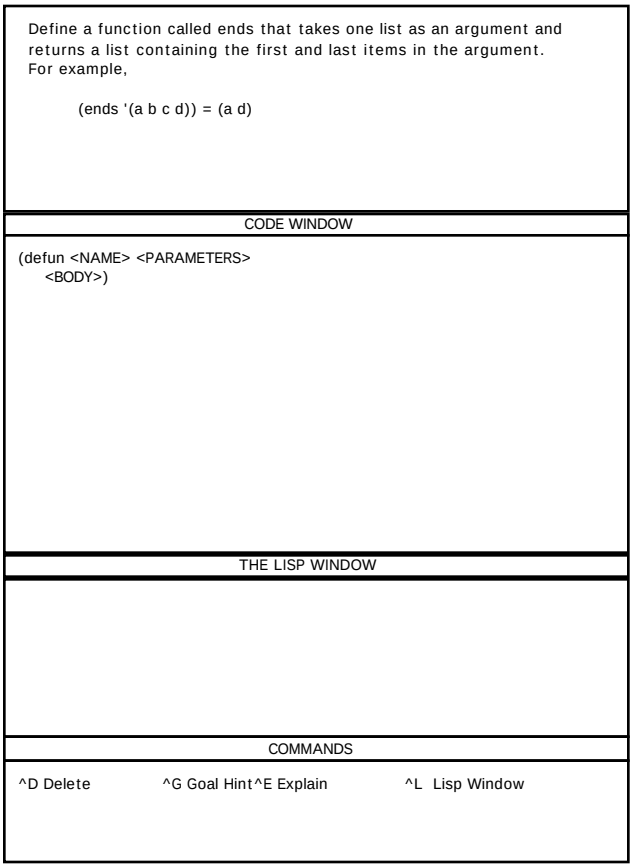


Figure 1. The APT Lisp Tutor interface (Immediate Feedback version)

The ACT Programming Tutor offers an interesting opportunity to examine the costs and benefits of relaxing the tutor's control over feedback and error correction for two reasons. First, expert programming involves a variety of code inspection, testing and debugging skills that are not directly exercised in the tutor. Second, the university students who work with the tutor represent a different population than the high school students. To explore feedback control we developed three new programming tutor variations, as described in the following sections, in which students have complete control over writing and editing their code with a full structure editor. The editor expands the function call templates just as in the immediate feedback condition, and it is possible for students in each of these conditions to follow exactly the same solution path as is required by the immediate feedback tutor. However, a set of cursor control and editing commands allows students to deviate from the standard coding order and to edit existing code.

### Error-Flagging

In this condition, the tutor provides immediate feedback on each coding step, displaying errors in bold on the screen,

but does not interrupt the student and require a correction. Instead, students are free to edit errors or to continue coding. At any time students can select any flagged node and ask for advice and the same advice is available as in the immediate feedback tutor. Students may also test their code at any time in the Lisp Window. Since step-by-step advice is available if needed, students are required to complete a correct solution to each problem, but unlike the immediate feedback condition, the student may turn in a working solution that the tutor cannot generate and does not recognize. The tutor evaluates such variant solutions by applying them to test cases and accepts code that works correctly even though some symbols may be flagged.

### Demand-Feedback

In the demand-feedback condition, the tutor only provides help upon request of the student. At any time students can ask the tutor to check over the code for errors or can ask for advice on any symbol. In either case, the same advice is available as in the immediate feedback and error flagging conditions. When students submit a problem solution, the tutor tests the code and will accept any working solution. If the program does not work correctly, the tutor checks through the code and reports the first error it detects. Since step-by-step help is available upon request, the student is required to reach a correct solution to each problem.

### No-Tutor

In this condition, students receive no advice on how to write the programs. When the student signals that he or she is done working, the tutor tests the code and indicates whether or not the solution is correct. If not, the student can continue working. Students are encouraged to keep working until reaching a correct solution, but are not required to, since it may not be possible without assistance. This is an important contrast with the three feedback conditions in which students are required to achieve working solutions. If a student in the no-tutor condition gives up with an incorrect answer, the tutor presents a canonical solution to the problem. This approximates a common homework situation in which students can look up correct solutions in the back of the book.

## DESIGN OF THE STUDY

These four tutor versions were evaluated in a study in which students worked through five chapters in a Lisp programming curriculum, completing a fixed set of programming problems. Three measures of tutor impact were employed: (1) student performance in completing the tutor problems, (2) performance in three different test environments and (3) student attitude questionnaires.

### Participants

Forty undergraduates participated in the study for pay. Each participant was assigned to one of the four feedback conditions. The students in this sample had an average Mathematics SAT score of 693 and had taken an average of 1.9 programming courses prior to this experiment, although none had prior experience with Lisp. These variables were controlled in assigning students to the four conditions.

### Procedure

The students worked at their own pace through five chapters of a Lisp programming text [3], and completed accompanying exercises with the tutor. Following the second lesson, subjects filled out a short attitude questionnaire and completed a paper-and-pencil programming test. Following the fifth lesson, students again filled out a short questionnaire and completed three programming posttests. The first of these tests was paper-and-pencil, the second test required students to complete exercises on-line with a text editor and Lisp interpreter, and in the third posttest all students completed exercises with an immediate-feedback version of the tutor.

### Curriculum

Five lessons were selected to bring students to the most challenging topic in the tutor curriculum, recursion, as rapidly as possible. The five lessons cover function calls, function definitions, conditionals, basic recursion and advanced recursion. Students completed a total of 42 tutor problems in completing these lessons.

### Tutor Versions

All four tutor versions display four windows on the screen as described in the introduction. All four versions provided access to a Lisp interpreter in the third window. Students could enter this Lisp environment as often as they wished and at any time in the course of completing a problem. When students enter the Lisp environment, their problem code is automatically loaded into the environment. The bottom window on each screen provides a reminder of the editing and help commands available to the student.

#### *Code Entry.*

Students in the immediate-feedback condition are constrained to enter their code top-down, left-to-right and depth-first. They are required to enter a correct symbol at each goal before proceeding to the next and they can not go back and modify correct symbols. Students in the error-flagging, the demand-feedback and no-tutor conditions can freely enter and modify their code with a full structure editor.

#### *Help Facilities.*

The immediate-feedback tutor interrupts students immediately when a mistake is made. It provides a feedback message, deletes the error and requires the student to try again. If the students makes three errors that the tutor cannot diagnose or if the student makes repeated errors of the same type, the tutor provides an explanation of the correct step and inserts the correct symbol into the code. The error-flagging tutor immediately redisplays any incorrect code in bold, but does not interrupt the student. The feedback-on-demand tutor never volunteers information. The no-feedback version provides no feedback at the level of individual symbols in the student's code.

Two types of help are common to the immediate feedback, error flagging and demand-feedback versions of the tutor. Students can ask for a *goal hint* or an *explanation.* A goal hint reminds the student of the pending goal, but does not provide information on how to achieve that goal. An explanation describes how to achieve the current goal and

inserts the correct symbol into the student's code. Students in the error-flagging and feedback-on-demand versions can each request one other type of help. In the error-flagging condition, students can ask the tutor to *comment* on an error the tutor has flagged. In response, the tutor displays the error feedback message that is presented automatically in the immediate-feedback condition. Students in the feedback-on-demand condition can ask the tutor to *verify* their code. In response, the tutor searches the code top-down, left-to-right and depth-first, highlights the first error that is found and provides the corresponding error feedback message. If there are no errors, the tutor notifies the student. If in executing either of these two commands the tutor finds that a student has made three undiagnosed errors at the same goal or is repeating the same type of error at a goal, the tutor will provide an explanation of the correct action and insert the correct symbol into the code.

*Exercise Completion.*

In the immediate-feedback version an exercise ends automatically as soon as the student has completed a correct solution. Students in the other three conditions notify the tutor when they think they have finished an exercise. If the code is complete and recognized as correct, the tutor advances the student to the next exercise. If the code is a syntactically complete program but not a recognized solution, the tutor tests the code on sample cases. If the code works, the tutor notifies the student that the code works but is unrecognized. In this case, the tutor displays a canonical solution to the exercise, then advances the student to the next exercise. If the code does not work, the tutor notifies the student. In the error-flagging condition, the tutor points out that there are still errors flagged on the screen. In the feedback-on-demand version the tutor automatically executes a verification, as described above, and points out the first error that is encountered. Students in these two conditions cannot finish an exercise until they have code that works correctly. In the no-feedback condition, the program simply notifies the student if the code does not work and encourages the student to keep trying. Since no assistance on subgoals is available in this condition there is no guarantee the student will reach a correct solution and the program will allow the student to complete an exercise even if the code is incorrect. In this case, the program displays a canonical solution to the exercise.

### Programming Tests

The first paper-and-pencil test, following the second lesson, contained ten exercises, three code evaluation questions, three code debugging questions and four code generation exercises. The code generation questions were similar to the tutor exercises and required students to write short Lisp programs. In the code evaluation questions students determined the results of executing short Lisp programs with sample values. The debugging questions presented short programs with mistakes that the student was required to fix. The posttests that followed the final lesson consisted exclusively of code generation exercises. The paper-and-pencil test contained eight questions, while each on-line test required the student to complete six exercises. In the on-line editor test, students entered their code with a text editor and had access to a Lisp environment. They could readily load their code into the Lisp environment for testing. The final posttest employed an immediate-feedback version of the tutor, but students did not have access to a Lisp environment as they typed their code. All tests were open book.

### Questionnaires

Each questionnaire contained the first seven questions displayed in Table 4 below. The first two questions were intended to assess the students' self-knowledge of the learning process. The next five questions asked their opinions on the tutor. The eighth question, asking whether students would like to learn more Lisp, only appeared on the second questionnaire. Students responded to all questions on a 7-point Likert scale.

## RESULTS

Students required an average of 12.4 hours to complete the experiment, distributed over an average of 7.4 sessions.

### Tutor Performance – Learning Time

As displayed in Figure 2, average time to complete the tutor problems is inversely related to the level of support provided by the tutor. Students in the immediate feedback condition finished the problems fastest, followed in order by students in the error flagging, demand feedback and no-tutor conditions. In a two-way analysis of variance (tutor version x lesson) this main effect of tutor version is reliable $F(3,36) = 11.69$, $p < .01$. The main effect of lesson is also reliable $F(4,144) = 41.98$, $p < .01$, and the interaction of feedback condition and lesson is marginally reliable, $F(12,144) = 1.74$, $p < .07$. Note that learning time for the immediate feedback, error flagging and demand-feedback conditions is similar for the first two, (easiest) lessons and substantially shorter than in the no-tutor condition. For

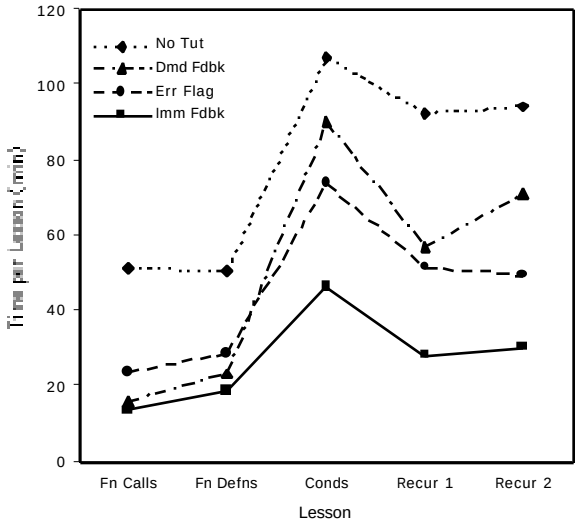


Figure 2. Mean time per lesson to complete the programming problems in the five tutor lessons.

the final three harder lessons, the advantage of immediate feedback over the other three conditions increases. It is important to note that students in the no-tutor condition gave up without generating a correct solution in 30% of the problems across the lessons, so Figure 2 underestimates the true learning time in this condition.

### Tutor Performance – Use of Feedback

We examined the tutor protocol files for two lessons in detail, to see how students made use of feedback in the error-flagging and demand-feedback conditions. We selected lesson 2, an easy lesson and lesson 9, with the hardest exercises. Students in both conditions responded passively to the available assistance. That is, students in the error-flagging condition tended to edit errors almost immediately. In lesson 2 they modified 66% of their errors immediately and 79% after coding at most one additional symbol (which may reflect type-ahead). By lesson 9, these percentages rose to 78% and 87% respectively. Students in the demand-feedback condition, however, tended to wait until they had typed a complete solution before seeking feedback. In lesson 2, only one student requested feedback in an exercise prior to coding a complete solution. In lesson 9, only three subjects asked for feedback prior to completing a solution.

### Test Performance

Students completed posttests in three environments, (1) paper-and-pencil, (2) a text editor and Lisp environment and (3) an immediate-feedback tutor. The accuracy measure for paper-and-pencil and on-line editor posttests is the number of errors at the level of individual code symbols in the students' final answer to the exercises, corresponding to the number of incorrect production firings. In the case of the immediate-feedback tutor posttest, students necessarily arrive at a correct answer for each exercise. The accuracy measure for this test is the probability of making a mistake at each goal in the course of completing the exercises. Elapsed time was recorded for both the on-line editor and tutor tests. Data was lost for two students in the no-tutor condition for the two on-line tests due to disk failure.

#### *Paper and Pencil Posttests.*

Table 1 displays the mean number of errors per problem in the two paper-and-pencil posttests. The first test, following lesson 2, contained code evaluation problems, code debugging problems and code writing problems. The second test, following the fifth lesson contained code writing exercises only. Students in the no-tutor condition are making substantially more errors than students in the other three conditions. The effect of tutor version was marginally significant in a two-way analysis of variance, $F(3,36) = 2.48$, $p < .08$. The effect of question type was significant, $F(3,108) = 27.99$, $p<.01$ and the interaction was marginally significant, $F(9,108) = 1.77$, $p<.09$.

To explore the effect of tutor version further, we performed a two-way analysis of variance that collapsed across the three feedback conditions, to contrast an overall "feedback" condition with the no-tutor condition. In this analysis, the effect of feedback was significant, $F(1,38) = 6.81$, $p < .05$. Students in the no-tutor condition made reliably more errors than students who had feedback available. The effect of question type was again significant, $F(3,114) = 28.73$, $p < .01$ as was the interaction of feedback and question type $F(3,114) = 4.39$, $p < .01$. Finally, we performed an analysis that excluded the no-tutor condition and compared just the three feedback conditions. The effect of tutor version in this analysis was not significant. The effect of question type was again significant, $F(3,81) = 14.84$, $p < .01$, but the interaction was not.

To explore the reliable interaction of tutor version and question type, we performed an analysis of variance on each of the four question types separately, collapsing across the three feedback conditions to compare a "feedback" condition with the no-tutor condition. The effect of feedback was not significant for either the evaluation or debugging questions. However, students who received feedback conditions were reliably more successful than the no-tutor students on the Test 1 coding problems, $F(1,38) = 6.78$, $p < .05$ and on the Test 2 coding questions $F(1,38) = 5.21$, $p < .05$. Finally, we performed an analysis of variance on each of the four questions in which we excluded the no-tutor condition and compared the three feedback conditions. There were no reliable differences among the feedback conditions on any of the four problem types.

| | Immed Feedbk | Error Flag | Demand Feedbk | No Feedbk |
|---|---|---|---|---|
| Test 1 | | | | |
| Evaluation | 0.33 | 0.40 | 0.40 | 0.60 |
| Debugging | 0.30 | 0.50 | 0.27 | 0.57 |
| Coding | 1.00 | 0.98 | 0.58 | 1.83 |
| Test 2 | | | | |
| Coding | 4.16 | 2.76 | 2.44 | 6.58 |

Table 1: Mean number of errors per answer in the paper and pencil tests.

#### *On-Line Editor/Lisp Interpreter Test*

Results of the on-line editor test are displayed in Table 2. The mean number of coding errors per answer is reported, along with mean elapsed time to complete each problem. The difference in error rates in this comparison is non-significant. However, a number of students exceeded the time limit for this test and failed to even begin the final exercise. This necessarily distorts estimates of error rates at the level of individual symbols. As a result, we computed the error count for just the first five exercises. The group means are ordered identically and in this case, the effect of tutor version was marginally significant $F(3,34) = 2.55$, $p < .09$ Again, we performed an analysis that collapsed across the three feedback conditions to contrast "feedback" with the no-tutor condition. In this analysis the effect of feedback was significant; students in the no-tutor condition made reliably more errors, $F(1,36) = 7.00$, $p < .05$. In an analysis of variance that excluded the no-tutor condition,

the error rate differences among just the three feedback conditions was not significant.

The effect of tutor version on elapsed time is reliable $F(3,34) = 3.13$, $p < .05$. Students in the immediate feedback and no feedback conditions are about 21% slower finishing the exercises than students in the two student-controlled feedback conditions. Four pairwise T-tests comparing each of the error flagging and demand-feedback conditions to each of the immediate-feedback and no-feedback conditions were all at least marginally significant. Note, however, that the difference between the two student-controlled conditions and the immediate feedback condition represents a speed-accuracy tradeoff. While students in the immediate feedback conditions are finishing more slowly than students in the error flagging and demand feedback conditions, the immediate-feedback students have almost 40% fewer errors in their answers.

| | Immed Fdbk | Error Flag | Demand Fdbk | No Fdbk |
|---|---|---|---|---|
| Errors | 2.63 | 3.61 | 3.68 | 5.25 |
| ElapsedTime | 8.2 | 6.9 | 6.6 | 8.1 |

Table 2: Mean number of errors and mean elapsed time per problem (min) for the on-line editor/Lisp interpreter test.

*Immediate Feedback Tutor Posttest.*

Finally, all students completed six "test" exercises with the immediate feedback tutor. This environment offers the purest assessment of students' coding skills, distinct from their code inspection and debugging skills. Table 3 displays the probability that the students' first coding action is correct at each problem solving step, and displays students' mean elapsed time to complete each problem. There is little difference among the four groups in the probability of a correct coding action, but students in the no-tutor group are taking substantially longer overall to complete these exercises. The effect of tutor version on elapsed time is significant in an analysis of variance, $F(3,34) = 3.14$, $p<.05$. An analysis contrasting the three feedback conditions with the no-tutor condition was also significant, $F(1,36) = 9.62$, $p < .01$. An analysis that excluded the no-tutor condition to compare elapsed time in the three feedback conditions was non-significant.

| | Immed Feedbk | Error Flag | Demand Feedbk | No Feedbk |
|---|---|---|---|---|
| p(Correct) | 0.93 | 0.91 | 0.94 | 0.90 |
| Elapsed Time | 3.0 | 3.0 | 3.2 | 4.2 |

Table 3: Probability of a correct response at each coding step and elapsed time per problem (min) in completing the immediate feedback test problems.

In summary, the students in the three feedback conditions achieve equivalent accuracy levels in all tests. They perform more accurately than the no-tutor group in the paper and pencil coding exercises and in the on-line editor test. All four groups are equally accurate in the immediate feedback posttest, although the no-tutor group took longer to complete that posttest.

**Questionnaires.**

Students completed attitude questionnaires after lessons 2 and 5. The results are displayed in Table 4. A separate two-way analysis of variance was performed on each one of the first seven questions with feedback condition and questionnaire (first vs. second) as factors. A one-way analysis of variance was performed on the eighth question, which only appeared in the second questionnaire. The most striking pattern in these results is the overall similarity of the ratings across the four feedback groups.

*Self-monitoring of Learning*

The first two questions were designed to assess students' monitoring of the learning process. The questionnaire factor was significant in both analyses. The first question asked students to judge exercise difficulty and students accurately perceived that the exercises were more difficult in the final three lessons, $F(1,36) = 69.04$, $p < .01$. Question 2 asked how well students learned the material and students were less confident they had learned the material well in the final three lessons, $F(1,36)=27.10$, $p < .01$. There was a marginal effect of tutor version in the first question. Students who received more feedback from the tutor (the immediate feedback and error flagging conditions) generally perceived the exercises as easier, $F(3,36)=2.41$, $p<.09$. There was no effect of tutor version on the second question, and no interaction in either analysis.

*Feedback Opinions*

Questions 3-7 asked students' opinion of the tutor. The surprising result is that there was no reliable main effect of tutor version on how much students liked the tutor (question 3), and more specifically, how much they liked the tutor's assistance (question 6), whether or not they would like more assistance (question 7) and whether they thought the tutor helped them understand better (question 5). There is a reliable interaction of questionnaire and tutor version for question 3, however, $F(3,36)=3.12$, $p<.05$. Students' overall rating of how much they like the tutor declined from the first to second questionnaire, but the decline was smaller for the immediate feedback and error flagging groups who received more assistance. In fact, this rating actually went up slightly from the first to second questionnaire for the immediate feedback group.

There was only one significant main effect of tutor version among questions 3-7. In question 4 students' perception of whether the tutor helped them finish more quickly was directly related to the level of feedback control exercised by the tutor, $F(3,36)=3.32$, $p<.05$. The more help the tutor provided, the more students believed the tutor helped them finish faster. Note that these beliefs are consistent with the learning time data displayed in Figure 2. There was also a significant interaction in this question. The difference

| *Questions* | Questionnaire 1 | | | | Questionnaire 2 | | | |
|---|---|---|---|---|---|---|---|---|
| | **Imm Fdbk** | **Err Flag** | **Dmd Fdbk** | **No Fdbk** | **Imm Fdbk** | **Err Flag** | **Dmd Fdbk** | **No Fdbk** |
| 1. How Difficult were the exercises? (1=Easy, 7=challenging) | 3.3 | 2.9 | 3.8 | 4.1 | 5.2 | 4.9 | 5.7 | 5.7 |
| 2. How well did you learn the material? (1=Not Well, 7=Well) | 5.8 | 5.4 | 5.7 | 5.3 | 4.8 | 5.3 | 5.1 | 4.5 |
| 3. How much did you like the tutor? (1=Disliked, 7=Liked) | 5.1 | 4.9 | 5.7 | 4.9 | 5.5 | 4.6 | 4.3 | 4.5 |
| 4. Did the tutor help you finish more quickly? (1=Slower, 7=Faster) | 5.3 | 4.3 | 4.7 | 4.2 | 6.5 | 5.7 | 4.3 | 4.5 |
| 5. Did the tutor help you understand better? (1=Interferred,7=Helped) | 5.5 | 4.8 | 5.3 | 5.0 | 4.5 | 4.2 | 4.8 | 4.5 |
| 6. Did you like the tutor's assistance? (1=Disliked, 7=Liked) | 5.3 | 4.6 | 5.5 | 4.8 | 5.2 | 4.7 | 4.3 | 4.5 |
| 7. Would you like more or less assistance? (1=Less, 7=More) | 3.8 | 4.0 | 4.5 | 4.8 | 4.1 | 4.5 | 4.4 | 5.1 |
| 8.Would you like to learn more Lisp? (1=No, 7=Yes) | - | - | - | - | 6.5 | 5.0 | 4.9 | 5.9 |

Table 4. Mean responses to the attitude questionnaire of the students in each of the four feedback conditions.

between the high feedback groups (immediate and error flagging) and low feedback groups (demand and no-feedback) was more pronounced in the second questionnaire $F(3,36)=3.11$, $p<.05$.

The main effect of the first vs. second questionnaire was at least marginally reliable for four of the five tutor feedback questions. In the second questionnaire, students liked the tutor less (question 3) $F(1,36)=4.11$, $p<.06$, liked the tutor's assistance less (question 6) $F(1,36)=3.68$, $p<.07$, were less convinced that it helped them understand the material (question 5) $F(1,36)=6.19$, $p<.05$, but were more convinced that the tutor helped them finish faster (question 4) $F(1,36)=3.11$, $p = .10$.

The final question in questionnaire 2 asked students whether they would like to learn more Lisp. The effect of tutor version is marginally significant for this question, $F(3.36) = 2.82$, $p < 06$. Students in the immediate feedback and no-tutor conditions were more eager to continue than in the error flagging or feedback-on-demand conditions. This pattern of results is surprising, since the immediate-feedback and no-tutor conditions are quite dissimilar. The only quality these two conditions have in common is that students have no control over the tutorial assistance.

## DISCUSSION

There are three principal conclusions concerning the impact of the four feedback conditions. First, time to complete the fixed set of tutor problems was inversely related to the control exerted by the tutor in problem solving. Students in the immediate feedback condition finished fastest, followed in order by students in the error flagging, demand-feedback and no-tutor conditions. Second, students in the three tutor-supported conditions, all of whom were required to obtain correct solutions to the tutor problems, performed equivalently across three test environments, while students in the no-tutor condition who failed to solve 30% of the tutor problems also scored about 25% worse across test environments. Thus, test achievement is strongly related to the set of problems students successfully solved, while the solution paths students followed in reaching those solutions had little or no impact. Finally, students did not show any strong preferences among the four feedback conditions. There could hardly be a wider range of tutor support and feedback control across conditions, but there were no differences among the four groups in how much they liked the tutor, how much they liked the tutor's assistance and whether they wanted more assistance. The performance of the error flagging students and demand-feedback students on the tutor problems provides converging evidence for this conclusion. When immediate feedback was provided in the error flagging condition, students were largely content to fix the errors immediately, and when no feedback was offered in the demand feedback condition, students were content to wait until they had entered a complete program before requesting help.

This pattern of results is particularly surprising, because the students in this study are high-achieving and well-prepared to learn a new computer language. These students, on the whole, might be expected to prefer more control over learning and should be well-prepared to take advantage of learning opportunities afforded when tutor control is relaxed.

Two instructional design principles follow from these results. First, immediate feedback on individual problem solving steps can be an efficient and effective form of tutorial support for students learning a complex problem solving skill such as programming. Students in the immediate feedback condition worked through the tutor problems fastest, while there was little difference among the three step-by-step feedback conditions in test performance. The decision about when to deploy immediate feedback is essentially a curriculum decision. Immediate feedback on each symbol is efficient when students are focused on writing programs, but would have to be relaxed if we want students to focus on debugging the code they write.

Second, these results demonstrate that relaxing immediate feedback on coding performance may be necessary, but is not sufficient to promote error detection and other process-monitoring skills. Instead, it is important to provide direct training and support for such skills. Bielaczyc, Pirolli and Brown [6], for example, have demonstrated that metacognitive skills in programming can be directly trained. When such skills are targeted in intelligent computer tutors, immediate feedback on those skills should be a useful pedagogical tool.

## ACKNOWLEDGMENTS

This research was supported by NSF grant number 9720359 to CIRCLE: Center for Interdisciplinary Research on Constructive Learning Environments.

## REFERENCES

1. Anderson, J.R., Conrad, F.G. and Corbett, A.T. (1989). Skill acquisition and the LISP Tutor. *Cognitive Science, 13*, 467-505.
2. Anderson, J.R., Corbett, A.T., Koedinger, K.R. and Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences, 4*, 167-207.
3. Anderson, J.R., Corbett, A.T. and Reiser, B.J. (1987) *Essential Lisp*. Reading, MA: Addison-Wesley.
4. Anderson, J.R. and Lebiere, C. (1998*). The atomic components of thought*. Mahwah, NJ: Erlbaum.
5. Bangert-Drowns, R.L., Kulik, C.C., Kulik, J.A. & Morgan, M. (1991). The instructional effect of feedback in test-like events. *Review of Educational Research, 61*, 213-238.
6. Bielaczyc, K., Pirolli, P.L., and Brown, A.L. (1995). Training in self-regulation strategies: Investigating the effects of knowledge acquisition activities on problem solving. *Cognition and Instruction, 13*, 221-252.
7. Corbett, A.T. and Anderson, J.R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction, 4*, 253-278.
8. Corbett, A.T. and Trask, H. (2000). Instructional interventions in computer-based tutoring: Differential impact on learning time and accuracy. *CHI 2000 Conference Proceedings: The Future is Here*, 97-104.
9. Chi, M.T.H., Bassok, M., Lewis, M, Reimann, P., Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 13*, 145-182.
10. Fox, B. (1991). Cognitive and interactional aspects of correction in tutoring. In P. Goodyear (ed.*) Teaching knowledge and intelligent tutoring*. Norwood, NJ: Ablex.
11. Kulhavy, R.W. (1977). Feedback in written instruction. *Review of Educational Research, 47*, 211-232.
12. Kulik, J.A. and Kulik, C.C. (1988). Timing of feedback and verbal learning. *Review of Educational Research, 58*, 79-97.
13. Lepper, M.R., Woolverton, M., Mume, D.L. and Gurtner, J. (1993). Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors. In S. Lajoie & S. Derry (eds.*) Computers a cognitive tools*. Hillsdale, NJ: Erlbaum.
14. Merrill, D.C., Reiser, B.J., Ranney, M. and Trafton, J.G. (1992). Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *Journal of the Learning Sciences, 2*, 277-305.
15. Munro, A. Fehling, M.R. and Towne, D.M. (1985). Instruction intrusiveness in dynamic simulation training. *Journal of Computer-Based Instruction, 12*, 50-53.
16. Pressey, S.L. (1926). A simple apparatus which gives tests and scores - and teaches. *School and Society, 23* (586), 373-376.
17. Schmidt, R.A., Young, D.E., Swinnen, S. and Shapiro, D.C. (1989). Summary knowledge results for skill acquisition: Support for the guidance hypothesis. *Journal of Experimental Psychology: Learning, Memory and Cognition, 15*, 352-359.
18. Schofield, J.W. (1995). *Computers and classroom culture.* Cambridge, England: Cambridge University Press.
19. Schooler, L.J. and Anderson, J.R. (1990). The disruptive potential of immediate feedback. *The Proceedings of the Twelfth Annual Conference of the Cognitive Science Society,* Cambridge, MA.