The Effect of Using Problem-Solving Tutors on the Self-Confidence of Students

Amruth N Kumar

Ramapo College of New Jersey,
Mahwah, NJ 07430
amruth@ramapo.edu

Abstract. We have been developing software tutors to help students learn programming concepts by solving problems. We evaluated the effect of these tutors on the self-confidence of the students in fall 2004 and spring 2005. We found that the self-confidence of the class as a whole increases. However, there is no correlation between the change in learning and the change in the self-confidence of the students.

1 Introduction

We have been developing software tutors for various topics in Computer Science. The topics include introductory programming topics (e.g., expression evaluation [4,6], loops [1]), advanced programming topics (e.g., pointers [9,10], classes [3]) and topics in the junior/senior level Comparative Programming Languages course (e.g., scope and its implementation [2,11], parameter passing mechanisms [13]). We have evaluated our tutors over several semesters, in multiple schools, and under differing conditions, and in each case, our evaluation has shown that the tutors have helped students learn. The evidence that we have accumulated supports the following conclusions:

1. Using software tutors helps Computer Science students learn programming concepts [1,3,8].
2. Tutors that provide narration of step-by-step execution of programs are more effective in helping students learn than those that do not [2,5,8,9].
3. When presented with a choice between software tutors and printed workbooks/traditional textbooks for learning, students prefer to use software tutors; and they prefer tutors that provide feedback to those that do not [7].

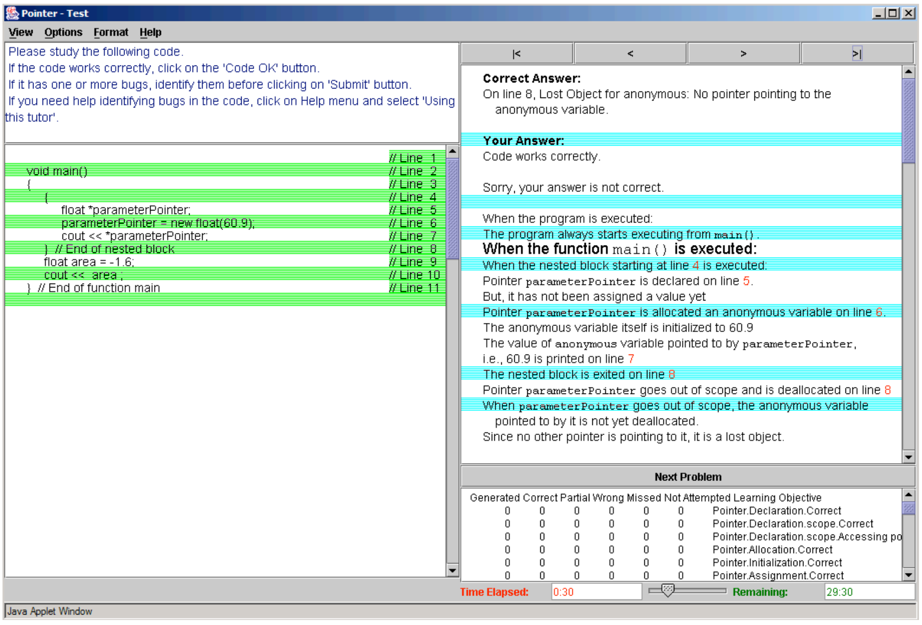
In this paper, we will examine the effect of using software tutors on the self-confidence of students. In the next section, we will discuss the software tutors that we evaluated, and the protocol we used for evaluation. We will present the methods in section 3, the results of our evaluation in section 4, and discuss the results in section 5.

2 The Tutors

We have been developing software tutors to help students learn Computer Science topics by solving problems. The features of these tutors include:

- The tutors generate problems as instances of parameterized templates. This enables the tutors to present multiple un-identical instances of a problem, either to the same user on different occasions to provide for repetitive practice, or to different users on the same occasion to prevent plagiarism / cheating when the tutors are used for assignments and tests.
- The tutors provide detailed feedback. This includes narration of the step-by-step execution of the program. This helps students learn from their mistakes. The tutors can be used with feedback for tutoring, and without feedback for testing.
- The tutors adapt to the needs of the learner. They present problems on concepts that the student does not yet understand, and skip the concepts over which the student has demonstrated mastery.

The tutors run on the Web. The interface of the tutor on C++ pointers is presented in Figure 1. In the figure, the program is shown in the left panel, and the feedback is presented in the right panel. The student enters answers by clicking on the code in the left panel and selecting an applicable bug from a drop-down box that appears.



Please study the following code.
If the code works correctly, click on the 'Code OK' button.
If it has one or more bugs, identify them before clicking on 'Submit' button.
If you need help identifying bugs in the code, click on Help menu and select 'Using this tutor'.

```
                                                // Line  1
   void main()                                  // Line  2
   {                                            // Line  3
       {                                        // Line  4
           float *parameterPointer;             // Line  5
           parameterPointer = new float(60.9);  // Line  6
           cout << *parameterPointer;           // Line  7
       }  // End of nested block                // Line  8
       float area = -1.6;                       // Line  9
       cout <<  area ;                          // Line 10
   }  // End of function main                   // Line 11
```

Correct Answer:
On line 8, Lost Object for anonymous: No pointer pointing to the anonymous variable.

Your Answer:
Code works correctly.

Sorry, your answer is not correct.

When the program is executed:
The program always starts executing from `main()`.
When the function `main()` is executed:
When the nested block starting at line 4 is executed:
Pointer `parameterPointer` is declared on line 5.
But, it has not been assigned a value yet
Pointer `parameterPointer` is allocated an anonymous variable on line 6.
The anonymous variable itself is initialized to 60.9
The value of `anonymous` variable pointed to by `parameterPointer`,
i.e., 60.9 is printed on line 7
The nested block is exited on line 8
Pointer `parameterPointer` goes out of scope and is deallocated on line 8
When `parameterPointer` goes out of scope, the anonymous variable pointed to by it is not yet deallocated.
Since no other pointer is pointing to it, it is a lost object.

Generated	Correct	Partial	Wrong	Missed	Not Attempted	Learning Objective
0	0	0	0	0	0	Pointer.Declaration.Correct
0	0	0	0	0	0	Pointer.Declaration.scope.Correct
0	0	0	0	0	0	Pointer.Declaration.scope.Accessing po
0	0	0	0	0	0	Pointer.Allocation.Correct
0	0	0	0	0	0	Pointer.Initialization.Correct
0	0	0	0	0	0	Pointer.Assignment.Correct

Time Elapsed: 0:30 Remaining: 29:30

Fig. 1. Screen shot of a tutor on C++ pointers

Our evaluations have shown that students do learn from these tutors. We wanted to find out whether the improvement in learning affected by the tutors resulted in improvement in the self-confidence of the students. In order to answer this question, we conducted studies during the evaluation of four tutors in fall 2004 and spring 2005 – two on introductory programming, and two on intermediate programming topics. The tutors that we evaluated were:

1. A tutor on arithmetic operators that covered all the arithmetic operators, precedence, associativity, integer division, coercion, and inapplicability of real operands to the remainder operator in C++.
2. A tutor on relational operators that covered all the relational operators, their precedence and associativity with respect to arithmetic operators, the implicit treatment of Boolean constants as numbers in C++, and the inadvisability of using equality operator to compare real operands.
3. A tutor on `while` loops that covered the following cases: relational conditions, condition variables modified within the loop, updating the loop variable before the action, empty loop, pass-through loop, nested independent loops and nested dependent loops.
4. A tutor on `for` loops that covered the following cases: relational conditions, simple/compound statement as the body of the loop, updating the loop variable before the action, empty loop, pass-through loop, nested independent loops and nested dependent loops.

We evaluated all four tutors in the *Computer Science I* course offered by another instructor (not the author) at our institution in fall 2004 and spring 2005. The instructor assigned the tutors roughly one every two weeks, over the course of a semester. Students were given up to two weeks to work with each tutor. The students worked with the tutors asynchronously after class. Using the tutors was mandatory for course completion. But, the tutors did not contribute to the final grade in the course. This design ensured that students took the tutor sessions seriously, but not seriously enough to cheat.

3 The Method

When a student used a tutor, the student was run through a pre-test-practice-post-test protocol. The tutors themselves administered the protocol and stepped through the following stages:

1. Registration – the students were asked to enter their name, status, and demographic information;
2. Pre-Survey – the students were asked to rate their knowledge about the topic of the tutor on a 5-point Likert scale;
3. Trial Run – the students were shown instructions for using the tutor, and given an opportunity to use the tutor on a few sample problems not relevant to the current topic. This helped the students get used to the interface of the tutor.
4. Pre-test – the students were asked to solve a pre-determined set of problems, and were given limited time to complete the test. The students did not receive any feedback during the test.

5. Practice – based on the performance of the students on the pre-test, the tutor identified the concepts on which the students needed additional practice. It presented problems on only these concepts. Students were provided detailed feedback during this stage. The practice session lasted a fixed amount of time.
6. Feedback – the students were asked to enter feedback about the usability, learnability and usefulness of the tutor that they used for practice.
7. Post-test – the students were asked to solve problems based on the same sequence of problem templates as on the pre-test. Since the tutor generates problems as random instances of problem templates, the problems on the post-test were un-identical to, but closely matched with the problems on the pre-test for their content and the order in which they appeared. The time allowed for the post-test was the same as that for the pre-test.
8. Post-Survey - the students were asked to again rate their knowledge about the topic of the tutor on a 5-point Likert scale. The same questions were asked on the post-survey as on the pre-survey.

Since the students used the tutor after class, no instructor was present during the tutoring session. Students could have consulted their textbook or other reference material during the evaluation. But, since the tutor administered the above stages back to back, was timed and could not be suspended, students did not have much time to consult external sources.

We used the difference between pre-test and post-test scores as a measure of the change in the learning of the student. We compared the student responses on the pre and post-survey to assess their level of self-confidence in the topic before and after using the tutor. Since the pre-survey and post-survey were conducted immediately before and after using the software tutor, and students had little time in between to engage in any other activity that may have influenced their self-confidence, any difference in the pre and post-survey scores may be attributed to the use of the software tutors.

4 The Results

As mentioned earlier, we conducted this study during the evaluation of four tutors in fall 2004 and spring 2005 – tutors on arithmetic operators, relational operators, `while` loops and `for` loops. In the next four sections, we will list the types of questions we asked on the pre and post-survey of each tutor, and discuss the results of analyzing the collected responses.

4.1 Tutor on Arithmetic Operators

The pre and post-survey consisted of the following questions:

1. How well do you know Arithmetic Operators (+,-,*,/,%)?
2. How well do you know operator precedence concepts?
3. How well do you know operator associativity concepts?

Students answered these questions on a five-point Likert scale of "Very Well" (1), "Well" (2), "Average" (3), "Not Well" (4) and "Not at all" (5).

In Table 1, we have listed the average score of the class on the pre-test and post-test, and the paired t-test 2-tailed p-value of the difference between pre and post-test score. Next, we have listed the average of the class responses aggregated over all three questions on the pre- and post-survey, the average for each of the three questions, followed by paired t-test 2-tailed p-value of the difference between pre and post-survey response.

The self-confidence of the class improved from the pre-survey to the post-survey on the aggregate of all three questions (6.73 to 5.41), and the improvement was statistically significant. This warranted post-hoc analysis which showed that the improvement was significant on questions 2 and 3. The learning of the class improved from the pre-test (3.61 points per problem) to the post-test (4.31 points per problem) and the improvement was statistically significant. However, there was no correlation between the improvement in learning and the improvement in the self-confidence of the students (correlation coefficient r = 0.44376).

Fall 2004 (N=22)	Pre	Post	Stat. Significance
Average Score	3.61	4.31	0.00011
Confidence	6.73	5.41	0.01221
Q1	2.05	1.86	0.25748
Q2	2.32	1.64	0.00177
Q3	2.36	1.91	0.03794

Table 1: Change in Scores and Self-Confidence - Arithmetic Operators - Fall 2004

Spring 2005 (N=30)	Pre	Post	Stat. Significance
Percentage Score	0.49	0.67	0.00003
Confidence	8.33	6.87	0.00585
Q1	2.20	2.27	0.69026
Q2	2.93	2.27	0.01119
Q3	3.20	2.33	0.00005

Table 2: Change in Scores and Self-Confidence - Arithmetic Operators – Spring 2005

Table 2 shows the corresponding figures from the spring 2005 evaluation of the tutor. In spring 2005, we considered the average percentage correctness rather than the average score per problem. Once again, although there was a statistically significant improvement in the learning, and the self-confidence of the students in aggregate as well as on questions 2 and 3, there was no correlation between the improvement in learning and the improvement in the self-confidence (correlation coefficient r = 0.1505).

4.2 Tutor on Relational Operators

The pre and post-survey consisted of the following questions:

1. How well do you know Relational Operators (<,<=,>,>=,==,!=)?
2. How well do you know operator precedence concepts?
3. How well do you know operator associativity concepts?

Table 3 lists the figures from the fall 2004 evaluation of the tutor on relational operators. Although there was a statistically significant improvement in the learning as well as self-confidence, there was no correlation between the two (correlation coefficient r = -0.37738).

Fall 2004 (N=27)	Pre	Post	Stat. Significance
Average Score	3.45	4.20	0.00012
Confidence	6.70	5.11	0.00021
Q1	2.19	1.63	0.00129
Q2	2.22	1.74	0.00265
Q3	2.30	1.74	0.00012

Table 3: Change in Score and Self-Confidence - Relational Operators – Fall 2004

Table 4 lists the figures from spring 2005 evaluation of the tutor. Once again, the improvements in learning and self-confidence were statistically significant, but there was no correlation between the two (correlation coefficient r = 0.2853).

Spring 2005 (N=30)	Pre	Post	Stat. Significance
Percentage Score	0.74	0.79	0.01883
Confidence	7.60	5.53	0.00000
Q1	2.27	1.73	0.00040
Q2	2.63	1.90	0.00000
Q3	2.70	1.90	0.00000

Table 4: Change in Score and Self-Confidence - Relational Operators – Spring 2005

The improvement of self-confidence on question 1 was never statistically significant for arithmetic tutors, but always so for relational tutors. The reason for this could be that relational operators are easier than arithmetic operators, and the 45 minutes time-limit allowed for the use of the tutors is adequate for relational expressions, but not so for arithmetic expressions.

Questions 2 and 3 on arithmetic and relational tutor surveys were the same, and pertained to cross-cutting issues – precedence and associativity. It is remarkable that the improvement in confidence on these questions was always statistically significant.

4.3 Tutor on Logic-Controlled Loops

The pre and post-survey consisted of the following questions:

1. How well do you know loops?

2. How well do you know WHILE loops?
3. How well do you know NESTED while loops?

Table 5 lists the figures from fall 2004 evaluation of the tutor. Once again, there was no correlation between the improvement in learning and the improvement in the self-confidence of the students (correlation coefficient r = -0.12841). We repeated this experiment in spring 2005, but the sample size was too small to analyze.

Fall 2004 (N=23)	Pre	Post	Stat. Significance
Percentage Score	0.69	0.81	0.02840
Confidence	7.29	5.86	0.00037
Q1	2.38	1.95	0.00095
Q2	2.38	1.85	0.00209
Q3	2.52	2.04	0.00167

Table 5: Change in Score and Self-Confidence - while Loops – Fall 2004

4.4 Tutor on Counter-Controlled Loops

The pre and post-survey consisted of the following questions:

1. How well do you know loops?
2. How well do you know FOR loops?
3. How well do you know NESTED for loops?

In fall 2004, the change in the aggregate of self-confidence questions was not statistically significant, so we did not perform post-hoc analysis. Table 6 lists the figures from spring 2005 evaluation. Once again, there was no correlation between the improvement in learning and the improvement in the self-confidence of the students (correlation coefficient r = -0.44487 in spring 2005).

Spring 2005 (N=23)	Pre	Post	Stat. Significance
Percentage Score	0.45	0.56	0.07538
Confidence	9.70	8.04	0.00222
Q1	3.09	2.70	0.01642
Q2	3.30	2.65	0.00292
Q3	3.30	2.70	0.00520

Table 6: Change in Score and Self-Confidence - for Loops – Spring 2005

5 Discussion

We can draw the following conclusions from the above data:

1. The self-confidence increases with the use of software tutors.
2. There is no correlation between the change in learning and the change in the self-confidence of the users of software tutors.

In order to explain why is there no correlation between the change in learning and the change in self-confidence, we need to consider two cases:

- Students whose learning improved, but who did not report an increase in self-confidence. The software tutors help these students learn, but do not promote self-confidence in the students. This may point to the need for the software tutors to provide more affective feedback during the learning process to encourage and reward learning.
- Students whose learning did not improve, who reported an increase in self-confidence any way. These students may be mistaking increased familiarity for improved learning. Students can grow more familiar with a topic without learning it, if they engage in shallow processing of knowledge [14]. However, our software tutors were designed to promote deep processing of knowledge by getting the students to solve problems. This might suggest that students in this category are not engaged in the problem-solving process. Again, this points to the need for the software tutors to provide more affective feedback to motivate and focus the students.

We plan to incorporate affective feedback into our software tutors and re-evaluate the correlation between learning and self-confidence. Finally, literature in Psychology suggests that we should not expect a correlation between learning and the belief that learning has occurred, because testimonials can mislead [12]. Our current results support this finding, counter-intuitive as it is.

We introduced two new questions during the spring 2005 evaluation of counter-controlled loops:

1. How well do you know variables?
2. How well do you know recursion?

These questions were meant to serve as control questions – by the time students attempt problems on `for` loops, they should be very familiar with the concept of variables and were not expected to report any increase in self-confidence. Recursion is an advanced topic that students would not yet have seen; they were not expected to report any change in the self-confidence on recursion. Yet, much to our surprise, the self-confidence of the students improved from 2.96 to 2.57 on variables and 3.78 to 2.78 on recursion, both the improvements being statistically significant. It is unlikely that students intentionally inflated their post-survey responses, since the pre-survey and post-survey were separated by several stages (pre-test, practice, feedback and post-test), and it is unlikely that students could remember their pre-survey responses until the post-survey. This could be Hawthorne effect. We plan to investigate.

We plan to continue to evaluate our tutors for their effect on the self-confidence of the students. This includes evaluating additional tutors as well as evaluating tutors in additional classes. The tutors are currently available for academic use free of charge, and may be accessed at www.problets.org.

Acknowledgments

Partial support for this work was provided by the National Science Foundation's Educational Innovation Program under grant CNS-0426021.

References

[1] Dancik, G. and Kumar, A.N., A Tutor for Counter-Controlled Loop Concepts and Its Evaluation, Proceedings of Frontiers in Education Conference (FIE 2003), Boulder, CO, 11/5-8/2003, Session T3C.

[2] Fernandes, E. and Kumar, A.: A Tutor on Scope for the Programming Languages Course, Proceedings of 35th SIGCSE Technical Symposium, Norfolk, VA, (March 2004), 90-95.

[3] Kostadinov, R. and Kumar, A.N. A Tutor for Learning Encapsulation in C++ Classes, to be presented at ED-MEDIA 2003 World Conference on Educational Multimedia, Hypermedia and Telecommunications}, Honolulu, HI, 6/23-28/2003.

[4] Krishna, A., and Kumar A.: A Problem Generator to Learn Expression Evaluation in CS I and Its Effectiveness, The Journal of Computing in Small Colleges, Vol 16, No. 4, (May 2001), 34-43.

[5] Kumar, A.N., Explanation of step-by-step execution as feedback for problems on program analysis, and its generation in model-based problem-solving tutors, Technology, Instruction, Cognition and Learning (TICL) Journal, to appear

[6] Kumar, A. Results from the Evaluation of the Effectiveness of an Online Tutor on Expression Evaluation, Proceedings of 36th SIGCSE Technical Symposium, St. Louis, MO, (February 2005).

[7] Kumar, A.N., Using Online Tutors for Learning - What do Students Think?', Proceedings of Frontiers in Education Conference (FIE 2004), Savannah, GA, 10/20-23/2004, Session T3C.

[8] Kumar, A.N., Learning Programming by Solving Problems, in Informatics Curricula and Teaching Methods, L. Cassel and R.A. Reis ed., Kluwer Academic Publishers, Norwell, MA, 2003, 29-39.

[9] Kumar, A.N., A Tutor for Using Dynamic Memory in C++, Proceedings of 2002 Frontiers in Education Conference (FIE 2002), Boston, MA, 11/6-9/2002, Session T4G.

[10] Kumar A. Learning the Interaction between Pointers and Scope in C++, Proceedings of The Sixth Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE '01), Canterbury, UK, June 2001, 45-48.

[11] Kumar A.N.: Dynamically Generating Problems on Static Scope, Proceedings of The Fifth Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2000), Helsinki, Finland, (July 2000), 9-12.

[12] McCord, J. A thirty-year follow-up on treatment effects. American Psychologist. Vol 33. 284-289. 1978.

[13] Shah, H. and Kumar, A.N., A Tutoring System for Parameter Passing in Programming Languages, Proceedings of The Seventh Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2002), Aarhus, Denmark, (June 2002).

[14] Willingham, D.T. Why Students Think They Understand - When They Don't, American Educator, Winter 2003-04.