Factors Affecting the Success of Non-Majors in Learning to Program

Susan Wiedenbeck
Drexel University

ICER 2005

Introduction

- End-user programmers increasingly important
  - 55 million end users
  - Programming spreadsheet formulas, macros, style sheets, web applications, domestic technologies
  - Programming education is a strong facilitator to end-user programming
  - But dropout and failure rate in introductory programming courses of 15-30% (Guzdial & Soloway, 2002)
- How do cognitive and social-cognitive factors affect success and failure in an introductory programming course?

Factors Suggested in the Literature

- Previous computer programming experience (secondary school)
- Mathematics or science background
- Learning styles
- Course outcome expectations
- Playfulness (studied in training courses)
- Self-efficacy

Factors Studied

Previous experience and two important factors in cognitive and social cognitive theory:

- Self-efficacy: the individual's judgment of his or her ability to perform a task in a specific domain
- Knowledge organization: a person's internal organization of concepts

Goals of this research

- Study previous experience, self-efficacy, and knowledge organization of novice non-majors
- Explore the relationship between these factors
- Investigate their combined influence on course performance

Background on Self-Efficacy

- Albert Bandura's theory 1977
- Basic Principle: "*people are likely to engage in activities to the extent that they perceive themselves to be competent at those activities*"
- Self-beliefs a key element in human performance

Self Efficacy and Cognitive Strategies

- Self-efficacy influences behavior in poblem solving:
  - effort expended
  - persistence
  - coping strategies adopted
  - performance outcomes

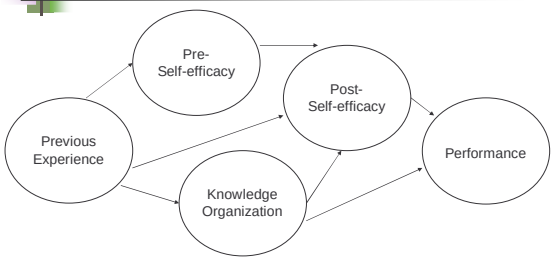
Four Sources of Self-Efficacy

- Performance attainments
- Experiences of observing the performance of others
- Verbal persuasion
- Physiological reactions people use to judge their capableness and vulnerabilities (anxiety, fatigue)

Knowledge Organization in Programming

- Effective knowledge organization supports reasoning, comprehension, and explaining the behavior of systems
  - Skilled programmers chunk knowledge into meaningful groups in memory, recognize patterns, uses context of chunks to plan
  - Less skilled programmers don't chunk related knowledge, each piece is an independent unit, doesn't support pattern recognition and planning

The Proposed Model



Methodology - Participants

- 120 non-majors (34%F, 66%M)
- Average Age: 20
- Second and third year undergraduates from a variety of majors
- 5 sections of C++ programming course
- 4 different instructors coordinated to cover the same material

Methodology - Materials

- Background questionnaire
- Self-efficacy scale (Ramalingam & Wiedenbeck, 1998)
- Knowledge organization
  - Program memorization/recall task (Shneiderman, 1976; Di Persio, Isbister, Shneiderman, 1980)
    - Indicator of well-organized knowledge
    - Predicts exam grades

Methodology - Procedure

- Two parts during one 16-week semester
  - Part 1: Weeks 1-8
    - Background questionnaire
    - Pre-self-efficacy score
    - Program memorization/recall task
  - Part 2: Weeks 9-16
    - Post-self-efficacy score
    - Debugging score on final examination
    - Final course grade

Results - 1

Previous experience data

	Mean
Number of courses using computers	2.62
Number of programming courses	1.15
Number of computer programs developed	~10
Number of programming languages	1.43
Longest length of program (lines)	~20

Results - 2

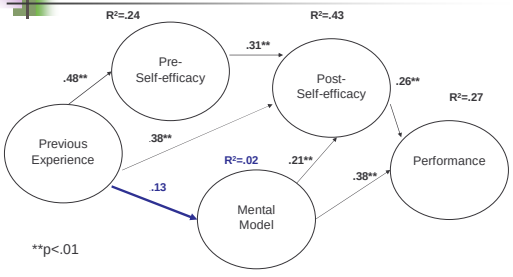
Self-efficacy data – mean question response on scale of 1-7

	Mean	StdDev
Pre-SE	3.73	1.54
Post-SE	5.11	1.17

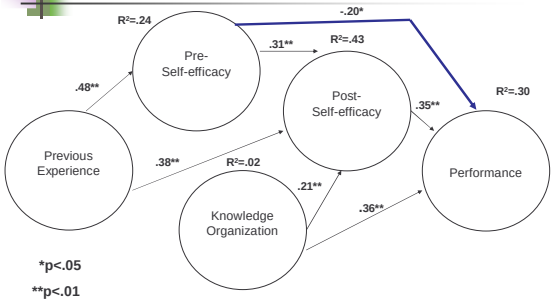
Analysis of Model

- Path Analysis
  - Way to analyze predictive relationships of variables in a model
  - Uses series of multiple regressions
  - Also concerned with analyzing the fit of the overall model with the data

Results of the Analysis



Revised Model of Factors



Discussion - 1

- Self-Efficacy increased substantially over the course of the semester
  - Frequent hands on programming tasks
  - Malleability of self-efficacy
- Previous experience influences pre- and post-self-efficacy (indirect relationship to performance)
- Previous experience did not affect students' knowledge organization

Discussion - 2

- Strong knowledge organization increases beliefs in self-efficacy
- Self-efficacy and knowledge organization influenced course performance
- **But** high pre-self-efficacy combined with low previous experience was predictive of lower course performance

Limitations

- The time span of a 16 week semester
- Self-reporting of previous experience
- Use of course grade as the performance outcome
- What happened to the students who dropped out during the semester?

Pedagogical Interventions - 1

- Build a student's self-efficacy
  - Frequent hands-on programming activities – learner-appropriate exercises with performance successes
  - Observation of the performance of peers – video or live observation of student working out a programming problem

Pedagogical Interventions - 2

- Social persuasion – collaborative exercises, groups of students with varying degrees of abilities
- Monitor students physiological state – design the classroom to decrease anxiety, reduce competition

Thank you