NEWS RELEASE

CONTACT: Adrienne Decker
Adrienne@buffalo.edu
(585) 475-4653

TOP TEN COMPUTER SCIENCE EDUCATION RESEARCH PAPERS OF THE LAST 50 YEARS RECOGNIZED

At 50th Anniversary SIGCSE Symposium, Leading Computer Science Education Group Highlights Research that Has Shaped the Field

New York, NY, March 2, 2019 – As a capstone to its 50th annual SIGCSE Technical Symposium, leaders of the Association for Computing Machinery (ACM) Special Interest Group on Computer Science Education (SIGCSE) are celebrating the ideas that have shaped the field by recognizing a select group of publications with a "Top Ten Symposium Papers of All Time Award." The top ten papers were chosen from among the best papers that were presented at the SIGCSE Technical Symposium over the last 49 years.

As part of the Top Ten announcement today in Minneapolis, the coauthors of each top paper will receive a plaque, free conference registration for one co-author to accept the award and up to a total of $2,000 that can be used toward travel for all authors of the top ranked paper.

"In 1969, the year of our first SIGCSE symposium, computing education was a niche specialty" explains SIGCSE Board Chair Amber Settle of DePaul University, of Chicago, USA. "Today, it is an essential skill students need to prepare for the workforce. Computing has become one of the most popular majors in higher education, and more and more students are being introduced to computing in K-12 settings. The Top Ten Symposium Papers of All Time Award will emphasize the outstanding research that underpins and informs how students of all ages learn computing. We also believe that highlighting excellent research will inspire others to enter the computing education field and make their own contributions."

The Top Ten Symposium Papers are:

1. "Identifying student misconceptions of programming" (2010)
Lisa C. Kaczmarczyk, Elizabeth R. Petrick, University of California, San Diego; Philip East, University of Northern Iowa; Geoffrey L. Herman, University of Illinois at Urbana-Champaign
Computing educators are often baffled by the misconceptions that their CS1 students hold. We need to understand these misconceptions more clearly in order to help students form correct conceptions. This paper describes one stage in the development of a concept inventory for Computing Fundamentals: investigation of student misconceptions in a series of core CS1 topics previously identified as both

important and difficult. Formal interviews with students revealed four distinct themes, each containing many interesting misconceptions.

2. “Improving the CS1 experience with pair programming” (2003)
Nachiappan Nagappan, Laurie Williams, Miriam Ferzli, Eric Wiebe, Kai Yang, Carol Miller, Suzanne Balik, North Carolina State University
Pair programming is a practice in which two programmers work collaboratively at one computer, on the same design, algorithm, or code. Prior research indicates that pair programmers produce higher quality code in essentially half the time taken by solo programmers. The authors organized an experiment to assess the efficacy of pair programming in an introductory Computer Science course. Their results indicate that pair programming creates a laboratory environment conducive to more advanced, active learning than traditional labs; students and lab instructors report labs to be more productive and less frustrating.

3. “Undergraduate women in computer science: experience, motivation and culture” (1997)
Allan Fisher, Jane Margolis, Faye Miller, Carnegie Mellon University
During a year-long study, the authors examined the experiences of undergraduate women studying computer science at Carnegie Mellon University, with a specific eye toward understanding the influences and processes whereby they attach themselves to or detach themselves from the field. This report, midway through the two-year project, recaps the goals and methods of the study, reports on their progress and preliminary conclusions, and sketches their plans for the final year and the future beyond this particular project.

4. “A Multi-institutional Study of Peer Instruction in Introductory Computing” (2016)
Leo Porter, Beth Simon, University of California, San Diego; Dennis Bouvier, Southern Illinois University; Quintin Cutts, University of Glasgow; Scott Grissom, Grand Valley State University; Cynthia Lee, Stanford University; Robert McCartney, University of Connecticut; Daniel Zingaro, University of Toronto
Peer Instruction (PI) is a student-centric pedagogy in which students move from the role of passive listeners to active participants in the classroom. This paper adds to this body of knowledge by examining outcomes from seven introductory programming instructors: three novices to PI and four with a range of PI experience. Through common measurements of student perceptions, the authors provide evidence that introductory computing instructors can successfully implement PI in their classrooms.

5. “The introductory programming course in computer science: ten principles” (1978)
G. Michael Schneider, University of Minnesota
Schneider describes the crucial goals of any introductory programming course while leaving to the reader the design of a specific course to meet these goals. This paper presents ten essential objectives of an initial programming course in Computer Science, regardless of who is teaching or where it is being taught. Schneider attempts to provide an in-depth, philosophical framework for the course called CS1—Computer Programming 1—as described by the ACM Curriculum Committee on Computer Science.

6. “Constructivism in computer science education” (1998)
Mordechai Ben-Ari, Weizmann Institute of Science
Constructivism is a theory of learning which claims that students construct knowledge rather than merely receive and store knowledge transmitted by the teacher. Constructivism has been extremely influential in science and mathematics education, but not in computer science education (CSE). This

paper surveys constructivism in the context of CSE, and shows how the theory can supply a theoretical basis for debating issues and evaluating proposals.

7. “Using software testing to move students from trial-and-error to reflection-in-action” (2004)
Stephen H. Edwards, Virginia Tech
Introductory computer science students have relied on a *trial and error* approach to fixing errors and debugging for too long. Moving to a *reflection in action* strategy can help students become more successful. Traditional programming assignments are usually assessed in a way that ignores the skills needed for reflection in action, but software testing promotes the hypothesis-forming and experimental validation that are central to this mode of learning. By changing the way assignments are assessed--where students are responsible for demonstrating correctness through testing, and then assessed on how well they achieve this goal--it is possible to reinforce desired skills. Automated feedback can also play a valuable role in encouraging students while also showing them where they can improve.

8. “What should we teach in an introductory programming course” (1974)
David Gries, Cornell University
Gries argues that an introductory course (and its successor) in programming should be concerned with three aspects of programming: 1. How to solve problems, 2. How to describe an algorithmic solution to a problem, and 3. How to verify that an algorithm is correct. In this paper he discusses mainly the first two aspects. He notes that the third is just as important, but if the first two are carried out in a systematic fashion, the third is much easier than commonly supposed.

9. “Contributing to success in an introductory computer science course: a study of twelve factors” (2001)
Brenda Cantwell Wilson, Murray State University; Sharon Shrock, Southern Illinois University
This study was conducted to determine factors that promote success in an introductory college computer science course. The model included twelve possible predictive factors including math background, attribution for success/failure (luck, effort, difficulty of task, and ability), domain specific self-efficacy, encouragement, comfort level in the course, work style preference, previous programming experience, previous non-programming computer experience, and gender. Subjects included 105 students enrolled in a CS1 introductory computer science course at a midwestern university. The study revealed three predictive factors in the following order of importance: comfort level, math, and attribution to luck for success/failure.

10. “Teaching objects-first in introductory computer science” (2003)
Stephen Cooper, Saint Joseph's University; Wanda Dann, Ithaca College; Randy Pausch Carnegie Mellon University
An objects-first strategy for teaching introductory computer science courses is receiving increased attention from CS educators. In this paper, the authors discuss the challenge of the objects-first strategy and present a new approach that attempts to meet this challenge. The approach is centered on the visualization of objects and their behaviors using a 3D animation environment. Statistical data as well as informal observations are summarized to show evidence of student performance as a result of this approach. A comparison is made of the pedagogical aspects of this new approach with that of other relevant work.

Annual Best Paper Award Announced

Today SIGCSE officers also announced the inauguration of an annual SIGCSE Test of Time Award. The first award will be presented at the 2020 SIGCSE Symposium and recognize research publications that have had wide-ranging impact on the field.

About SIGCSE
The Special Interest Group on Computer Science Education of the Association for Computing Machinery (ACM SIGCSE) is a community of approximately 2,600 people who, in addition to their specialization within computing, have a strong interest in quality computing education. SIGCSE provides a forum for educators to discuss the problems concerned with the development, implementation, and/or evaluation of computing programs, curricula, and courses, as well as syllabi, laboratories, and other elements of teaching and pedagogy.

About ACM
ACM, the Association for Computing Machinery**,** is the world's largest educational and scientific computing society, uniting computing educators, researchers and professionals to inspire dialogue, share resources and address the field's challenges. ACM strengthens the computing profession's collective voice through strong leadership, promotion of the highest standards, and recognition of technical excellence. ACM supports the professional growth of its members by providing opportunities for life-long learning, career development, and professional networking.

###