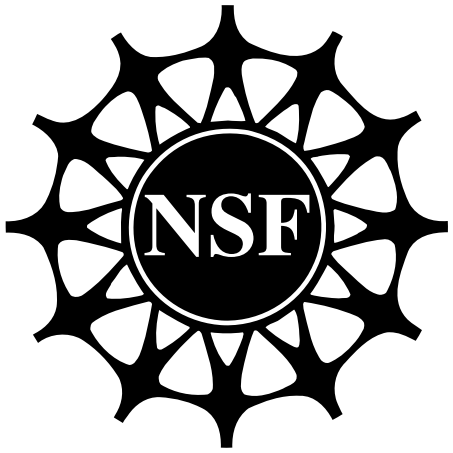

The NSF CCLI showcase is a yearly event at the SIGCSE conference. If you are working on an NSF CCLI grant, or have recently completed one and would like to present at the showcase, please contact Aaron Bloomfield at asb (at) cs (dot) virginia (dot) edu. The selection process will begin in the early fall.

http://www.cs.virginia.edu/~asb/nsfcclishowcase

NSF
CCLI Project Showcase

SIGCSE 2007
Covington, Kentucky

March 7-10, 2007

Booth 231
Exhibition Hall

National Science Foundation
Course, Curriculum and Laboratory Improvement Program

Program at a Glance

Thursday, 10:30 a.m.—12:00 p.m.

The Discrete Math Concept Inventory Project (page 4)
Vicki Almstrum, Southwestern University
David Klappholz, Stevens Institute of Technology

The Development of Student Electronic Portfolios for Curriculum Improvement in Practice-Oriented Biology and Computer Science Programs (page 5)
Kostia Bergman, Northeastern University
Veronica Porter, Northeastern University
Viera K. Proulx, Northeastern University
Mel Simms, Northeastern University

Media Computation as an Approach to Attract and Retain Students (page 6)
Mark Guzdial, Georgia Institute of Technology

Friday 10:30 a.m.—12:00 p.m.

CLICS: Computer Security Workshops for Faculty (page 7)
Paul J. Wagner, University of Wisconsin, Eau Claire
Andrew T. Phillips, University of Wisconsin, Eau Claire

An Interactive Approach to Formal Languages and Automata with JFLAP (page 8)
Susan H. Rodger, Duke University

Redesigning Introductory Computing: The Design Discipline (page 9)
Stephen Bloch, Adelphi University
John Clements, California Polytechnic University, San Luis Obispo
Kathi Fisler, Worcester Polytechnic Institute
Matthew Flatt, University of Utah
Viera K. Proulx, Northeastern University

Problets: Online Programming Tutors for Computer Science I (page 10)
Amruth Kumar, Ramapo College of New Jersey

Friday 2:00 p.m.—3:30 p.m. (continued on next page)

TinkerNet and TinkerNet 2 (page 11)
Michael Erlinger, Harvey Mudd College

SEED: Developing Instructional Laboratories for Computer SEcurity EDucation

Saturday, 10:30 a.m.—12:00 p.m.

Wenliang Du, Syracuse University

ꕥ • ꕥ

The security and assurance of our computing infrastructure has become a national priority. To address this priority, higher education has gradually incorporated the principles of computer and information security into the mainstream undergraduate and graduate computer science curricula. To achieve effective education, learning security principles must be grounded in experience. This calls for effective laboratory exercises (or course projects). Although a number of laboratories have been designed for security education, they only cover a small portion of the fundamental security principles. Moreover, their underlying lab environments are different, making integration of these laboratories infeasible for a semester-long course. Currently, security laboratories that can be widely adopted are still lacking, and they are in great demand in security education.

We have developed a novel laboratory environment (referred to as SEED). The SEED environment consists Minix, an instructional operating system (OS), and Linux, a production OS. It takes advantage of the simplicity of Minix and the completeness of Linux, and provides a unified platform to support a rich set of laboratories for computer security education. Based on the SEED environment, we have developed a list of laboratories that cover a wide spectrum of security principles. These labs provide opportunities for students to develop essential skills for secure computing practice. We have been using these labs in our courses during the last four years.

Program at a Glance

The Discrete Math Concept Inventory Project

Thursday, 10:30 a.m.—12:00 p.m.

Vicki Almstrum, Southwestern University
David Klappholz, Stevens Institute of Technology

ꕥ • ꕥ

The focus of the Discrete Math Concept Inventory project is to study how student learning of Discrete Mathematics (DM) is influenced by pedagogy (e.g. the way DM instruction is sequenced and integrated with instruction in the Computing core) and attitude towards DM (e. g. student assessment of how and whether mastery of DM will contribute to their intended careers, generally in IT). Immediate objectives are to construct and validate two types of tools for use in the study: a Views About Discrete Mathematics instrument (VADM) and a coordinated collection of Concept Inventories (CIs) covering core DM concepts, including set theory, propositional logic, predicate logic, relations, functions, recursion, and mathematical induction. We expect that these tools will prove to be useful in both formative and summative assessment of DM learning. On the formative side, if a CI for a specific concept is administered after that concept has been taught and the results reveal common misunderstandings, the instructor could use these insights to address the concept again, possibly by simply discussing the misunderstandings in class. On the summative side, post-core administration of the CIs can reveal how well curricular choices have motivated students to learn DM and how well students have learned the concepts. We believe that use of the DMCI(s) and VADM will: (i) help improve DM pedagogy/instruction; (ii) help with ABET and other assessment of DM learning; and (iii) ultimately help improve motivation to learn and master DM concepts - which in turn could improve retention of computing majors.

Three Years of SOFTICE: Remotely Accessible, Load Balanced, Virtual Machines for Operating Systems and Networking Laboratories

Saturday, 10:30 a.m.—12:00 p.m.

Matt Rideout,, University of South Florida
Sarah Langevin, University of South Florida
Alessio Gaspar, University of South Florida
William Armitage, University of South Florida

ꕥ • ꕥ

The SOFTICE project (NSF CCLI award 0410696) started in 2004 with the intent of addressing classroom management / technical and pedagogical issues involved in providing undergraduate students with hands-on experience in the operating systems and networking courses.

From the technical perspective, we provide students access to a Linux system on which they can spawn virtual machines (User Mode Linux - UML project), network them together (My Linux Network - MLN project) and even rebuild them from scratch with a modified kernel if need be. Because these virtual machines are running on the server, they can be accessed over the internet from any platform on which a SSH client (and optionally a x-server) can run (windows, Linux, Unix…). As the number of students grows, recycled classroom PCs can be federated into a cluster and attached to our single Linux server which will provide them, at boot time, with an OS to run and use them to dispatch and load balance students connection. From the students perspective, still a single IP to connect to, form the administrator perspective, all these extra nodes scale up the computing power without requiring extra attention.

From the pedagogical perspective, our work focused on leveraging Linux technologies; in networking, students can assemble, using an abstract configuration language, arbitrary networks to monitor protocols traffic, learn about routing… In operating systems, they can insert code into running kernels to explore its components one by one thus circumventing traditional problems stemming form using a production-level OS in the classroom.

Improving Student Learning in Multimedia Programming

Saturday, 10:30 a.m.—12:00 p.m.

Chris Stein, CUNY Borough of Manhattan Community College
Jody Culkin, CUNY Borough of Manhattan Community College

Multimedia Programming majors at the Borough of Manhattan Community College, CUNY have historically had difficulty learning the basic concepts behind programming and digital media, and applying these concepts to real-world tasks. Our grant, "Improving Student Learning through the use of 3D Simulation Activities and Case Studies in Multimedia Programming (NSF-DUE-0511209)," addresses these issues by adapting and implementing exemplary educational materials and pedagogical strategies from three sources: the National Center for Case Study Teaching in Science at University at Buffalo SUNY, St. Joseph's University (NSF-DUE-0302542) and Wake Forest University (NSF-DUE-0127280, NSF-DUE-0340969). Also, as part of the grant, a tutoring program and a faculty development program were instituted to extend student learning and ensure integration of the newly developed courseware into the teaching practices of the faculty.

The materials from Wake Forest teach students the basic scientific concepts behind digital media and use those concepts to give them a deeper understanding of multimedia software. The team sponsored by St. Joseph's created curricular materials that use the Alice programming environment to make object-oriented programming more enjoyable and accessible to under-prepared students. We adapted their materials to teach a course that begins with Alice and then transitions students to the Flash IDE and ActionScript programming. To move an application development course away from a lecture-based pedagogy, we are developing a hands-on Case Study approach with the help of the National Center for Case Study Teaching in Science.

The Development of Student Electronic Portfolios for Curriculum Improvement in Practice-Oriented Biology and Computer Science Programs

Thursday, 10:30 a.m.—12:00 p.m.

Kostia Bergman, Northeastern University
Veronica Porter, Northeastern University
Viera K. Proulx, Northeastern University
Mel Simms, Northeastern University

This project is intended to move the assessment of practice-oriented science education from indirect measures of learning to direct indicators of student achievement. A review of science programs that combine academic instruction with cooperative education work placements shows that the assessment of student achievement still relies heavily on self-reports from employers and students to determine whether or not students attain desired skills and abilities. A carefully crafted electronic portfolio can be a vehicle through which the students present direct evidence of attainment of program learning goals. Electronic portfolios have become widely adopted in K-12 science education, but have not as yet become common in university science instruction. This may be because the first wave of higher education electronic portfolios focused on expanded resume presentations and personal development without the rigor necessary to be useful indicators of student academic achievement. The creation and maintenance of a performance-based portfolio can become a part of a science student's learning experiences, and not just a documentation of learning that has already taken place. Portfolios may be especially useful for first-generation and second language students who need as much opportunity as we can provide to improve their skills in communicating and reflecting on their scientific understanding.

Media Computation as an Approach to Attract and Retain Students

Thursday, 10:30 a.m.—12:00 p.m.

Mark Guzdial, Georgia Institute of Technology

The research findings on why students avoid CS or drop-out those that try seems to point much of the blame on what we teach and how we teach it. A common theme of several studies is that students find introductory computer science classes too abstract and lacking relevance for real-world problems. To address this problem, we have developed a two semester sequence of introductory computer science courses that contextualize computing education around the manipulation and creation of media. Students in these classes learn about loops by writing progrms negating and generating grayscale version of pictures; they learn about array manipulation by writing programs to splice sounds; they learn about managing larger programs by writing 100+ line programs to generate animations and collages; they learn about linked list manipulation by composing music through nodes filled with MIDI notes; and they meet their first tree as a "scene graph." The results are compelling: Dramatically higher retention rates, particularly among women and non-technical majors, and renewed interest in computing degrees. We'll present student work and research findings in this presentation.

Computing Educators Oral History Project

Saturday, 10:30 a.m.—12:00 p.m.

Barbara B. Owens, Southwestern University
Vicki Almstrum, Southwestern University

The Computing Educators Oral History Project (CEOHP) was conceived as a means to address factors that affect girls' and women's decisions about studying computing and working in the field. This collection of career stories and artifacts from a variety of computing educators will provide inspirational role models at levels ranging from middle school to late career, and serve as a rich multimedia database for social scientists studying career paths and influences.

Since 2004, a grass-roots movement has brought together a broad international group of women and men passionate about this project, who have pushed it forward. Concrete results to date include an in-depth formative report, a solid protocol for carrying out interviews, training several interviewers, a temporary website with sample interviews, and a baseline collection of 12 interviews with computing educators from across the globe. The work during 2007 is being supported by a grant from NSF, with the goal of establishing the project's foundation for the next five years.

The work this year, which will clearly define the expertise and infrastructure for this project, includes two strategy meetings, two project reviews, and creation of an advisory board that includes technical and visionary leaders. This planning phase is using students enrolled in a capstone project at the PI's home institution to develop a prototype web portal and tools for making the collection accessible. The results will ensure the project's short-term technical success as well as its long-term sustainability.

The current status of the project can be followed at http://cs.southwestern.edu/OHProject.

State-Wide Undergraduate Grid Computing Course

Friday, 2:00 p.m.—3:30 p.m.

Barry Wilkinson, University of North Carolina, Charlotte
Clayton Ferner, University of North Carolina, Wilmington

Grid computing has become an important concept for high performance computing. By taking advantage of the Internet, geographically distributed computers can be used collectively for collaborative problem solving. In Grid computing, different organizations can supply resources and personnel, and the Grid infrastructure can cross organizational boundaries. This concept has many benefits including solving problems that could not be solved previously because of limited computing resources (e.g. searching for new drugs). We have developed an undergraduate grid computing course that crosses organizational boundaries using resources at several North Carolina universities. The course is broadcast across North Carolina using the televideo facilities of the North Carolina Research and Education Network. Fourteen universities and colleges participated included minority-serving universities, state universities, and private colleges.

Most grid computing courses are graduate-level courses within a single department. Our course is unique both because it targets undergraduates and because many universities participated. The course was first taught in Fall 2004 and again in Fall 2005. A newly revised version of the course is currently being taught in Spring 2007 using a top-down approach starting with the use of a grid computing portal, leading through details of grid computing infrastructure with seven hands-on assignments, finally cumulating in a team mini-project.

This work is funded by the National Science Foundation through their CCLI program, grant #0410667/053334, and the University of North Carolina Office of President.

CLICS: Computer Security Workshops for Faculty

Friday, 10:30 a.m.—12:00 p.m.

Paul J. Wagner, University of Wisconsin, Eau Claire
Andrew T. Phillips, University of Wisconsin, Eau Claire

This NSF CCLI Showcase presentation describes a workshop that we have used to provide computer security education to CS/IT professionals and students, and that has been effective in communicating basic computer security principles as well as an understanding of some of the significant tools and techniques in this area. Evaluation of the workshop has been very positive, and we have been offering the workshop nationally since 2004.

The workshop focuses on computer system security, as opposed to network security, and it concentrates on technological issues, although there is some limited discussion of social engineering and physical security. It also focuses on defensive issues, though some discussion of attacking strategies is presented in order to help the participant understand the mindset of an attacker.

We present the workshop information in six modules, and finish with a cyberwar exercise. The modules include material on (1) information gathering and packet sniffing, (2) port scanning, (3) password policies and password cracking, (4) vulnerability assessment and analysis, (5) system hardening, and (6) intrusion detection. All topics include hands-on exercises with common tools on both Linux and Windows systems.

In the cyberwar exercise, our workshop staff initiate a series of controlled exploits against all participant systems based on a variety of attack scenarios. The status of all participant machines is displayed for all participants so that they can see if the attacks successfully penetrated their system(s). This allows participants to see the attacks developing and to defend against them in real time.

An Interactive Approach to Formal Languages and Automata with JFLAP

Friday, 10:30 a.m.—12:00 p.m.

Susan H. Rodger, Duke University

ꕥ • ꕥ

Traditionally, formal languages and automata have been presented in a formal manner. Students generally have difficulty understanding abstract concepts presented in this manner. We have developed an instructional software tool JFLAP (www.jflap.org) to complement the formal approach to the formal languages and automata course by allowing one to explore the abstract topics in a visual and interactive manner. With JFLAP one can construct and test finite automata, pushdown automata, Turing machines, and grammars. In addition, one can interactively explore many proofs such as pumping lemma, NFA to DFA to minimal state DFA, and CFG to NPDA. JFLAP allows one to explore applications of formal languages. In the area of parsing, one can construct an SLR parse table, including the DFA that models the stack, and compare the parsing process with an NPDA for the same grammar. In the area of biology, one can build L-system grammars to model the growth of plants and fractals.

JFLAP has been used around the world in over 160 countries in automata theory courses, compiler courses, discrete math courses and artificial intelligence courses. We demonstrate JFLAP, describe how it can be used in teaching, and discuss a 2-year study to evaluate JFLAP's effectiveness in learning involving a dozen universities.

Learning Computer Graphics Programming Through Examples

Friday, 2:00 p.m.—3:30 p.m.

Kelvin Sung, University of Washington, Bothell
Peter Shirley, University of Utah
Becky Reed Rosenberg, University of Washington, Bothell

ꕥ • ꕥ

A few years ago, due to practical constraints, we started looking for answers to the question that, "if students can only schedule one elective computer graphics course in their undergraduate education, what are the most important and useful knowledge we should cover?"

When answering this question, we strived to achieve: practicality and essentiality. The knowledge should be practical with potential applicability in students' chosen filed of profession, and the knowledge should be essential concepts that support students' future self-learning.

These objectives guided us to the topics that are relevant to popular computer graphics (CG) applications. To address time restriction, only topics related to interactive applications are covered. To maximize potential applicability, example concept demonstration applications (CDAs) based on popular APIs implementing and utilizing the relevant concepts are provided. To ensure students learn the concepts together with the APIs, the CDAs are implemented based on more than one APIs (OpenGL and D3D).

The result is a course that emphasized on programming CG concepts in moderately complex interactive applications. We cover CG topics with exemplified CDAs in OpenGL and D3D. Students demonstrated their understanding by implementing and integrating CG concepts in their projects. These projects are implemented based on a combination of languages C++/C#/Java, and APIs (OpenGL/D3D/Java3D/XNA, and MFC/WinForm/JavaSwing). Currently, there are more than 100 CDAs implemented in OpenGL and D3D.

In our presentation we will explain the organization of our CDAs; describe example CDAs of CG concepts; demonstrate example student projects; and discuss student learning outcome and challenges we have encountered.

Integration and Assessment of Pair Programming, Unit Testing and Lab Practica in an Introductory Computer Science Course

Friday, 2:00 p.m.—3:30 p.m.

Grant Braught, Dickinson College
Tim Wahls, Dickinson College
Louis Ziantz, Dickinson College

The objective of this project has been to design, implement and assess a new introductory computer science course at Dickinson College. The new course focuses on programming fundamentals and integrates three exemplary practices: unit testing, pair programming and laboratory practica. In developing and assessing this course we have focused on three distinct goals: (1) Adapting and integrating pair programming, unit testing and the use of lab practica. (2) Assessing (a) the success and retention rate of students in the new course, (b) the development of students' individual programming skills, (c) students' use and mastery of unit testing and (d) the effects of pair programming on (a), (b) and (c). (3) To refine and disseminate the materials developed for our course.

During the first year of this project, four sections of our new introductory course (24 students each) were offered with half of these sections using pair-programming for lab assignments. A comparison of the data collected from the paired and individual sections showed:

- No evidence for a difference in individual programming ability as measured by performance on laboratory practica.
- No evidence for a difference in performance on any individually completed course work.
- Students in paired sections had greater confidence in their individual ability to create complete unit test suites for their programs.

Redesigning Introductory Computing: The Design Discipline

Friday, 10:30 a.m.—12:00 p.m.

Stephen Bloch, Adelphi University
John Clements, California Polytechnic University, San Luis Obispo
Kathi Fisler, Worcester Polytechnic Institute
Matthew Flatt, University of Utah
Viera K. Proulx, Northeastern University

The goal of this project is to disseminate the TeachScheme! - ReachJava! introductory computer science curriculum to faculty members through a series of summer workshops held at four locations throughout the USA over the next three years.

The curriculum provides an integrated pedagogic and technical solution to teaching a disciplined program design that supports novice learners and challenges all students. Our approach uses multiple programming languages over multiple semesters with a well-developed methodology for transitioning between languages. Students develop skills at iterative refinement through a series of extended exercises supported by Teachpacks that encapsulate infrastructure code while allowing students to work in pedagogically-motivated language subsets. Each of these features is critical to creating a powerful yet novice-friendly introductory curriculum.

Students start programming in functional languages, yet through the use of Teachpacks implement an interactive game within the first three weeks of the semester. By the end of the second semester they develope a solid understanding of the architecture of the Java Collections Framework including the design of algorithms that leverage the collections interfaces for data structure access and the functional operations on the data set elements. Test-driven design and strict documentation requirements enforce disciplined programming while illustrating in a concrete way the program's desired behavior.

Curriculum materials include software, Teachpacks, textbooks (published first part, the second part in a draft form), a wealth of lab materials, lecture notes, and assignments with solutions.

Problets: Online Programming Tutors for Computer Science I

Friday, 10:30 a.m.—12:00 p.m.

Amruth Kumar, Ramapo College of New Jersey

Problets are web-based software that provide practice exercises for Computer Science I. The exercises include programming problems such as "Debug this program", "What is printed by this program?", "Show the changes to the array after the loop", and "Evaluate the expression step-by-step." The problems are not multiple-choice questions, and cannot be solved without a deep understanding of programming concepts.

Problets grade the student's answer and provide instant feedback. In order to help the student learn from mistakes, problets: 1) illustrate the correct answer by explaining the step-by-step execution of the problem code; 2) graphically visualize the step-by-step execution of the program. These features help students learn on their own, at their own pace.

Problets log the student data on a central server. So, faculty can get reports of how well each student did on each topic, how well the class did, and the concepts that are not yet clear to each student/class.

Problets are designed to be used as supplements to classroom instruction and complements to programming projects. They can be used as exercises during closed labs, as assignments after class, or for in-class testing with the feedback turned off. Since problets are self-administering, they can be used with or without faculty supervision.

Currently, problets are available for arithmetic expressions, relational expressions, logical expressions, variables and scope, if and if-else statements, while loops, for loops, and C++ pointers. They are available for C, C++, Java and C# programming.

Problets are free for educational use. They run on any recent Java-enabled browser. Using problets in a course is simple: 1) faculty specify the programming language and the topics for which they would like to use problets; 2) a dedicated web site is set up for their class; 3) faculty ask students to use problets from this web site; 4) after the students use each problet, faculty request a report, which is emailed to them within three working days.

To try out problets, please visit http://www.problets.org.

TinkerNet and TinkerNet 2

Friday, 2:00 p.m.—3:30 p.m.

Michael Erlinger, Harvey Mudd College

TinkerNet1 was developed as a low-cost platform for teaching bottom-up, handson networking at the undergraduate level. In the original TinkerNet "throw away" PCs, cheap components, and free software were used to enable students to build their own networking protocol stack from Ethernet up to the Application level, and to have their packets actually transmitted on the wire. Operationally, students: write network specific code; integrated that code into a new OS kernel; downloaded the new kernel to a waiting host machine; and then evaluated the traffic produced by that kernel. Since nothing is emulated, standard networking tools such as packet sniffers can be used to test student generated traffic from a host located on the TinkerNet network. The directions and all code for creating a TinkerNet are available on the web.

A new TinkerNet, TinkerNet 2, has been created as a software only system based on User Mode Linux (UML) (http://user-mode-linux.sourceforge.net). In this version only a server is needed. Students again write network specific code and integrate the code into an OS kernel, but in TinkerNet 2 the student kernel is executed as a virtual kernel within UML. From the student view, TinkerNet 2 behaves the same as original TinkerNet. TinkerNet in both forms includes a set of laboratory exercises based on building a network protocol stack, e.g., Ethernet packet recognition and generation, ARP packet recognition and generation, etc. Included with the laboratory exercises is an automatic grading program which evaluates each exercise.

This work was supported in part by the National Science Foundation under grant NSF-DUE-0443012 to Harvey Mudd College.