The Impact of Instructor Initiative on Student Learning: A Tutoring Study

Kristy Elizabeth Boyer[a]*, Robert Phillips[ab],
Michael D. Wallis[ab], Mladen A. Vouk[a], James C. Lester[a]
[a] Department of Computer Science, North Carolina State University, Raleigh, NC, USA
[b]Applied Research Associates, Inc. Raleigh, NC, USA
*keboyer@ncsu.edu

ABSTRACT
In the quest to find instructional approaches that benefit student learning, engagement, and retention, evidence suggests providing students with hands-on practice is a worthwhile use of class time. This paper presents results from an exploratory study of two different instructional approaches that were encountered in a study of experienced human tutors working with novice computing students engaged in a programming exercise. No difference in average learning gains was found between a *moderate* approach, in which students were given control of problem solving nearly half the time, and a *proactive* approach in which the tutor took initiative nearly three-fourths of the time. Implications of this finding for fine-grained instructional strategy, as well as for broader classroom management decisions, are discussed. This paper also makes the case for the value of one-on-one tutoring studies as an exploratory research methodology for the comparative evaluation of computer science teaching strategies.

Categories and Subject Descriptors
K.3.2 [**Computer and Information Science Education**]: *Computer Science Education*

General Terms
Design, Human Factors

Keywords
Computing education research, tutoring, research study, active learning, problem-solving, teaching strategies

1. INTRODUCTION
Computing educators have a plethora of instructional approaches at their disposal. For a given course, they may choose among textbooks, presentation formats, programming languages, and communication platforms. Beyond these instructional choices that often persist throughout the duration of a course lie the finer-grained decisions of teaching. Such decisions include the particular approach (e.g., lecture, discovery learning) that will be used to teach course material.

SIGCSE'09, March 3-7, 2009, Chattanooga, Tennessee, USA.

It is helpful to have empirical results that highlight the differential effects of specific strategies in particular situations. Because it has been established that active learning is a valuable classroom approach that can enhance student learning and motivation [6], one important teaching scenario for computing educators is hands-on practice time provided to students in a classroom setting. Evidence suggests that introducing an active component to a computing course may benefit students, particularly with regard to engagement and retention [3].

As with most instructional approaches, in-class practice activities involve pedagogical decisions such as when to intervene for students who appear to be struggling, and how to choose the specificity and elaborateness of answers to student questions. This paper presents the results of an empirical study that explores teaching strategies for helping students solve introductory programming problems.

One-on-one instruction in computer science has been recognized as a technique in which effective teaching strategies can be applied [10]. This paper reports on a study that investigates the impact of two different instructional approaches that were employed in a controlled study of experienced human tutors working with novice computer science students who were engaged in a programming exercise. The two instructional approaches differed with respect to the level of control and direction, or *initiative* [5], that the tutor exerted during the problem-solving activities. It was found that for the two levels of instructor initiative studied here (73% and 55% instructor initiative, corresponding to 27% and 45% student initiative, respectively), there was no significant difference in student learning gains. In addition to contributing to our understanding of instructional techniques for guiding problem solving, the findings highlight important future work on classroom management techniques for facilitating active learning in computer science.

2. TUTORING STUDY
This paper describes an exploratory research study conducted during a university CS1 course. In the study, each student worked on a programming exercise while interacting with a dedicated human tutor via remote collaboration software.

2.1 Participants
Study participants consisted of 61 students enrolled in a university CS1 course. Students were obtained on a volunteer basis through in-person visits by a researcher to course sections taught by three different instructors. Students earned a small amount of extra course credit for participating in the study. An alternate

assignment for extra credit was offered to students who did not participate in a tutoring session.

2.2 Programming Exercise

Each student in the experiment completed a programming exercise that focused on one-dimensional arrays, *for* loop constructs, and parameter passing. These were all timely concepts for participants with respect to upcoming programming projects and exams in the class. The programming problem was scaffolded with a fully-implemented graphical component that allowed students to test and see the results of their work visually. The context of the problem, which consisted of analyzing emergency call response times for a county that was considering replacing their fleet of ambulances, was designed with social relevance in mind [7]. The material required for the programming exercise had been presented in the students' class lectures.

2.3 Treatment Groups

Students were assigned randomly to one of the two tutors. These tutors, one female graduate student and one male undergraduate student, were both experienced in tutoring students in introductory programming. The two tutors were chosen because of their demonstrated effectiveness in two prior pilot studies using numerous experienced tutors. No instructional strategies were prescribed for the tutors. As discussed in Section 3, analysis of the tutoring logs revealed the two tutors chose significantly different teaching strategies while working with the students.

2.4 Logistics

During the course of problem solving, the student constructed the solution in the programming interface while the tutor observed the student's actions remotely in real time. Tutors were not allowed to edit the student's programming code. The tutor and student engaged in dialogue through a textual chat message interface as part of a software tool designed to facilitate remote collaboration [2]. Tutors were not made aware of any student characteristics such as achievement level in the course, gender, or age. Reciprocally, no information about the tutor's identity was provided to the student.

Students completed a pre-survey and pretest upon arrival to the study, and were then shown a brief instructional video containing an orientation to the software. After beginning work on the programming problem, the students were allowed to work until completion except for one subject who was stopped due to time constraints. This subject was omitted from the analyses presented here, as were three subjects whose logs were incomplete due to technical difficulties. At the end of the tutoring session, students completed a posttest with items identical to those in the pretest.

Both the pretest and posttest consisted of free-response questions for which students wrote Java code using the same concepts that were relevant to the programming exercise. Time to complete this free-response test ranged from 8 to 20 minutes. The pretest and posttest scores are reported as percentages correct out of the total possible points on each test. It should be noted that although there was likely a test-retest effect present from administering the same test before and after the instruction, this effect is present in both groups and therefore does not have a bearing on the differential analysis presented here. [1]

3. ANALYSIS AND RESULTS

Over the course of the tutoring sessions, all programming actions and typed dialogue were logged to a database. Because no pedagogical strategies had been prescribed, a preliminary qualitative analysis of the logs was conducted to search for strategies that appeared to differ systematically between tutors. The qualitative analysis led to the hypothesis that the tutors differed significantly in the percentage of time the student was allowed to direct and control the problem-solving task. In dialogue analysis, control of the dialogue and problem-solving is referred to as *initiative* [5]. The dialogue data described here is of a *mixed-initiative* nature because both the student and the tutor are able to take and relinquish control during the tutoring session, within limits (i.e., the software interface permits the tutor only to view, not edit, the solution to the programming problem).

In order to explore whether there was a significant difference in the teaching strategies with respect to initiative, the analysis employs a standard *corpus annotation* approach: an annotation scheme was applied manually to the data, and quantitative methods were used to discover whether statistically significant differences were present in the resulting relative frequencies. The remainder of this section describes the initiative annotation tags, provides example dialogue excerpts, and presents statistical results.

3.1 Annotation for Tutor Approach

In order to annotate the sessions as to whether the tutor or the student was leading the problem solving at a given time, the analysis employed two tags: *Student-Initiative* and *Tutor-Initiative.* These tags were applied to groups of dialogue messages as well as student programming actions.

Student-Initiative Mode. In Student-Initiative mode the student maintains control and direction over the problem-solving effort. Student-Initiative mode is characterized by one or more of the following attributes: 1) the student states his/her plan and (optionally) asks the tutor for feedback, 2) the student reads the problem description or constructs a portion of the solution independently (as indicated by no dialogue exchanged while the student is conducting these problem-solving activities), or 3) the student asks content-based questions (e.g., "I should start this index at 0, right?") as opposed to content-free questions (e.g., "What do I do now?").

Tutor-Initiative Mode. In Tutor-Initiative mode the tutor directs the problem solving effort. Only the student can construct program code in the problem-solving window, but the tutor can control and direct the student's work through dialogue strategy choices. Characteristics of the Tutor-Initiative mode include: 1) the tutor offers unsolicited advice or correction, 2) the tutor lectures on a concept, 3) the tutor explicitly suggests the next step in problem solving, or 4) the tutor poses questions to the student.

To illustrate the different tutoring modes, consider two dialogue excerpts. The first excerpt, from the moderate tutor, illustrates

[1] Identical pretest and posttests were chosen over isomorphic versions to avoid a mismatch in difficulty that might have inflated or masked learning results.

Student-Initiative mode (Table 1). In this excerpt, the student asks a content-based question indicating he knows the problem lies in a *return* statement. The tutor provides an answer, the student acknowledges, and finally the student spends five minutes coding part of the problem solution. Lengthy periods of independent student work are common in Student-Initiative mode.

The second excerpt, from the proactive tutor, illustrates Tutor-Initiative mode (Table 2). In this excerpt, the tutor provides unsolicited advice and asks questions of the student. The student spends a brief time repairing the problem solution, and the tutor once more provides unsolicited feedback. As illustrated in this excerpt, brief periods of student work interspersed with frequent dialogue are common in Tutor-Initiative mode.

Table 1: Excerpt of *Student-Initiative* mode[2]

Student:	What am I not typing right in the return statement?
Tutor:	You only need to return the identifier.
Tutor:	In other words, you just need to return newtimes
Student:	Ok.
	[student works on solution independently for five minutes]

Table 2: Excerpt of *Tutor-Initiative* mode

Tutor:	Hmm, that doesn't look quite right.
Tutor:	Do you see the projected array output?
Student:	Yes.
Tutor:	It looks like it's only getting the first value…
Tutor:	So your loop must be stopping before it's done with its work.
Tutor:	Do you see what might be causing that?
	[…tutor-led conversation continues…]
Tutor:	But it's coming out 1.0 instead of 4.3.
Tutor:	Anything else look wrong on the graph, compared to the instructions?
Student:	The second bar is not right
Tutor:	I think fixing the length might be the only thing you need to change.
	[student works on solution for ten seconds]
Tutor:	Much better.
Student:	Yeah!!
Tutor:	=)

3.2 Proportion of Tutor-Led Problem Solving

A total of 61 tutoring sessions were conducted. Out of these, a subset of 30 (15 randomly selected from each treatment group) were annotated for initiative. From the time stamps logged on each textual dialogue message and problem-solving action, boundaries between Student-Initiative and Tutor-Initiative mode were identifiable as points in time. Total elapsed time in each mode was calculated for each session. This calculation revealed that the two tutors did in fact take significantly different levels of initiative. One tutor, whom we refer to as the *proactive* tutor, directed the problem solving an average of 73% of the time. The second tutor, referred to here as the *moderate* tutor, took initiative only 55% of the time on average (Figure 1). This difference in means is statistically significant with a two-sample *t*-test using pooled variance (p=0.029): the moderate tutor allowed students to take initiative significantly more often than the proactive tutor.

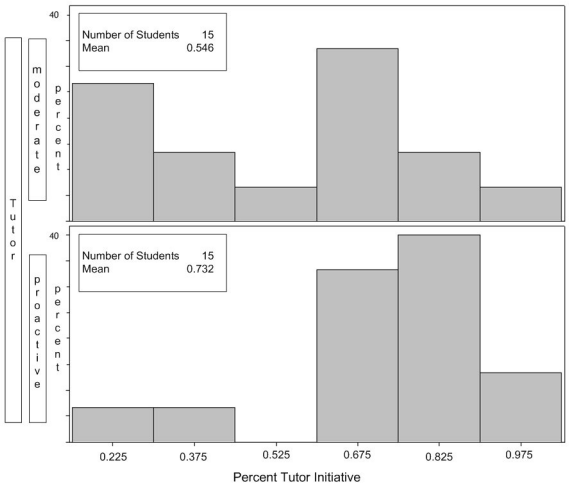

Figure 1: Comparison of Tutor Initiative

3.3 Random Distribution of Students

One possible explanation for the difference in tutor approach might be that there was a systematic difference in the groups of students assigned to each tutor. For instance, if the proactive tutor had been assigned weaker students than the moderate tutor, this could explain not only the need for more control on the part of the tutor, but also a lack of discrepancy in learning outcomes. However, there is no evidence of a systematic difference in the samples of students tutored. Initially, students were randomly assigned to one of the two tutors. To confirm that no systematic difference in student preparedness was created, we conducted a post hoc analysis of pretest scores (Figure 2). The mean pretest score for students assigned to the moderate tutor was 79.5%, compared with a mean pretest score of 78.9% for students assigned to the proactive tutor. A *t*-test with pooled variance indicates there is no evidence of a statistical difference in the pretest scores (p=0.930).

3.4 Equivalence of Learning Gains

Learning gain is calculated as the difference between posttest score and pretest score (in terms of percent correct). Despite a significant difference in the mean level of tutor initiative between treatment groups, there was no significant difference between the mean learning gains for the two groups (Figure 3). Students who worked with the moderate tutor had an average learning gain of 6.9 percentage points. Students who worked with the proactive tutor had an average 6.0 percentage point learning gain. This difference is not statistically significant (p=0.796).

[2] In both excerpts, capitalization and punctuation have been added to textual dialogue messages for readability.

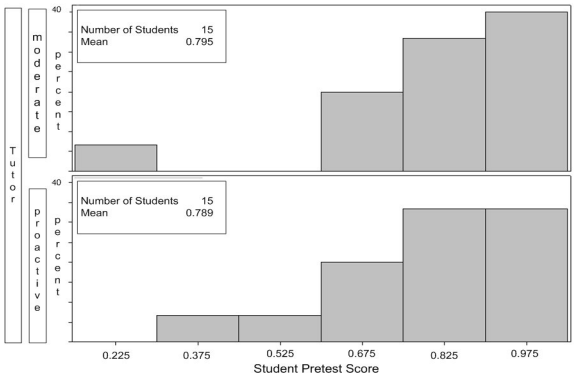

Figure 2: Comparison of Student Pretest Score by Tutor

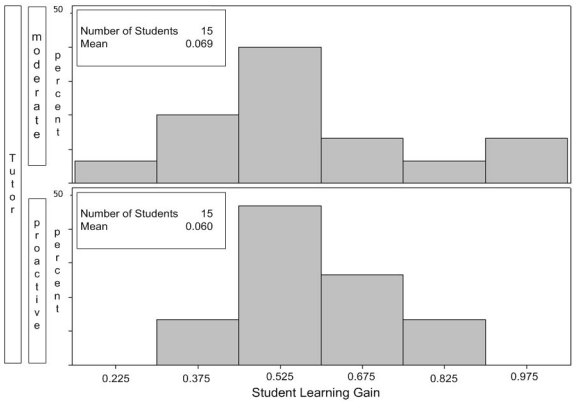

Figure 3: Student Learning Gains by Tutor

4. DISCUSSION

It was found that there was no significant difference in student learning gains between proactive instruction, in which the tutor retained control and direction of problem solving 73% of the time, compared to moderate instruction in which the tutor led the problem solving only 55% of the time. The findings of this exploratory study have important implications at both a fine-grained teaching-strategy level and a broader classroom-management level. This section discusses those implications and addresses limitations of the study.

4.1 Tutoring Methodology

While it is the case that most college and university-level computing education takes place in group instruction settings, studying one-on-one tutoring is a valuable research methodology for exploring the effects of different instructional strategies. First, tutoring studies permit experimental *control*: by collecting data involving one instructor and one student, investigators can achieve control beyond what is easily attained if data is collected at the classroom level only (e.g., when instructors vary their teaching methods between classes and then compare aggregate data by class to discern the differential impact of the teaching methods). Second, because tutor-student interactions can be readily recorded using basic logging techniques, tutoring studies facilitate data *capture*. Finally, because each learner is provided with what the tutor deems the most suitable instructional approach, one-on-one instruction permits greater *coverage* of teaching strategies. In addition, a greater number of different instructors can be observed teaching the same material compared to what would be feasible in classroom implementation. Particularly in exploratory studies where it may not be necessary to have classroom-sized samples of individual students for each tutor, a one-on-one research approach can reveal trends and patterns worthy of future investigation.

By aggregating tutoring data based on certain factors (e.g., instructional technique), we can discover differential implications of the factor levels for the entire sample of students, which can be extrapolated to the student population as a whole. This type of inference is used regularly in whole-class educational research, and many issues (e.g., representativeness of the student sample, hidden individual differences) are shared between whole-class and one-on-one research situations. In both cases, such confounding factors are important to address.

4.2 Threats to Validity

One artifact of the design of this exploratory study is that because different tutors were used, it is plausible that the difference in instructor initiative did have an effect on student learning, but that unseen differences between tutors offset the impact of the different levels of initiative. This potential threat to validity is necessary in such an exploratory study because had only a single tutor been utilized, the study could not have revealed the systematic differences in the individual tutors' natural strategies. A more controlled design can be implemented in subsequent confirmatory experiments.

A second possible confound pertains to making extrapolations to group teaching strategies based on tutoring data. One-on-one instruction provides highly contextualized, individualized assistance. This constant source of effective help is part of the power of human tutoring, and is at least partially responsible for the dramatic learning results achieved in some tutoring studies that compare one-on-one instruction to group instruction [1]. For this reason, among others, it is not reasonable to suggest replacing whole-class research studies with one-on-one tutoring studies. However, because of the control, capture, and coverage that are readily achievable in tutoring studies, we can use this approach to identify important patterns that warrant whole-class studies. The results reported here are an example of just such a pattern.

4.3 Open Questions

This study of initiative raises an important hypothesis worthy of future investigation. With the proactive tutor, the student was in control of the problem solving an average of 27% of the time. With the moderate tutor, the student directed the problem solving significantly more often: 45% of the time, with no significant increases in learning gains observed. This result, in which higher student initiative did not translate into higher learning gains, can be contextualized with other research in which student initiative is allowed to vary. For example, in the very different context of pair programming (e.g., [8, 9]), it has been found that when students

complete projects in pairs (presumably dividing the possible 100% initiative between both students since only one student is in control at a given time), learning gains over the semester are as good or better than students who completed the projects individually (close to 100% initiative on the part of the student). These findings suggest that the level of student initiative has an interaction effect with other aspects of the instructional setting, and that further investigation is warranted to fully understand these phenomena.

In a large tutoring study with the investigation of learning in multiple disciplines, results suggest the effectiveness of tutoring lies not solely with a tutor's skill in what to say and when, nor solely in the student's active construction of knowledge, but rather with the interaction between the two parties [4]. Interactivity as a catalyst for student learning has also been studied by proponents of active learning [6], who find a strictly lecture format (presumably close to 0% student initiative) is not as desirable as classroom approaches that do allow some level of student initiative. Our results suggest there may be a "plateau" effect of sorts, in which providing room for a modest level of student initiative could have appreciable benefits for student learning. This hypothesis can be further investigated using tutoring experiments as well as whole-class comparison studies.

A final noteworthy benefit of using one-on-one instruction as a research methodology for the comparative evaluation of teaching strategies is that tutoring is known to increase the skill of the tutors as well as the students [4]. Utilizing tutoring as a pedagogical training ground for new computer science teachers is an active area of application and research [10]. By capturing multiple tutoring sessions, it may be possible to investigate how the tutor's pedagogical style changes over time and the corresponding impact on student learning.

5. CONCLUSIONS & FUTURE WORK

Hands-on practice in class is a promising instructional approach in which empirical results such as those presented here can help the computing education community understand the impact of alternative instructional approaches. One-on-one human tutoring is a viable exploratory research approach for investigating teaching strategies because it readily lends itself to control of confounding factors, capture of the complete instructional interaction, and coverage of a variety of instructors and teaching strategies. In the study reported here, it was found that a moderate approach, in which the tutor holds the initiative just over half the time, yielded learning gains comparable to a proactive approach in which the tutor directed the problem solving nearly three-fourths of the time.

Conducting a confirmatory tutoring experiment, as well as whole-class research studies with prescribed levels of instructor initiative, are promising directions for future work. These investigations can shed light on the fine-grained cognitive and affective processes at work as students acquire computing concepts and skills.

6. ACKNOWLEDGMENTS

Thanks to Carolyn Miller for valuable feedback. This work is supported in part by the NC State University Department of Computer Science along with the National Science Foundation through Grants REC-0632450 and IIS-0812291, a Graduate Research Fellowship, and the STARS Alliance Grant CNS-0540523. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the participants, and do not necessarily represent the official views, opinions, or policy of the National Science Foundation.

7. REFERENCES

[1] Bloom, B. S. The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher, 13*, 6 (1984), 4-16.

[2] Boyer, K. E., Dwight, A. A., Fondren, R. T., Vouk, M. A. and Lester, J. C. A development environment for distributed synchronous collaborative programming. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education.* (Madrid, Spain, June 30 - July 2, 2008). ACM Press New York, NY, 2008, 158-162.

[3] Boyer, K. E., Dwight, R. S., Miller, C. S., Raubenheimer, C. D., Stallmann, M. F. and Vouk, M. A. A case for smaller class size with integrated lab for introductory computer science. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education.* (Covington, Kentucky, March 7-10, 2007). ACM Press New York, NY, 2007, 341-345.

[4] Chi, M. T. H., Siler, S. A., Jeong, H., Yamauchi, T. and Hausmann, R. G. Learning from human tutoring. *Cognitive Science, 25*, 4 (2001), 471-533.

[5] Core, M. G., Moore, J. D. and Zinn, C. The role of initiative in tutorial dialogue. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics.* (Budapest, Hungary). Association for Computational Linguistics, Morristown, NJ, 2003, 67-74.

[6] Felder, R. M. and Brent, R. Learning by Doing. *Chemical Engineering Education, 37*, 4 (2003), 282-283.

[7] Layman, L., Williams, L. and Slaten, K. Note to self: make assignments meaningful. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education.* (Covington, Kentucky, March 7-10, 2007). ACM Press New York, NY, 2007, 459-463.

[8] McDowell, C., Werner, L., Bullock, H. and Fernald, J. The effects of pair programming on performance in an introductory programming course. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education.* (Covington, KY). 2002, 38-42.

[9] Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C. and Balik, S. Improving the CS1 experience with pair programming. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education.* (Reno, Nevada, February 19-23, 2003). ACM Press, New York, NY, 2003, 359-362.

[10] Ragonis, N. and Hazzan, O. Tutoring model for promoting teaching skills of computer science prospective teachers. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education.* (Madrid, Spain, June 30 - July 2, 2008). ACM Press New York, NY, 2008, 276-280.