2017

Get Rid of Your Student's Fear and Intimidation of learning a Programming Language

Christina Frederick
Embry-Riddle Aeronautical University

Matt B. Pierce
Embry-Riddle Aeronautical University, piercem5@my.erau.edu

Andrew Calvin Griggs
Embry-Riddle Aeronautical University

Lulu Sun
Embry-Riddle Aeronautical University, sunl@erau.edu

Li Ding
Embry-Riddle Aeronautical University, dingl@erau.edu

ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION

Paper ID #20923

Get Rid of Your Student's Fear and Intimidation of learning a Programming Language

Dr. Christina Frederick, Embry-Riddle Aeronautical Univ., Daytona Beach

Dr. Frederick is currently a Professor and Graduate Program Coordinator in the Human Factors and Systems Department at Embry-Riddle Aeronautical University in Daytona Beach, Florida. Dr. Frederick received her Ph.D. in 1991 from the University of Rochester with a major in Psychological Development. She previously taught at the University of Rochester, Southern Utah University and the University of Central Florida. In 2000, Dr. Frederick joined the Human Factors and Systems Department at Embry-Riddle, where her work focused on applied motivation and human factors issues in aviation/aerospace. Dr. Frederick also served in various roles in University administration between 2004-2012, including Vice President for Academics and Research. Dr. Frederick's current research interests examine how individual differences interact with technology to enhance educational engagement and performance. Dr. Frederick is the author of more than 50 research publications, 4 book chapters and over 60 regional, national and international conference presentations on a wide range of topics in human factors and psychology. She is active in a number of professional associations, and is a Consultant for Psi Chi, the National Honor Society in Psychology.

Mr. Matthew Pierce, Embry-Riddle Human Factors and Systems
Mr. Andrew Calvin Griggs, Embry-Riddle Aeronautical University
Dr. Lulu Sun, Embry-Riddle Aeronautical Univ., Daytona Beach

Lulu Sun is an associate professor in the Engineering Fundamentals Department at Embry-Riddle Aeronautical University, where she has taught since 2006. She received her B.S. degree in Mechanical Engineering from Harbin Engineering University (China), in 1999, and her Ph.D. degree in Mechanical Engineering from University of California, Riverside, in 2006. Before joining Embry-riddle, she worked in the consulting firm of Arup at Los Angeles office as a fire engineer. Her research interests include second language acquisition in programming languages, flipped classroom, and virtual training. She is a professional member of the Society of Fire Protection Engineers, and a member of the American Society for Engineering Education.

Dr. Li Ding, Embry-Riddle Aeronautical University

Li Ding is a visiting professor of the Department of Engineering Fundamentals at Embry-Riddle Aeronautical University, where she has been since 2012. She received her Ph.D in Environmental Engineering from the University of Illinois at Urbana-Champaign in 2010. She taught several undergraduate courses in engineering and in science, and she currently teach Introductory to Programming for Engineers. From a background of an engineer, she is transitioning into an educator, and has been working with other principle researchers on education studies since 2015.

Get Rid of Your Students' Fear and Intimidation of Learning a Programming Language

Christina Frederick, Matthew Pierce, Andrew Griggs, Li Ding, Lulu Sun
Embry-Riddle Aeronautical University, frederic@erau.edu, piercem5@my.erau.edu, griggsa2@my.erau.edu, dingl@erau.edu, sunl@erau.edu

***Abstract* – Knowledge of computer programming is very beneficial and often required for engineering students. Unfortunately, students frequently experience fear and intimidation regarding introductory programming courses. Second language acquisition (SLA) techniques have shown promise as a means of content delivery in programming courses. Blended learning environments are also becoming increasingly popular in course frameworks. This workshop will discuss the application of second language acquisition in a blended learning environment (SLA-aBLe) and will examine the effectiveness of using SLA techniques to teach introductory programming. The proposed workshop will also share instructor experience(s), provide course materials, and review student outcomes from this two year study. Workshop participant involvement is encouraged through interactive elements of the presentation such as live polling, discussion, and a question and answer forum.**

Index Terms – Education, Performance, Programming, Second-Language-Acquisition

INTRODUCTION

Computer programming knowledge is an essential learning criterion for computer science and engineering degrees. However, students often find learning programming to be difficult, especially without prior knowledge entering their first year in college [1]. Motivation and learning style are often causal factors for shortcomings in student performance when first exposed to programming [2]. A variety of students, each with their own learning style, will take introductory programming courses and strive for success [3]. Students may learn material through "deep" learning or "surface" learning [1]. Deep learners seek to understand all of the material presented while surface learners wish to just memorize the information needed to get by in class. Programming courses should be taught with both of these learning styles in mind [4].

DIFFICULTIES IN TEACHING PROGRAMMING

Programming functions as a culmination of multiple skills which follow a hierarchical model rather than a linear model [1]. Due to time constraints in introductory programming courses, the order in which concepts are presented is often less than intuitive [1]. For example, algorithms are often tacked onto other topics rather than getting their own lectures. This is unfortunate, as algorithms play a vital role in programming language fluency. Students are expected to not only understand computational algorithms, but how to apply them to problem solving and computer code. Concepts in programming are often very interdependent and course instruction does not always mirror these shared relationships. There are many languages to select from when teaching programming; however there is little evidence to support which is best for teaching students introductory programming concepts [1]. The goal in these courses is not to teach students Python or Java, for example, as there are other courses which specialize in those languages; introductory programming courses' overarching goal is to teach students to program. Despite educators being fully aware of this, students can find it hard to make the same distinction which can lead to difficulty when learning more abstract concepts [1].

SECOND LANGUAGE ACQUISITION TECHNIQUES FOR TEACHING

There are many theories and recommendations as to how computer programming should be taught. Programming has its own linguistic terms, grammar, and syntax as with a foreign language [5]. Second Language Acquisition (SLA) refers to the process of learning a new language, typically within children or adolescents [6]. SLA considers multiple complexities which factor into learning a new language. In an article by Chris Panell (2003), he claims he is teaching at his best when he teaches programming like a Language Arts course with emphasis on writing and "speaking" [7]. Having students simply memorize syntax proves to be less effective for fostering a deeper understanding of material in students. Panell states this has led him to adopt different teaching styles in his programming courses. SLA offers a novel approach to the problem of teaching introductory programming courses, as it functions as a framework to parse the wealth of new content students are exposed to in their first year of programming courses. When paired with familiar language rules and syntax, students find it easier to learn programming languages when they are presented from an SLA orientation and demonstrate deeper levels of understanding than would be found following rote memory tasks. The primary issue regarding course instruction is not which language to select for introductory courses, rather it is discerning effective strategies for content delivery that works around student's understanding.

THE SLA-aBLe PROJECT

EGR115, Introduction to Computing for Engineers has been taught utilizing a blended learning approach featuring both online course work and face to face class meetings. Blended learning environments offer numerous benefits for students such as working at their own pace and in settings of their choice. Despite the increased autonomy offered by blended learning environments, difficulties associated with learning programming language are rigorous. In an attempt to improve student learning outcomes and reduce stress associated with learning programming language, EGR115 has been altered to incorporate second language acquisition techniques into its curriculum.

To better understand the effects of second language acquisition techniques on programming language instruction and allow for between groups analysis, some instances of EGR115 were taught utilizing the contemporary non-SLA methods. Sections of EGR115 taught using SLA techniques in addition to a blended environment are referred to as SLA-aBLe sections, while those utilizing just the blended environment alone are referred to as non-SLA-aBLe sections. Over the course of 4 semesters (2 years) EGR115 was taught from an SLA orientation 11 times and a non-SLA orientation 11 times (N = 22). These courses did not vary in course content, only content delivery. Both sections covered introductory topics such as data type, input and output, conditional statements, and loops. SLA-aBLe sections adopted a framework that divides the learning process into five main stages: preproduction, early production, speech emergence, intermediate fluency, and advanced fluency. The division of course content into these stages aims to further illustrate the similarities between programming language and foreign language. Students begin with minimal comprehension in the preproduction stage and, as they advance through stages of development, gradually sharpen varying cognitive skills and components of fluency.

The preproduction stage is where the basic concepts are introduced. The use of simple diction, pictures, and other visual tools are used to reinforce learning at this stage. Early production skills are developed through the use of short answer and multiple choice questions in addition to an online discussion panel. At this stage students begin writing their first simple programs and demonstrate limited comprehension of course concepts. The speech emergence stage is characterized by lab activities focusing on application, comprehension, and problem solving. Exercises during this stage utilize a "think, pair, share," activity, which involves student collaboration and discussion. Intermediate fluency is achieved as students are able to compare and contrast varying concepts within programming and the ability to begin explaining their problem solving process. Advanced fluency is achieved during the presentation of an open-ended project. Final projects serve as a tool for students to express their comprehension as they develop a novel program which utilizes concepts across the course instruction.

Extensive PowerPoint presentations were created to accompany topics in each stage of the course. These slides featured demonstrations of MATLAB code, pictures, animations, and questions all aimed to facilitate understanding. Following their design, these slideshows were then recorded with narration into multiple short videos (10-20 minutes). The SLA-aBLe sections utilized these video lessons to teach topics within each stage of the course. These video lessons were designed with fluency in mind, and featured opportunities to practice new commands and apply previous learning to concepts introduced in the lesson. Programming concepts such as commands or syntax are compared to grammar and vocabulary. EdPuzzle was used to track the usage of these video lessons by students.

At each stage of learning in the SLAaBLe course, there is a greater focus on problem solving and fluency in content design and delivery. Typical face to face class meetings consisted of a brief review of common errors in previous online quizzes, think pair share exercises, and individual programming assignments. Following this model, at the end of each stage of learning, students should be able to demonstrate comprehension and application of various concepts within each topic. These demonstrations become increasingly complex throughout the course, culminating in a sophisticated end of course project. Stressing fluency, application, and problem solving throughout instruction encourages a deeper level of understanding than simple wrote memory.

GOALS OF THE WORKSHOP

The proposed workshop will provide an overview of this NSF funded project, examining the effectiveness of Second Language Acquisition (SLA) techniques to teach introductory programming courses in colleges and universities. Through the presentation and audience interaction, this workshop will provide an overview of the research project, the techniques utilized in course design, insight from educators involved in the project, pedagogical lessons learned thus far, and copies of course materials used so that participants may utilize them in their own course instruction.

WORKSHOP STRUCTURE

I. Project Description (10 Minutes)

This workshop will present the results of a two year applied project that integrated SLA techniques into an introductory programming class (EGR115 Introduction to Computing for Engineers), and then compare course effectiveness and outcomes to sections of the same course being taught without SLA techniques.

II. SLA Techniques (15 Minutes)

Presenters will discuss the utilization of SLA in programming course content delivery and design. Course content is divided into five main stages of learning, with each stage

characterized by increasing fluency and deeper understanding. Workshop participant interaction will be conducted by allowing participants to work on varying levels of programming problems and experience this project design first-hand.

III. Professor Commentary (25 Minutes)

Presenters will discuss the techniques used in each section, and how their experience varied between the sections. Commentary from Dr. Li Ding, who taught both SLA-aBLe and Non-SLA-aBLe sections of EGR115, will be provided to better understand differences between instruction methods. Dr. Ding will also demonstrate the techniques utilized in course instruction through audience engagement in a hands-on exercise.

IV. Lessons Learned & Next Step(s) (10 Minutes)

Workshop presenters will discuss the challenges and opportunities associated with SLA course implementation, upcoming steps in project development, and recommendations for future SLA course integration.

V. Project Materials (20 Minutes)

Presenters will share the project website, course PowerPoint videos, quizzes, surveys, and programming problems developed for this project. Components of course material development will be discussed, such as second language experience and blended learning design experience. Workshop participants will receive a flash drive containing these same materials used in class for implementation in their own institutions' future courses.

VI. Question & Answer Discussion (10 Minutes)

The workshop will conclude with a brief question and answer forum between the audience and presenters. Potential discussion topics may include programming language study, teaching experience, student perception, feedback, online course design techniques, and recommendations for future work. Audience participation is highly encouraged.

WORKSHOP SUMMARY

This workshop aims to provide the information and materials necessary to adapt programming language courses for first-year engineering students into a SLA format to improve student outcomes. SLA course instruction divides programming language content into familiar terms and exposes students to new concepts in a more intuitive manner. SLA participants are engaged in proven strategies and techniques through active discussion, collaboration, and sharing of experience. Workshop attendees will be engaged with the presentation through multiple interactive components such as group discussion and audience polls. Workshop participants will leave with a better understanding of programming course design and methods of SLA-based content delivery.

INTENDED AUDIENCE

This workshop offers the most benefit to individuals who work with first-year engineering students and courses. This workshop is primarily intended for audiences comprised of engineering education researchers, administrators, and faculty involved in the design and delivery of first year engineering programs or courses. However, the members of this project welcome feedback from all perspectives and hope to deliver continuous innovation in pedagogical strategies; members of all disciplines are welcome to attend.

ACKNOWLEDGMENT

This project and workshop are made possible by the support provided by the National Science Foundation, Division of Engineering Education and Centers, grant number EEC 1441825. Any conclusions, findings, recommendations, or opinions expressed in this project are those of the authors, and do not necessarily reflect the National Science Foundation's views. The authors would also like to acknowledge the contributions of Dr. Li Ding, Ms. Caroline, Liron, and Dr. Matthew Verleger, who assisted in conducting the project in their classes, Dr. James Pembridge, who offered suggestions regarding project design and implementation, and finally the Embry-Riddle Aeronautical Institution Research for their assistance in survey data collection.

REFERENCES

[1] Naraghi, M. H., & Litkouhi, B. (2001). An effective approach for teaching computer programming to freshman engineering students. *age*, *6*, 1.

[2] Jenkins, T. (2002, August). On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences* (Vol. 4, No. 2002, pp. 53-58).

[3] Solomon, J. (2004). Programming as a Second Language. Learning & Leading with Technology, 39(4), 34-39.

[4] Kathryn D. Sloane and Marcia C. Linn. Instructional Conditions in Pascal Programming Classes. In R. E. Mayer (ed), "Teaching and Learning Computer Programming", pp 207-235, Lawrence Erlbaum Associates, 1988.

[5] F. Marton and R. Säljö. On Qualitative Differences in Learning I: Outcome and Process. British Journal of Educational Psychology, Vol. 46, pp 4-11, 1976.

[6] Ortega, L. (2014). *Understanding second language acquisition*. Routledge.

[7] Panell, C. (2003). Teaching computer programming as a language. *Tech Directions*, *62*(8), 26.