Program at a Glance

Thursday, 10:00 a.m.—11:30 a.m.

Thursday, 3:00 p.m.—4:30 p.m.

Friday 10:00 a.m.—11:30 a.m.

Program at a Glance (cont.)

Friday 10:00 a.m.—11:30 a.m. cont.

Friday 3:00 p.m.—4:30 p.m.

Saturday 10:15 a.m.—11:45 a.m.

Computing Principles for All Students' Success (ComPASS)

Thursday, 10:00 a.m.—11:30 a.m.

Diane Baxter (UC San Diego)

The University of California, San Diego (UCSD), its San Diego Supercomputer Center (SDSC), and San Diego State University (SDSU), are expanding the computer sciences curriculum in San Diego County's high schools, community colleges, and universities. Through a project called 'Computing Principles for All Students' Success' or *ComPASS,* the team aims to strengthen the region's educational capacity for preparing high school and college students of all backgrounds and disciplinary interests to contribute to and participate in what has become a computationally driven economic future. *ComPASS* contributes to a nationwide goal of training approximately 10,000 high school teachers to teach advanced placement (AP) Computer Science (CS) Principles courses by the year 2015. This project strategically targets both in-service and pre-service teachers with professional development to teach CS Principles. The course provides a solid conceptual understanding of the ideas, logic, and principles that underlie computing, benefitting all students, not just the computer science majors. Taught at the college level to both freshman and senior levels, it will also provide an excellent background for soon-to-be teachers in all fields. And taught in high schools, it will prepare students for a wide variety of future careers with the problem-solving skills that will be essential for all students to succeed.

The *ComPASS* project is testing a sustainable model for introducing computer science principles into general education at the high school level, evaluating strategies and methods for preparing teachers to teach computer science while developing a strong peer support network in partnership with the San Diego chapter of the Computer Science Teachers Association (CSTA) of the Association of Computing Machinery (ACM). Together, they are also gaining support for the value of this course among college-bound students, their parents, and their school administrators.

ComPASS is supported by NSF grants 1138512 (UC San Diego; D. Baxter, P.I.) and 1138492 (SDSU; L. Beck, P.I.).

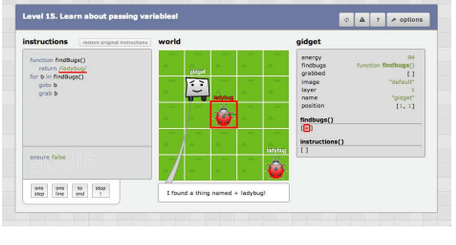
CER: Collaborative Research: Computing Education through Collaborative Debugging

Thursday, 10:00 a.m.—11:30 a.m.

Andrew Ko, Margaret Burnett, and **Catherine Law** (University of Washington)

This research project is investigating how to teach debugging and computing skills in personalized, social, online learning settings. We are creating an interactive website that presents a series of debugging puzzles to learners, allowing them learn diagnostic strategies and computing concepts in a self-paced manner and in collaboration with their peers. The website will also allow learners to create their own debugging puzzles to share with friends, family, and the world, resulting in an online community of social learning. Using this design, the project will investigate the effect of explicit instruction on debugging strategies on learners' ability to learn basic programming concepts and the effect of social learning features on learning outcomes and engagement. The project will also contribute basic discoveries on how to teach debugging and how such teaching can be integrated into instruction on other aspects of computing. Evaluations will include both controlled laboratory experiments, annual summer camps involving over 200 U.S. teens, and a worldwide deployment of the website.

This project is part of a larger effort to increase both computing literacy and interest in computing careers on the part of U.S. teens. This project's focus on social learning is particularly important for broadening participation in computing, allowing teens to learn computing in supportive and collaborative settings, rather than the highly competitive settings that tend to deter many teens today. Moreover, because the website will only require an Internet connection and a web browser, and not software installations or special devices, the project will also improve access to effective, evidence-based computing education, reaching learners who only have Internet access at public libraries or at school. These efforts will not only improve U.S. citizens' global competitiveness for information technology jobs, but also help the broader citizenry better understand and debug the increasingly software-based world around them.



Inclusive Exploring CS Curriculum Enhancement as Face-to-Face and Online Support for Visually Impaired, High School Students

Thursday, 10:00 a.m.—11:30 a.m.

Stephanie Ludi (Rochester Institute of Technology)

The I-ECS project strives to increase the participation of students with visual impairments through better preparation and support. The overall goal is to increase access to an established curriculum (Exploring Computer Science) and increase success within the area of computing for students with visual impairments at the secondary level. The structured approach, building on the foundation of the ECS curriculum, will enable those with visual impairments to have greater access to the diverse areas of computer science through face-to-face and online resources that foster continuous learning and community, as the means to transition to college degrees in computing. The project components are designed to promote growth by all participants, including parents. The educator seminar and online community will foster greater accommodation of visually impaired students in the computer science classroom.

Game-Themed CS1/2: Empowering the Faculty

Thursday, 10:00 a.m.—11:30 a.m.

Kelvin Sung, Jason Pace (U. Washington Bothell)**, and Michael Panitz** (Cascadia Community College)

Despite the proven success of using computer video games as a context for teaching introductory programming (CS1/2) courses, barriers including the lack of adoptable materials, required background expertise (in graphics/games), and institutional acceptance still prevent interested faculty members from experimenting with this approach. Our project is designed specifically to address these issues by conceiving, designing, implementing, and testing CS1/2 game-themed teaching modules with accompanied tutorials that enable faculty to gradually acquire new knowledge and skills with minimal overhead while enlisting the institutional support to successfully implement and deploy these educational innovations. The teaching modules will abstract and hide the details of computer graphics and games so that neither the adopting faculty nor the students need to have any background in these areas. Additionally, these teaching modules will be self-contained and limited in curriculum scope so that faculty members can pick and choose a subset to integrate into their existing classes for limited scope experimentation. The accompanied tutorials will detail the implementations of the modules such that faculty can learn to develop their own game-themed modules over time. This work is described as both student- and faculty-centric because it is designed to address the needs of both.

The following figure shows a simple example of super breakout game. The three lines in the InitializeGame() function creates the game. This example shows how breakout games can be used as a vehicle for teaching conditional statements in a CS1 class.

```
void InitializeGame() {
   LifeSet.Add(5);    // 5 life
   PaddleSet.Add(1);  // 1 paddle
   BlockSet.Add(21);  // 21 blocks
}
```

CT4TC—Computational Thinking for Teaching Computing: Validating a Theory of Broadening Participation

Thursday, 3:00 p.m.—4:30 p.m.

Alex Repenning (University of Colorado)

A fundamental challenge to computer science education is the difficulty of broadening participation of women and underserved communities. The idea of game design and game programming as an activity to introduce children at an early age to computational thinking in a motivational way is quickly gaining momentum. A pedagogical approach called Project First has allowed the Scalable Game Design project to reach a large group of middle schools students including a large percentage of female (45%) and underrepresented (48%) students. With over 8000 students in inner city, remote rural, and Native American communities Scalable Game Design has investigated the impact of pedagogical approaches employed by teachers such as mediation and scaffolding onto students' levels of interest. Findings suggest strong gender effects based on classroom scaffolding approaches. For instance, girls are substantially less likely to be motivated through scaffolding based on direct instruction. Conversely, guided discovery scaffolding approaches are highly motivating to the point where they can even overcome other negative predictors such as small girls to boys class participation ratios. This presentation introduces the project, discusses different scaffolding approaches and presents data connecting gender specific motivational levels with scaffolding approaches.

SCALABLE GAME DESIGN

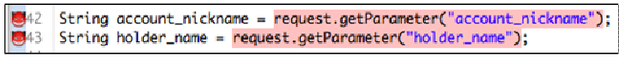
Supporting Secure Programming in Education in the IDE

Thursday, 3:00 p.m.—4:30 p.m.

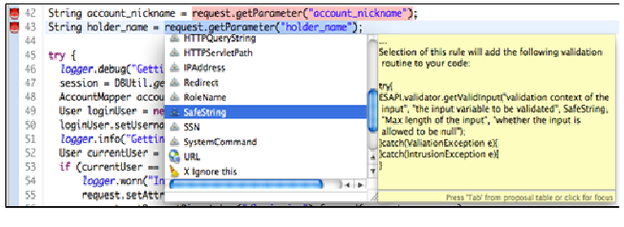
Bill Chu and **Michael Whitney** (UNC Charlotte)

Software flaws are a root cause of many of today's information security vulnerabilities. Current curricula emphasis on traditional information security issues does not address this root cause. We propose educating students on secure programming techniques through interactive tool support in the Integrated Development Environment (IDE). We believe this approach can complement other curricula efforts by teaching and providing continuous reinforcement of practices throughout programming tasks. In this paper, we evaluate our prototype tool, ASIDE, which provides instant security warnings, detailed explanations of vulnerabilities, and code generation. We report the results of an observational study on 20 students from an advanced Web programming course. The results provide early evidence that our tool could potentially help students learn about and practice secure programming in the context of their programming assignments.

The ASIDE prototype is a proof-of-concept Eclipse plugin for Java, which integrates secure programming support and education into the IDE. ASIDE works as a long running process in the background and scans a selected project for code patterns that match pre-defined heuristic rules of security vulnerabilities. ASIDE is an OWASP open source project. Presentations, papers and source code can be found at https://www.owasp.org/index.php/OWASP_ASIDE_Project.



ASIDE Vulnerable Code Warnings with Suggestions



CS Unplugged: Encourage Computing without Computers

Thursday, 3:00 p.m.—4:30 p.m.

Cyndi Rader, **Tracy Camp** (Colorado School of Mines), and **Wendy DuBow** (NCWIT)

Many countries, including the United States, have recently faced the problem of declining numbers of students majoring in computing, despite an increasing demand from employers for such skills. The Computer Science Unplugged activities provide ways to expose students to computer science (CS) ideas without using computers. Many K-12 school students have had limited exposure to CS concepts, and therefore, have a very poor understanding of what computing education and careers involve. The CS Unplugged activities teach students the kind of thinking that is expected of a computer scientist via activities, games, magic tricks, and competitions. The Unplugged activities involve problem solving to achieve a goal, and in the process introduce students to fundamental concepts from computer science.

Many believe teaching computing concepts without computers is effective in increasing students' confidence that they can understand computing concepts, as well as their interest in computing careers. Thus, CS Unplugged has recently enjoyed widespread adoption around the world, e.g., CS Unplugged has been translated into 12 languages and is recommended in the ACM (Association for Computing Machinery) K-12 curriculum. *But does it work?* This effort will deploy and evaluate CS Unplugged activities in five middle schools in order to determine whether students understand the fundamental computing concepts presented to them and show an increased interest in computing (with special attention to effects on female and racial/ethnic minority students), and whether teachers can effectively incorporate CS Unplugged activities into their classrooms. Depending on the results, in the third year of the grant this effort may be extended to middle schools across the state.

A Curriculum-wide Software Development Case Study

Thursday, 3:00 p.m.—4:30 p.m.

Massood Towhidnejad (Embry-Riddle Aeronautical University)

An excellent way to address the challenges of introducing real-world exposure through a curriculum is the use of a robust life-cycle engineering case study – that is, a case that engages the student in the engineering of a complex system throughout its "life": system definition, preliminary and detailed design, implementation, verification and validation, and operation and maintenance.

The case study project has concentrated on building a foundation for full development: research into case study teaching; identifying a case study problem; creating a scenario framework; describing the launch of the software development team; fashioning a software development plan to guide development of the DigitalHome System; establishing a development process; creation of a DigitalHome need statement; analysis, modeling and specification of the DigitalHome requirements; development of a system test plan; development of a software architecture; and specification of the system components.

A set of DH mini-case studies (that we call "case modules") have been created; they are designed to engage students in active learning software engineering activities related to the DigitalHome System. With this life-cycle engineering case study, we advocate a "Team Learning Format". Our approach involves the following activities:

- Students are assigned prior reading or other preparation.
- The instructor introduces the case, providing motivation and giving background.
- The instructor divides the class into teams, and if fitting, assigns roles.
- Teams work on a case module exercise: discussing problems/issues, answering questions and making decisions, and prepare a report of their conclusions.
- Teams present their report to the class, and the class discusses results.

Existing DigitalHome scenarios, artifacts, and case modules can be viewed at http://www.softwarecasestudy.org/.

Production and Assessment of Student-Authored Wiki Textbooks

Friday, 10:00 a.m.—11:30 a.m.

Jennifer Kidd (Old Dominion University) and
Ed Gehringer (North Carolina State University)

Traditionally, students study from textbooks written by "experts" in the field. But there are important pedagogical advantages to having them write part or all of their textbook. Until Web 2.0, however, student-authored textbooks were infeasible because of the overhead in reviewing contributions and making them available to the rest of the class. A wiki meets both of these needs very nicely. But substantial administrative overhead remains. Our Expertiza system has features for managing this overhead. Students sign up for topics, and form teams, online through the application. Individual students review the work of teams, based on a rubric that may have multiple sections and multiple kinds of questions (numeric ratings, checkbox, prose answer). Authors can communicate with reviewers in double-blind fashion. There can be multiple rounds of review, allowing authors a chance to revise their work in response to reviewer comments. Either the instructor or other students can metareview reviews submitted by students. Team members can rate each other on contributions to the team. All of this data is available to help the instructor assess the work of each student.

Expertiza has been used to produce wikitexts in a half-dozen different courses. Large majorities of students report that they put a lot of effort into their work, benefit from reviewer feedback, and learn a lot from writing their textbook chapters and from reading their fellow students' contributions.

Establishing the Exploring Computer Science Course in Silicon Valley High Schools

Friday, 10:00 a.m.—11:30 a.m.

Ralph Morelli and **Chinma Uche** (Trinity College)

The Mobile CSP project is a partnership between Trinity College, the Hartford Public School System, the Connecticut Chapter of the Computer Science Teachers Association, and other Hartford area high schools. The goal is to design and develop an Advanced Placement (AP) computing course based on a Mobile Computer Science Principles curriculum, *Mobile CSP*, which was tested at Trinity College and the Greater Hartford Academy of Mathematics and Science (GHAMAS) as Phase 2 pilot courses of the College Board's CS Principles project. The Mobile CSP curriculum uses the new mobile computing language, *App Inventor for Android*, to provide a rigorous, programming-based introduction to computational thinking. The project will train and support teachers in participating Connecticut high schools to implement the Mobile CSP curriculum starting in summer 2013. The presentation will highlight some of the resources that will be used in the summer training.

Preparing CS Students for the Multicore Era: Teaching Parallel Computing in the Undergraduate Curriculum

Friday, 10:00 a.m.—11:30 a.m.

Apan Qasem, Martin Burtscher, Wuxu Peng, Hongchi Shi, Dan Tamir (Texas State Univ.)

The widespread deployment of multicore-based computer systems over the last decade has brought about drastic changes in the software and hardware landscape. However, most undergraduate computer science (CS) curricula have not embraced the pervasiveness of parallel computing. In their first years, CS undergraduates are typically exclusively trained to think and program sequentially. However, too firm a root in sequential thinking can be a non-trivial barrier for parallel thinking and computing. Thus, there is an urgent need to teach multicore and parallel computing concepts earlier and often in CS programs.

This project addresses the rapidly widening gap between highly parallel computer architectures and the sequential programming approach taught in traditional CS courses. It proposes to systematically integrate parallel computing into current undergraduate curricula. Specifically, its goals are to develop course modules and projects for introducing parallel computing concepts in several early computer science courses, to design an upper-level multicore programming course that serves as a capstone for parallel computing concepts, and to promote this model by making all relevant material freely available.

The enhanced curriculum will equip CS students with skills that are highly sought after by the computing industry. The planned outreach activities will help broaden the participation of female and Hispanic student groups in STEM education. The free teaching modules and the dissemination thereof at professional meetings will encourage the building of a community of educators interested in introducing parallel computing in the undergraduate curriculum. Finally, the developed teaching material will be contributed to ACM/IEEE's model CS undergraduate curriculum.

The Next Generation of Practice Exercises for Computer Science I

Friday, 10:00 a.m.—11:30 a.m.

Amruth Kumar (Ramapo College of New Jersey)

Problets are web-based software tutors designed to help *Computer Science I* students learn programming concepts by solving problems. They present problems, grade the student's answer and provide instant feedback.

- **Problems:** Problets present problems such as debugging programs, predicting their output, predicting their state, and evaluating expressions. They are adaptive – they present problems on only the concepts that the student does not already know. This helps minimize the problems solved by the students while maximizing their learning. The problems are randomized - typically, no two students get the same problem, and no student gets the same problem twice.
- **Feedback:** Problets explain the correct answer to each problem step-by-step. In controlled evaluations, this unique feature of problets has been shown to improve learning.
- **Usage:** Instructors can use problets for closed lab exercises, after-class assignments, and in-class testing. Since problets run in any Java-enabled browser, students can use them 24X7 to learn on their own time, at their own pace, and as often as they please. A typical problet takes 30-40 minutes to complete. Problets are free for educational use.
- **Instructor support:** Problets require no software installation. A dedicated web site is set up for each adopter. Adopters direct their students to use problets off this site – they decide which problets to use and when. After their students use each problet, they can request a report, which is provided in Excel format.
- **Topics:** Problets are available for expressions (arithmetic, relational, logical, assignment, bit-wise), if-else, switch, while, for, do-while, advanced loop concepts, functions, arrays, classes and C++ pointers. They are available for C, C++, Java and C#.

For additional information, and to try out problets, please visit http://www.problets.org. If you are interested in using problets, please send email to Amruth Kumar, amruth@ramapo.edu

Acknowledgments: This work was supported in part by the National Science Foundation under grant DUE-0817187.

OpenDSA: Open Source Interactive Data Structures and Algorithms

Friday, 3:00 p.m.—4:30 p.m.

Clifford A. Shaffer (Virginia Tech)

The OpenDSA Project's goal is to develop a complete online interactive textbook for data structures and algorithms (DSA) courses. Once completed, OpenDSA will include:

- Hundreds of instructional modules, where each module is the equivalent to one topic, such as one sorting algorithm, corresponding to a couple of pages in a standard textbook.
- Every algorithm or data structure is illustrated by an interactive algorithm visualization. Students could enter their own test cases to see how the algorithm or data structure works on that input, and they can control the pacing of the visualization.
- Every module contains multiple interactive assessment activities that give students immediate feedback on their proficiency with the material. This means many hundreds of exercises.

We will accomplish this through an open-source, creative commons environment. "Open Source" means that not only can anyone use the materials, they can also access the source code that generates the materials. "Creative Commons" means that anyone has permission to modify or remix the materials for their own purposes. You as an instructor (or even a student or professional who wants to self study) will be able to pick and choose from the selection of modules and exercises, automatically generating a custom textbook that contains exactly the topics you want. Since all materials are open source, you can rewrite any part if you don't like what is already there. Infrastructure will be included that lets you register students and then track their progress through the modules and exercises.

A major complaint of students in DSA classes is that they do not get enough practice problems, or sufficient other means of testing their proficiency. One of the most important aspects of our vision is a rich set of exercises to ensure that the student understands the material as he/she progresses through the book. Our modules will contain a mix of content, visualizations, and exercises. We make extensive use of the Khan Academy exercise infrastructure to build interactive exercises.

Transforming the Freshman Experience of Computing Majors

Friday, 3:00 p.m.—4:30 p.m.

Penny Rheingans and **Marie desJardins** (UMBC)

We describe a project to develop, deliver and evaluate a new first-year course for computing majors designed to increase retention, completion, and success among students, especially women and those from underrepresented groups. This course includes specific elements designed to achieve seven core learning goals: (1) increase understanding of the discipline, in terms of different majors and careers, (2) clarify students' personal interests and motivations about their choice of major and career, (3) increase confidence, self-efficacy, and community, (4) expose students to, and allow them to practice, key design and development skills, (5) strengthen writing, presentation, and teaming skills, (6) teach skills in problem solving, algorithmic analysis, and computational thinking, and (7) help students learn how to study effectively and how to access campus academic resources.

The approach synthesizes elements from successful first-year engineering courses, traditional introductory computing courses, general first-year seminars, and the new AP CS Principles course. Course content is woven into four curricular arcs designed to provide an overview of the discipline, build technical skills, provide a group design experience, and strengthen professional skills. Learning infrastructure includes collaborations among experienced computing faculty, staff members with student affairs experience, peer mentors to facilitate cohort-building and give informal advice, and undergraduate peer teachers to directly support learning.

Peer Instruction in Computer Science

Friday, 3:00 p.m.—4:30 p.m.

Beth Simon (UC San Diego)

What a course “is” and “does” can be viewed through the lens of instructional design. Any course should be based around the learning goals we have for students taking the course – what it is we want them to know and be able to do when they finish the course. Describing how we go about supporting students in achieving those goals can be broken into two parts: a) the content (e.g. for loops, methods) and materials (e.g. Java or Scratch) we choose and b) the methods or pedagogical approaches (e.g. lecture, lab-centric) we employ.

Peer Instruction (PI) is one such method that supports active learning -- on which this project focuses. In a PI-designed course, students prepare before class (often by reading the textbook and completing a quiz), and lecture is based around a series of multiple choice questions designed to engage students in developing deep understanding of core concepts. During the lecture a question is posed, students think about it individually and vote on an answer (often using a “clicker”), then discuss in peer groups, and then re-vote after discussion. A class-wide discussion follows where students are encouraged to share their reasoning and the instructor can model her cognitive processes.

Many of our resources can be found at http://peerinstruction4cs.org Here at SIGCSE 2013 we are presenting several papers in conjunction with this project. They document:

- The adoption of Peer Instruction in an introductory Matlab course for non-majors
- A look at reading quizzes as supporting Peer Instruction
- A 50% reduction in course fail rates via a study of 16 instances of 4 computing courses (theory of computation, architecture, CS2, and CS1)
- A 6% increase in final exam score for students taking a Peer Instruction CS0 based on a quasi-experimental study comparing students in the same course using standard lecture methods.
- A 30% increase in retention in the computing major after switching our CS1 to adopt a trio of best practices including Peer Instruction, Pair Programming, and Media Computation.

Transforming Experience of CS Software Dev through Multiplayer Online Game Classroom Collaboration in Industrial Format

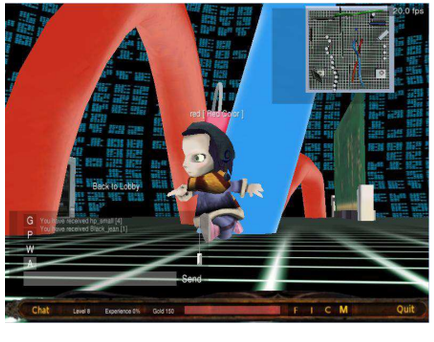
Friday, 3:00 p.m.—4:30 p.m.

Ilmi Yoon (San Francisco State University)

This innovative software development course is developing and testing a novel approach to computer science instruction at San Francisco State University and Cal State University Los Angeles. In classrooms emulating typical industry work environments, students are collectively creating and building a Multiplayer Online Game using a variety of complex software components. In the process, students are developing: (1) effective communication, presentation, collaboration and trouble-shooting skills; (2) heightened motivation to master core computer science content; (3) a robust end product; and (4) practical insights to prepare them for successful careers in industry.

The scalable game programming course includes a suite of instructional materials with accompanying documentation, all of which are being made freely available to the educational community. Materials can be flexibly adopted by computer science instructors including those who do not have sufficient background to develop multiplayer online game programming or senior-level software design courses on their own.

The project is a collaboration of two campuses in the California State University--one of the nation's largest and most ethnically diverse systems of higher education--and is being disseminated regionally and nationally. The innovative course design appeals to students who have not traditionally been attracted to STEM education, including many from minorities underrepresented in the sciences that enroll in large numbers at the two pilot campuses. The project reaches out to the public by making the student-produced games freely available, thereby playfully exposing non-scientists to essential computer science concepts and potentially inspiring young people to consider careers in software development.



Collaboration across Disciplines and Organizations: Enhancing Research on Diversity and Equity in K-12 CS Education

Saturday, 10:15 a.m.—11:45 a.m.

Jill Denner (ETR Associates), **Linda Werner** (UC Santa Cruz), and **Shannon Campe** (ETR Associates)

Addressing issues of equity and diversity in the K-12 computer science pipeline requires strong collaborations between computer scientists, social and learning scientists, and educators. But these partnerships are not easy to form or sustain. The co-authors have been working together and with other partners on NSF-funded projects for over ten years. These collaborations reach across organizations, including non-profit research agencies, schools, community-based programs, and faculty and students from multiple departments in universities.

We will showcase specific ways these collaborations strengthened our efforts to increase diversity and equity in K-12 CS, and how to build effective collaborations. This includes the development of pedagogical approaches to engage students in computational thinking, learning assessments, exemplary strategies for increasing diversity in computing, and research findings on a range of topics, including the computer science concepts that students learn by game programming. The presentation will include examples of student games, game programming curriculum, and research-based approaches to integrating social justice into K-12 computer science curriculum and fostering effective pair programming. Projects include:

- *Computer Science for the Social Good: Using Near-Peers to Engage Latino/a Students* CNS 1240756
- *Girls Creating Games: Increasing Middle School Girls' Interest in Technology* HRD 0217221
- *Computer Game Design: An Interdisciplinary Approach to Addressing Underrepresentation in Computing* DRL-1042944
- *The Development of Computational Thinking among Middle School Students Creating Computer Games* DRL 0909733
- *The Computer Science Collaborative Project* CNS 0940646

Casting a Wide Net: Applied Computational Thinking in STEM

Saturday, 10:15 a.m.—11:45 a.m.

David Weintrop, Elham Beheshi, and Kemi Jona (Northwestern University)

In this project, we are developing computational thinking (CT) activities to be embedded throughout existing high school STEM curricula. Our approach casts a wide net, introducing a diverse range of students (including those from traditionally underrepresented groups) to CT skills through courses that they are already taking. Our goal is to spark student interest in CT coursework and career routes by portraying CT as a vital and broadly applicable skillset that pervades modern STEM fields. By blending our CT instruction with STEM content, our activities have a place in existing school curricula, reach a broad student audience, and ground CT concepts in STEM materials teachers already have experience teaching.

Thus far, we have completed a number of key goals. The first has been the development of a Computational Thinking in STEM (CT-STEM) taxonomy. The taxonomy catalogs the CT skills that are central to STEM disciplines. The taxonomy was derived through analyzing CT-STEM activities collected from a variety of sources and through interviews with STEM practitioners engaged in CT. Second, we have developed and refined 16 classroom-ready CT -STEM activities across the disciplines of math, chemistry, biology, and physics. In summer 2012, we conducted two professional development workshops for area teachers, from which we recruited 12 pilot teachers who are currently testing our CT-STEM activities in their classrooms. Along with the development of the taxonomy, and our CT-STEM activities, we are also developing CT-STEM assessment items that we will use to measure students' learning of CT-STEM skills.

This work is supported in part by National Science Foundation under NSF grant CNS-1138461. However, any opinions, findings, conclusions, and/or recommendations are those of the investigators and do not necessarily reflect the views of the Foundation.

Developing Elementary (Learning) Progressions to Integrate Computational Thinking

Saturday, 10:15 a.m.—11:45 a.m.

Diana Franklin (UC Santa Barbara)

Computing will become an increasingly important part of driving innovation, and the next generation of inventors need to understand how to develop and use software. The ability to analyze and express problems in a way compatible with software solutions will become critical. Schools need to facilitate students' development of computational thinking skills.

There are many questions that need to be answered in order to develop children's computational thinking skills. Current work has explored how students can be taught computer science as well as proofs of concept that show elementary school students are capable of programming in elementary school. In order to advance, two fundamental questions need to be answered. First, how do children learn computational thinking? That is, what is the logical learning progression children follow so that we know the order in which skills should be developed? Second, how can we integrate computational thinking lessons with existing curricula in a way that engages students, builds confidence, and teaches them computational thinking?

We propose first steps in answering both questions. Computational thinking itself is very complex, with elements of algorithms, abstraction, modeling, and computation. With the CSTA CS K-12 computing standards as a beginning framework, we will develop a learning progression for aspects of computational thinking using empirical evidence for grades 4-6.

We will begin by formulating low anchor points for the two populations. These will be formulated based on interviews with lower and upper elementary students from schools with varying demographics. We will then develop learning progressions based on student learning of using existing elementary school curriculum resources developed for K-6 and for 4-6. We will use the anchor points and learning progressions to develop a complete curriculum, with special emphasis on delivery methods known to appeal to students with diverse backgrounds, as well as examples and projects that students with varying backgrounds can relate to.

Developing Faculty Expertise in Information Assurance through Case Studies and Hands-on Labs

Saturday, 10:15 a.m.—11:45 a.m.

Xiaohong Yuan, Kenneth Williams, Huiming Yu (NC A&T State)
Li Yang, Joseph Kizza, Kathy Winters (UT Chattanooga)
Bill Chu (UNC Charlotte)

Incorporating hands-on exercises and case studies is a proven effective pedagogy that can increase student interests and enhance their learning experience. However, they have not been widely adopted in IA education because their development takes significant amount of time and effort. We are implementing two faculty summer workshops to provide training for 36 faculty members with hands-on exercises and case studies on IA with the objective to build faculty capacity, increase student interest and learning, and increase partnership among instructors in IA. The instructors are active educators and researchers whose areas of expertise complement each other and broadly span the scope the IA knowledge domain. The workshop topics cover cryptography, security ethics, security policy, digital forensics, access control, security architecture and systems, network security, risk management, attack/defense, and secure software design and engineering. The PIs will help workshop participants design a personalized plan to integrate IA hands-on labs and case studies into their curriculum, and provide continuous follow-up assistance.

We successfully held the first workshop in the summer of 2012. Nineteen faculty members attended the workshop. Workshop participants included current IA faculty or those from institutions with a strong interest in, and with appropriate institutional commitment to introducing hands-on exercises and case studies during the academic year following each workshop. We are conducting on-going assessment on the effectiveness of the hands-on exercises, visualizations, and case studies in cooperation with workshop participants.

Topic Index

Topic Index

Topic	Sessions
Case Studies	Thurs 3:00, Sat
Computational Thinking	Thurs 3:00, Sat
CS Principles	Thurs 10:00, Fri 3:00
Curriculum Development	Thurs (all) and Fri (all)
Digital Textbooks	Friday (all)
Diversity	Thurs (all), Fri 3:00, Sat
Faculty Development	Thurs 10:00, Sat
Games	Thurs (all), Fri 3:00, Sat
Introductory CS	Thurs 10:00, Fri (all)
K-12 Education	Thurs (all), Fri 10:00, Sat
Pairs / Peer Learning	Thurs 10:00, Fri (all)
Teaching Methods	Thurs 10:00, Fri (all), Sat
Tools	Thurs 3:00 and Fri 10:00

Meet with Program Officers

This year, NSF Program Officers will be available during most presentation sessions in the meeting area outside the NSF Showcase booth. A specific schedule of times and Program Officers is available at the NSF Showcase booth.

While there will be some open time to meet with Program Officers, some will work by appointment.

Please visit http://www.cs.virginia.edu/~sherriff/nsfshowcase to sign up for an appointment time to meet with a Program Officer or come by the NSF Showcase booth.

About the NSF Showcase

Every year, twenty projects that are currently being sponsored by NSF are asked to present their work in an interactive, personal format during the break sessions and open slots at SIGCSE. The SIGCSE Symposium provides a forum for educators from K-12 through college to discuss issues and new ideas related to the development and implementation of computing curricula, along with other elements of teaching and pedagogy. The goal of the showcase is to share information about programs and research opportunities that attendees might not otherwise hear about.

The NSF Showcase is an annual event at the SIGCSE conference. If you are working on an NSF computer science education related grant, or have recently completed one and would like to present at the showcase, please contact Mark Sherriff or Aaron Bloomfield. The selection process will begin in the fall.

http://www.cs.virginia.edu/~sherriff/nsfshowcase

Mark Sherriff - sherriff@virginia.edu
Aaron Bloomfield - aaron@virginia.edu

For more information, please visit:

http://www.cs.virginia.edu/~sherriff/nsfshowcase

Cover Image:
Denver Skyline
Credit: GoodFon.com