Self-Efficacy and Mental Models in Learning to Program

Vennila Ramalingam
Media Lab
Massachusetts Institute of Technology
Cambridge, MA 02139 USA
vennila@media.mit.edu

Deborah LaBelle
Pennsylvania State University
Delaware County
Media, PA 19063 USA
+1 610 892 1343
dml19@psu.edu

Susan Wiedenbeck
College of IST
Drexel University
Philadelphia, PA 19104 USA
+1 215 895-2490
sw53@drexel.edu

ABSTRACT
Learning to program is a unique experience for each student, and it is not fully understood why one person in an introductory programming course learns to program better and more quickly than the next. Self-efficacy is an individual's judgment of his or her ability to perform a task within a specific domain [1]. A mental model is a person's internal (mental) representation of real world objects and systems [9]. Research has shown that high self-efficacy and a good mental model are important to knowledge acquisition and transfer. This research investigates the effects of students' self-efficacy and mental models of programming on learning to program. The results show that self-efficacy for programming is influenced by previous programming experience and increases as a student progresses through an introductory programming course. The results also show that the student's mental model of programming influences self-efficacy and that both the mental model and self-efficacy affect course performance.

Categories and Subject Descriptors
K.3.2 [**Programming Languages**]: Computer and Information Science Education – *Computer science education.*

General Terms
Experimentation

Keywords
Learning to program, self-efficacy, mental models

1. INTRODUCTION
The dropout and failure rates in introductory programming courses at the university level are evidence to the fact that learning to program is a difficult task. Some studies suggest that the dropout and failure rate is as high as 30 percent [6]. Success in the entry level programming course often determines whether the student will continue to pursue a computer related major. These individual decisions, in turn, affect the future of the computing profession. In spite of research on factors that influence the enrolment and success of novices in introductory programming, it is not well understood what makes programming an enjoyable and motivating experience for some, while others find it a painful struggle to complete the course.

The literature has suggested several factors that may influence novices' success in computing. These include previous computing experience [2, 3, 15], comfort level [18], computer playfulness during training [11], computer self-efficacy [7], mathematics or science background [3, 18], students' attributions of success [18], learning style [3, 16], and the student's mental model of programming [4, 14, 17].

Two key constructs in modern cognitive and social cognitive theory are mental models and self-efficacy. The goal of this research is to study self-efficacy and mental models of beginning level programmers, explore the relationship between these two very important concepts, and study their combined influence on the student's course performance.

ITICSE'04 June 28-30, 2004, Leeds, United Kingdom.

2. PRIOR RESEARCH
Bandura ([1], p. 391) defines self-efficacy as "people's judgments of their capabilities to organize and execute courses of action required to attain designated types of performance." Self-efficacy is important in learning because "competent functioning requires both skills and self-beliefs of efficacy to use them effectively" ([1], p. 391). In learning situations, self-efficacy influences the amount of effort expended, type of coping strategies adopted, use of cognitive strategies while solving problems, persistence in the face of failure, and performance outcomes [1]. Bandura posits that judgments of self-efficacy are based on four principal sources of information: the individual's performance attainments, vicarious experiences of observing the performance of others, verbal persuasion and associated types of social influences, and physiological states from which people partly judge their capableness, strength, and vulnerability. The most important of these is performance attainments.

It should be noted that self-efficacy is specific to a certain activity. Therefore, a person may have high self-efficacy in one domain, such as gardening, and low self-efficacy in another, such as computer programming. Nevertheless, transfer of self-efficacy

[1] Current address: 640 E. 11th Street #3, New York, NY, 10009. Telephone number: +1 212 979 7335

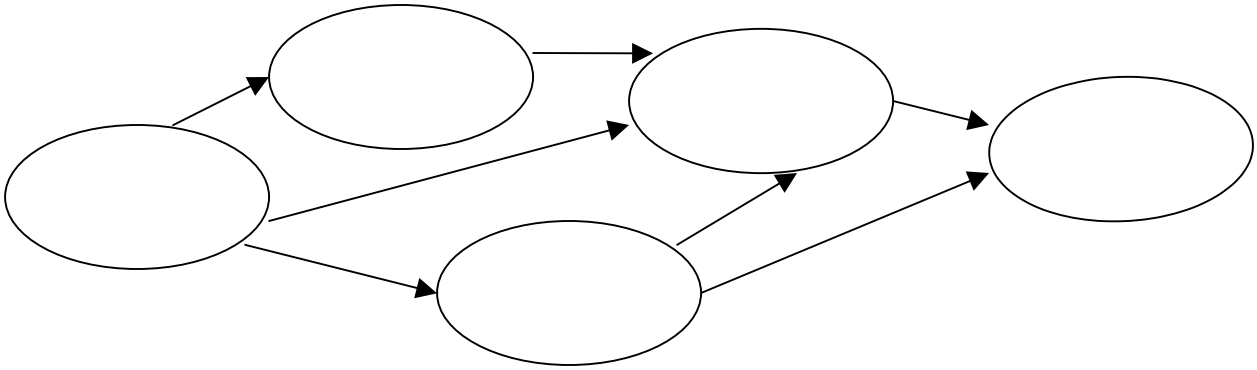

Figure 1. Proposed model of factors affecting student performance in an introductory programming course

beliefs across related activities may occur [1], for example, across different programming languages or tasks.

Because skills and self-beliefs are so intertwined, one way of improving student performance is to improve student self-efficacy. While self-efficacy can be enhanced by the use of behavior modeling, cooperative learning environments and verbal persuasion [1, 5], the greatest influence of all is positive personal experience [1].

Norman [9] defines mental models as predictive representations of real world systems. That is to say, people create internal representations of objects and information in the world, and they use these mental representations to reason about, explain, and predict the behavior of external systems.

Programming is a highly cognitive activity that requires the programmer to develop abstract representations of a process in the form of logic structures. Having a well-developed and accurate mental model may affect the success of a novice programmer in an introductory programming course. Such a model could include knowledge about how programs work in general, as well as knowledge about the syntax and semantics of a specific language [4]. Mental models (also referred to as schemas) play an important role in program comprehension [10, 14, 17] and correspondingly in comprehension-related tasks, such as modification and debugging.

This study proposes a model of performance of novice programmers based on their self-efficacy and mental models (see Fig. 1). The ovals represent variables and the arrows represent predicted relationships of the variables. In line with the existing research [2, 3, 15], we expect that previous experience will be a significant predictor of both students' self-efficacy and mental models of programming. Based on self-efficacy theory [1], we further expect that students' self-efficacy will increase as a result of instruction and continued hands-on exposure to programming (post-self-efficacy higher than pre-self-efficacy). We also hypothesize the students' mental models of programming will be significantly related to their perceptions of self-efficacy. Finally, it is expected that both mental model and self-efficacy will explain a significant amount of course performance, i.e., in keeping with Bandura, students' knowledge content and self-perceptions will be intertwined in successful performance.

3. METHODOLOGY

3.1 Participants

Seventy-five students enrolled in four sections of a CS1 course at a public university took part in this study. The students were undergraduates majoring in computer science, as well as a wide variety of other majors ranging from agricultural science to management information systems.

3.2 Materials

The materials included a background questionnaire, a self-efficacy scale, and two instruments to measure mental models. All of the materials except the self-efficacy scale were developed specifically for this study.

The background questionnaire used five questions measuring the breadth of participants' prior computer and programming background: number of courses taken that used computer applications as a mandatory part of course work (e.g., spreadsheets, databases), number of programming courses taken, number of programming languages used, number of programs written, and length of the programs written.

Self-efficacy was measured using the Computer Programming Self-Efficacy Scale [12]. This validated instrument was used previously by Wilson and Shrock [18] in their research on success factors in introductory computer science courses. The scale consists of thirty-three items that ask students to judge their ability in a wide range of programming tasks and situations, e.g., "I would be able to write syntactically correct statements," "I would be able to write a program that computes the average of 3 numbers," "I would be able to comprehend a long, complex multi-file program," "I would be able organize and design my program in a modular manner." Responses are marked on a 7-point Likert scale.

The students' mental models were evaluated by using two measurements, program comprehension and program recall. The program comprehension booklet consisted of six short C++ programs (each 15-20 lines long). The programs consisted of a class definition, a constructor, a member function of the class, and a main function. Each of the programs was followed by a list of five true/false questions covering each of the information categories originally developed by Pennington [10] to measure the mental model of programmers: elementary operations,

control flow, data flow, program function, and program state. These same categories have been used in more recent research on mental models in OO programming [17]. Recall (measured as the number of lines recalled correctly) has been used as a measure of mental organization or mental models in past programming research, e.g., Shneiderman [13]. Our program recall booklet, modeled on Shneiderman's, contained a C++ program that dealt with temperature conversion. The program had 27 lines of code.

3.3 Procedure

The study was conducted over the course of a fifteen week semester in two parts, one in the second week of the semester, and the other in the thirteenth week of the semester.

The first phase of the study involved collecting the student background information and having students complete the self-efficacy scale, which yielded the pre-self-efficacy score.

The second phase of the study involved completion of the same self-efficacy scale (yielding the post-self-efficacy score) and the two tasks designed to assess the students' mental model. The students completed the self-efficacy scale first and then were given the program comprehension booklet. For each of the six programs, they had 1.5 minutes to study the program and two minutes to answer the questions. The students were not allowed to look back at the programs while answering the questions. Finally, students were given the recall booklet. They had five minutes to study the program, then closed the booklet and had five more minutes to recall and reproduce the program from memory, to the best of their ability.

The performance measure was the student's final course grade and was obtained from the instructor at the end of the semester.

4. RESULTS

4.1 Self-Efficacy of Novice Programmers

The alpha-reliability of the self-efficacy scale was .98, indicating a highly reliable scare. The mean pre-self-efficacy score was 94.63 and the mean post-efficacy score was 163.37 out of a maximum possible score of 231 (Table 1). Self-efficacy of participants increased significantly over the course of a semester of instruction ($t = 12.78$, $p<.0001$). The mean increase in self-efficacy was 68.75.

To evaluate the effect of pre-self-efficacy on the amount of change in self-efficacy during the course, we divided the participants into quartiles of equal size based on their pre-self-efficacy scores (Table 2). Change in self-efficacy was calculated as the difference between the post-self-efficacy and the pre-self-efficacy score and was subjected to a one-way ANOVA. The ANOVA was significant, $F(3,71) = 12.83$, $p<.0001$. The results show significant effects of time of measurement (pre vs. post), quartile, and the interaction between the two. The ANOVA was followed by a Tukey range test to determine specifically how the groups differed from one another. Results suggest that the group with the highest pre-self-efficacy (group mean = 165.05) experienced the least increase in self-efficacy and differed significantly ($p<.05$) from all other groups, which registered much larger increases in efficacy (Table 2). The other three groups did not differ significantly from one another.

Table 1. Self-efficacy descriptive data (N=75)

Variable	Mean	StdDev	Min	Max
Pre-SE	94.63	49.51	33	219
Post-SE	163.37	41.36	52	229

Table 2. Descriptive data for the four quartiles divided on initial self-efficacy

Group	Pre-SE Mean	Post-SE Mean	Change Mean	Change StdDev	N
1	46.47	134.42	87.95	46.68	19
2	66.39	159.94	93.56	32.01	18
3	99.11	170.05	70.95	31.98	19
4	165.05	188.89	23.84	40.25	19

4.2 Relationship of Self-Efficacy and Mental Models

The mental model measure was a cumulative sum of participants' program comprehension scores and the recall scores. Analysis of the data revealed a simple Pearson correlation of $r = .3227$, $p<.01$ between the mental model measure and post-self-efficacy score. There was no significant correlation between pre-self-efficacy and mental models.

4.3 Analysis of the Model

A major goal of this study was to evaluate the effect of self-efficacy and mental models on students' performance. To test if our model was supported by the data, a path analysis was conducted [8]. Path analysis consists of a series of multiple regressions used to analyze the relationships of variables in a model. Each arrow in our model (Fig. 1) represents a possible relationship of a predictor (independent) variable on a response (dependent) variable. Figure 2 shows the results of the multiple regressions. The strengths of each relationship are depicted as "path coefficients," or regression weights, which vary between 0 and 1 (shown on the arrows in Fig. 2). A significant path coefficient indicates that there is indeed a reliable relationship between the predictor and response variable. All paths predicted in the model were significant ($p<.05$) *except* for the path from previous experience to mental model. The R^2 values associated with the dependent variables indicate how much of the variance in the variable is explained by the predictor variables feeding into it. As Fig, 2 shows, the variance explained is substantial except for the effect of previous experience on mental model.

In addition to analyzing the individual relationships, path analysis seeks to analyze the "fit" of the overall model to the data. This is done using a Chi-square test. Eliminating the non-significant path from previous experience to mental model, the Chi-square test of this model with four degrees of freedom yielded $\chi^2=1.35$, $p>.85$. Contrary to the usual interpretation, in this analysis the non-significant result represents a good fit. Literally, it means that our model containing a theory-based subset of all possible paths is just as good a predictor of performance as a model containing *every* possible path, i.e., this more parsimonious subset of relationships adequately represents the important influences on student performance.

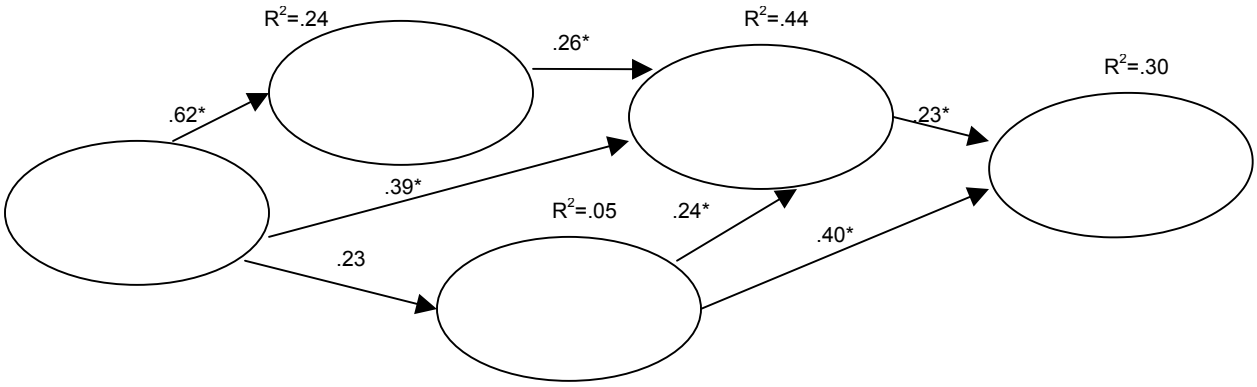

Figure 2. Results of the analysis of the relationships in the model (*p<.05)

5. DISCUSSION

5.1 Self-Efficacy, Mental Models, and Programming Performance

The self-efficacy of students increased significantly over the course of a semester of instruction. The results suggest that individuals' changes in self-efficacy are a function of their pre-self-efficacy. The three lower quartiles showed a significant increase in efficacy with group 2 registering the greatest increase. However, the group with the highest initial self-efficacy experienced the least increase in efficacy as a result of a semester of instruction in C++ programming. This is a sensible result because a strong sense of self-efficacy is not easily changed, but weak self-efficacy is more malleable [1].

The highest quartile is different from the other groups in another aspect as well. The standard deviation for this group was almost double its mean change in self-efficacy (Table 2). The implication is that, while some students in the group registered an increase in efficacy, others decreased. Further analysis showed that about one-fifth of this group (21.1%) experienced a decrease in self-efficacy. This suggests that these students overestimated their ability to cope with the challenges of the introductory course.

Therefore, as depicted in the results (Fig. 2), we can say that previous experience is a strong predictor of pre-self-efficacy, and interestingly it also predicts post-self-efficacy. This indicates that students' prior high school experience continues to affect their perceptions of their capabilities even near the end of the university course. The results also show that having developed a strong mental model increases feelings of self-efficacy. Finally, both what student know, as represented by their internal mental model, and what they believe about themselves, as represented by their self-efficacy, affect their performance in the course. While instructors have always recognized the importance of what students know, the results of this study underline the parallel importance of students' self-beliefs.

5.2 Implications for Pedagogy

An ongoing challenge in computer science education is to attract students to introductory computer science courses and support them in what, for many, are the difficult early days. This applies to computer science majors, as well as to students taking a single course in programming.

Computer science instructors are well aware of the importance of students internalizing good mental models of programming. This study shows that a well-developed and accurate mental model directly affects course performance and also increases self-efficacy, the other key element in course performance. Given this double impact, helping students develop good mental models should remain a goal in introductory programming courses. Teaching from the object-oriented perspective may in itself assist the goal of developing the mental model, because the high salience of objects, their attributes, and the relationships of objects highlights the correspondence of computing objects to real world objects [17]. Apart from the programming language, the goal of building good mental models could also be achieved by instruction that engages the student in experiential learning tasks that involve tracing the logic of a program [4]. Tasks such as debugging and modifying programs often involve tracing and are likely to promote development of the mental model. In general, assignments that involve both program comprehension and creation strengthen the mental model through reasoning about consequences, e.g., adding a new module to a program that involves interactions with other parts of the program.

The other path to increasing student performance is direct self-efficacy interventions. When teaching an introductory programming course we must challenge students but not overwhelm them with complex programming tasks that undermine their self-efficacy. Interventions that support and increase self-efficacy have been identified by Bandura [1]: performance attainments, observation of the performance of others, social persuasion, and monitoring of one's own physiological state. The importance of performance attainments on self-efficacy indicates that students need to incrementally build up a history of success at increasingly difficulty tasks. This suggests that frequent assignments with quick and ample feedback are more desirable than a smaller number of longer-term projects. In terms of observation of the performance of others, it is known that watching another person carry out a difficult task increases self-efficacy [1, 5]. In learning computer applications [5] it has been shown that peer modeling is most likely to aid learners, because it shows the learner that someone "like me" can do the job. In computer programming we could build the students' self-

efficacy by modeling how to build a complex program. A video of students at work planning, creating, and debugging a program might provide appropriate modeling. Social persuasion from peers also promotes self-efficacy [1]. This implies a classroom that encourages group work and strong interaction with other students. Social connections may also be built by augmenting the traditional class setting with online work groups. Finally, a calm physiological state, as opposed to anxiety and apprehension, increases feelings of self-efficacy. This may be promoted by taking measures to increase comfort in the classroom [18] and decrease student competition. It may also be promoted by evaluation methods that incorporate measures of student improvement rather than just absolute measures of achievement.

As a final comment, it is worth noting that any students in introductory CS courses are not computer science majors. These students often, however, enter careers in which they develop end-user programs, such as spreadsheets or interactive web sites, to support their own work. Since end-user programming is usually voluntary, motivation is the key to whether the user programs. Having strong self-efficacy beliefs may be especially important for future end users.

6. FUTURE DIRECTIONS

Limitations of this study include the short time span in which the study took place, the self-reporting of previous experience, and the performance measure of based on the final course grade.

An important area of future research is the study of students who drop out of introductory programming course (and thus are not even included in studies such as this that require pre and post measurement). For such students, it may be important to plan early interventions to support development of students' self-efficacy and mental model, before the student gives up.

7. ACKNOWLEDGMENTS

This work was supported in part by the EUSES Consortium via NSF grant CCR-0324844.

8. REFERENCES

[1] Bandura, A. Social Foundations of Thought and Action. Prentice Hall, Englewood Cliffs, NJ, 1986.

[2] Bunderson, E.D. & Christensen, M.E. An analysis of retention problems for female students in university computer science programs. Journal of Research on Computing in Education, 28(1) (1995), 1-15.

[3] Byrne, P. & Lyons, G. The effect of student attributes on success in programming. Proceedings of ITiCSE 2001, (2001). ACM Press, NY, 49-52.

[4] Cañas, J.J., Bajo, M.T. & Gonzalvo, P. Mental models and computer programming. International Journal of Human-Computer Studies, 40 (5) (1994), 795-811.

[5] Compeau, D.R. & Higgins, C.A. Computer self-efficacy: development of a measure and initial test. MIS Quarterly (June 1995), 189-211.

[6] Guzdial, M. & Soloway, E. Log on education: teaching the Nintendo generation to program. Communications of the ACM, 45(4) (2002), 17-21.

[7] Karsten R. & Roth R.M. Computer self-efficacy: a practical indicator of student computer competency in introductory IS courses. Informing Science. 1(3) (1998), 61-68.

[8] Kerlinger, F.N. & Pehazur, E.J. Multiple Regression in Behavioral Research. Holt, Reinhart & Winston, New York, 1973.

[9] Norman, D.A. Some observations on mental models. In D. Gentner and A.L. Stevens, Eds., Mental Models, Erlbaum, Hillsdale, NJ, 1983.

[10] Pennington, N. Comprehension strategies in programming. In E. Soloway and S. Iyengar, Eds., Empirical Studies of Programmers: Second Workshop. Ablex, Norwood, NJ, 1987, 100-113.

[11] Potosky, D. A field study of computer efficacy beliefs as an outcome of training: the role of computer playfulness, computer knowledge, and performance during training. Computers in Human Behavior, 18 (2002), 241-255.

[12] Ramalingam V. & Wiedenbeck S. Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. Journal of Educational Computing Research, 19(4) (1998), 365-379.

[13] Shneiderman, B. Exploratory experiments in programmer behavior. International Journal of Computer and Information Sciences, 5 (2) (1976), 123-143.

[14] Soloway, E. & Ehrlich, K. Empirical studies of programmer knowledge. IEEE Transactions of Software Engineering, SE-10 (5) (1984), 595-609.

[15] Taylor, H. & Mounfield, L. Exploration of the relationship between prior computing experience and gender on success in college computer science. Journal of Educational Computing Research, 11(4) (1994), 291-306.

[16] Thomas, L., Woodbury, J. & Jarman, E. Learning styles and performance in the introductory programming sequence. Proceedings of SIGCSE 2002 (Covington KY, Feb. 2002). ACM Press, NY, 33-37.

[17] Wiedenbeck, S., Ramalingam, V., Sarasamma, S. & Corritore, C.L. A comparison of the comprehension of object-oriented and procedural programs by novice programmers. Interacting with Computers, 11 (1999), 255-282.

[18] Wilson, B.C. & Shrock S. Contributing to success in an introductory computer science course: s study of twelve factors. Proceedings of SIGCSE 2001 (2001), ACM Press, NY, 184-188.