Poster Abstracts – Session 1

ICER 2015

We are pleased to introduce a collection of posters as part of the ICER conference program. These posters intend to further expand the ICER community and spark discussion among conference participants. The goal is for these posters to provide a venue in support of both new ideas and newcomers to our community.

Like the main ICER program, the posters cover a wide range of computing education related work, and the presenters are from a variety of different institutions around the world. Short abstracts for each of the posters to be presented at this year's conference in Poster Session 1 can be found on the pages to follow.

Leo Porter
2015 Lightning Talks and Posters Chair
UC San Diego

Poster Session 1

1. *A Preliminary Study on Undergraduate Students' Learning Experiences While Solving Basic Cybersecurity Challenges*
   Ikseon Choi, Sejin Kim, Younseok Lee, and Kang Li
   The University of Georgia, USA

2. *Teaching an Introductory Undergraduate Course in Programming: Theory and Practice for Enhancing Cognition, Memory and Learning*
   Alexander Christou
   Indiana University, USA

3. *Policy Matters*
   William E. J. Doane
   IDA's Science & Technology Policy Institute, USA

4. *What Factors Influence Program Plans in CS1*
   Kathi Fisler
   Worcester Polytechnic Institute

5. *A Continuous Integration Framework for Promoting Software Engineering Best Practices*
   Sarah Heckman
   North Carolina State University, USA

6. *Applying Advanced Econometric and Marketing Techniques to Analyze Various Issues in Computer Science Education*
   Lianen Ji, China University of Petroleum, Beijing , China
   Jean Mayo and Ching-Kuang Shene, Michigan Technological University, USA

7. *What Does Intrepid Exploration Look Like Among Girl Pair Programmers?*
   Louise Ann Lyon
   ETR

8. *EIP: Engaging Laboratory Experiences for the Introduction to Programming course* Kathi Fisler
   José R. OrtizUbarri, Rafael ArceNazario, and Ivelisse Rubio
   University of Puerto Rico, Rio Piedras

9. *Determining the Impact of Teacher Professional Development on Perceived Ability to Teach a Computer Science Principles Course*
   Thomas Price, Veronica Cateté, Jennifer Albert, and Tiffany Barnes
   North Carolina State University, USA

10. *ClassTranscribe*
    Jia Chen Ren, Mark Hasegawa-Johnson, and Lawrence Angrave,
    University of Illinois at Urbana-Champaign, USA

A Preliminary Study on Undergraduate Students' Learning Experiences While Solving Basic Cybersecurity Challenges

Ikseon Choi, Sejin Kim, Younseok Lee, and Kang Li
The University of Georgia
ichoi@uga.edu

Abstract:
Teaching students to develop active knowledge and the skills to deal with real-world cyberattacks has been challenging. The overarching goal of this project is to develop meaningful learning resources and teaching methods for basic cybersecurity at a college level. We initially propose two assumptions. First, understanding how systems would be vulnerable to cyberattacks is one of the prerequisites for gaining successful cybersecurity expertise. Secondly, students will develop more meaningful knowledge and skills through solving problems in authentic contexts rather than through abstract lectures.

These assumptions led us to develop challenge-based learning resources that mimic the challenges used in Capture-the-Flag (CTF) competitions, to introduce undergraduate students to cybersecurity. CTFs are popular computer security competitions among hackers. The learning resources we developed are a series of stepwise, reusable challenges in the area of binary vulnerabilities, web security, cryptography, and network security. Each challenge includes the description of the security scenarios (for students), the setup instructions (for instructors), and the source code of the challenges, hosted at http://tunablectf.com

A preliminary evaluation study was implemented (1) to understand how students learn about cybersecurity while solving challenges and (2) to find ways to improve the learning resources and the implementation methods. A think-aloud protocol was used while three participants were independently or collaboratively solving a series of cybersecurity challenges. The preliminary results revealed that the students were engaged in various types of complex knowledge while completing the cybersecurity challenges, including conceptual/system knowledge, procedural knowledge, strategic knowledge, and creativity. Further anecdotal data indicated that these experiences awakened them to the importance of cybersecurity. We also observed that providing relevant information while they were solving challenges was helpful for their effective learning. A refined research method will be implemented to obtain deeper understanding of students' learning experiences and to improve the learning resources for effective cybersecurity education.

Teaching an Introductory Undergraduate Course in Programming: Theory and Practice for Enhancing Cognition, Memory and Learning

Alexander Christou
Indiana University
agchrist@indiana.edu

Abstract:
Relative to other academic disciplines, computer science has existed for a relatively short period of time and has focused on preparing students for jobs in the private sector. As a result, theories of learning in computer science have been underexplored. The formality and maturity of computer science education is still developing, both in theory and in practice. In this paper, contemporary theories of cognition, memory and learning will be discussed in the context of an introductory programming course intended for first-year undergraduate computer science majors.

The paper begins with an overview of three "grand" theories of learning as applied to programming education: associational theory, constructivist theory, and sociocultural theory. Next, strategies for enhancing student memory and retrieval in programming are explored, including contextualization and cumulative learning. Finally, the role of motivation and engagement is discussed as a method of preventing learning "bottlenecks" in programming. Strategies for increasing student motivation and engagement are proposed, including game-based assignments and regular feedback.

Policy Matters

William E. J. Doane
IDA's Science & Technology Policy Institute
Washington DC 20036
wdoane@ida.org

Abstract:
Public policy affects us all. It affects how and what we are taught [1]. It affects what we're allowed to research and what we're allowed to publish [2]. It affects our use and expectations of everyday technologies [3]. Public policy affects us as researchers, scientists, and citizens. It's vital that computationally trained individuals be involved in all levels of public policy discourse regarding the capabilities and uses of and restrictions on computational technologies.

Public policy is any action of the government that is enforced through social expectation or governmental authority. It is constructed by stakeholders who contribute their perspective to the discussion in a way that is both practical and addresses problems of societal relevance.

To engage in public policy debate requires only that you be willing to communicate your specialized knowledge so that the general public and policymakers alike can make better and better informed decisions.

Engaging computer science students in public policy discussions takes what they learn about computing technologies, asks them to bring both their critical thinking and communications skills to bear, and translates their knowledge into action for the public good.

Computer science education has a special role to play by broadening the range of voices involved in public policy debates regarding technology issues. We can invite students to examine critically the policy implications of a given technology; to explore how policy affects their work, lives, and community; or to engage directly in public policy internships and organizations.

[1] Lisa Kaczmarczyk & Renee Dopplick. 2014. Rebooting the Pathway to Success. ACM, New York, NY, USA.
[2] Federal Register. May 20, 2015. Wassenaar Arrangement 2013 Plenary Agreements Implementation: Intrusion and Surveillance Items. 80 Federal Register 28853.
[3] Kristina Peterson & Damian Paletta. June 2, 2015. Congress Reins In NSA's Spying Powers. Wall Street Journal Online Edition.

What Factors Influence Program Plans in CS1?

Kathi Fisler
(joint work with Shriram Krishnamurthi and Janet Siegmund)
WPI Department of Computer Science
kfisler@cs.wpi.edu

Abstract:
What factors influence students' program plans in CS1 and CS2? The constructs and program development techniques that students have learned are obviously significant, but other factors—such as choice of programming language, libraries taught, and general design principles—likely matter as well. At ICER 2014, Fisler presented data [1] showing that CS1 students using functional programming via the How to Design Programs [2] curriculum produced diverse high-level program plans on Soloway's classic Rainfall problem. Some (but not all) of this diversity correlated with the built-in functions that students used in their solutions.

To further explore the influences of curriculum and language on program plans, we conducted a follow-up study in two equivalently-positioned CS1 courses in the same department: one course used Racket and introduced data structures and Big-O analysis while the other focused on Java programming for interactive applications. Students produced code for three planning problems and preference-ranked given solutions for three different planning problems. Findings include richer appreciation for diverse program plans in the Racket course, a noticeable concern for efficiency in the Java-based course even though the course had not discussed performance, and additional evidence that library functions heavily influence program structure. We noticed discrepancies between students' valuing of efficiency and their use of library functions that would increase running time. This raises interesting questions about the cost models that students develop and how these models influence programming decisions.

[1] Kathi Fisler. 2014. The Recurring Rainfall Problem. In *Proceedings of the international conference on Computing education research (ICER '14)*. ACM, New York, NY, USA.

[2] Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. How to Design Programs. MIT Press, 2001.

A Continuous Integration Framework for Promoting Software Engineering Best Practices

Sarah Heckman
North Carolina State University
sarah_heckman@ncsu.edu

Abstract:
Software engineering best practices, like test driven development (TDD) and frequent pushes to version control, are common in industry and increasingly common topics in computer science courses. However, students may not apply these practices while completing coursework outside of the classroom. We are utilizing professional software engineering tools to promote the use of software engineering best practices and provide rapid feedback to students on the graded aspects of their work in three courses at NC State. There is also research into tools that promote the adoption of software engineering best practices by professional developers. We seek to combine and modify industrial-strength continuous integration tools and best practice adoption systems to encourage and enforce student adoption and use of software engineering best practices during in-class labs and assignments.

Our framework utilizes Jenkins for continuous integration, evaluation of student work, and collection of student practice data. Evaluation consists of pulling student code from version control, compiling the student's project, running the student's test cases, running teaching staff test cases, and generating feedback from tools like static analysis. Initial surveys show that 76% of students reported increased productivity and 84% of students reported increased code quality when using the framework. We will report on how students have used the tooling in practice through data mining metrics collected during project development. From the data, we will identify productive and unproductive software engineering best practice patterns and identify or create adoption systems to move students from unproductive to productive practices. For example, in a CS1.5 course offered in Fall 2015, we will check student test code for asserts before running teaching staff test cases to encourage TDD.

Applying Advanced Econometric and Marketing Techniques to Analyze Various Issues in Computer Science Education

Lianen Ji
Department of Computer Science and Technology
China University of Petroleum, Beijing
Changping, Beijing 102249
People's Republic of China
jilianen@cup.edu.cn

Jean Mayo and Ching-Kuang Shene
Department of Computer Science
Michigan Technological University
Houghton, MI 49931
USA
{jmayo,shene}@mtu.edu

Abstract:
What are the causes of my low teaching evaluations? Why are students leaving the CS departments? Why didn't students take my section of a multi-section course? These are typical and frequently asked questions. For decades studies have found that grade inflation, class size, class meeting time, level of difficulty of a course, toughness of an instructor, students' ability to answer evaluation questions honestly, etc. may AND may not affect evaluation, retention, etc. However, these issues have not been addressed fully by educators in the discipline of computer science. We will present a framework that uses advanced econometric and marketing methods to explore these issues on various fronts to determine the causation among various factors.

Two types of datasets are usually available. The *aggregate* datasets (e.g., average score of an instructor in teaching evaluation) lack the in-group variances due to aggregation. The *disaggregate* datasets are the raw responses from the subjects being surveyed. Simultaneous equation models, in which each equation models the cause of a particular endogenous variable (e.g., evaluation scores, student expected or received grades, etc.), will be used on the aggregate datasets. The more innovative part of our framework is the use of disaggregate (rather than the less accurate aggregate) data directly from students. We shall apply discrete (cardinal or ordinal) choice modeling techniques to study how students rate their instructors, what factors cause students to leave the CS discipline, etc. We shall compare the stated expectation (i.e., stated preference) and the actual outcome (i.e., stated choice). The conjoint analysis methods in marketing analysis will be used for survey design and data analysis, while discrete choice modeling will be used for modeling the choice behavior of students. Finally, we will study if our models and model structures can be used at other schools and institutions (i.e., transferability).

Acknowledgments. This work is partially supported by a grant from the China Scholarship Council (CSC No.201306445011) and grants from the National Science Foundation (DUE-1140512, DUE-1245310 and IIS-1456763).

What Does Intrepid Exploration Look Like Among Girl Pair Programmers?

Louise Ann Lyon
ETR
LouAnn.Lyon@etr.org

Abstract:
Pair programming (PP) has been widely touted as a promising strategy to engage students in computer science, and the assumption that girls like to collaborate suggests that PP should be a particularly effective means of attracting girls to the field. However, there is insufficient empirical evidence to show how pair programming might foster collaboration between middle school students and how working in pairs might inspire girls—in particular—to pursue computing. In this poster, we posit that PP can be used to foster what Sherry Turkle (cited in Margolis and Fisher [1]) has called "intrepid exploration" (IE) among girls and that the resulting experiences of fearlessness, curiosity, persistence, and innovation can embolden girls to pursue computing.

To date, we have collected video data of nine pairs of girls programming an Alice game during one semester of an after school course. We present here instances of IE that occurred in this class in order to better define and illustrate the concept. This lightning talk and poster open a dialog on what intrepid exploration looks like, under what conditions it appears, and what we can do to create settings that will support intrepid exploration among students—particularly girls—in the classroom while students use pair programming techniques to construct creative computer programs.

The larger study currently in progress aims to address the overall question: *What are the conditions under which PP can foster the kinds of thinking and problem solving that will prepare middle school students to pursue and persist in computing fields?* The study is being conducted in four middle schools—two predominately white middle class and two predominately Latino/a working class—to investigate variations in IE based on gender and cultural background.

[1] Jane Margolis and Allan Fisher. 2002. *Unlocking the clubhouse: Women in computing*. The MIT Press, Cambridge, Massachusetts.

EIP: Engaging Laboratory Experiences for the *Introduction to Programming* course

José R. Ortiz-Ubarri
jose.ortiz23@upr.edu

Rafael Arce-Nazario
rafael.arce@upr.edu

Ivelisse Rubio
iverubio@gmail.com

Department of Computer Science
University of Puerto Rico – Rio Piedras
PO Box 70377
San Juan, PR 00936–8377

Abstract:
There is a wealth of online educational materials for CS1 concepts. However their adoption in laboratory experiences is uphill, especially in resource-constrained departments. We present a set of well-structured laboratory experiences in C++ that are designed to be engaging, applied, meaningful, easily accessible and transferable. All materials are available in English and Spanish and provide a turnkey solution that can ease instructors' transition toward more hands-on, active and engaging courses.

The exercises use a framework that simplifies the implementation of computing tasks that students are not expected to understand at the CS1 level and could divert attention from the main concepts.

In one of the lab experiences, the students implement a green screen image processing app. They are provided with a GUI that allows them to load an image with a solid background and a background image. Throughout the exercise they apply their knowledge of nested loops, decision structures and expressions to implement an algorithm that complete the app and creates a third, merged image.

For each of the laboratories we administered pre, post tests and a questionnaire to assess student learning. The students also rated the experiences in terms of engagingness, applicability and meaningfulness. Results show that the students gain knowledge by performing the labs and found them to be engaging, practical and meaningful. The students also ranked the laboratories in order of preference, and the results show that there is not a common preference among all students which suggest that the variety of the applications capture the different interests or preconception that CS students have before taking the course.

[1] Molenda, Michael (May/June 2003). "In Search of the Elusive ADDIE Model". Performance improvement 42:5, May-June, 2003, pp. 34–36.
[2] Parlante, Nick, et al. "Nifty assignments." *Proceedings of the ACM Technical Symposium on Computer Science Education*. ACM, 2015.

Determining the Impact of Teacher Professional Development on Perceived Ability to Teach a Computer Science Principles Course

Thomas Price, Veronica Cateté, Jennifer Albert and Tiffany Barnes
NC State University
{twprice, vmcatete, jlsharp, tmbarnes}@ncsu.edu

Abstract:
During the summer of 2014, 57 teachers attended Professional Development (PD) training on the Beauty and Joy of Computing (BJC) curriculum for the AP Computer Science Principles course. The PD consisted of 6 weeks, with the first and last weeks held in person, and the middle four weeks consisting of projects and assignments in an online edX course. This poster evaluates the effectiveness of that PD through pre- and post-survey data collected from the 57 attendees at the NC State University and UC Berkeley sites.

We attempt to address the following questions in the poster through our evaluation of the PD:

- What aspects of the PD did teachers find effective and ineffective?
- How did the PD affect teachers' perceived ability and intentions to teach a CS principles course?
- How did a teacher's background and level of participation in the PD relate to the effectiveness of the PD?

We found that the PD significantly improved participants' perceived fluency in each of the PD's four core content areas (equity, content, inquiry/engagement and differentiation). Teachers completed BJC programming labs to understand the material from their students' perspective, and we saw the strongest improvement in teachers' perceived ability to use the SNAP programming language. We also found that teachers with prior CS experience showed a greater improvement in perceived knowledge of BJC content, indicating that some PD material may have been inaccessible to novices.

A primary goal of the PD was to encourage teachers to use the BJC curriculum in their classes; however, there was no significant increase in teachers' intentions to use BJC after the PD. Interestingly, there was little correlation between how much of the BJC curriculum the teachers intended to use before and after the PD, indicating that teachers' intentions did change, but not uniformly.

ClassTranscribe

Jia Chen Ren, Mark Hasegawa-Johnson, and Lawrence Angrave
University Of Illinois at Urbana-Champaign
jren4@illinois.edu, jhasegaw@illinois.edu, angrave@illinois.edu

Abstract:
ClassTranscribe is an open-source, web-based platform that leverages crowdsourcing to address the problem of accurate, reliable and fast transcriptions of lectures. Lecturers specify a particular audio or video file and a series of associated student transcribers. The file is then broken up into transcription tasks and distributed with redundancy amongst the student transcribers. Lastly, the completed tasks are intelligently reassembled into a complete transcription. Completed transcriptions provide search functionality that augments existing lecture recordings and enable enhanced educational features including closed captioning. The resulting conceptual understanding gained from the act of transcribing has been shown to be enticing [1] and serves as a core motivation for students to participate. ClassTranscribe has been deployed in the University of Illinois at Urbana-Champaign CS 241 Systems Programming course. 35 students collectively transcribed every CS 241 lecture recorded during the Spring 2015 semester using the ClassTranscribe platform. 141 students, or 68% of enrolled students in the course, used the transcription-enabled search and video captioning features on ClassTranscribe. On average, students used ClassTranscribe for 7 minutes and 50 seconds and averaged 8.85 searches over the course of the semester. Student search term frequency data illuminated areas of confusion and enabled more targeted and effective review sessions. Test scores of participating student transcribers were recorded and used to obtain preliminary results pertaining to correlations between learning and transcribing that will be presented at the conference. We have published the source code of ClassTranscribe online at [2] and a working demonstration at http://classtranscribe.com.

[1] Tim Causer, Justin Tonra, Valerie Wallace, “Transcription maximized; expense minimized? Crowdsourcing and editing The Collected Works of Jeremy Bentham,” Literary and Linguistic Computing, 27(2), pp. 119-137.
[2] ClassTranscribe, ClassTranscribe [Online] 2015. Available: http://www.github.com/cs-education/classTranscribe (Accessed: 29 May 2015)