View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by AIS Electronic Library (AISeL)

Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2000 Proceedings

Americas Conference on Information Systems (AMCIS)

2000

Investigating Computer Self-Efficacy with Students in COBOL Programming

Michael A. Chilton
Southwest Missouri State University, mac927f@mail.smsu.edu

Cynthia K. Riemenschneider
University of Arkansas, criemen@comp.uark.edu

Follow this and additional works at: http://aisel.aisnet.org/amcis2000

Recommended Citation
Chilton, Michael A. and Riemenschneider, Cynthia K., "Investigating Computer Self-Efficacy with Students in COBOL Programming" (2000). *AMCIS 2000 Proceedings*. 232.
http://aisel.aisnet.org/amcis2000/232

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2000 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

Investigating Computer Self-Efficacy with Students in COBOL Programming

Michael A. Chilton, CIS Department, SW Missouri State University, mac927f@mail.smsu.edu
Cynthia K. Riemenschneider, CISQA Department, University of Arkansas, criemen@comp.uark.edu

Abstract

The theory of self-efficacy (Bandura, 1982) has been used in the field of Information Systems to test its predictive nature of computing outcomes (Compeau & Higgins, 1995, Gist, Schwoerer & Rosen, 1989; Murphy, Coover & Owen, 1989). It has been shown to be a successful measure of performance and indeed, is now perceived as a "practical indicator of student computer competency" (Karsten & Roth, 1998). This study attempts to further quantify and qualify such a measure and investigates the degree to which self-efficacy can be manipulated. Additionally, it has been traditionally thought that programming maintenance is more difficult when a programmer must modify someone else's program rather than his or her own. This study also investigates this phenomenon within the context of self-efficacy. Findings show that self-efficacy is an important indicator of outcome performance and that it can be artificially manipulated. Some surprising results occurred when students were asked to modify someone else's code.

Introduction

Self-efficacy has been used in studies in information technology (e.g., Igbaria & Iivari, 1995, Compeau & Higgins, 1995) to investigate the effects of perceived self-efficacy on computer usage. Compeau & Higgins (1995) developed a measure for computer self-efficacy.

Programming maintenance is a large part of the IS budget, comprising anywhere from 40 to 75% of it (Vessey & Weber, 1983). It is largely assumed that a programmer who must modify someone else's code faces a more daunting task than one who writes the code from scratch simply because the programmer must now first decipher the original programmer's code and then modify it. Had the code been his/hers in the first place, the time to modify it should be much less.

Given these two concepts—the effects of self-efficacy on task completion and the difficulty of programming maintenance—the current study attempts to gain some insight into these two areas. The resulting experimental design was a 2 x 2 factorial Analysis of Variance in which two experimental groups received a single manipulation (either for self-efficacy or having to modify another person's code) and one group received both. The model is shown in figure 1. The purpose of this study is to measure the effects of such manipulations on outcome performance.

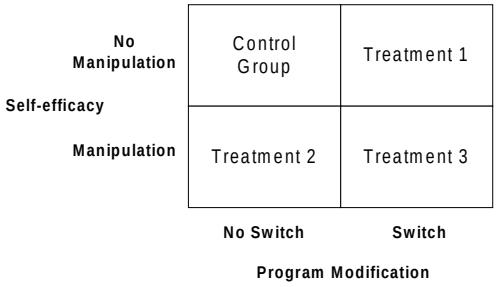

Figure 1

Hypotheses

The research model depicts four basic groups consisting of those students who received no manipulation for self-efficacy or program complexity (the control group), those who received either a manipulation for self-efficacy or were required to modify someone else's program (treatment groups 2 and 3) and those who received both treatments (treatment group 4). Additional tests were run to investigate the effects of gender, age, programming experience and instructional technique. The last comparison was necessary because there were two instructors among the four classes participating in the study.

Hypotheses are based on social cognitive theory. This theory would suggest that those who had higher self-efficacy would perform better than those with lower amounts. Whether self-efficacy can be manipulated and whether the subsequent change has an effect is the research question posed here. Additionally, for those students required to modify someone else's code, experience would suggest that these students would have more difficulty and produce code whose quality is not as high had they produced it themselves from scratch. The hypotheses are shown below.

H1: COBOL students with greater self-efficacy will write programs of higher quality than COBOL students with significantly lower self-efficacy.

H2a: Self-efficacy of student COBOL programmers can be artificially inflated using a written accolade.

H2b: Students whose self-efficacy has been artificially inflated will write higher quality programs.

H3: COBOL students who must modify someone else's program in a fixed amount of time will write lower quality programs.

Research Design

Sixty-nine students from four programming courses were selected and randomly assigned to one of four groups. There were two classes of introductory COBOL programmers and two classes of advanced. There were also two classes instructed by one person and two classes instructed by another. Due to logistic and communication problems, one class was eliminated. The first step in the research was to administer the computer self-efficacy instrument to provide a baseline score for each subject. The experiment was conducted late in the semester (the 13th and 15th weeks of a 16 week semester) so that students had a chance to acquire some skills in programming and therefore have a better basis to judge themselves. Students were then assigned a computer program as part of the normal learning process and given two weeks to complete it. They were not allowed to seek outside help nor collaborate with each other in the writing of the program. On the due date, the programs were collected and graded. Two researchers performed additional checks to more precisely measure the quality of coding. The programs were then returned to the students along with a note from the grader. The note for the control groups was a simple explanation of how the grade was computed. The note for the treatment groups included a complimentary statement regarding the quality of their program (regardless of the actual quality). After receiving their programs, a second measurement of self-efficacy was obtained. Another programming assignment was then made that was a modification to the one just returned. Although not a difficult program modification, it was also non-trivial. The control groups were allowed to retain and modify their own programs (no switch), but the treatment groups were randomly assigned someone else's program to modify (switch). Students were not told whose program they were to modify, and so only those who received their own programs back actually knew.

Methodology

First, the demographic information regarding the participants is reported. The average age of the participants was 23.7 years with a standard deviation of 6.4. The youngest respondent was 19 and the oldest was 55 years. Forty of the participants were male and 29 were female. The sample was split fairly evenly between the two instructors; one instructor taught 35 students and the other taught 34 students. Twenty-three of the students were taking advanced COBOL and 46 were taking introductory COBOL.

Second, a comparison of the groups was made to see if there was a difference between the groups based on the instructional style. A series of t-tests were conducted to determine if there was a difference in the groups assigned to each instructor. The results of the t-tests are shown in Table 2. There were no significant differences between the two groups based upon the instructor. An additional comparison was made between the groups to see if there was a difference based on the level of the course the student was taking-either introductory or advanced COBOL. Table 3 shows the results of these comparisons. The significant differences between the two groups were the number of months of COBOL programming experience and the self-efficacy scores at both time 1 and time 2. The difference in the number of months of COBOL programming experience was expected to be significant since students in the advanced classes had a full semester more of experience than those in the introductory classes. This supports the theory that personal experience is an important source of self-efficacy evaluation.

Analysis

To test the hypotheses, several techniques were used. Two multivariate regressions for time 1 and time 2 were performed in order to test hypothesis 1. Should this hypothesis be true, we would expect to see significance in each coefficient of determination (r^2) and a significant coefficient for the self-efficacy independent variable. We included instructional technique and class level as additional independent variables, but did not expect these variables to significantly contribute to the model. The regression at time 1 was significant with a p-value < 0.05 as shown in Table 4, and as expected, the only independent variable coefficient that was significant was the efficacy score at time 1. Thus, at time 1 the student's self-efficacy score was a predictor of his/her programming assignment score. At time 2, however, the regression model was not significant and none of the independent variables contributed to the variance of the students' programming assignment score.

In order to test hypothesis 2a, a t-test was conducted to see if there was a significant difference in the self-efficacy score at time 2 between those students who had received the manipulation and those who had not. The results are reported in Table 5. The students who received the manipulation had a significantly higher self-efficacy score at time 2, thus lending support for hypothesis 2a, that the self-efficacy of student COBOL programmers can be artificially inflated using a written accolade. Hypothesis 2b was tested by a multi-phase approach. First, a t-test was performed to determine if there was a difference in the programming assignment 2 score for those subjects who had received the manipulation and those who had not. These results are shown in Table 6. There was not a significant difference between the programming assignment 2 scores for those that had been manipulated and those that had not. Further analysis was done to see if there was a difference between the first and second self-efficacy scores of the students who had received the manipulation. A paired difference t-test was performed; overall, the mean efficacy score at time 2 was higher than time 1 for those who were manipulated, but it was not significantly higher. A paired difference t-test

was performed to compare the self-efficacy scores for those students who did not receive the manipulation. Again, there was no significant difference.

To test hypothesis 3, a t-test was conducted to determine if there was a significant difference in the programming assignment 2 score for those students who had to modify someone else's program (switch equals yes) versus modifying their own program. The results are reported in Table 7 and they indicate there was a significant difference between the scores. Surprisingly, however, the significance is not in the direction we hypothesized. The students who modified someone else's program had a higher mean score than those who modified their own programs.

Some additional analyses were performed to see if there were any gender differences in the efficacy scores or in the programming assignment scores. Table 8 gives a summary of the t-tests that were performed. No significant differences between the males and females were found in any comparison.

Discussion

This study provided some very interesting findings. First, an individual's self-efficacy can initially be a predictor of his/her ability to write COBOL programs as indicated by our findings regarding time 1 in the test of hypothesis 1. It is difficult to conclude here that hypothesis 1 is sustained, however, because of the non-significant outcome at time 2. Because one class was eliminated from the study, the sample size became reduced to the point that this may have affected these findings. Additional effort is planned for this study to further investigate this result.

Second, it was found that it is possible to manipulate an individual's self-efficacy based upon positive written feedback. Those students who received manipulation had a higher mean self-efficacy score than those who did not receive manipulation. This is an essential finding regarding the importance of giving positive feedback to students to help them increase their confidence in their abilities to perform programming assignments. This is particularly applicable to academicians facing the challenges of teaching larger numbers of students in courses that may have traditionally been smaller in size due to the subject matter or nature of the course. Instructors can influence a student's computer self-efficacy by providing positive comments regarding programming assignments. Admittedly, this may not ultimately affect the programming assignment score based on the results of this study. However, since the findings regarding hypotheses 2a and 2b were conflicting, we are in the process of collecting additional data to increase our sample size and rerun the statistical analyses.

The surprising result regarding hypothesis 3 was the direction of the statistical significance. Those students who were required to modify someone else's program ended up with a higher mean score than those who modified their own programs. One possible explanation for this finding is that the student who modified someone else's program expected to expend more time and effort since the program was not his/her initial creation. Therefore, these students over-compensated for the experimental treatment of the researchers and out-performed those who were modifying their own programs. Again, additional data is being collected to further investigate this phenomenon.

The lack of significant difference between the males and females was consistent with prior research. Karsten and Roth (1998) also used predominantly traditional students and did not find any significant difference in computer self-efficacy based on gender.

References

Bandura, A. "Self-efficacy Mechanism in Human Agency," *American Psychologist*, (37:2), 1982, pp. 122-147.

Bandura, A. *Self-efficacy: The exercise of control.* New York: W.H. Freeman & Co., 1997.

Compeau, D. R. and Higgins, C. A. "Computer Self-Efficacy: Development of a Measure and Initial Test," *MIS Quarterly*, (19:2), June 1995, pp. 189-211.

Cronbach, L. J. "Coefficient Alpha and the Internal Structure of Tests," *Psychometrica*, (16), 1951, pp. 297-334.

Gist, M.E., Schwoerer, C., and Rosen, B. "Effects of Alternative Training Methods on Self-efficacy and Performance in Computer Software Training," *Journal of Applied Psychology*, (74:6), 1989, pp. 884-891.

Igbaria, M. and Iivari, J. "The Effects of Self-efficacy on Computer Usage," *Omega International Journal of Management Science*, (23:6), 1995, pp. 587-605.

Karsten, Rex and Roth, R. "Computer Self-Efficacy: A Practical Indicator of Student Computer Competency in Introductory IS Courses," *Informing Science*, (1:3), 1998, pp. 61-68.

Lientz, B. P., Swanson, E.B. and Tompkins, G.E. "Characteristics of Application Software Maintenance," *Communications of the ACM*, (21:6), 1978, pp. 466-471.

Murphy, C. A., Coover, D. and Owen, S.V. "Development and Validation of the Computer Self-efficacy Scale," *Educational and Psychological Measurement*, (49), 1989, pp. 893-899.

Nunnally, J. *Psychometric Theory*. New York: McGraw-Hill, 1978.

Vessey, I. and Weber, R. "Some Factors Affecting Program Repair Maintenance: An Empirical Study," *Communications of the ACM*, (26:2), 1983, pp. 129-134.

Tables may be obtained from the first author.