A Study to Identify Predictors of Achievement in an Introductory Computer Science Course

Sandra Katz
Learning Research and Development Center, University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260
1-412-624-7054
katz+@pitt.edu

John Aronis
Computer Science Dept., University of Pittsburgh
6211 Sennett Square
Pittsburgh, PA 15260
1-412-624-9185
aronis@cs.pitt.edu

David Allbritton
Dept. of Psychology, DePaul University
2219 N. Kenmore Ave.
Chicago, IL 60614
1-773-325-4799
dallbrit@depaul.edu

Christine Wilson
Learning Research and Development Center, University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260
1-412-624-9583
clwilson@pitt.edu

Mary Lou Soffa
Computer Science Dept., University of Pittsburgh
6401 Sennett Square
Pittsburgh, PA 15260
1-412-624-8425
soffa@cs.pitt.edu

ABSTRACT

In the study reported on here, 65 prospective computer or information science majors (47 male, 18 female) worked through a tutorial on the basics of Perl. All actions were recorded and time-stamped, allowing us to investigate the relationship between six factors that we believed would predict performance in an introductory computer science (CS) course (as measured by course grade) and how much students would learn from the tutorial (as measured by gain score from pre-test to post-test). These factors are: preparation (SAT score, number of previous CS courses taken, and pre-test score), time spent on the tutorial as a whole and on individual sections, amount and type of experimentation, programming accuracy and/or proficiency, approach to materials that involve mathematical formalisms, and approach to learning highly unfamiliar material (string manipulation procedures). Gender differences with respect to these factors were also investigated.

Predictors of grade and gain score included SAT score, pre-test score (negatively correlated with gain), time (negatively correlated with gain and grade), and various measures of programming accuracy and/or proficiency—for example, the total number of program runs that contained errors (negatively correlated with grade and gain). Several measures of experimentation predicted gain score. Experimentation also predicted grade, but only as applied to the least familiar tutorial material. Although experimentation was practiced throughout the tutorial by both sexes, male and female students differed with respect to the types of tutorial topics and tasks they experimented with and the degree of experimentation—for example, male students were more likely to write programs not suggested by the tutorial. These findings suggest that a tutorial such as the one used in this study could serve as an instrument to identify students who are likely to succeed (or not) in an introductory CS course, and that instructional interventions to promote achievement should encourage experimentation, reflection on the results of experiments, care and accuracy.

SIGMIS Conference '03, April 10-12, 2003, Philadelphia, Pennsylvania.

Keywords

Computer science instruction, programming instruction, gender and computer science.

1. INTRODUCTION

Since satisfactory performance is a requirement for retention in undergraduate computer science programs, many studies have been conducted to identify factors that predict achievement in CS courses. A complex array of experiential, affective, personality, socio-cultural, and cognitive factors have been shown to predict achievement—for example, simply owning a computer (e.g., [8], [12]); using a computer in pre-college computing classes (e.g., [5]); prior programming experience (e.g., [6]); confidence, intrinsic motivation, and having clear career goals (e.g., [2], [4], [9], [11], [13]); and various aptitudes, such as math ability, spatial reasoning ability, verbal reasoning ability, and Piagetian formal operations (e.g., [1], [3], [4], [7], [14]).

The research discussed in this paper investigates a piece of the "achievement and retention puzzle" that has received very little attention to date: how learning strategies and behaviors affect performance in undergraduate computer science programs. Since

programming is one of the first skills that computer science students learn, and a stumbling block for many, we focus on students' strategies for learning how to program. Research by Recker and Pirolli [10] suggests that learning strategies can have a strong impact on programming skill. In particular, students who approached learning materials in a reflective manner—e.g., by elaborating on example programs—outperformed less reflective students on laboratory programming tasks.

To get a firsthand look at students' approach to learning a new programming language, we developed a laboratory study in which students work through a tutorial on the basics of Perl.[19] By logging all actions taken by each student, we were able to investigate correlations between six learning factors and two dependent variables representative of achievement—course performance (as measured by final grade), and how much students learned from the tutorial (as measured by gain score from pre-test to post-test). These factors are: preparation (SAT score, number of previous CS courses taken, and pre-test score), time spent on the tutorial as a whole and on individual sections, amount and type of experimentation, programming accuracy and/or proficiency, approach to programming tasks that involve mathematical formalisms, and approach to learning new material (pattern matching). (See Table 1.) Gender differences with respect to these factors were also investigated.

2. METHODS

2.1 Materials

Five topics are covered in the Perl programming tutorial, with one brief chapter per topic: input/output basics, arithmetic expressions, conditional execution, while loops, and pattern matching (regular expressions). Each chapter consists of explanatory text, sample programs, and recommended practice exercises.

2.2 Subjects

65 students (47 males, 18 females) participated in the study. 56 students (43 males, 13 females) were prospective computer science students at the University of Pittsburgh, enrolled in an introductory CS course. CS students were recruited from eight sections of the course, spanning three academic semesters. All eight sections were taught by the same instructor. Nine students (4 males, 5 females) were prospective information science (IS) majors, recruited from one section of an introductory IS course at the same university. Eighteen of the 65 students were Black (14 males, 4 females); 47 were White or Asian (33 males, 14 females).[20] Students were paid a nominal amount for their participation.

2.3 Procedure

Five to ten students participated in each experimental session, which took approximately four hours. In each session, we first administered a pre-test to measure students' ability to program in Perl. After the pre-test, students were told that they would receive a bonus payment if they met a criterion score on the post-test; this was done to motivate students to work hard on the tutorial. Students then went through a step-by-step lesson on how to use the programming interface. This lesson was done as a group. Students then worked through the tutorial individually, at their own pace, so that we could track the time that they spent on each chapter, example and exercise. Finally, students took the post-test, which was identical to the pre-test.

2.4 Data Analysis

The two dependent variables for achievement in this study are pre-test to post-test gain score and course grade. Representative measures for the six factors we considered as possible predictors of achievement are shown in Table 1. Additional measures include sub-categories and aggregations of those shown in Table 1—for example, number of runs of practice exercises involving pattern matching (examples excluded), number of runs of correctly modified examples and practice exercises, respectively.

3. RESULTS AND DISCUSSION

We predicted that students who would do well in the course and earn high pre-test to post-test gain scores would have the following characteristics, relative to less successful students:

- Be better prepared, with respect to previous programming courses taken and aptitude (SAT scores)
- Spend less time on the tutorial as a whole and on individual chapters
- Write more experimental (modified) programs, and run tutorial-provided and self-generated programs on various inputs
- Exhibit better coding accuracy and/or proficiency—that is, have higher correct run/total run ratios
- Experiment more with relatively challenging tutorial material involving mathematical functions and pattern matching, and do so more accurately

Findings for gender comparisons and achievement are reported below. All findings are significant at the .05 level or less, except where otherwise noted.

3.1 Gender Comparisons

Male students had more prior programming experience than female students (t (59) = 232). Both men and women showed evidence of experimenting, though on different types of material. Men spent more time working on self-designed practice exercises ("free form" experimentation; t (46) = 1.97, p = .06). Whereas men had more correct runs of modified examples (t (62) = 2.94), women had more correct runs of modified practice exercise programs (t 63) = 2.18). Whereas men had more correct runs of modified pattern-matching examples (t (61) = 2.49), women had more correct runs of modified examples and exercises involving mathematical functions (t (63) = 2.04). Thus, both male and female students experimented on the more challenging tutorial material, though on different types of challenging material. Further research is needed to determine whether a tendency to experiment with certain types of material is indicative of interest, difficulty (i.e., an attempt to gain proficiency on topics one is weak on), or other factors.

[19] We eliminated students who said that they had prior experience with Perl or who scored high on the pre-test.

[20] Analyses based on race are in progress and will not be presented in this paper.

Table 1: Representative Measures of Factors Considered as Possible Predictors of Achievement (Course Grade and Gain Score)

Preparation	• Number of previous programming courses taken • SAT score (math, verbal, and total) • Pre-test score
Time	• Total time spent on the tutorial (5 chapters) • Time spent writing code in all 5 chapters (excludes reading time, short breaks between chapters, etc.) • Time spent on each tutorial chapter • Time spent only on writing code in each chapter
Experimentation—type and degree (shown in increasing order)	**Experimenting with Tutorial-provided Examples** • Number of runs of sample code on various inputs[21] • Number of attempted modifications of sample programs—series of runs that worked towards a single, functional change—whether or not the program ultimately compiled and was free of logical errors • Number of runs of correctly modified examples **Experimenting with Recommended Practice Exercises** • Number of runs attempting recommended practice exercises—whether or not they compiled and were free of logical errors • Number of runs of correct practice exercise programs • Number of attempted modifications of practice programs—series of runs that worked towards a single, functional change—whether or not they ultimately compiled and were free of logical errors • Number of runs of correctly modified practice programs **Free-form Experimentation** • Number of student-designed practice exercises—programs not tied to any tutorial example or exercise
Accuracy and/or coding proficiency[22]	• Number of correct runs—that is, runs free of syntactic and logical errors • Number of runs with errors • Percent of correct runs (correct runs/total runs)
Approach to examples and exercises involving mathematical functions (ease with formalism)[23]	**Experimentation with Mathematical Material** • Number of runs of examples and practice exercises involving mathematical functions—whether or not they ultimately compiled and were free of logical errors • Number of runs of correct programs involving mathematical functions • Number of runs of modified programs involving mathematical functions—whether or not they ultimately compiled and were free of logical errors **Accuracy with Mathematical Material**

[21] Our logs do not reveal the inputs that students used to run programs. We assume that if students ran the same, functioning program more than once, they used different inputs.

[22] The ratio of correct runs to total runs (correct plus incorrect runs) is an ambiguous measure. It might indicate how much the student had to struggle to get programs to work (coding proficiency). Alternatively or simultaneously, it might indicate how careful or accurate the student is.

[23] Three tutorial tasks involved mathematical functions—quadratic equations, geometric series, and harmonic series—one exercise and two examples.

	• Percent of correct program runs involving mathematical functions
Approach to new material (pattern matching)[24]	• Same measures of experimentation and accuracy as for ease with formalism

3.2 Predictors of Gain Score

Pre-test score correlated negatively with gain (r = -.46), whereas post-test score correlated positively with gain (r = .76). Thus, students who were initially the least proficient programmers, at least in Perl, seemed to benefit the most from the tutorial. Aptitude, according to all three SAT scores, also predicted gain scores (math, r = .33; verbal, r = .38; total, r = .25). Various measures of time were negatively correlated with gain (e.g., total time spent on the tutorial, r = -.28), suggesting that students who took longer to work through the tutorial as a whole and individual chapters were having difficulty grasping the material.

Accuracy, as measured by the total number of runs with errors, correlated negatively with gain (r = -.29). This finding was supported by analyses of students' approach to the least familiar material (on pattern matching). The percent of correct unmodified runs of pattern-matching examples and exercises predicted gain score (r = .25). When coupled with the negative correlation between pre-test score and gain, these findings suggest that accuracy was indicative of care rather than coding proficiency. It seems as though less skilled coders (in Perl) who were careful and strove for accuracy learned more than less careful students. Further research is needed to test this interpretation.

Students who experimented more, and to a higher degree, also got more out of the tutorial. Measures of experimentation that predicted gain include: the total number of runs on modified examples, whether or not correct (r = .28); the number of runs of correctly modified examples (r = .27); and the number of runs of "free form" experiments (r = .25).

3.3 Predictors of Course Grade

The nine IS students were excluded from the analysis of grade, since comparisons across the two disciplines would not have been meaningful. Gain score was uncorrelated with grade. This suggests that the overlap between the content of the tutorial and the course was relatively small. As with gain score, predictors of course grade included pre-test score (r = .28), post-test score (r = .37), aptitude (SAT total, r = .40), and various measures of time (negatively correlated; e.g., total time, r = -.30).

The same measure of accuracy that predicted gain score also predicted grade—the total number of runs with errors (r = -.38). This finding was supported by several other correlations between accuracy measures and grade: the percent of correct compilations, overall (r = .46); the percent of correct unmodified runs of examples and practice exercises (r = .42); and the percent of correct modified runs of examples and exercises (r = .31). Analyses of students' approach to new material are consistent with these findings for the tutorial as a whole. For example, the percent of correct modified and unmodified runs of pattern-matching exercises and examples correlated with grade (r = .35). As with gain score, these correlations between accuracy and grade are difficult to interpret and warrant further investigation. Which way does the relation point: Does accuracy indicate an acquired level of proficiency with coding that is *reflected* in final course grade, or do students earn high grades in this course partly *because* they are careful and accurate?

Experimentation also predicted grade, but only with respect to learning the least familiar material (as opposed to the tutorial as a whole). For example, the number of correct runs of pattern-matching examples and exercises correlated with grade (r = .30).

3.4 Regression Analysis of Achievement

We tested a regression model that included predictors related to the factors we thought would matter. Predictors and associated factors (shown in parentheses) include the following:

Preparation
- Pre-test score
- Overall SAT score
- Number of prior CS courses taken

Time
- Overall time spent on programming tasks—examples, exercises, and modifications of the same

Experimentation
- Number of functional modifications made to programs—examples and practice exercises
- Number of free-form experiments

Accuracy
- Overall number of runs with errors
- Percent of correct compilations

Ease with Formalism

[24] The final chapter covered pattern matching. Because most students in the experiment had taken at least one prior programming course, we expected them to have been exposed to the other tutorial topics, though in a different language than Perl. The pattern-matching chapter contained three examples and two practice exercises.

- Number of correct runs of programs involving mathematical functions (experimentation)

Approach to New Material

- Number of correct runs of pattern-matching examples and exercises (experimentation)
- Percent of correct runs of pattern-matching examples and exercises (accuracy)

These predictors were entered into step-wise regression analyses (probability required to enter = .075, probability to exclude = .10), with course grade and gain score as dependent variables. In the regression for course grade, only computer science students were included. The overall model predicted a significant proportion of the variance in grades: adjusted $R^2 = .36$, $F(3, 46) = 10.23$, $p < .001$. The only predictors retained in the model were SAT overall score (Beta = .31, $t = 2.67$, $p < .01$), pre-test score (Beta = .27, $t = 2.37$, $p < .05$), and the percent of compilations that were correct (Beta = .44, $t = 3.79$, $p < .001$). Thus the main finding was that coding accuracy and/or proficiency predicted course grades. The overall model also predicted a significant proportion of the variance in gains: adjusted $R^2 = .39$, $F(4, 55) = 10.37$, $p < .001$. The only predictors retained in the model were pre-test score (Beta = -.50, $t = -4.79$, $p < .001$), SAT score (Beta = .31, $t = 2.99$, $p < .01$), number of compilation errors (Beta = -.25, $t = -2.46$, $p < .05$), and number of free-form experimental runs (Beta = .20, $t = 1.99$, $p = .052$), partially supporting the finding from the grades analysis that accuracy predicts learning, and suggesting that experimentation predicts learning as well.

4. CONCLUSION

A tutorial such as the one used in this study could serve as an instrument to identify students early on who are likely to succeed (or not) in introductory courses that focus on programming concepts and skills. This information could be used by instructors to determine which students are likely to need extra help. In conjunction with prior research, this study also suggests what types of interventions are likely to be effective: exercises that promote coding accuracy and proficiency, and encourage experimentation and reflection on the results of experiments [10]. In future studies, we will develop and assess interventions that have these features, and attempt to specify more precisely what types of experimentation promote achievement in undergraduate programming courses.

5. ACKNOWLEDGMENTS

This research was supported by a grant from the National Science Foundation (grant number EIA 0089963). The data presented and views expressed are not necessarily endorsed by this agency. We thank the anonymous reviewers for helpful comments on a previous version of this paper.

6. REFERENCES

[1] Cafolla, R. Piagetian formal operations and other cognitive correlates of achievement in computer programming. Journal of Educational Technology Systems, 16(1), 1987, 45-55.

[2] Charlton, J.P., and Birkett, P.E. Psychological characteristics of students taking programming-oriented and applications-oriented computing courses. Journal of Educational Computing Research, 18(2), 1998, 163-182.

[3] Clement, C.A., Kurland, D.M., Mawby, R., and Pea, R.D. Analogical reasoning and computer programming. Journal of Educational Computing Research, 2(4), 1986, 473-86.

[4] Jagacinski, C.M., LeBold, W.K., and Salvendy, G. Gender differences in persistence in computer-related fields. Journal of Educational Computing Research, 4(2), 1988, 185-202.

[5] Kagan, D. Learning how to program or use computers: A review of six applied studies. Educational Technology, 28(3), March 1988, 49-51.

[6] Koohang, A.A., and Byrd, D.M. A study of selected variables and future study. Library and Information Science Research, 9(1), 1987, 214-288.

[7] Lai, S., and Repman, J. The effects of analogies and mathematics ability on students' programming learning using computer-based learning. International Journal of Instructional Media, 23(4), 1996, 355-364.

[8] Levin, T., and Gordon, C. Effect of gender and computer experience on attitudes towards computers. Journal of Educational Computing Research, 5(1), 1989, 69-88.

[9] Marcoulides, G.A. The relationship between computer anxiety and computer achievement. Journal of Educational Computing Research, 4(2), 1988, 151-158.

[10] Recker, M.M., and Pirolli, P. Student strategies for learning from a computational environment. In C. Frasson, G. Gauthier, and G.I. McCalla (Eds.), Intelligent tutoring systems (pp. 382-394). Berlin: Springer-Verlag, 1992.

[11] Reed, W.M., and Overbaugh, R. The effects of prior experience and instructional format on teacher education students' computer anxiety and performance. Computers in the schools, 9(2/3), 1993, 75-89.

[12] Taylor, H.G., and Mounfield, L.C. Exploring the relationship between prior computing experience and gender on success in college computer science. Journal of educational computing research, 11(4), 1994, 291-306.

[13] Volet, S.E., and Styles, I.M. Predictors of study management and performance on a first-year computer course: The significance of students' study goals and perceptions. Journal of Educational Computing Research, 8(4), 1992, 423-449.

[14] Webb, N.M. Microcomputer learning in small groups: Cognitive requirements and group processes. Journal of Educational Psychology, 76, 1984, 1076-1088.