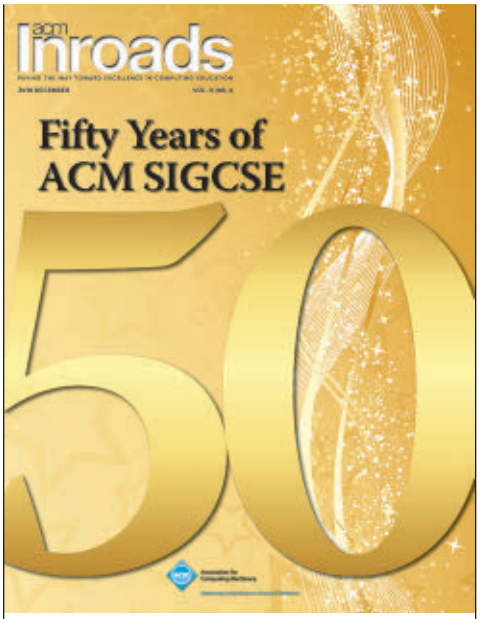
acm Inroads
PAVING THE WAY TOWARD EXCELLENCE IN COMPUTING EDUCATION
2018 DECEMBER VOL. 9, NO. 4

Fifty Years of ACM SIGCSE

Association for Computing Machinery
Advancing Computing as a Science & Profession

acm Inroads
PAVING THE WAY TOWARD EXCELLENCE IN COMPUTING EDUCATION
Association for Computing Machinery
Advancing Computing as a Science & Profession
2018 December | volume 9, number 4

CONTENTS



Publication Information

ACM Inroads is published four times a year:
March; June; September; December by ACM
Print (ISSN 2153-2184) | Online (ISSN 2153-2192)

Acknowledgment

Thanks to ACM's Special Interest Group on Computer Science Education (SIGCSE) for making Inroads' publication and distribution possible.

ACM Publications

Publications Office

ACM, 2 Penn Plaza, Suite 701
New York, New York 10121-0701 USA
+1-212-869-7440 (Tel) | +1-212-869-0481 (Fax)

Annual Subscriptions

				Single Copies
Members print:	$ 49	e-only: $ 39	p+e: $ 59	$ 9
Students print:	$ 26	e-only: $ 21	p+e: $ 34	$ 4
Non-members:	$ 135	e-only: $108	p+e: $162	$ 25

SIGCSE members receive *ACM Inroads* as a membership benefit.

Please send orders to
ACM, General Post Office, P.O. Box 30777
New York, New York 10087-0777 USA
or call +1-212-626-0500

For credit card orders, call +1-800-342-6626
Order personnel available 08:30–16:30 EST
After hours, please leave message and order personnel will return your call.

Change of Address

acmcoa@acm.org

Other Services, Questions, or Information

acmhelp@acm.org

Periodicals postage paid in New York, New York 10001 USA and at additional mailing offices.

Postmaster: Please send address changes to:
ACM Inroads
ACM
2 Penn Plaza, Suite 701
New York, New York 10121-0701 USA

Introducing ACM Transactions on Internet of Things (TIOT)

A new journal from ACM publishing novel research contributions and experience reports in domains whose synergy and interrelations enable the IoT vision

Now Accepting Submissions

ACM Transactions on Internet of Things (TIOT) publishes novel research contributions and experience reports in several research domains whose synergy and interrelations enable the IoT vision. TIOT focuses on system designs, end-to-end architectures, and enabling technologies, and on publishing results and insights corroborated by a strong experimental component.

Submissions are expected to provide experimental evidence of their effectiveness in realistic scenarios and the related datasets. The submission of purely theoretical or speculative papers is discouraged, and so is the use of simulation as the sole form of experimental validation.

Experience reports about the use or adaptation of known systems and techniques in real-world applications are equally welcome, as these studies elicit precious insights for researchers and practitioners alike. For this type of submissions, the depth, rigor, and realism of the experimental component is key, along with the analysis and expected impact of the lessons learned.

For more information and to submit your work, please visit https://tiot.acm.org.

Association for Computing Machinery

Advancing Computing as a Science & Profession

Mark Bailey
Hamilton College

Laurie Smith King
College of the Holy Cross

EDITORS' MESSAGE

Welcome to the December 2108 *Inroads*, a special issue that celebrates the 50th anniversary of the founding of the SIGCSE! This culminates work that began in March of 2017 when the SIGCSE Board asked us to celebrate the special interest group's semicentennial with a special edition of *Inroads*. We, along with our guest editor Jane Prey, have been working steadily since then to bring this issue to fruition. SIGCSE's history, current challenges, and possibilities for the future are explored and enticingly spiced with personal reflections from educators who cover the career spectrum, old-timers and first-timers alike. Jane's guest editor's message gives further details. Enjoy.

The familiar *Inroads* elements throughout the issue have a fiftieth anniversary twist. Notably, on the Back Page, John Barr offers a puzzle containing images from the last five decades (including ones of us!) that you must match to the correct decade. Check your work with the solution on page 108.

Barbara Boucher Owens describes the two prizes that ACM awards each year to computer science educators. Appropriate to this special issue, she has also gathered photos of past award winners. We thought this was so interesting that we asked her to run a pair of contests as well. You can find the URLs for the contests at the end of her article and on the *Inroads* website. These links will go live upon the arrival of this issue to your doorstep/inbox. Winners of the contests will receive free registrations to the SIGCSE Symposium this year. The deadline for submissions is before early registration for SIGCSE 2019 begins.

It seems just yesterday that we were welcoming Daniel Zingaro to our Editorial Advisory Board. Alas, Daniel is stepping down from the EAB in order to focus on achieving tenure. Daniel has provided many excellent reviews with key insights to help authors improve their work. We will miss Daniel on our Board and ask that you forward names on to us to fill his shoes!

Finally, this truly special issue could not have been accomplished without our guest editor, Jane Prey. From the outset, we knew we needed a guest editor for this issue, and we knew just whom to ask. In serving the SIGCSE community for well over 25 years, Jane knows everybody! Her special knack of twisting arms, cajoling, and sweetly nudging was just what we needed to get busy special-issue authors to write and submit their articles on time. Jane has done a beautiful job and we want to express our heartfelt thanks for taking on this project despite the pressures of professional duties, retirement, and 'grandmotherdom.' Jane, thank you from all of us—the readers, the authors, and, especially, Mark and Laurie. ❖

Mark Bailey and Laurie Smith King
Editors-in-Chief

DOI: 10.1145/3284635

GUEST EDITOR'S MESSAGE

It's hard to believe we're celebrating the 50th anniversary of SIGCSE this year. What a remarkable achievement, and I can't believe I was there for almost all of it!! This issue of *Inroads* brings together many views of the SIGCSE we know and love.

We have several different sections in this issue.

SIGCSE: The Beginning—our history and reflections from several of those who were there.

SIGCSE: Now and Moving Forward—comments about who we are and future aspirations.

SIGCSE: Trends and Reflections—from people who have greatly influenced how we think about computer science education today.

And a section that gives us things to ponder about.

SIGCSE: The Next Fifty Years—thoughts about us as an organization and as a profession.

SIGCSE: The Open Challenges—what are some of the roadblocks coming down the pipe?

And my favorite section.

My SIGCSE: Personal Reflections of Computing Educators—the sharing of experiences and thoughts on the impact of SIGCSE on their careers from some members of our community.

Thank you to all the contributing authors—being asked to think about an ill-defined topic and sharing personal thoughts and experiences is very different from writing about current research results. But you all did it with great enthusiasm and passion. I have learned so much from our fellow SIGCSE members.

I am most grateful to our reviewers—Andrew McGettrick, Bob Beck, Dan Joyce, Deepak Kumar, Ellen Walker, JD Doughty, Judy Sheard, Mark Allen Weiss, Paul Tymann, Tom Cortina, and Tracy Camp. They had to put on a different reviewer cap reading these articles, and they didn't require much nagging!

And, of course, a big THANK YOU to Laurie King and Mark Bailey, the current editors of *Inroads* for helping me pull together this issue. They offered great advice, held my hand while I worked through the paper submission system and kept me on task! They were GREAT!

I hope you take the time to read through these articles. They reflect our history—who we are, what we care about, and our thoughts about what's coming up.

And finally, thanks to all of you for being part of the SIGCSE family and making it the dynamic organization it is. Looking forward to another 50 years of computing education sharing and friendship. ❖

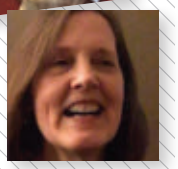
Jane Chu Prey

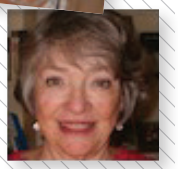
Jane Chu Prey
Co-Chair
ACM Education Board
Fort Meyers, FL 33912
janeprey@gmail.com

DOI: 10.1145//3284633

SIGCSE Annual Awards:
Call for Nominations and Contest

by Barbara Boucher Owens




Where is your picture? In order to encourage all SIGCSE members to actively participate in honoring those colleagues who make our organization great, I want to highlight both those who have already received awards and to underscore the responsibility of members to join in the nomination process.

You may be aware that SIGCSE has two major awards, both presented at the annual SIGCSE Technical Symposium.

The *SIGCSE Award for Outstanding Contribution to Computer Science Education* honors an individual or group in recognition of a significant contribution to computer science education. The contribution should have had long lasting impact on, and made a significant difference in, computing education.

The *SIGCSE Award for Lifetime Service to the Computer Science Education Community* honors an individual who has a long history of volunteer service to the computer science education community.

Although not a SIGCSE Award, per se, ACM also annually awards computing educators with the *Karl V. Karlstrom Educator Award* since its inception in 1989.

Additionally ACM members may be recognized with advanced member grades of *senior member*, *distinguished member*, and the pinnacle *ACM Fellow designation*.

As SIGCSE members, I want to emphasize that it is your responsibility to consider nominating your colleagues for these awards and designations.

I am announcing upcoming contests to identify the SIGCSE award winners' pictures. There will be two online contests, one for each type of award that SIGCSE gives. The prize for each will be a one-year free membership to SIGCSE.

SIGCSE has given one *Outstanding Educator Award* per year since 1981 except 1984 none and 2007 two (38 awards) and has given one *Lifetime Service Award* per year since 1997 (22 awards).

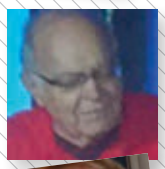
The location of the contest is *https://cs.hamilton.edu/SIGCSE50contest/*.

Keep your eyes peeled for an announcement the contests are open and use the URL above to access both Contests! Good luck! ❖

Barbara Boucher Owens, PhD
Emeritus Faculty, Southwestern University
Georgetown, TX USA
owensb@southwestern.edu

DOI: 10.1145/3284631





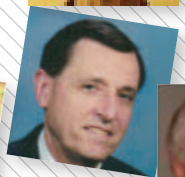













AWARD NOMINATIONS SOLICITED

As part of its mission, ACM brings broad recognition to outstanding technical and professional achievements in computing and information technology.

ACM welcomes nominations for those who deserve recognition for their accomplishments. Please refer to the ACM Awards website at **https://awards.acm.org** for guidelines on how to nominate, lists of the members of the 2018 Award Committees, and listings of past award recipients and their citations.

Nominations are due **January 15, 2019** with the exceptions of the Doctoral Dissertation Award (due **October 31, 2018**) and the ACM – IEEE CS George Michael Memorial HPC Fellowship (due **May 1, 2019**).

A.M. Turing Award: ACM's most prestigious award recognizes contributions of a technical nature which are of lasting and major technical importance to the computing community. The award is accompanied by a prize of $1,000,000 with financial support provided by Google.

ACM Prize in Computing (previously known as the ACM-Infosys Foundation Award in the Computing Sciences): recognizes an early-to mid-career fundamental, innovative contribution in computing that, through its depth, impact and broad implications, exemplifies the greatest achievements in the discipline. The award carries a prize of $250,000. Financial support is provided by Infosys Ltd.

Distinguished Service Award: recognizes outstanding service contributions to the computing community as a whole.

Doctoral Dissertation Award: presented annually to the author(s) of the best doctoral dissertation(s) in computer science and engineering, and is accompanied by a prize of $20,000. The Honorable Mention Award is accompanied by a prize totaling $10,000. Winning dissertations are published in the ACM Digital Library and the ACM Books Series.

ACM – IEEE CS George Michael Memorial HPC Fellowships: honors exceptional PhD students throughout the world whose research focus is on high-performance computing applications, networking, storage, or large-scale data analysis using the most powerful computers that are currently available. The Fellowships includes a $5,000 honorarium.

Grace Murray Hopper Award: presented to the outstanding young computer professional of the year, selected on the basis of a single recent major technical or service contribution. The candidate must have been 35 years of age or less at the time the qualifying contribution was made. A prize of $35,000 accompanies the award. Financial support is provided by Microsoft.

Paris Kanellakis Theory and Practice Award: honors specific theoretical accomplishments that have had a significant and demonstrable effect on the practice of computing. This award is accompanied by a prize of $10,000 and is endowed by contributions from the Kanellakis family, and financial support by ACM's SIGACT, SIGDA, SIGMOD, SIGPLAN, and the ACM SIG Project Fund, and individual contributions.

Karl V. Karlstrom Outstanding Educator Award: presented to an outstanding educator who is appointed to a recognized educational baccalaureate institution, recognized for advancing new teaching methodologies, effecting new curriculum development or expansion in computer science and engineering, or making a significant contribution to ACM's educational mission. The Karlstrom Award is accompanied by a prize of $10,000. Financial support is provided by Pearson Education.

Eugene L. Lawler Award for Humanitarian Contributions within Computer Science and Informatics: recognizes an individual or a group who have made a significant contribution through the use of computing technology; the award is intentionally defined broadly. This biennial, endowed award is accompanied by a prize of $5,000, and alternates with the ACM Policy Award.

ACM – AAAI Allen Newell Award: presented to individuals selected for career contributions that have breadth within computer science, or that bridge computer science and other disciplines. The $10,000 prize is provided by ACM and AAAI, and by individual contributions.

Outstanding Contribution to ACM Award: recognizes outstanding service contributions to the Association. Candidates are selected based on the value and degree of service overall.

ACM Policy Award: recognizes an individual or small group that had a significant positive impact on the formation or execution of public policy affecting computing or the computing community. The biennial award is accompanied by a $10,000 prize. The next award will be the 2019 award.

Software System Award: presented to an institution or individuals recognized for developing a software system that has had a lasting influence, reflected in contributions to concepts, in commercial acceptance, or both. A prize of $35,000 accompanies the award with financial support provided by IBM.

ACM Athena Lecturer Award: celebrates women researchers who have made fundamental contributions to computer science. The award includes a $25,000 honorarium.

For SIG-specific Awards, please visit **https://awards.acm.org/sig-awards**.

Vinton G. Cerf, ACM Awards Committee Co-Chair

John R. White, ACM Awards Committee Co-Chair

Insup Lee, SIG Governing Board Awards Committee Liaison

Rosemary McGuinness, ACM Awards Committee Liaison

News from the SIGs

by Ellen Walker
Hiram College

For this special issue celebrating the 50th anniversary of SIGCSE, we bring you greetings from Amber Settle, who also previews the conferences that SIGCSE will present as it begins its second half-century.

From our SIGCSE reporter and SIGCSE chair, Amber Settle.

The past year was a significant milestone for us as SIGCSE celebrated its fiftieth anniversary in 2018. Thanks to the efforts of guest editor Jane Chu Prey and the many SIGCSE leaders who authored the articles in this issue, you will learn a lot about what SIGCSE has done since 1968 and what the future may bring for our SIG. I would also like to take the opportunity to thank Briana Morrison for her hard work in creating and sharing her weekly posts to the SIGCSE mailing list celebrating the anniversary.

As SIGCSE moves into its second half-century the organization continues to evolve, and some of the most exciting things to come in the next year concern the SIGCSE conferences.

The 2019 SIGCSE Symposium will be the fiftieth, and the organizers are planning a variety of special events during the conference to celebrate. The conference is taking place February 27th to March 2nd in Minneapolis, Minnesota, USA, and the organizers plan to honor our history while sharing their vision of the next 50 years. The SIGCSE Board is providing $50,000 to support events at the conference, and part of the support will be for 50 Travel Grants to be provided for attendees to the conference. The conference will offer the usual papers, panels, special sessions, workshops, birds-of-a-feather sessions, demos, lightning talks, posters, nifty assignments, and pre-symposium events that make the conference a whirlwind of activity, but the social events and other activities will have special meaning thanks to the anniversary. It is a conference you simply do not want to miss.

The next SIGCSE conference of the year is a new one. For the first time ever SIGCSE will sponsor SIGCSE Global, which will be held in Chengdu, Sichuan, China from May 17th to May 19th. Initially the conference will be held every other year, and it is designed to be hosted in countries that are not already served by an existing SIGCSE conference. The details of what types of submissions will be accepted will be announced to the SIGCSE mailing list as the conference and program committees are finalized and the call for proposals written. You will want to watch for announcements about the conference, since it provides a unique opportunity to visit one of the nicest cities in China. Chengdu is the capital of the Sichuan province and well known for its conservation centers for giant pandas. The SIGCSE Global conference will be offered alongside the ACM Turing Celebration Conference, which will provide opportunities to hear keynotes from Turing Award winners as well as other prominent computing researchers.

As SIGCSE moves into its second half-century the organization continues to evolve, and some of the most exciting things to come in the next year concern the SIGCSE conferences.

Next up is the 2019 ITiCSE conference held in Aberdeen, Scotland from July 15th to July 17th. While the activities at the conference remain the same as in previous years and include working groups, keynotes, papers, panels, posters, and tools,

The end of 2018 and the start of 2019 mark a push for SIGCSE into its next 50 years, and the many changes coming to the SIGCSE conferences show that our SIG is moving forward with as much energy as went into its first half-century.

tips, and courseware sessions, the conference is benefitting from collaboration with two European computing organizations. Beginning in July 2018 the ITiCSE steering committee was expanded to include representatives from Informatics Europe and ACM-Europe, and the collaboration should improve the experience for all attendees by ensuring that ITiCSE is the premiere computing education conference held in Europe. One immediate benefit of the collaboration in 2018 was a cash prize for the best paper which was sponsored by the ACM-Europe Council. Hopefully the collaboration will result in other positive changes, and I encourage you to attend and see for yourself.

The last 2019 SIGCSE conference is ICER located in Toronto, Ontario, Canada from August 12th to 14th. This is the first time that the conference will be located in Canada during its North American rotation. As usual ICER will offer papers, lightning talks, and posters as well as the Doctoral Consortium and workshops and the focus will remain on computing education research. The computing education research community has seen explosive growth in submissions during the past several years, and we expect that Toronto will continue the trend. Since the currently single-track conference is already a daunting three days in length, further expansion of the conference may involve a change in how sessions are offered. The ICER organizers are hard at work on a proposal for a change in format, and Toronto is likely the first place that any change would debut. These changes are promising for us as a SIG since the growth of the conference is a sign of the increasing maturity of the computing education research community. I hope that you will come to Toronto to celebrate the Canadian debut of ICER and see what exciting changes are in store for the conference.

The end of 2018 and the start of 2019 mark a push for SIGCSE into its next 50 years, and the many changes coming to the SIGCSE conferences show that our SIG is moving forward with as much energy as went into its first half-century. I hope that you will attend at least one SIGCSE conference in 2019 so that you can experience the excitement first hand. ❖

For more information on these SIGs, see the following websites

SIGCSE: *www.sigcse.org*

SIGITE: *www.sigite.org*

SIGMIS: *www.sigmis.org*

Ellen L. Walker
Professor of Computer Science
Hiram College
11715 Garfield Road
Hiram, OH 44234 USA
walkerel@hiram.edu

Amber Settle
School of Computing
College of Computing and Digital Media
DePaul University
Chicago, IL 60604 USA
asettle@cdm.depaul.edu

DOI: 10.1145/3284637

On Fifty Years of ACM SIGCSE

John R. White, *Past ACM CEO, Past ACM President*

SIGCSE is one of the oldest, largest, and strongest SIGs within the ACM, and epitomizes what ACM looks for and hopes for in a Special Interest Group. ACM gives SIGs significant autonomy to build and serve technical communities. With that autonomy comes a tremendous responsibility to not just declare the existence of a technical community, but to create and oversee the evolution of successful, high-quality technical activities for that community. SIGs are the heart and soul of ACM.

From its founding in 1968, SIGCSE has focused on computer science education, one of the pillars of ACM's mission. In 1968 there were very few departments of computer science, but the field was emerging. The first ACM curriculum in computer science, Curriculum '68 [1], had just been published, and SIGCSE almost immediately began to focus on helping educators understand and implement the recommendations in Curriculum '68. Over the intervening fifty years, SIGCSE has continued to contribute to ACM curriculum efforts, help shape ACM's accreditation activities, and, more importantly, define and grow what has become the world's most significant venue for exploring all aspects of computer science education—the SIGCSE Technical Symposium. Along the way, SIGCSE has expanded its mission and vision to include issues related to K-12 computing education and full inclusion and full access for all groups.

SIGs are the heart and soul of ACM.

To the founders of SIGCSE who are still with us, to the volunteers who shaped the early days of SIGCSE and defined its focus, and to the decades of SIGCSE leaders and volunteers who followed and shaped what SIGCSE is today—"Thank You." Thank you for making SIGCSE such a significant part of ACM. Thank you for shaping and improving computer science education at all levels. And thank you for contributing significantly to making ACM the premiere society in computing. ❖

References

1. ACM Curriculum Committee on Computer Science. 1968. Curriculum 68: Recommendations for Academic Programs in Computer Science. *Comm. ACM* 11, 3 (Mar. 1968), 151–197.

John R. White
Past ACM CEO,
Past ACM President
white@hq.acm.org

DOI: 10.1145/3233228

"A Hop, Skip and Jump" A Personal Journey Down SIGCSE Memory Lane

Robert M. Aiken, *Temple University*

Robert "Bob" Aiken was a founding member of SIGCSE in 1968 and became its first secretary. Between 1969 and 1973, Bob was the editor of the *SIGCSE Bulletin* where he generated publications volume 1, number 3, through volume 5, number 2. He served on the SIGCSE board of directors between 1975 and 1977 and then served as SIGCSE chair between 1977 and 1981. Between 1981 and 1985, he again served as a member of the board of directors. In 1995 he received the SIGCSE Outstanding Contribution to Computer Science Education award and in 1999 he received the SIGCSE Lifetime Service to Computer Science Education award. In 2002, Bob became an ACM Fellow.

PROLOGUE

Several months ago, the editors of the *ACM Inroads* invited me to submit an article that discussed the early days of SIGCSE for a special issue of the magazine titled, "Celebrating SIGCSE@50." Since then several articles and blogs have appeared covering various points on which I was basing my paper. Thus, in consultation with the guest editor, Jane Prey, my contribution focuses more on personal perspectives that reflect some of the interesting events and experiences that occurred as SIGCSE evolved. So, let me take you on a journey as we follow SIGCSE's evolution focusing on some of the key elements and milestones of its growth. I hope you enjoy this personal journey and learn some interesting snippets along the way.

GESTATION

Imagine that you are a 27-year-old freshly minted PhD. That was me in August 1968. I had just taken a position as an Assistant Professor of Computer Science at the University of Tennessee. My colleague (and boss), Gordon Sherman, encouraged me to attend the 23rd National ACM Conference, which was held in Las Vegas August 27–29. I jumped at the idea and during that visit, I met several early proponents of computer science education, including Elliot Organick. Elliot invited me to meet with a number of colleagues to establish a working group. Twenty of us signed a petition to get the ball rolling—many of them friends and colleagues of Elliot. The process for starting a special interest group (SIG) was that ACM would finance and support the work of a Special Interest Committee (SIC) for one year. To become a SIG, the SIC needed to show during that year that it was a viable entity by publishing at least one newsletter, appointing a set of officers, and recruiting enough members to make it self-supporting.

The setting was hardly auspicious—a smoky, small conference room in one of the hotels on the Las Vegas Strip. However, the people involved more than made up for the less than propitious surroundings! One can find a copy of the petition signers in "Celebrating SIGCSE's 50th Anniversary!" [6]

This visit to Las Vegas definitely expanded and enriched my computer science education horizons but left me a bit poorer in the pocketbook (i.e., having to borrow money to pay my room)! However, that is a story for another time.

Through announcements at the ACM Annual Meeting, the Spring and Fall Joint Computer Conferences (SJCC and FJCC), and via the *Communications of the ACM*, we were able to reach out to the CS education community to spread the word regarding our efforts to form SIGCSE. By late 1969 more than 200 colleagues had written to ACM headquarters signaling their support for chartering SIGCSE. We were successful in this endeavor and as noted in [3] we officially became a SIG in 1970. Later that year we had 143 attendees at our first technical symposium. Unbelievable that we now have over 2,700 members with a significant number of colleagues from outside North America and we attract more than 1,600 attendees to our annual technical symposia.

As noted in [2], significant activity was well under way including planning for the first SIGCSE technical symposium (Houston November 1970) and a meeting scheduled for November 18, 1969 at FJCC. One of the key topics being discussed was "Accrediting Computer Science and Computer Technology Programs – Pros and Cons!" Accreditation continued to be a hot topic for many years and ACM (with significant SIGCSE

support), in collaboration with the IEEE Computer Society, formed the Computing Sciences Accreditation Board (CSAB) in 1984.

A key point in making SIGCSE successful was the recognition that the *Bulletin* and proceedings were its foundation. They were central in keeping SIGCSE members informed and up-to-date, as well as providing a forum for exchanging ideas and publishing CS education research. With that in mind we also knew that our biggest cost was producing and mailing the Proceedings of the Technical Symposium to all the members as part of their dues. Our goal was to operate on a "break-even" basis yet continue to include the Proceedings as one of the four *Bulletin* issues. (Those members who attended the symposia thus received two copies though when they registered they were given the option of foregoing their second copy.) We were able to maintain this "balance" though the parameters changed when the *Bulletin* went on-line, and *Inroads* was established.

As early as the 25th ACM National Conference in September 1970, SIGCSE had an active and visible presence sponsoring five sessions.

PROMOTING SIGCSE

As early as the 25th ACM National Conference in September 1970, SIGCSE had an active and visible presence sponsoring five sessions [4].

1. General Education in Computer Science
2. Undergraduate Education in Computer Science
3. Graduate Programs in Computer Science
4. Organizing for Computer Science Education
5. Computer Science Education and Industry – A Dialogue

In addition, several members of SIGCSE participated in the First International Federation for Information Processing (IFIP) World Conference on Computer Education (Amsterdam, 1970) including yours truly. My primary purpose was to present a paper describing SIGCSE and to encourage participants from around the world to join and work with us. As noted in a *Bulletin* article [5] summarizing papers presented at this conference "SIGCSE is presented by Aiken. He reviews the organization accomplishments and goals and indicates a desire to see SIGCSE serve as an international forum for computer science education at the university level."

One of the most rewarding aspects of attending this conference was meeting leading European computer scientists and learning how they were addressing many of the same issues we were tackling in North America. A later highlight was a two month trip my wife and I took in the summer of 1973 driving through Eastern Europe, Turkey, Greece and Italy. This was during the "Cold War"—an era when it was often difficult to accomplish tasks such as crossing borders, changing money, getting necessary government approvals for vouchers, arranging meetings with colleagues at their offices, etc. We learned that patience really was a virtue! The one constant was that our colleagues, and people in general, were open and eager to develop relationships that transcended politics. SIGCSE was/is one such vehicle to overcome artificial barriers.

We mostly stayed in campsites and visited as many universities as possible. In Eastern Europe we were fortunate to visit colleagues teaching computer science at universities in Hungary, Rumania, (then) Czechoslovakia, and Bulgaria including Professors Boyan Penkov in Sofia, and Michal Chytil in Prague. Subsequently, I visited both of them giving lectures and discussing both SIGCSE and mutual research interests. I know that many of you have had similar experiences (perhaps without the camping aspect!) and have not only made numerous international friends but contacts for your departments and universities. Some of you have also shared mutual research interests that resulted in joint papers and collaborative research projects.

SIGCSE BULLETIN

The *Bulletin* has been the mainstay and lifeline for communicating among our members—particularly in the early days. The Calendar of Events was our only means for providing members with the latest information on key dates and places for upcoming events. The *Bulletin* also included Letters to the Editor, papers on current topics (often courses and curriculum) as well as observations from the Chair and Editor regarding items/topics of special interest. When I took over as Editor in 1969 (with Vol. 1 #3) we typed each issue on special mats. I hired undergraduates to do the typing and for four years we typed, edited and mailed the masters to ACM headquarters. Our ACM liaison then organized the printing and mailing. Irene Hollister, Bridget Gann and Fred Aaronson were among those at ACM HQ who were of tremendous assistance in our early days. It normally took 4–6 weeks from when we began typing and putting an issue together until our members received it. The issues were usually 20+ pages with several of them approaching 50 pages! We published three regular issues of the *Bulletin* per year plus the Proceedings as a fourth issue. It took numerous hours to put together an issue since any typing mistakes had to be painstakingly fixed, all graphs, etc. had to be hand drawn and fact checking was quite time consuming. Moving this process to a computer-based system was a HUGE plus!

John Impagliazzo was kind enough to send me some memories of when he took over the editing of the *Bulletin* in 1997. I quote from a recent note from him. "When I inherited the SIGCSE *Bulletin* from Jim Miller, he gave me two boxes of repro paper to mount articles, images, and the like. I figured it was time to throw out the old and ring in the new. So, I guess

we went from matte repro paper to computer typed processes in 1997. The magazine *ACM Inroads* began with the 2010 March issue. That was the point of bifurcation with the *SIGCSE Bulletin* continuing and *ACM Inroads* starting with Vol 1, No 1. *ACM Inroads* followed the publication policy of ACM. However, I did develop some guidelines before 2010 for the *SIGCSE Bulletin*." John later mentioned that ACM required the use of Quark software, which at the time was the professional software required by the publications division of ACM.

Even though communication modes have changed significantly with email, all forms of social media, and other assorted technology, the *Bulletin* remains (at least for me) a key way in which we share information with our members except now it is electronically rather than paper. In addition, *ACM Inroads* provides us a forum to broadly disseminate opinions, articles and research papers in computing education.

... it underlines the importance of the informal exchanges and meetings one has at a SIGCSE symposium that simply add an extra dimension to the experience.

HOT TOPICS AND DEBATES

Yes! There even were programming "language wars" and "heated discussions" on other topics in the early days of SIGCSE. For example, should one teach FORTRAN, ALGOL or, later, PASCAL as the programming language in the first course for computer science majors? Should machine or assembler language programming be included as a requirement for a computer science major?

However, even more strongly worded opinions surfaced in discussions regarding whether computer science should be strictly a graduate level offering—both at master and PhD levels—or whether it was robust enough as a discipline to be offered as an undergraduate major. Of course, we reached no definitive conclusions, but the opinions expressed were certainly eloquent and often vociferous. As time went on, these questions gave way as to how to best accommodate the increasing number of students who wanted degrees of all three types AND how best to provide the quality and quantity of professors and mentors to teach and guide them.

As noted above there were also early conversations regarding whether undergraduate computer science programs, IF developed, should follow some type of accreditation process. Some members felt we should develop a set of guidelines that would provide a minimum set of courses/topics required for a CS major. Others said that the discipline was not sufficiently mature to have agreement on what the minimum guidelines should be. The minutes of the SIGCSE meeting at FJCC in 1969 [3] already indicated that there was considerable interest in this topic though the only consensus was that accreditation in computer science "is not in the immediate future."

Various curricular debates spiced many SIGCSE meetings (as well as other CS educator symposia). For example, I had an ongoing "argument" with Niklaus Wirth regarding the role of Artificial Intelligence (AI) in computer science ... and as to whether it was really a "discipline." While I maintained it was, Niklaus saw it more as a facet of psychology rather than a field unto itself. I don't think either of us was able to convince the other of our differing positions. But our discussions, both formal and informal, were always lively and informative, and other colleagues were quick to offer their opinions! I certainly benefited from these discussions as it helped form my later research interests, particularly the role of AI in CS Education. Perhaps you have had a similar experience that led to sharpening your research focus and/or proposals.

Discussions such as these have been a mainstay of SIGCSE meetings. Many of them begin in formal settings such as a paper or panel session and then evolve into informal chats in lobbies, over drinks and/or dinner. I remember a particularly lively evening when Joe Weizenbaum elaborated on some of the major themes in his (then) recently published book, "Computer Power and Human Reason." [7] A small group of us sat around a table in the lobby of a (now forgotten) hotel listening to Joe discuss his thoughts about computing, what computers could do, might do and definitely what they should not do. This was based not only on his famous program, ELIZA, but the reactions to it and the different ways it was perceived by society. I mention it since for me it underlines the importance of the informal exchanges and meetings one has at a SIGCSE symposium that simply add an extra dimension to the experience.

COMPUTER SCIENCE CONFERENCE

After our first technical symposium in Houston (1970) and our second at Washington University in St. Louis (1972), SIGCSE decided it would be beneficial to collaborate with another group of computer educators. Starting in 1973 the technical symposia occurred in conjunction with the Computer Science Conference (CSC). This conference focused on research and primarily consisted of a few invited speakers and numerous short abstracts of ongoing computer science research activities. It provided a forum for doctoral candidates to discuss their work and for departments to recruit, a primary feature of this event! Originally, several large universities and corporations sponsored it and ACM handled many of the organizing details.

CSC was a larger and more financially secure organization. So, although SIGCSE had its own set of terrific volunteers to organize and host the symposia, SIGCSE needed to cede to the CSC organizers many of the decisions such as choice of lo-

cation, registration, exhibitions, and finances. The few notes I could find indicate this partnership ended with the joint conference in Philadelphia in 1996. Shortly thereafter CSC ceased operation.

Speaking of the 1996 SIGCSE Technical Symposium there were a number of significant aspects that made it noteworthy. For example, it was the start of the 50th anniversary of ACM; it hosted the last Computer Science Conference (Bob Beck was chair); and it witnessed the big chess battle between man and machine (Gary Kasparov against Deep Blue) [1]. It was also the 50th anniversary of the ENIAC and then Vice President Al Gore (who did not show up) was going to throw the switch at the University of Pennsylvania to commemorate that event; there was even a history session where many pioneers were present. John Impagliazzo, who chaired this symposium, reminded me of several of these activities and provided salient details, and also noted that Nell Dale engaged in her first snowball fight with none other than …. John ☺!

This was a particularly special symposium for me since I was living in Philadelphia and was able to host a number of friends who were attending the conference. I'm sure you have similar symposia experiences that strike a particularly pleasant chord with you … which leads to me to …

What do the following cities have in common: Houston, St. Louis, Columbus, OH, Detroit, Washington D.C., Anaheim, Atlanta, Detroit, Pittsburgh, Dayton, Kansas City, Indianapolis and Orlando? One could certainly say that they are cities with many attractions and/or have interesting histories. However, that is not a SIGCSE-related answer … so try again!

These cities were hosts to the first fourteen SIGCSE Technical Symposia (with St. Louis hosting two of them). As one can note, most of them are in the Midwest. This resulted from several factors including their proximity to numerous universities that strongly supported both the CSC and the Technical Symposia through providing volunteers (both faculty and students), as well as financial support. Additionally, these selections provided an opportunity for large numbers of attendees from these and surrounding regions to drive or bus to both conferences. In those days air travel was rather expensive, and many universities did not have large travel budgets. These locales also provided hotels and meeting space at prices more reasonable than in larger cities. Keeping travel costs as low as possible while providing attractive and reasonable accommodations were (and remain) critical factors in choosing sites for holding the symposia.

IMPACT OF SIGCSE

It is difficult to fully assess the impact that SIGCSE has had on its members as well as the field of Computer Science Education—both directly and indirectly. For me, it has had a significant impact both professionally and personally. I have already mentioned the value of the connections we have made with numerous colleagues and how much we learn at our symposia and through the *Bulletin* and *ACM Inroads*. Moreover, that does not take into account other factors such as learning about new technologies, books and practices in the exhibition area. Also, discovering new NSF programs through which we can submit proposals … never mind all the NSF programs that have funded SIGCSE related projects over the years.

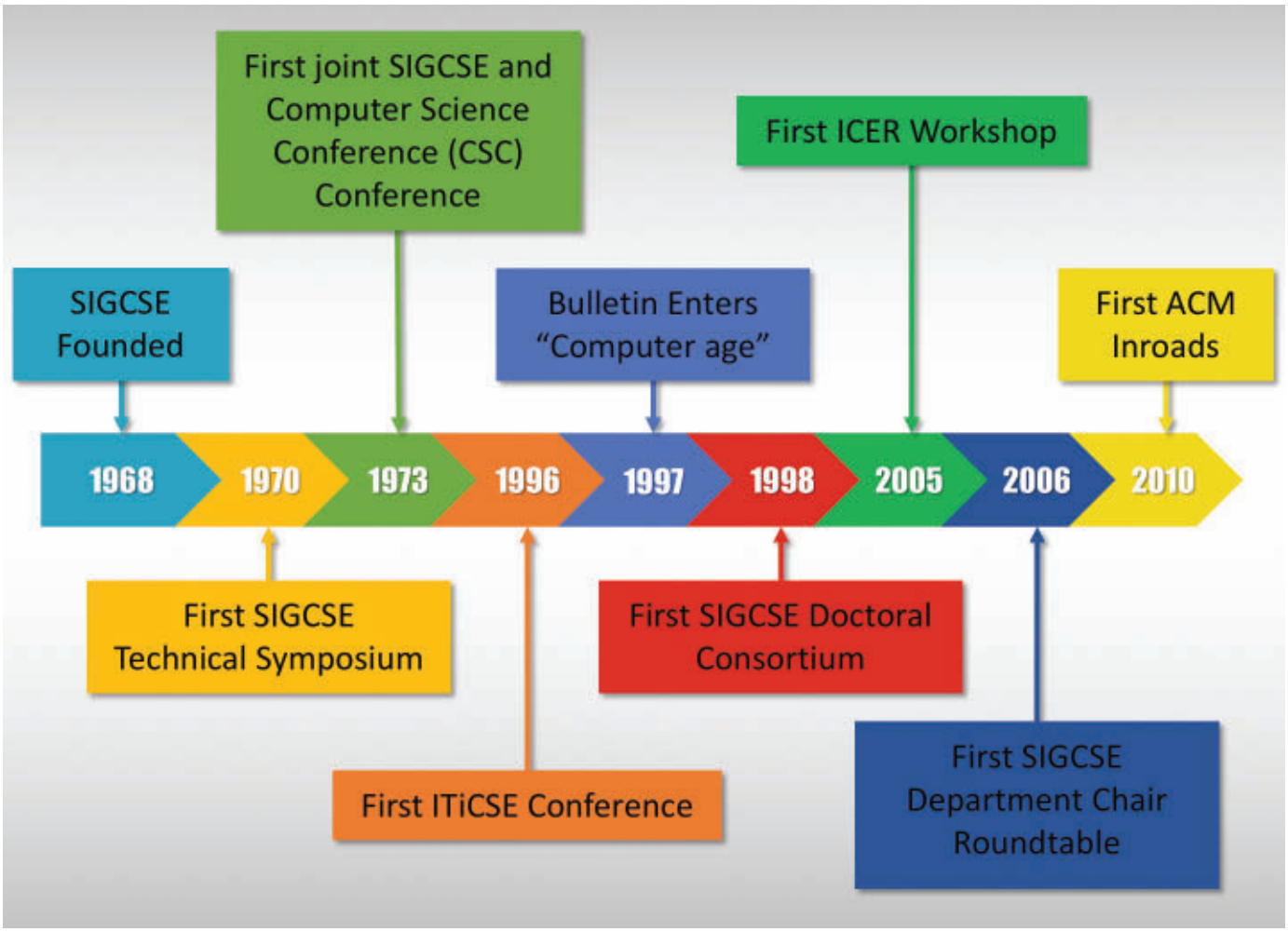
As an organization SIGCSE has worked with other entities to further our discipline. Foremost among these has been our close working relationship with the ACM Education Board (and its successors). SIGCSE, through the efforts of a number of its members, has also worked closely with other organizations, such as the IEEE Computer Society, ABET and the Educational Testing Service on a number of important projects including:

- Curriculum Development (and subsequent reports)
- Accreditation Activities
- Pre-college Education
- *ACM Transactions on Computing Education* (TOCE)
- Special Interest Group for Information Technology Education (SIGITE)
- Preparing and Grading CS Advancement Placement Tests

Not to mention that SIGCSE has supported/funded a variety of its own activities that have directly benefited and recognized our membership. In constructing this list, I was surprised at how many activities SIGCSE has initiated; among others:

- *Bulletin*
- *ACM Inroads*
- Annual SIGCSE Technical Symposium
- Innovation and Technology in CSE Conference (ITiCSE)
- Doctoral Consortium
- International Computing Education Research Workshop (ICER)
- Outstanding Contributions to Computer Science Education Award
- Lifetime Service to the Computer Science Education Community Award
- Committee on Computing Education in Liberal Arts Colleges
- Regional Conferences
- Local Chapters
- Travel Grants
- Special Projects Funding
- "First-Timer" Luncheons
- Listservs

Following is a figure providing a timeline of highlights from SIGCSE's growth and evolution. One should not regard this as a definitive list but rather a starting point. I know many of you can add your own interesting perspectives/experiences to this timeline—whether it is changes in the procedures for how papers were submitted and reviewed, to changes in registration processes, etc. SIGCSE continues to be a vibrant community of dedicated volunteers who impact and facilitate Computer Science Education in many different ways. Perhaps you can share your thoughts by sending a letter to the Editors!



BUILDING STRENGTH UPON STRENGTH THROUGH FRIENDS AND COLLABORATIONS

For me, the highlight of SIGCSE (especially the symposia) was meeting friends and making new ones. It was an especially fortuitous opportunity to visit and exchange information with my international colleagues. Unfortunately, as I have "matured," many of these colleagues have passed away. Yet, I still have memories of all the good times, especially the discussions and educational activities we shared. This spirit and camaraderie is perhaps the most important aspect we experience today and forms a legacy for SIGCSE's future.

Moreover, the social aspects of the meetings and the symposia provided highlights as well. In the early days, there was a small group of us that started a "What is the best restaurant in this town?" We would check with locals who would steer us to their 'secret tips' (before apps and sites like Open Table). We enjoyed stellar meals (and libations) in unique environments from Detroit to Cincinnati to Dayton to St. Louis and many locales in between … and beyond. Participants from the beginning included Della Bonnette, Norm Gibbs, Jim Miller, Joe Turner, and myself. Afterwards I would host a friendly poker game in my room in which several SIGCSE-ers participated over the years—some with more success than others ☺!

I hope you agree that the friends we make and the experiences we share with our SIGCSE colleagues rank at the top of our most memorable professional moments. Even in this day of social media, instant communication and electronic sharing of all types, there is nothing to compare to the face-to-face meetings, the opportunity to attend presentations and panel sessions, to peruse posters, and in general to interact directly with our colleagues.

There are so many members who have contributed to the success of SIGCSE that it is not practical to even begin to name all of them. However, let me mention a few from the very early days of SIGCSE. As we look back in history, we find that Elliot Organick (University of Houston) was the driving force behind SIGCSE and our first chair. Aaron Finerman (SUNY at Stony Brook and JPL) was our next and first elected chair. Ed Feustel (Rice University) was the local activities chair and Peter Calingaert (University of North Carolina) was the program chair for the first technical symposium in Houston. For SIGCSE's second technical symposium, the conference chair was Seymour Pollack (Washington University) and the program chair was Leland

Williams (Triangle Universities Computation Center). The first recipient for the SIGCSE 'Outstanding Contributions to Computer Science Education' award in 1981 was Bill Atchison (University of Maryland) and the second recipient was Alan Perlis (Carnegie-Mellon University). The SIGCSE Award for Lifetime Service to the Computer Science Education Community was initiated in 1997 and first awarded to Dick Austing (University of Maryland) and the second to Della Bonnette (University of Southwestern Louisiana). Moreover, one should not forget that Della and Jim Miller (University of Southern Mississippi) were diligent and dedicated early editors of the Bulletin.

Several couples were also major contributors. For example, early on Betty and Larry Jehn (University of Dayton) and later Mary Ann and Dick Austing headed up our symposium registration process. For those who are interested, you can browse through original back issues of the *SIGCSE Bulletin* through the excellent ACM digital archives of the *Bulletin*. One can discover countless colleagues who have been officers, committee chairs/members and those who served in many other volunteer positions for SIGCSE. That does not even take into account all the authors, reviewers, poster presenters, corporate sponsors and student members who contributed in myriad capacities.

I apologize to those who made important early contributions but whom I have not specifically noted. We have benefited significantly from all their efforts and continue to build on their early support. I would be remiss if I did not also "doff my chapeau" to all the SIGCSE officers and volunteers who over the past fifty years have contributed immeasurable time and effort. I hope that you will thank them when you have an opportunity—and volunteer yourself. You won't regret it! ☺ We need your efforts and enthusiasm as we embark on our next "50 Year Journey"!

EPILOGUE

After writing this piece I reflected on how much our field has changed. I cannot think of any discipline that has transformed/revolutionized and evolved as much in these past 50 years. Not only has the content changed dramatically but so has the way we teach. When we started SIGCSE computer mainframes were massive, such as the IBM 360 series. No one had a computer in his or her office. For large jobs we were fortunate if we had more than one hour on the massive university mainframes (often planned well in advance) and batch programs were run overnight. We moved on to a text-based internet which allowed us to communicate with our colleagues on simple bulletin boards and email. Then came the graphical WWW which again changed the paradigm and impacted the way we taught and communicated. We set out to create web sites and some simple interactive learning tools, still very limited by speed (dial up) with most people only having access to the internet at their local libraries or universities. This transition accelerated as we added broadband and Wi-Fi to homes, schools and businesses while losing the race for our best and brightest students to the first wave of startups (dotcom boom). We developed online learning platforms which have had to adapt to the next revolution of mobile devices, social media and MOOCs. Students nowadays find email and the "traditional" WWW outdated but I, for one, am still "stuck" in that world. As our field moves even more swiftly into the next fifty years no one can imagine where we are headed. For example, how will AI and unknown technologies change our field?

As our field moves even more swiftly into the next fifty years no one can imagine where we are headed. For example, how will AI and unknown technologies change our field?

As a 77-year old in 2018, I have learned that the relationships/friendships are the constant and SIGCSE has been the vehicle for the ride. As exciting as technology has been/is, the people who took this journey with me provided the greatest experiences and memories. My hope is that you also enjoy these exciting times and treasure the memories of your SIGCSE experiences and friendships. SIGCSE has and will continue to evolve with the new technologies and associated challenges (diversity and privacy among the many). I look forward to reading about what the new generation of 27-year-olds achieve and experience over their 50-year cycle! ❖

Acknowledgements

I would like to give a special "shout out" to John Impagliazzo for providing insight and information that added depth to this article. Also, David Aiken for reviewing and adding substance to several drafts, as well as preparing the figure. Finally, thanks to Laurie King for assisting with the formatting.

References

Most of the information included in this article stems from my fallible memory and items in the following *SIGCSE Bulletins* (from the ACM online digital archives).

1. Deep Blue; https://en.wikipedia.ofllowing rg/wiki/Deep_Blue_(chess_computer). Accessed 2018 May 1.
2. *SIGCSE Bulletin*, 1, 3 (1969).
3. *SIGCSE Bulletin*, 1, 4 (1969).
4. *SIGCSE Bulletin*, 2, 1 (1970).
5. *SIGCSE Bulletin*, 2, 5 (1970).
6. *SIGCSE Bulletin*, 50, 1 (2018).
7. Weizenbaum, J. *Computer Power and Human Reason*, (W. H. Freeman and Company, 1976).

Robert M. Aiken
Professor Emeritus
CIS Department
Temple University
Philadelphia, Pennsylvania USA
aiken@temple.edu

DOI: 10.1145/3230703

Where To From Here?

Our curriculum is not up to the challenges the world is throwing at us.

Peter J. Denning, *Naval Postgraduate School*

When I received my doctorate in EE from MIT in 1968, the name of the young field of computer science was just getting settled. I was amazed at the audacity of the dreams of the founders and pioneers. I was completely drawn in and I developed a romance with computing that has never faded.

I have had the good fortune to witness the growth and maturing of this field of education and research for the entirety of SIGCSE's fifty-year existence. My purpose here is to look at the high points of computing and computing education over the past fifty years. Several major forces shaped the computing curriculum we have today. Our curriculum is not up to the challenges the world is throwing at us.

SHAPING FORCES OF COMPUTING

Our modern age of electronic computing began in the late 1930s and spawned computing education in the late 1940s. Computing in the sense of methods and machines to automate calculation and logical deduction is much older—it evolved over at least 40 centuries before our age. Prior developments such as procedures for doing algebra, solving equations, evaluating series, Pascal's arithmetic calculator, Napier's logarithms, Newton-Leibniz calculus, LLull's logic wheels, Babbage-Lovelace analytical engine, slide rules, and human calculator teams set the framework of computational thinking that existed when computer science was founded in the 1940s. I will focus here on the main historical forces that shaped our field and how we approached our curriculum since that time.

Machinery and systems. The first electronic digital computers were built in the 1930s and early 1940s—Zuse in Germany in 1938, Atanasoff and Berry in the US in 1942, Eckert and Mauchly at Pennsylvania in 1945. All were engineers who believed that high speed computing would benefit science and engineering and would automate many human computational tasks that were prone to errors.

Their machines were great feats of engineering. They had to work out everything—how to represent data as signals in the machines, how to build reliable circuits that would perform logic operations on the data, how to store data, how to get data in and out of the machines, and how to design algorithms that would control the machines. There was no theory to guide them.

Although Alan Turing proposed his Turing machine theory of computation in 1936, his work was initially known primarily by a handful of mathematical logicians and was completely unknown to the engineers who built the first electronic computers [6]. Turing became more known among computer builders when he circulated his own detailed engineering design of his ACE computer, inspired by von Neumann's notes on the design of the stored program computer in 1945. It was not until the 1950s, when the first academic programs were being born, that Turing's work offered the theoretical basis to make computer science credible as a new department in universities. In other words, as important as Turing's work is, it did not inform or inspire the first electronic computers or the stored program concept. Instead, the success of the first stored-program electronic computers created the opening for Turing's theoretical work to become important.

For the first 40 years of computing, much of our energy was focused on advancing the technology for reliable computing and networking. Our early curricula reflected this by organizing around core technologies, such as programming languages, operating systems, and networks. The 1989 computing report named 9 core technologies [2]. The 2013 curriculum report named 16 core technologies among its 18 main knowledge areas [1]. Today's curriculum bears the imprint of the engineering concerns that started the field.

By 1982, there were about 120 departments in the US and Canada. Most of these early departments were formed amidst resistance from other departments in their universities, which saw computer science as a specialty of math or electrical engineering, but not as a separate department.

Academic Resistance. The first computing and programming courses appeared in the late 1940s. The first CS departments were Purdue and Stanford both in 1962. By 1982, there were about 120 departments in the US and Canada. Most of these early departments were formed amidst resistance from other departments in their universities, which saw computer

science as a specialty of math or electrical engineering, but not as a separate department. Most early departments were therefore founded within the most hospitable school—some in science, some in engineering, and few in business. The academic pioneers of the day—notably Forsythe, Newell, Perlis, Simon—spent a great deal of effort defending the new field against skeptics who thought it was neither a new field nor deserving to be called a science [5,7]. Their ideas guided the early development of computer science education. Because most CS departments were in schools of science or engineering, the term CS&E (computer science and engineering) became the collective term for all the departments. After 1989, at the recommendation of the ACM/IEEE computing-as-discipline committee, the term "computing" was used instead of "CS&E" [2]. Europeans preferred the name "informatics." By 2000, the resistance was pretty much gone as biology, physics, aeronautics, and other fields declared they dealt with natural information processes.

Formal methodists see mathematical proof as the only means to establish that software is error-free. Software engineers see formal methods as one tool of many—other tools are needed to address defects in hardware, deterioration of hardware, error confinement, and detection of malware.

Software Engineering. In 1968-69, software developers called for a new field, software engineering, because the existing approaches to software development were not able to take care of concerns for dependable, reliable, usable, safe, and secure (DRUSS) production software. The founders of software engineering believed that engineering perspectives such as fault tolerance, redundancy, and interface design could help. CS departments responded by setting up a software engineering course and over time some developed "tracks" containing several courses. Some universities set up a separate IT curriculum, sometimes as a track in the CS department and sometimes as a stand-alone department. In a few rare cases, software engineers formed their own departments where all the faculty could engage with engineering perspectives without being constrained by the more abstract ways of CS departments.

Networks. In the late 1970s many CS departments were unhappy that only a few of them were connected to the ARPANET, which was restricted to defense contractors. They banded together and won National Science Foundation support to design and build CSNET (computer science network). I was one of the four co-PIs. By 1986 we had adapted ARPANET technology and built a CS research community network of 50,000 users at 120 member institutions. The network significantly increased research productivity in all participating CS departments. CSNET gave NSF confidence it could manage a large network project, NSFNET, which became the backbone of the modern internet. Around 1989, ARPANET, CSNET, and NSFNET were decommissioned because everyone got the networking they needed from the internet.

Computational Science. Scientists had a long-standing interest in computation well before the 1940s. By the 1950s, with help from numerical analysts, they were using electronic digital computers for calculating prediction from complex models and for analyzing experimental data. By the 1980s they had gone well beyond this—they had developed their own brand of computational thinking that they saw as a new method of doing science. CS researchers responded with mixed reactions. Some embraced computational science as a welcome expansion of computing; others resisted it as the work of amateurs with little experience with computing. Eventually, CS researchers and educators overcame their reluctance and embraced it. Computational science appeared as a core area in the 2013 curriculum.

Formal Methods. A very large debate opened in the 1980s about the power of formal methods to give us reliable and dependable software [9]. Proponents called it the most important of all computing research. Skeptics cited all sorts of reasons that mathematical proof was insufficient for the DRUSS software objectives. The sharp words from all sides eventually quieted down, but the underlying tension endured. It is the same tension as between the traditional math-science oriented computer scientists and the software engineers. Formal methodists see mathematical proof as the only means to establish that software is error-free. Software engineers see formal methods as one tool of many—other tools are needed to address defects in hardware, deterioration of hardware, error confinement, and detection of malware. I wish the tension would go away, but it is still there. The two views are complementary and mutually reinforcing.

Artificial Intelligence and Machine Learning. AI was founded in 1954 in pursuit of the goal of general machine intelligence. It looked at language translation, image recognition, checkers, chess, problem-solving, robots, expert systems, and machine learning as steps on the way to that goal. By the mid-1980s the field had delivered so little of its big promises that the research funding agencies began to withdraw their support, precipitating the long doldrums dubbed "AI winter." The controversial 1987 book *Understanding Computers and Cognition* [11] argued that the quest for machine intelligence was fatally flawed and proposed that we devote our energy to designing machines that support human practices. That advice became the key to a resurgence of machine learning (ML) in the 2000s, when researchers found that they could apply the technology

of neural networks to large classes of pattern recognition and prediction, with astonishing success. In recent years Machine Learning and Big Data Analytics (BDA) have come to depend closely on one another. Referring to the success of ML and BDA as "AI" is a misnomer because neural networks are unintelligent pattern recognizing machines.

Parallel and distributed computing. From the 1960s computer scientists developed strong interests in computations performed by many processors working together. Because of their significantly greater speed, parallel processors became the mainstay of supercomputing, which became very popular in science, engineering design, medical and drug research, entertainment, and more, and fomented the revolution of computational science. Also, from the 1960s, computers distributed into many physical locations connected by a network provided access to remote services and showed great resiliency to failures. The 1960s dream of computer utility matured into the modern distributed systems making up "the cloud." This long line of developments left a permanent imprint on our curricula.

The idea that computing is universally valuable pervaded the thinking of the founders of computer science. Beginning in 1960, pioneer Alan Perlis repeatedly said that computer automation would spread to many fields ...

EVOLVING VIEW OF WHAT COMPUTING IS

The question of whether computer science was unique or was a science persisted for many years. Our views of what computing is evolved through four stages over the years [4].

- In 1960s, we said we studied phenomena surrounding computers.
- In the 1970s, we said we studied programming and all that entailed about algorithms, analysis, and correctness.
- In the 1980s, we said we studied automation, what could be efficiently automated by digital computers.
- In the 1990s, as other fields of science started to claim their fields included naturally occurring information processes, we said we studied information processes natural and artificial. After that, the old debate about whether computer science is science, or deserves its own academic department, completely faded.

The curricula we taught evolved along with these maturing views of the nature of the field. In 1968, ACM produced its first curriculum recommendation, the first attempt at standardizing what a computer science degree meant. In 1972 the NSF-sponsored COSINE (computer science in engineering) project advocated placing systems courses in the core curriculum, alongside the traditional math courses already there; operating systems was the first systems course to be accepted into the CS core. In 1989 the ACM and IEEE Computer Society (IEECS) cooperated on the first joint recommendation that emphasized the integration of theory, abstraction, and design—representing math, science, and engineering—in the core. They sought to ease the tensions between these three subgroups of the field. They began calling the field "computing" rather than "computer science and engineering." Those ideas dominated the 1991 ACM/IEEE computing recommendations. ACM and IEEECS continued their cooperation and produced major updates in 2001 and 2013. The growth of the field can be seen in the increasingly complex recommendations over the years. The 1968 curriculum had three major subdivisions; the 2013 curriculum had 175.

In recent times, our virtual machine technologies and platforms have improved so much, and chips and sensors shrank so much, that most designers of software are seldom aware of hardware. Some educators have argued that we no longer need to be concerned about hardware; we should drop our insistence that algorithms and software are intended to control machines. Instead, we should view algorithms and software as expressions of methods to solve problems that can be shared and communicated with others, a view that dominated the design of the ALGOL language in the 1950s. In his 1968 ACM A.M. Turing lecture, Richard Hamming took a dim view of the idea that we can abstract the machine out of the picture. He argued that the computer is at the heart of computing; without it, almost everything computing professionals do would be idle speculation. Hamming's insight remains valid today: there can be no computing without computers.

Along the way there has been an ongoing debate about what programming language(s) to use in CS courses. Should they be languages used heavily in industry, such as C, Java, or Javascript? Or languages designed for easy learning of basic programming concepts, such as Pascal or Python? It is ironic that on the one hand computer languages are equivalent in expressive power, while on the other hand language choice is the most fiercely debated issue in teaching computing. This debate is unlikely to end.

QUEST FOR COMPUTING EVERYWHERE

The idea that computing is universally valuable pervaded the thinking of the founders of computer science. Beginning in 1960, pioneer Alan Perlis repeatedly said that computer automation would spread to many fields and draw many people into "algorithmizing"—his term for what we now call computational thinking.

Computing educators became interested in the 1970s in bringing computing's general-purpose thinking tools into K-12 schools. That was a major challenge: few schools had teachers with computer science knowledge. Computer literacy was seen by many as a gentle first step toward getting computer courses into grade schools. The first attempts at literacy courses were little more than training in how to use word processors and spreadsheets. They were not popular with students or teachers. A turning point came in 1999, when a task force of the National Academy of Engineering issued a report reframing the goal from literacy to fluency. Larry Snyder, the chair of the task force, wrote *Fluency in Information Technology*, a textbook that became popular with high school teachers [8].

We arrive at our 50th anniversary of the founding of SIGCSE with a curriculum specifying what (and how) we teach computer science, a curriculum that evolved over half a century.

Also in the 1990s, the College Board became interested in an upgrade to the advanced placement test in computer science. With help from ACM and IEEE, they launched around 2000 an advanced placement (AP) curriculum focused on object-oriented programming with the Java language. Within a few years, AP enrollments plummeted as students and teachers discovered the material was too complex for beginners. The College Board, in cooperation with the US National Science Foundation, undertook a new advanced placement curriculum organized around computer science principles, which it hoped would provide a better return on their investment. The upgrade was rolled out in 2016.

In 2006 Jeannette Wing reframed the issue again around "computational thinking" (CT) which she characterized as the thought processes that computer scientists used to solve problems [10]. This formulation resonated with many people who saw computing permeating into their fields and wanted to learn how to harness the technology. As an assistant director for the Computing & Information Science & Engineering (CISE) directorate at the National Science Foundation, Wing mobilized many people and resources around the goal of getting a computing curriculum based around computational thinking into every K-12 school. They sought to train 10,000 teachers in computer science. They supported the development of the CS principles advanced placement curriculum and concurrently the development of a new genre of CS principles first courses in universities. Many organizations stepped up to define K-12 curricula around computational thinking, producing several proposals for teachers to choose [3].

Even with all this backing the new curricula have been slow to find their way into K-12 schools and some of the teachers are still concerned about what they should teach and how to assess whether students have learned it.

The definitions of CT in these proposals are quite narrow compared to the breadth of pressing computational issues in the world—they do not apply to complex systems, reliability concerns, hardware, or emerging technologies such as quantum computing. CT is not the defining characteristic of computer science. Neither is it "the way of thinking of computer scientists" because many in other fields have contributed significantly to our understanding of computation.

EDIFYING CONVERSATIONS ON BIG QUESTIONS

We arrive at our 50th anniversary of the founding of SIGCSE with a curriculum specifying what (and how) we teach computer science, a curriculum that evolved over half a century. The specification was shaped by many factors noted here.

- Strong emphasis on building technologies at the beginning
- Resistance to forming CS departments from other academic departments that did not accept computing as a legitimate field
- Developing our own community network at the dawn of the internet era
- Being torn by intense debates over the roles of science, math, and engineering in our field, manifested as struggles over how to teach software engineering and information technology, and how much to trust formal methods for software development
- Coming to grips with the emergence of computational science and now the penetration of computing into nearly every field of human endeavor
- The death of artificial intelligence and its resurrection as machine learning and its claims about automation and the future of humanity

This battle-hardened inheritance does not help us with many of the pressing issues of the world emerging around us. The worldwide connectivity we helped bring about through the Internet has brought many benefits from shrinking the world and globalizing trade. But it has also spawned conflicts between non-state organizations and traditional nations, trade wars, protectionism, terrorism, widespread detachment, fake news, political polarization, and considerable unease and uncertainty about how to move in the world. Access to troves of information via the internet has begun to show us that knowledge does not confer wisdom, and we long for wise leaders who have yet to appear. The world we encounter in our daily lives is full of surprises, unexpected events, and contingencies that not even our best learning machines and data analytics can help us with. We are now finding that many resources including sea and air access are contested among nations; we lack means to resolve the resulting disputes and we worry that the resulting conflicts

[H]ow shall we shape computing education so that our graduates can develop the design sensibilities, wisdom, and caring they will need to navigate in this world of which they will be citizens? Our current curriculum, chock full of courses covering the 2013 body of knowledge, is not up to this task.

could trigger wars or economic collapses. We see that collective human action affects the global environment but have yet to find ways to protect the environment we will bequeath to our children and grandchildren.

This leaves us with a big question—how shall we shape computing education so that our graduates can develop the design sensibilities, wisdom, and caring they will need to navigate in this world of which they will be citizens? Our current curriculum, chock full of courses covering the 2013 body of knowledge, is not up to this task.

A place to start would be to open space in our crowded curriculum to have conversations on big questions about the consequences of computing throughout the world. These conversations need to be interdisciplinary and intergenerational. Their purpose would not be to solve problems but to edify—develop mutual understanding, appreciation, and respect around these issues. Some examples of big questions are:

- How far can automation take us? Can everything be automated? Is there always something important left over that cannot be automated?
- Will AI displace more jobs through automation than it generates?
- How can we help people whose jobs are displaced by software and hardware we have designed?
- How do we cultivate good designers?
- Can we trust decisions by neural networks when given inputs outside their training sets?
- Will drones and robots combine to create an automated surveillance society?
- Is there a technological solution to the cybersecurity problem?
- Can we make our world work when computers have been embedded into almost all devices connected to the global network?
- Can blockchains and cryptocurrencies solve our problems with trust in central authorities? Are they too expensive to maintain?
- Is civilization so dependent on computing that an attack on a component of infrastructure, like electric grid, could collapse civilization?
- What is the difference between wisdom and knowledge? How are we fooled into thinking that massive internet information is wisdom?
- What are the social implications of brain-computer interfaces and implants into our brains and bodies?

I do not believe any of us has answers to any of these questions. But we need to be having the conversations about them. In so doing we need to embrace the mathematicians, scientists, and engineers in our field. It is time to give up the old tensions that we inherited from times long past, and work together as brothers and sisters, mothers and fathers, old and young on these big questions. ❖

Acknowledgements

I thank Matti Tedre for historical and technological insight in our conversations about the topics discussed here. I also thank Fernando Flores for giving the name "edifying" to the kind of conversation we need more of in education, and for edifying conversations to better understand the world our technology has shaped.

References

1. ACM. *Computer Science Curricula 2013*; https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf. Accessed 15 Feb 2018.
2. Denning, P., Comer, D.E., Gries, D., Mulder, M. C., Tucker, A., Turner, A. J, Young, P. R. Computing as a discipline. *Communications of ACM* 32, 1 (1989), 9–23.
3. Denning, P. Remaining trouble spots with computational thinking. *Communications of ACM* 60, 6 (2017), 33–39.
4. Denning, P. and Martell, C. *Great Principles of Computing*. (MIT Press, 2015).
5. Forsythe, G. E. (1968). What to do till the computer scientist comes. *American Mathematical Monthly* 75 (May 1968), 454–461.
6. Haigh, Thomas. Actually, Turing did not invent the computer. *Communications of ACM* 57, 1 (2014), 36–41.
7. Newell, A., Perlis, A. J. and Simon, H. A. Computer science. *Science* 157, 3795 (1967), 1373–1374.
8. Snyder, L. *Fluency with Information Technology: Skills, Concepts, and Capabilities*, 7th Edition (Pearson, NY, NY, 2017).
9. Tedre, M. *The Science of Computing: Shaping a Discipline*. (CRC Press / Taylor & Francis, New York, NY, USA, 2014).
10. Wing, Jeannette. Computational thinking. *Communications of ACM* 49, 3 (2006), 33–35.
11. Winograd, T. and Flores, F. *Understanding Computers and Cognition*, (Addison-Wesley Professional, 1987).

Peter J. Denning
Naval Postgraduate School
Monterey, California, 93943 USA
pjd@nps.edu

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for information innovation at the Naval Postgraduate School in Monterey, CA. He is Editor of ACM *Ubiquity* and is a past president of ACM. The views expressed here are his alone and are not necessarily those of his employer or the U.S. Federal Government.

DOI: 10.1145/3191833

Reflections on SIGCSE From the Past 30 Years

Susan H. Rodger, *Duke University*

Imagine attending a conference and going to a lunch that would change the direction of your life. That was me at SIGCSE 1994 finding out about a new teaching position. Since 1993, I have attended every SIGCSE Symposium and many other SIGCSE conferences; they have impacted my life in many ways. I have met colleagues, integrated ideas I learned into my teaching, and even changed the direction of my career from tenure-track research to teaching-track focused on computer science education. In what follows I reminisce about my experiences with SIGCSE and its impact on me over the past 30 years.

AN EARLY CLOSE ENCOUNTER WITH SIGCSE

My first encounter with SIGCSE was in 1989, but I didn't attend any of the sessions and didn't realize until later what I had missed. At the time, I was a graduate student at Purdue finishing my PhD and looking for a faculty position. With a two-body problem I was considering both research and teaching positions. Teaching positions at that time meant teaching at a 4-year college. I did not know of any such positions at research institutions. I heard about the 1989 ACM Conference on Computer Science that was held in Louisville, KY, close to Purdue. At the time, the conference was a general computer science research conference held in conjunction with the SIGCSE Symposium. SIGCSE had interview booths for teaching positions. I came just for the interviews, interviewing with several small colleges. I was nervous about the interviews and did not attend any of the conference. I would not go back to the SIGCSE Symposium for several years.

MY FIRST TEACHING POSITION

In Fall 1989, I started an Assistant Professor position at Rensselaer Polytechnic Institute (RPI). At RPI, I taught my own class for the first time and fell in love with teaching. RPI had decided that they would create a two-course sequence that weaved together the CS 2 course (data structures and analysis) with an automata theory course. The idea was to teach automata theory with related programming assignments and corresponding data structures. The CS Department gave this task to two of its new hires, Ellen Walker and me. Immediately I wanted to help students use software to visualize and experiment with theoretical concepts. I worked with students to develop software for visualizing automata theory concepts and for visualizing algorithms and data structures. In March 1992 I presented a software tool for pushdown automata at my first Computer Science Education (CSED) conference, the DIMACS workshop on Computational Support for Discrete Mathematics [6]. I met

Figure 1: Learning Scratch at SIGCSE 2008 Kid's Camp.

PHOTO: ©COLE RODGER PHOTOGRAPHICS

Figure 2: SIGCSE 2008 Conference Committee: (starting bottom L and clockwise): Susan Rodger, Don Kirkwood, Jeff Forbes, Dan Garcia, Tammy VanDeGrift, Lynn Degler, Larry Merkle, Henry Walker, John Dooley, Mark Guzdial, J.D. Dougherty, Lisa Kaczmarczyk, Sue Fitzgerald, Pam Cutter, and Steve Wolfman. (center L to R): Ellen Walker and Cary Laxer.

many new colleagues there, with many of them being researchers focusing on ways to visualize graphs. Alas, this was a one-time workshop and many of the colleagues I met were focused on their research careers.

MY FIRST SIGCSE SYMPOSIUM IN 1993

Two colleagues at RPI saw my passion for CSED and suggested I submit my work to the SIGCSE Symposium. At the time I did not realize this was the same conference I had interviewed at years before. So, in February 1993, I attended my "first" SIGCSE symposium, presenting a visualization for a parallel sorting algorithm [11]. This conference was incredibly different from the theory conferences I had attended. At theory conferences I was in the minority as a woman, and the conferences felt too competitive. At the SIGCSE Symposium everyone was friendly and eager to talk about ways to make their courses better and more interesting to students. The speakers that year were Alan Kay and Elliot Soloway, two very passionate people about CSED. At the conference I met several colleagues with similar interests, especially those who were interested in algorithm visualization. There were also high numbers of women. Five of the SIGCSE Board Members were women and several became role models for me. Nell Dale was Chair, Boots Cassel was Vice-Chair, and Harriot Taylor was Secretary-Treasurer. They were all participating heavily in the conference in panels and papers. I felt immediately that SIGCSE was a community that I wanted to be a part of.

How was the SIGCSE 1993 conference different than today? It was a two-day conference with 60 papers, 21 panels, BOFs, and 11 post-symposium workshops. That's right, 21 panels! In contrast, there were no panels at theory conferences. The panels at SIGCSE were stimulating and debated a wide variety of opinions on the direction of several CSED topics. How were the topics of women and K-12 in computing covered? There were two panels on women in computing (one with Ellen Spertus and one with Anita Borg), and two papers on women in computing. For K-12 there was one panel on the ACM curriculum for high school. The SIGCSE Symposium was still being held jointly with the ACM Conference on Computer Science then, a relationship that would continue through 1996.

RETURN TO THE SIGCSE SYMPOSIUM

Returning to the SIGCSE Symposium in 1994, I presented a paper on a software tool for visualizing parsing [5]. Algorithm visualization was exploding by then with over ten papers that involved software for algorithm visualization for all kinds of areas including CS 1 concepts, operating systems and genetic algorithms. I met many new colleagues who were dabbling in algorithm visualization. It was also at this time that I went to

PHOTO: ©COLE RODGER PHOTOGRAPHICS

lunch with Rocky Ross who brought along Alan Biermann from Duke. Alan talked about a new teaching position Duke had created called *Professor of the Practice*. Little did he or I know that I would go back to RPI, think about that position on and off for about 5 weeks, before finally applying. Two months later I moved to Duke, transforming my career from a tenure-track research position to a teaching-track position.

Figure 3: SIGCSE Board in 2008: Doug Baldwin, John Impagliazzo, Dan Joyce, Renée McCauley, Wanda Dann, Ingrid Russell, Alison Clear, Barbara Boucher Owens, and Henry Walker.

What were some of the topics covered at the SIGCSE 1994 Symposium? On women in computing there was one paper and one panel. On K-12 in computing there were two papers, two panels and a tutorial. Of those, one of the panels and one of the tutorials were about the AP CS program. The SIGCSE Symposium would be a hotbed for discussion on the direction of the AP CS program for years to come. There were also papers on how to deal with large enrollments such as peer learning and automatic grading. Enrollments were large then but not at the levels we have today.

COLLABORATIONS FLOURISH

It was exciting to be a part of the explosion of visualizations aiding the teaching of CSED. After meeting many colleagues interested in this topic in 1993 and 1994, six of us presented a tutorial on visual demonstrations at SIGCSE 1995 [8]. It was very exciting to be on my first tutorial (now called special sessions) at the Symposium. These collaborations led me to organize my first workshop, the *Workshop on Interactive and Visual Tools*, held at Duke in March 1996 with 35 attendees.

Around this time in 1996 SIGCSE started a new conference, ITiCSE, to be held in Europe every year during the summer. The conference was named *Integrating Technology into Computer Science Education*, though a few years later the name was changed to the *Conference on Innovation and Technology in Computer Science Education* (keeping the same acronym). What was ITiCSE like in its first year in 1996? The first conference was held in Barcelona, Spain and was a much smaller conference then the SIGCSE Symposium. It had 10 long papers, 36 short papers, posters, demonstrations and two panels. The conference format was a three-day conference with the afternoon off on the middle day for optional tours of the local area. That format was well received and continues today. The conference also had something new the Symposium did not—Working Groups. Working group members came early to the conference, worked for five days on a topic, presented during the conference to get quick feedback, and finished a substantial paper by the last day. I attended ITiCSE that summer and participated in the Working Group on Visualization [4]. Working groups are a great way to meet international colleagues, get to know the working group members well, and develop long-term collaborations. How were the working groups different then? That first year we started and finished the paper all in five days! I remember it as being quite intense, staying up late that last night. Now Working Groups begin their work before they arrive at the conference, sometimes collecting survey data in advance, and usually have significant work done when they arrive at the conference.

WHAT AND HOW SHOULD WE TEACH?

The SIGCSE community has been a great resource for ideas on what to teach and how to teach computer science, with a large focus on introductory programming and more recently on computing in K-12. There have been many fiery papers and panels over the years at the SIGCSE Symposium with views on which programming language to teach, and how best to teach specific languages (e.g., introducing objects early or late in the curriculum). One of my first experiences in introductory programming in K-12 was with the AP CS program, of which many in the SIGCSE community have been involved. My participation began in 1995 when I joined the AP CS Development Committee, the committee that writes the exam. The AP CS exam was quite controversial then as it switched from Pascal to C++ in 1998 and then to Java in 2004. Since exams are developed at least three years out, I was involved with exams in all three programming languages.

Figure 4: Erich and Markus at the first SIGCSE Kid's Camp in 2008.

PHOTO: ©COLE RODGER PHOTOGRAPHICS

AP CS has always had a large presence at the SIGCSE Symposium and the AP CS committees have always gotten input from the SIGCSE community. Back before I attended SIGCSE in 1990 there was a panel on explaining how to grade faster and more consistently using the experience of the detailed rubrics the AP CS program has for grading thousands of exams in a week [2]. With the transition of the exams from Pascal to C++ and then to Java just 5 years later, many panels were held at the SIGCSE Symposium during this time developing direction and collecting feedback from the community [1,9,10]. The SIGCSE community came together in 2009 [3] when CS enrollment numbers were low, and the AP CS AB exam was discontinued by the College Board. Many members from the SIGCSE community contributed to developing a new AP CS Principles exam that was given for the first time in 2017.

EMERGENCE OF THE TEACHING PROFESSOR POSITION IN RESEARCH UNIVERSITIES

Although teaching-focused positions have been around a long time at small colleges, they have only emerged as respected positions at research institutions in the last twenty-five years. Many research institutions were slow to add teaching faculty positions, but the number of positions has been increasing in the last six years as the number of college students interested in a computer science degree has increased and the need for teachers has expanded. My title at Duke is Professor of the Practice (PoP), a position created in the early 1990s to have a path to focus on education in the discipline. Around twenty percent of the faculty at Duke are in a PoP position. I learned about the PoP position through my networking with colleagues at that SIGCSE symposium back in 1994! As these types of positions began to appear at different research institutions, I participated in a panel session at SIGCSE 2004 on Teaching Faculty positions [7] with others who were in similar positions—Tom Horton at University of Virginia, Dan Garcia at UC Berkeley, and JD Dougherty at Haverford College (for the view at a small college). These positions with multiple levels of promotion are now at many Research Institutions with a variety of titles such as Teaching Professor and Clinical Professor. With the continued growth in CS majors, many research institutions are hiring in teaching-track positions.

PHOTO: ©COLE RODGER PHOTOGRAPHICS

Figure 5: SIGCSE 2008 Reception - Michael Kölling, Michael Caspersen, Carl Alphonce, Joe Bergin, and Joe Hummel.

Figure 6: SIGCSE 2008 Conference Co-Chairs: Susan Rodger and JD Dougherty.

VOLUNTEERING, A DIFFERENT VIEW FROM THE OTHER SIDE

In the past ten years I have had a different view of the SIGCSE community, from the volunteer side. For years I have benefitted from attending SIGCSE conferences. There's the networking and collaboration with colleagues, but there are also the ideas I've gotten from SIGCSE conferences to make my lectures more interesting and more engaging. In 2004 I started giving back by helping with conferences.

In 2008 I ran the SIGCSE Symposium with JD Dougherty in Portland, Oregon. Both of us had kids and understood the difficulties of balancing work and family life. I had struggled to attend SIGCSE with my young kids. At SIGCSE 1997, I brought my 5-week-old son, so I wouldn't miss the conference. I managed to give a talk while he slept in the back of the room under the watchful eye of a colleague. But then the audience clapped, he woke up screaming and I didn't get to attend many sessions! At SIGCSE 2000, I was a bit smarter and talked my sisters into coming to SIGCSE with me to watch my four-month old second son. I was able to enjoy my son and the conference that year! When JD and I co-chaired the SIGCSE 2008 Symposium, we created the first SIGCSE Kids Camp to make it easier for parents with young kids to attend SIGCSE by bringing their kids with them. The cost of the camp was kept low to help parents. We also had computer science activities such as Scratch programming. Both JD and I took advantage of the camp by bringing our families to the conference. My step-mother even volunteered as a helper in the camp while my two sons attended and built Scratch programs. We are glad to see that SIGCSE has continued the kid's camp at every SIGCSE Symposium since then.

Figure 7: SIGCSE 2008 Program Co-Chairs: Sue Fitzgerald and Mark Guzdial.

Organizing the SIGCSE 2008 conference was rewarding and a lot of work. I was quite busy running the conference, so I can't tell you much about what happened at the conference that year. Hopefully everything looked smooth to the attendees. I was able to attend all three keynotes: Marissa Mayer, Randy Pausch (given by Dennis Cosgrove and Wanda Dann), and Ed Lazowska. I tried to attend one panel session but was immediately pulled out of it to handle a situation that had arisen. Now when I attend a conference I appreciate all the hard work that goes into running such an event. We documented SIGCSE 2008 by hiring a photographer to take over 1000 pictures of all four days of the conference and have included a few photos from SIGCSE 2008 in this article.

FRIENDS, COLLABORATIONS, AND IDEAS

From my first close encounter with SIGCSE in 1989 to my recent experience with the hot beaches of Cyprus at ITiCSE 2018, my journey through SIGCSE conferences and SIGCSE volunteering over the years has led to great friends, intriguing collaborations, and the transfer of innovative ideas from colleagues into my courses. Since 2010 I have been on the SIGCSE Board. Between the Board and the conference committees, SIGCSE has amazing volunteers, many of whom have dedicated years of their time to the organization to make it better. It has been a pleasure to work with these volunteers. Consider getting involved as a SIGCSE volunteer so you can help make an impact for the next 50 years. ❖

Acknowledgements

I'm grateful to the many mentors who have encouraged me in CSED and to the many volunteers I have worked with over the years in the SIGCSE community. Many thanks to Cole Rodger Photographics.

References

1. Astrachan, O., Clancy, M., Nevison, C., Owens, B. and Trees, F. Advanced Placement and C++, Opening a Dialogue, Twenty-seventh *SIGCSE Technical Symposium on Computer Science Education*, 28, 1 (1996), 393–394.
2. Astrachan, O., Levine, D., Reges, S. and Walker, H. Faster, fairer, and more consistent grading, using techniques from the Advanced Placement reading, *ACM SIGCSE Bulletin*, 22, 1 (1990), 266.
3. Astrachan, O., Walker, H., Stephenson, C., Diaz, L. and Cuny, J. Advanced Placement Computer Science: The Future of Tracking the First Year of Instruction, *Fortieth SIGCSE Technical Symposium on Computer Science Education*, 41, 1 (2009), 397–398.
4. Bergin, J., Brodie, K., Goldweber, M., Jimenez-Peris, R., Khuri, S., Martinez, M., McNally, M., Naps, T., Rodger, S. and Wilson, J. An Overview of Visualization: Its Use and Design, Report of Visualization Working Group, *ACM SIGCSE/SIGCUE Conference on Integrating Technology in Computer Science Education*, Barcelona, Spain, (1996), 192–200.
5. Blythe, S., James, M. and Rodger, S. LLparse and LRparse: visual and interactive tools for parsing, *Twenty-fifth SIGCSE Technical Symposium on Computer Science Education*, 26, 1 (1994), 208–212.
6. Dean, N. and Shannon, G., Editors, Computational Support for Discrete Mathematics, DIMACS Workshop March 12–14, 1992, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 15, (American Mathematical Society, 1994).
7. Dougherty, J., Garcia, D. Horton, T. and Rodger, S. Panel on Teaching Faculty Positions. *Thirty-fifth SIGCSE Technical Symposium on Computer Science Education*, 36, 1 (2004), 231–232.
8. Grissom, S., Naps, T., Ross, R., Hunkins, D., Rodger, S. and Schweitzer, D. Using visual demonstrations to teach computer science. *Twenty-sixth SIGCSE Technical Symposium on Computer Science Education*, 27, 1 (1995), 370–371.
9. Nevison, C., Corica, T., Kmoch, J., Fix, S., Noonan, R, and Kay, D. Changes in the Advanced Placement Computer Science Course Case Studies and C++, *Twenty-sixth SIGCSE Technical Symposium on Computer Science Education*, 27, 1 (1995), 374–375.
10. Stehlik, M., Fix, S., Rodger, S., Nevison, C., and Weiss, M. Advanced placement transition to C++, *Twenty-ninth SIGCSE Technical Symposium on Computer Science Education*, 30, 1 (1998), 372.
11. Trahan, R. and Rodger, S. Simulation and Visualization Tools for Teaching Parallel Merge Sort, *Twenty-fourth SIGCSE Technical Symposium on Computer Science Education*, 25, 1 (1993), 237–241.

Susan H. Rodger
Computer Science Department
Duke University
Durham, NC USA
rodger@cs.duke.edu

PHOTO: ©COLE RODGER PHOTOGRAPHICS

DOI: 10.1145/3230695

SIGCSE: Now and Moving Forward

Amber Settle, *DePaul University* and Renée A. McCauley, *College of Charleston*

The Association for Computing Machinery (ACM) Special Interest Group for Computer Science Education (SIGCSE) was founded in 1968, and in its first 50 years has grown to be a global organization with broad impact across all levels of computer science education. SIGCSE has three influential conferences, a strong volunteer base, and activities and awards that involve participants from over 70 countries. To continue building on its strengths, we argue that SIGCSE needs to continue to engage its diverse constituencies, improve its connections with other ACM Special Interest Groups, manage growth effectively, and continue with careful governance. In doing these things SIGCSE will be well positioned for its next half century.

INTRODUCTION

General purpose computers have been in use since the 1940's, general-purpose programming languages emerged in the 1950's, and the first computer science degrees were awarded in the 1960's [2,3,5]. It is not surprising that the ACM SIGCSE was founded in the 1960's, and it celebrates its 50th anniversary in 2018. In this article, we discuss the current state of the SIGCSE organization and the directions we predict for its future.

NOW: AN ORGANIZATION WITH BROAD IMPACT

Since its founding in 1968, SIGCSE has grown tremendously in mission, membership, and impact.

Currently, SIGCSE has the third largest membership of 36 Special Interest Groups (SIG) of the ACM. SIGCSE is an organization with impact across the globe. Members hail from over 70 countries, and Figures 1 and 2 show the current percentages and counts of SIGCSE members by geographic and cultural region. While many SIGs are looking to expand outside of North America, SIGCSE has been an organization with a global footprint for decades. Its global impact is further exemplified by the influence of its international conferences, that is discussed in the next section.

SIGCSE is organized by an elected board of eight voting members. In recognition of the importance of participation and representation of members from across the globe, the slate of candidates for the SIGCSE Board has in recent years been carefully chosen to enable representation reflective of our global membership.

The current SIGCSE Board consists of six members from the United States, one member from Canada, and one member from Australia [6].

There is such strong interest in SIGCSE activities and events in certain regions of the world that four regional SIGCSE chapters are currently active, one in each of Australasia, China, India, and the United States [7]. SIGCSE chapters allow educators to foster local connections and share ideas and experiences with comput-

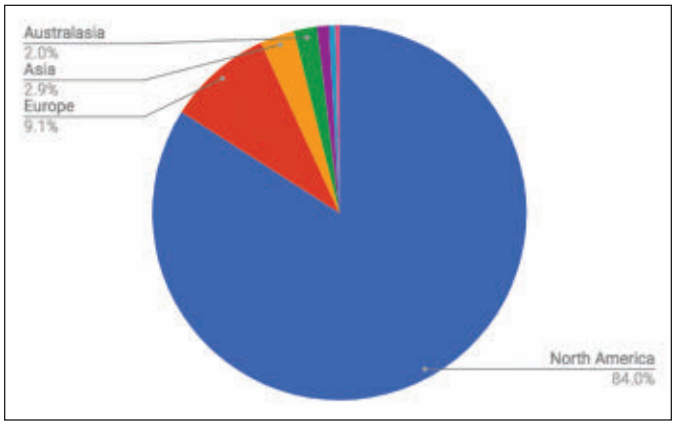


Figure 1: Membership (percentages) by geographic and cultural region

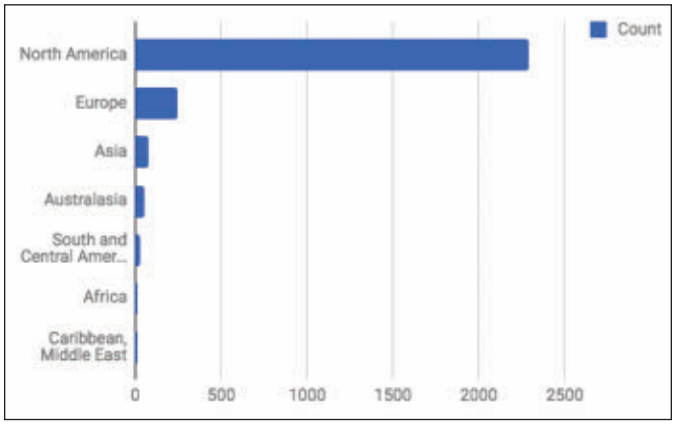


Figure 2: Membership counts by geographic and cultural region

ing educators from their local region. Notably two of the local chapters, Australasian ACM SIGCSE and China ACM SIGCSE, host their own regional computing education conferences.

A major benefit that SIGCSE provides for the computing education communities around the world is its sponsorship of three premier conferences. The oldest and largest conference is the SIGCSE Technical Symposium (typically called "SIGCSE" but referred to here as the Symposium to distinguish it from the organization), which is in its 49th year! This four-day event, usually held in North America in February or March each year, includes workshops, plenary sessions, papers, panels, and exhibits, and is attended by more than 1200 educators from all levels of the education spectrum, and from all over the world. In its 24th year is the conference on Innovation and Technology in Computer Science Education (ITiCSE). Twenty-three of the twenty-four ITiCSE conferences have been held in or around Europe, usually in June or July, and SIGCSE recently committed to continuing to host ITiCSE in this region of the world. The newest SIGCSE offering is the research-focused International Computing Education Research (ICER) conference, which is in its 14th year. Unlike its other conferences, the ICER location rotates between North America, Europe, and Australasia and is usually held in August or September each year. All SIGCSE conferences are vibrant, exhibiting their largest attendance numbers in 2017. Figure 3 shows the attendance numbers for each conference in each of the last three years, and for comparison purposes, the average attendance over the previous decade.

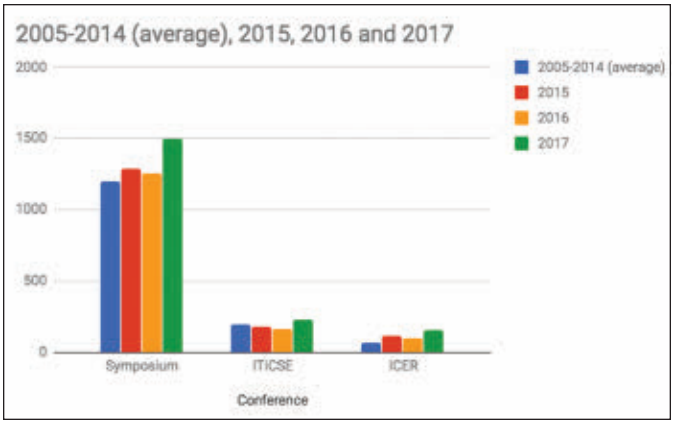


Figure 3: Conference attendance since 2005

In addition to conferences, SIGCSE serves the computing education community through an annual doctoral consortium, a workshop for department chairs, a workshop for new faculty, special projects and travel grant programs, an email list server, a quarterly newsletter, a print magazine, and much more. The print magazine (the magazine in which this article appears), *ACM Inroads*, is a particularly significant benefit provided by SIGCSE. As an ACM magazine it focuses broadly on computing education, with articles, columns, and opinion pieces by computer science, information systems, and information technology educators. SIGCSE fully funds the magazine and provides access to it for all its members.

Every level of computing education from primary through tertiary is represented among the papers, posters, panels, and other contributions that appear in SIGCSE conferences.

One of SIGCSE's strengths is the willingness of its members to voluntarily serve the computing education community by serving the SIGCSE organization. The clear majority of SIGCSE work is done through the efforts of volunteers, including organization leadership, conference organization, reviewing for grant programs, listserv management, and committee organization/membership. While smaller SIGs might struggle to find volunteers, SIGCSE thrives due to the hard work of hundreds of dedicated members from around the world.

SIGCSE serves diverse constituencies. Every level of computing education from primary through tertiary is represented among the papers, posters, panels, and other contributions that appear in SIGCSE conferences. SIGCSE has long served the needs of the computing education practitioner community. As the computer science research community has grown, so has their participation at SIGCSE conferences and at the annual SIGCSE Doctoral Consortium, offered since 1998. Currently, SIGCSE supports 20 doctoral students each year at an event that allows them to present and get feedback on their research.

MOVING FORWARD

While SIGCSE has flourished in its first half century, there are several areas in which the organization can build upon its strengths and further evolve. To be as successful in its next fifty years as it has been in its first fifty years, SIGCSE needs to balance the needs of its diverse constituencies, collaborate with other ACM SIGs, manage growth, and balance the need for executive oversight with the flexibility possible through bottom-up change.

DIVERSE CONSTITUENCIES

The SIGCSE community is composed of both educational practitioners and educational researchers, and of course many of its members are both. SIGCSE recognizes the value of and wishes to serve the needs of both communities. While much of the early work that appeared in SIGCSE conferences and newsletters focused primarily on the needs computer science education practitioners, SIGCSE has evolved as the computing research

community has grown—the Doctoral Consortium and ICER conferences were both developed to meet the needs of the research community. To further recognize the impact of SIGCSE and SIGCSE members on computing education research, it is important to note that SIGCSE members currently serve as editors and on editorial boards of the two major computing education journals, *ACM Transactions on Computing Education and the Computer Science Education* journal [1,9]. While growth in computer science education research is crucial to the health of the community, remaining relevant for computing education practitioners will remain an important focus of SIGCSE. Indeed, many SIGCSE members have primary research interests in discipline-specific areas not in education, and SIGCSE wants to continue to serve their needs as practitioners. Symposium and ITiCSE organizers work to meet the needs of both groups, for example, by creating special tracks for research-focused and practitioner-focused submissions. Balancing benefits and events to meet the needs of both researchers and practitioners is important for the health of the SIGCSE organization.

How to connect SIGCSE with members of other SIGs remains an open question.

In meeting the needs of practitioners alone, SIGCSE faces challenges in serving its diverse practitioner communities. The needs of members vary according to educational level of focus (primary through tertiary) as well as geographic region. The interests of university faculty in Africa are significantly different from the interests of elementary school teachers in Europe, and yet as an organization SIGCSE seeks to serve all groups. Large conferences like the Symposium provide multiple parallel tracks, which address the issues only to some extent. Unfortunately, Symposium attendance is overwhelmingly North American, and educators from other geographic regions often cannot fully benefit [10]. Thus, it is important that SIGCSE offer conferences and events that are accessible and relevant to members that are diverse in both interests and geographic locations.

Interest in SIGCSE globally has reached the point where local chapters and the conferences they support are insufficient to meet demand. Further, with members from more than 70 countries, the expansion of SIGCSE conferences into other parts of the world is particularly important to the future of the organization. One of the main reasons for this is that the conferences often serve as a recruiting tool for SIGCSE membership, and the presence of SIGCSE members across the globe is a critical component of the organization's success. With all of this in mind the SIGCSE Board recently voted to pilot a fourth conference. The conference will be modeled after the ITiCSE conference and hosted in countries outside of Europe and North America, with a projected start date in 2019 or 2020. The conference will initially be held every other year, and India and China, with particularly strong computer science education communities, are potential locations for the first two conferences.

In creating events that are relevant for diverse audiences, conference organizers themselves should come from diverse locations and populations. Here SIGCSE also faces challenges because the organization depends heavily on member-volunteers and that membership is composed primarily of North Americans (82% in 2017), the majority of whom are university-level educators. As one might expect the SIGCSE volunteer base skews according to its membership. Finding ways to nurture volunteers outside of North America and outside of university-level institutions is important moving forward.

OUTREACH TO OTHER SIGS

As mentioned previously, currently there are 36 SIGs, classified into eleven categories (with some falling into more than one category)—applications, artificial intelligence, digital content, education, hardware design, interaction, networking, ops and management, performance, software, and theory. Arguably, the focus of most SIGs is a topic area—such as artificial intelligence, programming languages, computers and society, graphics, computer architecture, to name only a few—that is the focus of coursework at some level of education. Thus, most members of many of the SIGs could benefit from the services of SIGCSE.

A challenge of the SIGCSE organization is to make connections with the members of other SIGs and offer services that would benefit them. For example, while some research conferences already have strong educational communities, others might be enhanced by adding an education-track. In another example, graduate students attending research conferences might benefit from attending the SIGCSE new faculty workshop. Also, SIGCSE members benefit when discipline-based researchers contribute to education, by sharing courses, offering workshops, and participating on curriculum-writing committees.

A challenge for SIGCSE is to figure out how to recruit these potential constituents/collaborators. In the past two years the SIGCSE Board and Symposium organizers have worked with SIGCSE members active in the SIGGRAPH conference to cross-pollinate presentations between the two conferences. In an expansion of that collaboration six SIGs (specifically SIGITE, SIGCAS, SIGGRAPH, SIGHPC, SIGCHI, and SIGPLAN) were invited to present at the 2018 Symposium. But these collaborations are too new to evaluate, and thus far there are no other cross-SIG activities that involve SIGCSE. How to connect SIGCSE with members of other SIGs remains an open question.

GROWTH

While SIGCSE has experienced steady growth through its five decades of existence, that growth is poised to accelerate due to several phenomena. First, growth in undergraduate education, particularly in the U.S. but also globally, has spiked over the last decade. Between 2009 and 2015 bachelor degree production in computer and information science at not-for-profit institutions in the United States grew 74 percent, in contrast with the 16

percent increase seen in overall bachelor degree production during the same period [4]. Some institutions have seen even more dramatic growth, for example, a 300% increase in computer science majors at PhD-granting institutions in the United States and Canada [4].

Growth in opportunities for school-age children to learn about computer science has occurred in recent years through the efforts of many international organizations. Computer science is mandated for children in several countries around the world, including the United Kingdom and Finland. The importance of computer science education for school-age children has also been recognized by many countries, several of which do not yet require it of all students, including the United States, Australia, and New Zealand, among many others.

The growth of educational opportunities throughout all levels of education has resulted in increased demand for SIGCSE conferences, both in terms of submissions and attendance. Two of the three SIGCSE conferences have reached record attendance and submission levels in the past two years. The record submissions have generally resulted in more competitive acceptance rates, which is productive for those wishing to use such publications for promotion and evaluation purposes. However, it restricts the number of people able to publish in such venues. Finding new outlets for scholarship in computer science education is therefore crucial. As mentioned previously, it is a positive development that SIGCSE members make up the editorial staff at the two leading computer science education journals, *ACM Transactions on Computing Education* and *the Computer Science Education* journal [1,9]. Yet, conferences remain crucial outlets for computer science education publications, and as mentioned above the SIGCSE Board hopes that the development of a fourth conference will improve the situation.

GOVERNANCE

As mentioned previously, SIGCSE is a volunteer-organization and one of its strengths is the number and diversity of its volunteer base. Volunteers run conferences, review papers and grant applications, and facilitate access to the information shared by SIGCSE members across the globe. As a result, these volunteers have knowledge of what is working well within the organization's services and what could be improved.

Changes suggested by volunteers are typically well integrated into the structure of SIGCSE events and programs. But they can also impact a large organization in unforeseen ways, which is why an elected SIGCSE Board remains important. The volunteers serving on the SIGCSE Board have typically been serving the organization for many years, often as conference organizers or in other leadership roles in the SIG. As a part of their service on the Board, they are privy to information necessary to understand the broader impact of each conference, event, and program offered by the SIG, as well as how the SIG fits with relation to other SIGs and ACM. For the Board to be effective, board members need to understand the entirety of the SIGCSE membership, including geographic and institutional differences. A representational Board, with the entire Board up for election every three years, is therefore crucial to the health of the organization.

Effective communication among the Board, volunteers, and other members is an important way to leverage the strengths that each group brings to SIGCSE. Currently the SIGCSE Board holds a business meeting at the Symposium, minutes of each Board meeting are published on the SIGCSE web site, and the SIGCSE chair writes a quarterly article that appears in *ACM Inroads*. Volunteer leaders also meet with the SIGCSE Board during its meeting at the Symposium. But each of these communication methods is limited in significant ways. Finding new ways to engage each group with each other is crucial to ensuring that SIGCSE can adapt to the changes it faces in the next half century.

WHAT CAN YOU DO?

As mentioned previously, what SIGCSE offers is accomplished primarily through its volunteers. To grow its services and meet the needs of its diverse membership, SIGCSE must continue to grow its volunteer-base not only in terms of numbers, but also in geographic locale, education community, interests, etc. Be a part of the next 50 years of growth. Volunteer! Get involved! ❖

References

1. *Computer Science Education* Editorial Board; http://www.tandfonline.com/action/journalInformation?show=editorialBoard&journalCode=ncse20. Accessed 2018 June 4.
2. History of Computing; https://en.wikipedia.org/wiki/History_of_computing. Accessed 2018 June 4.
3. History of Programming Languages; https://en.wikipedia.org/wiki/History_of_programming_languages. Accessed 2018 June 4.
4. National Academies of Sciences, Engineering, and Medicine. *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*. (The National Academies Press, Washington, DC, 2017); https://doi.org/10.17226/24926.
5. Purdue University, Department of Computer Science. History of the Department; https://www.cs.purdue.edu/history/. Accessed 2018 June 4.
6. SIGCSE Board; http://sigcse.org/sigcse/about/board. Accessed 2018 June 4.
7. SIGCSE local chapters; http://sigcse.org/sigcse/programs/local-chapters. Accessed 2018 June 4.
8. SIGCSE web site; http://sigcse.org/sigcse/. Accessed 2018 June 4.
9. *Transactions on Computing Education* Editorial Board; https://toce.acm.org/editorial.cfm. Accessed 2018 June 4.
10. Walker, E., Settle, A., and Zilora, S. News from the SIGs, *ACM Inroads*, 8, 2 (2017), 6–8.

Amber Settle
School of Computing
DePaul University
Chicago, IL USA
asettle@cdm.depaul.edu

Renée A. McCauley
Department of Computer Science
College of Charleston
Charleston, SC USA
mccauleyr@cofc.edu

DOI: 10.1145/3230687

The SIGCSE Symposium: A Brief History

Robert E. Beck, *Villanova University* and Henry M. Walker, *Grinnell College*

Historically, within the framework of the Association for Computer Machinery (ACM), events were labeled according to their anticipated size. For example, large events were "conferences," smaller events (a few hundred attendees) were "symposia," and quite small events were "workshops." Decades ago, events sponsored by SIGCSE drew only a few hundred attendees (143 at SIGCSE 1970), and this participation level led SIGCSE's annual event to be called a "Technical Symposium." Although attendance now regularly exceeds 1200 (and was over 1500 in the past two years), the title, "Technical Symposium," remains and its formal name, the SIGCSE Technical Symposium on Computer Science Education, is shortened to SIGCSE <year>.

This document reviews the history of the SIGCSE Technical Symposia, from their first beginnings in 1970 through current times. To organize the review, we divide the history of the SIGCSE symposia into four main phases:

- the early years (Section 1),
- co-location with the ACM Computer Science Conferences (and ACM Computing Week) (Section 2),
- the transition from SIGCSE as a co-located conference to a standalone event (Section 3), and
- growth to a regular attendance of 1200 [and more] (Section 4).

After this historical overview, the article focuses upon several components of the symposia, as they have evolved over the years:

- symposium scope and statistics (Section 5) and
- conference sessions, exhibits, and other events (Section 6).

We conclude by looking ahead to future symposia (Section 7) and offer a list of references (Section 8).

1. THE EARLY YEARS

The first SIGCSE Technical Symposium was held 48 years ago on November 16, 1970, at the Astrodome in Houston, Texas. The 1960's brought a cultural shift in the US, and it seems only right that Unix time and SIGCSE Symposia both began in 1970. Many universities were starting computer science degree programs, and, as noted by Nell Dale as part of the CS oral history project [6], SIGCSE provided a community for computer scientists interested in effectively teaching post-secondary computer science. SIGCSE 1970 co-chairs were Peter Calingaert, at the time teaching for the University of North Carolina at Chapel Hill, and Edward A. Feustel from Rice University. It is a testimony to their dedication to service to computer science that neither can remember too many details; however, Ed remembers that he and Bob Jump handled the local arrangements. Peter was responsible for the technical content, and Robert M. Aiken from the University of Tennessee served as Editor of the Proceedings. More than 40 papers were submitted, with 18 accepted. According to Bob Aiken there were 143 attendees at the first Technical Symposium.

The program of the first Technical Symposium shows that the areas of concern then remain areas of concern now. Our understanding of computing has evolved as have the platforms on which we perform the computing. But the general questions, paraphrased from the first Symposium program, have not changed.

- What is computing? Do we take a "big tent" view (e.g., do we consider computing as a broad subject that includes numerous elements from partner disciplines)?
- What should be taught in our computing courses? Where does assembly language fit? How about machine learning? Big data?
- How do we design and execute the first course in computing? Fortran? COBOL? Basic?

2. CO-LOCATION WITH ACM COMPUTER SCIENCE CONFERENCE

The ACM Computer Science Conference (CSC) over its history from 1973 through 1996 was closely associated with the SIGCSE Technical Symposium. The focus of CSC was computing technology interchange [1]. A supporting goal was communication between researchers and educators across the spectrum of areas that made up computer science in the 1970's to the 1990's. This two-way communication would lead to continually updated curricula and educational practices.

The first ACM Computer Science Conference, labeled optimistically as the "First Annual," was held in February 1973 in Columbus, Ohio. It overlapped on Thursday with the third SIGCSE Technical

Symposium, a tradition that continued throughout the existence of CSC. Those who registered for the SIGCSE Symposium were welcomed to attend the plenary sessions of CSC held on Thursdays. This session in 1973 featured the invited address given by Allen Newell from Carnegie-Mellon University, titled "MERLIN and the Problem of Understanding." The Friday sessions for SIGCSE 1973 were held jointly with the American Society for Engineering Education Committee on Education (ASEE/CoED).

This close association with the Computer Science Conference was helpful to SIGCSE members in many ways. It allowed those interested in teaching computer science an opportunity to participate in a research conference and an education conference in the same venue, so an attendee could pay for travel to one destination and partake in a wide range of activities and events. The power of CSC to draw exhibitors produced a rich connection with industry and with publishers. However, CSC planning dictated many things for the Technical Symposia, including the site, the dates, the structure and format of the exhibits, and the need for careful space coordination on the overlap day.

The last year of Computing Week was 1996, held in Philadelphia, PA, and included [4]:

- the ACM Computer Science Conference (CSC '96),
- the Computers and the Quality of Life Symposium (CAL '96), sponsored by SIGCAS (Computers and Society),
- the Symposium for Applied Computing (SAC),
- the 27th SIGCSE Technical Symposium on Computer Science Education,
- three days of hardware, software, and textbook exhibits,
- the ACM Student Programming Contest,
- a Student Poster Session, and
- the first ACM Chess Challenge, with World Chess Champion, Garry Kasparov, and IBM's Deep Blue.

As this listing suggests, Computing Week evolved from two overlapping conferences in 1973 to a much larger program that still included the Technical Symposium as a major component. Following tradition, SIGCSE 1996 started with a reception on Wednesday evening, February 14. Thursday and Friday included a full program with 78 papers, 16 panels, 9 seminars, 20 posters, and six birds-of-a-feather (BOF) sessions.

In addition to the regular Thursday-Friday program, attendees could expand their background by registering for workshops at the end of the regular sessions. Thus, four workshops were held on Friday evening, eight on Saturday morning, seven on Saturday afternoon, and three on Sunday morning. (In a few cases, one topic spanned two sessions, and, overall, the workshops covered 16 distinct topics.) Altogether, workshops regularly enrolled 200+ people and served at least three purposes.

1. Attendees gained background, experience, and professional development at only a modest cost.
2. The SIGCSE Symposium gained revenue, allowing the registration fee for the main conference to be relatively low (as befitting an event for academics).
3. Some airlines gave substantial discounts to travelers who stayed over Saturday night, so attendees often could obtain very attractive fares by attending a Saturday workshop and returning home on Sunday.

The program of the first Technical Symposium shows that the areas of concern then remain areas of concern now.

In retrospect, two elements of ACM Computing Week 1996 also had a substantial impact on later events, both for ACM and for the SIGCSE Symposia. Although submissions to SIGCSE Symposia increased steadily over the years, submissions to the ACM Computer Science Conference showed steady decline. As an example, SIGCSE 1996 could accept only 78 papers of 205 submitted (acceptance rate of 38%) given the time and space available for parallel sessions. However, to fill its program, CSC 1996 accepted all 51 of its submissions (acceptance rate of 100%).

Philadelphia was chosen for the site of Computing Week 1996 and the second week of February was chosen as the date. These choices were important for historical reasons—the ENIAC computer was dedicated on February 15, 1946, at the Moore School of Electrical Engineering at the University of Pennsylvania in Philadelphia. Also, ACM was founded in September 1947. In celebration of both landmarks, ACM planned "a yearlong celebration of the 50th anniversary of modern computing that will culminate with the celebration of the 50th anniversary of the ACM at Computing Week '97 in San Jose in March 1997." [4] Several of these events (e.g., a "History Retrospective," "Electronic Global Village," and "Electronic Education Event") were held during ACM Computing Week 1996.

3. ON OUR OWN

Over the course of two years, 1997 and 1998, the world of the SIGCSE Symposium changed dramatically!

SIGCSE 1997

The year 1997 marked the culmination of ACM's 50th anniversary celebration. As reported by Charles H. House in the *Communications of the ACM*,

> *ACM97: The Next 50 Years of Computing* incorporated a professional conference where 1,800 attendees gathered to listen to key technologists share their visions of the future, with a spirited, interactive exposition spotlighting some of the most outstanding projects on tap for future generations [5, p. 31].

Since Computing Week 1996 had few submissions and declining attendance, and since ACM wanted to celebrate its 50th anniversary, effectively ACM97 became a replacement for a

Computing Week 1997. Also, ACM decided to reorganize its schedule, and ACM97 was held Saturday-Wednesday, March 1–5, 1997. In this restructuring, SIGCSE 1997 had several new elements.

- SIGCSE 1997 was held Thursday–Saturday, February 27-March 1, *before* ACM97, rather than being scheduled afterwards.
- SIGCSE 1997 was run largely as an independent conference, with its headquarters in a hotel, and not at the conference center.
- SIGCSE 1997 had a full program, with up to six parallel sessions; some discussion considered whether sessions should extend to Saturday as well, but this seemed more than could reasonably be handled at this stage of the conference's evolution.
- In line with some previous experience with conferences on the west coast, submissions and attendance were down somewhat. Overall, 75 of 177 papers were accepted (including 18 of 39 from outside the United States).
- Although largely independent, ACM had arranged contracts for SIGCSE 1997, ACM97, and related activities. Also, the same management company supporting ACM97 was also utilized for SIGCSE 1997.
- As separate SIGCSE and ACM conferences evolved during 1997, some issues related to communications and misunderstandings arose. However, when accounts closed, SIGCSE 1997 finished with a small budget surplus, and SIGCSE had gained much new experience.

SIGCSE 1998

Starting in 1998, the SIGCSE Symposia have been completely on their own—not affiliated with separate ACM conferences, although SIGCAS has held a co-located meeting. This independence allowed some expansion of the overall program.

- Regular sessions ran all day Thursday and Friday, and on Saturday morning.
- A Doctoral Consortium was organized for all day on Wednesday.
- Workshops were spread throughout the conference (but not during other symposium sessions).
- SIGCSE 1998 coordinated with the ACM Programming Contest.
- As was the case in SIGCSE 1996, SIGCSE cooperated with faculty groups from two-year schools to encourage attendance and involvement.
- A/V costs increased substantially from the past, so conference registration was raised from $115 to $130, and workshop registration from $40 to $45—still quite low for national conferences!

For the next seven years the SIGCSE Symposium welcomed, as a co-located conference, the IEEE Conference on Software Engineering Education and Training (CSEET). After this initial collaboration CSEET followed a different path for its annual event.

4. OVER 1200, AND RECENTLY OVER 1500 ATTENDEES

At the beginning, the SIGCSE Technical Symposium attracted a modest group of computing professionals who had interests in computing education—both at the undergraduate and graduate levels. Records of attendance during the early years are spotty. The earliest records of SIGCSE attendance show 143 for SIGCSE 1970 in Houston, TX, 179 at SIGCSE 1972 held in St. Louis, MO, 200 for SIGCSE 1973 at Ohio State University, and 300+ for SIGCSE 1974 in Detroit, MI. Interestingly, this attendance level of a few hundred is consistent with some historic ACM guidelines for the size of conferences labeled as a "symposium."

Records of attendance during the combined ACM Computing Week also require interpretation, since attendees could register for one conference or for combinations.

Starting in the mid-1990s, attendance records are consistently available, although different sources (e.g., [7,8]) sometimes report slightly different numbers. (Perhaps an individual could not attend because of a medical condition, or an invited speaker received a complimentary registration.) The graph in Figure 1 represents merged data from both [7] and [8].

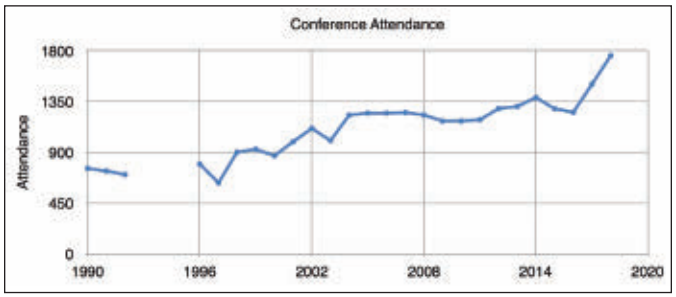


Figure 1: Conference Attendance: 1990-2018

Overall, symposium attendance was generally consistent at 700 to 800 during ACM Computing Week. The number decreased for SIGCSE 1997, when the symposium started on its own, but the next several years showed a steady increase. Finally, in 2004, attendance broke 1200 (reaching 1228), and attendance hovered between 1180 and 1280 through 2015 (except 1304 attended 2013—slightly above range). Most recently SIGCSE 2017 showed a clear and substantial increase to 1501 and SIGCSE 2018 had 1731 registered attendees.

In addition to this overall pattern for attendance at the symposium, three additional factors reflect the level of attendee involvement.

- Symposium attendees often want to make the most of their time; travel can be expensive, and participants often sign up for workshops both before and after the symposium itself. Specifically, over the years, many attendees have shown strong and increasing interest in both pre-symposium workshops (e.g., on Wednesday) and post-symposium workshops (e.g., on Saturday afternoons or Sunday mornings).

- Many groups are using Wednesday as a time for pre-Symposium meetings. These include a gathering of department chairs, meeting of ACM-W, activities involving SIGCAS and the ACM Committee on Professional Ethics, and the BlueJ community, among others.
- In the 1990s, many airlines set fares so that travelers could obtain dramatically cheaper tickets with a stay over a Saturday night. Thus, beyond attendee interest in workshops, attendees also had strong incentives to enroll in one or more workshops to take advantage of cheap, restricted fares. However, in 1996, Southwest Airlines removed this restriction for relatively inexpensive seats, and other airlines followed over the next few years. Pragmatically, this change in airfare structure removed the economic incentive to register for workshops, although interest in taking advantage of professional opportunities still provides strong incentives for workshop attendance.

As the map in Figure 2 shows, the location for the Symposium has been spread throughout the lower 48 states. Future sites are selected under a variety of constraints:

- a large enough city with a convention facility to accommodate 1600 attendees;
- enough hotel rooms close by the meeting site available at a reasonable price;
- ease of access by air, with consideration for international attendees;
- a number of restaurant choices near the meeting site; and
- reasonable weather during the first week of March.

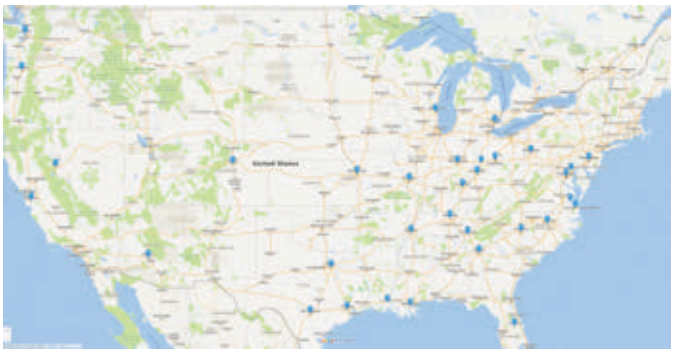


Figure 2: Locations of the SIGCSE Technical Symposia

5. SYMPOSIA SCOPE AND STATISTICS

We now turn to reflecting on the technical content of the Symposia. Over the 48+ years of its existence, the SIGCSE Technical Symposium has continued to evolve and expand, and symposium leadership has had to balance various types of sessions to yield an attractive and varied program. For example, plenary sessions generally have expanded from just one or two keynote speakers to one address each day plus assorted community events (e.g., a reception, a first-timers luncheon, a closing luncheon). Also, with a limited number of time slots available over a conference, conflicting demands arise in the mix of papers, panels, special sessions, etc. For example, the addition of one panel session might mean reduction in the number of sessions for papers.

5.1 SUBMISSIONS, ACCEPTANCES, AND ACCEPTANCE RATES

Papers consistently comprise a substantial component of each SIGCSE Technical Symposium. The first symposium, held on November 16, 1970 in Houston, Texas, included the presentation of 18 papers, selected from over 40 submitted. Over the years, the length of the symposium has expanded several times, with regular programming progressing from one day to two days, to two and a half days. Pre-symposium and post-symposium activities have supplemented the main program substantially and attracted increased attendance and paper submissions.

In what follows, the first graph (Figure 3) presents both the number of papers submitted and the number accepted. Numbers of submissions for some years are not available, and different sources (e.g., [7,8]) sometimes report slightly different numbers, perhaps due to duplicate submissions or papers withdrawn. As indicated, the number of submissions has varied somewhat over the years, and with a reasonably consistent level of about 300 submissions between 2004 and 2015. Submissions grew to 348 for SIGCSE 2017 and to over 500 for SIGCSE 2018, and it will be interesting to observe if these increases will continue in the future.

Turning to papers accepted, the graph in Figure 3 shows a modest increase over the years, largely reflecting longer conferences and increased number of parallel tracks. For any year, a program committee must balance the number of papers, panels, and other sessions, so the number of paper sessions may vary modestly. However, even with adjustments from year to year, the number of accepted papers has remained quite steady, between 100 and 111, for about 18 years (2005-2017).

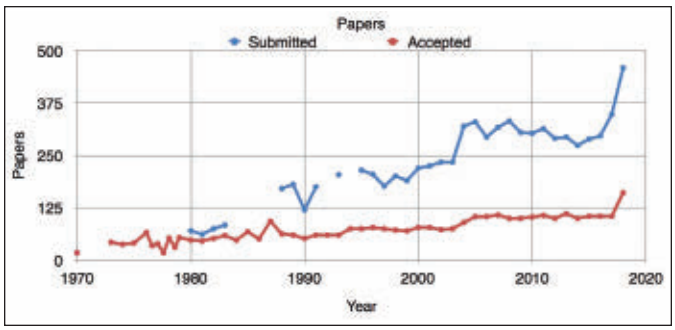


Figure 3: Papers Submitted and Accepted

With both the number of papers submitted and the number accepted being reasonably steady for several years, the acceptance rate for papers has remained reasonably steady (mostly between 30% and 35% since 1995 or so, as shown in Figure 4.

5.2 THE PAPER REVIEWING PROCESS

For at least several decades, papers submitted to a SIGCSE sym-

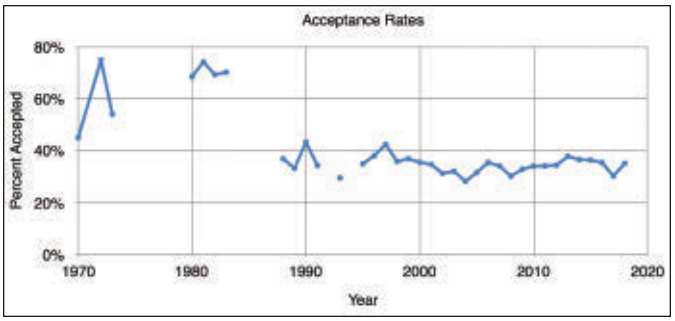


Figure 4: Paper Acceptance Rates

posium were evaluated and selected through a peer-review process, and, as previously reported, only about one-third of submissions have been accepted for almost three decades. Details of the review and selection process, however, have evolved substantially through this period, following at least two challenging principles.

- Program committees have encouraged all interested SIGCSE members to collaborate in the reviewing process. In contrast to many conferences, in which reviewers are screened by conference leaders, SIGCSE has valued the wide range of perspectives of its full membership.
- Program committees, of course, sought to accept the papers most appropriate, based both on overall quality and on the desire to organize a wide-ranging program. For example, if numerous papers discussed CS1/CS2 and few considered operating systems, a balanced program might include the best of both CS1/CS2 and operating systems, even if overall ratings for one area might be higher or lower than those for papers in the other area.

As an example of the challenges of paper selection, SIGCSE 2000 received 219 submissions, and each paper was distributed to at least four of 482 reviewers. With the scope of this process, attention needed to be paid to different perspectives and priorities among the many reviews received—altogether a challenging task for symposium leadership.

As might be expected, details of the submission and reviewing process have evolved substantially over the years, often reflecting practical opportunities and limitations of technology as it evolved. Although program committees refined the submission and review process regularly over the years, logistics, policies, and procedures may be divided into roughly three main phases.

- *Reviewing in the 1990s:* Throughout the 1990s, paper submission and review primarily utilized physical paper.
- An author would print n copies of a paper and mail the copies to the program chair. The submission deadline might be based on when the mail was postmarked, when the submission was received, or some combination.
- The program chair would retain one copy for later processing, assign reviewers, and mail the remaining $n - 1$ copies to reviewers.
- Reviewers would send reviews to the program chair. Originally all correspondence was done through the postal system, although SIGCSE 1999 Program Chair, Robert Noonan, worked to allow some processing of email reviews.
- Ratings and reviewer comments would be compiled (manually for most symposia), papers ranked and selected, and comments returned to authors via the postal system.

Overall, the paper submission and reviewing process was dictated, largely, by paper-based technology, with papers and reviews transmitted by postal mail.

- *SIGCSE 2000–SIGCSE 2008:* In 1999 when planning SIGCSE 2000, Symposium Co-chairs, Nell Dale and Lillian "Boots" Cassel, asked Program Chair, Henry Walker, about the possibility of allowing the online submission and reviewing of papers. In response, over the summer 1999, Walker worked with two students, Weichao Ma and Dorene Mboya, and two technical consultants, Wayne Twitchell and Theresa P. Walker, to develop a system that allowed either online or paper submissions. (At this stage, electronic submission was encouraged, but paper served as an alternative for those accustomed to the traditional approach or those with limited internet access.) The next year, almost all papers were submitted electronically, and with SIGCSE 2002, electronic submission was required. Within a few years, John Dooley joined the collaborative team, and this system was used extensively for over a decade. Pragmatically, the introduction of an online system had several anticipated and important consequences.
  - Logistics of submission, paper handling, reviewer assignment, paper distribution to reviewers, recording of reviews, paper selection, and feedback to reviewers were greatly streamlined.
  - The number of reviewers per paper could be expanded, since paper duplication and mailing were no longer required.
  - After reviews were completed, they could be distributed easily to authors, allowing efficient refining of accepted papers based on feedback.

Program committees have encouraged all interested SIGCSE members to collaborate in the reviewing process. In contrast to many conferences, in which reviewers are screened by conference leaders, SIGCSE has valued the wide range of perspectives of its full membership.

- After reviewing deadlines had passed, reviewers of a paper could view other reviews of the same paper, in effect providing some feedback to reviewers as well as authors.
- Within just a few years, each paper could be sent to six or seven reviewers to obtain a particularly broad range of perspectives.

- *SIGCSE 2009–present:* Although an online system allowed extensive feedback for papers by many reviewers, nervousness was sometimes expressed regarding divergent views and ratings. Inherently within the process, reviewers provided their views about each paper, but reviewers did not have an opportunity to compare differing perspectives, to discuss strengths and weaknesses, and to work toward a consensus. In response, the reviewing process was expanded to yield a multi-tiered system.
  - Initially with SIGCSE 2009—a small group of "Associate Program Chairs" served to oversee sets of papers—reading reviewers' reviews and providing a second stage in the overall reviewing process. By handling several papers, this second tier could add a global perspective to the individual papers and reviews.
  - Shortly thereafter, a group of people were designated as "meta-reviewers," each of whom read both a group of papers and their reviews, and then prepared an informed summary and reflection for each paper. Overall, a "meta-review" could highlight perspectives and insights from several reviewers, while also providing a broad perspective.
  - Within a few years the meta-reviewing process was expanded to include discussion of each paper by all its reviewers, under the direction of an "Associate Program Chair" (APC). With discussion, reviewers might refine their own reviews, adjust their ratings, and/or provide insights. Often, this process reduced variability and provided authors (and the program chairs) with clearer direction and feedback [3].

5.3. FURTHER CONSEQUENCES FROM ONLINE SUBMISSIONS

As already noted, the shift to the online submission and reviewing of papers for SIGCSE 2000 and later resolved many logistical challenges and yielded several anticipated benefits. Three additional elements may or may not have been anticipated.

- *International Authors and Reviewers:* The paper-based system in place before SIGCSE 2000 relied upon postal mail for the transmission of papers, both for submission and for reviewing. Within North American, postal mail took some time, but mail usually was delivered within a few days. However, mail to and from Europe or other international destinations often was much slower. As a result, authors from outside North America often had to mail their submissions significantly before a stated deadline.

  Further, mail from a program chair (in North America) to a reviewer outside North America could take a few weeks, and such mailing delays did not fit within timing constraints for reviewing. Consequently, before SIGCSE 2000, papers could not be reviewed by computing educators outside North America. With the possibility of online submissions for SIGCSE 2000, however, electronic submissions could be assigned to reviewers world-wide—greatly extending the range of people who could participate in the reviewing process.
- *Reviewer Variability Study:* When many reviewers evaluate papers, questions naturally arise regarding variability in ratings from one reviewer to the next. To help address this question, as part of system development for SIGCSE 2000, 10 papers (largely selected at random) were assigned to 100 reviewers each, in addition to the normal process of reviewing all the submissions. In addition to basic questions regarding overall rating variability, the study also considered what impact several variables might have on overall ratings. In summary, the findings are largely reassuring, but also may suggest directions for further exploration.
  - In summary, "the following factors are NOT statistically significant in contributing to overall paper ratings: referee gender, referee's country (U.S. versus non-U.S.), familiarity of referee with subject, or paper format (electronic submission versus hardcopy)." [13, p. 181]
  - Papers with either low or high median scores had relatively low standard deviation scores—reviewers often agreed that papers were quite good or quite weak.
  - Papers with intermediate median scores displayed higher standard deviation scores; reviewers had relatively broader reactions to papers that were not obviously wonderful or obviously weak.

Reference [13] provides a more complete discussion of this variability study.

- *Timing of Author (and Reviewer) Submissions:* Before SIGCSE 2000, all paper submissions had to be mailed. Thus, to be confident that mail would arrive in time for the reviewing process, authors typically completed and mailed their manuscripts several days before a deadline. Even if consideration of a submission was based upon a postmark, the paper still had to arrive within a time interval, so it could be distributed and reviewed. Pushing a deadline too closely might result in a paper not being eligible for consideration.

  Electronic submissions, naturally, changed such timing considerations [10,11]. Papers could be submitted the night of a deadline and still be included within the reviewing process. As a result, many authors would submit papers close to a stated deadline. By SIGCSE 2010 and SIGCSE 2011, the clear majority of papers were submitted within a few days, or even within a few hours, of the stated deadline. For example, the graph in Figure 5 shows the number of submissions received in the days before the deadline for these two conferences. In this graph, the horizontal axis measures the days

before the deadline; 0 represents the deadline itself, -1 designates the day before deadline, etc. The vertical axis shows the total number of submissions received. Overall, this graph shows that the largest number of papers for each of these conferences arrived on the last day! [10,11]

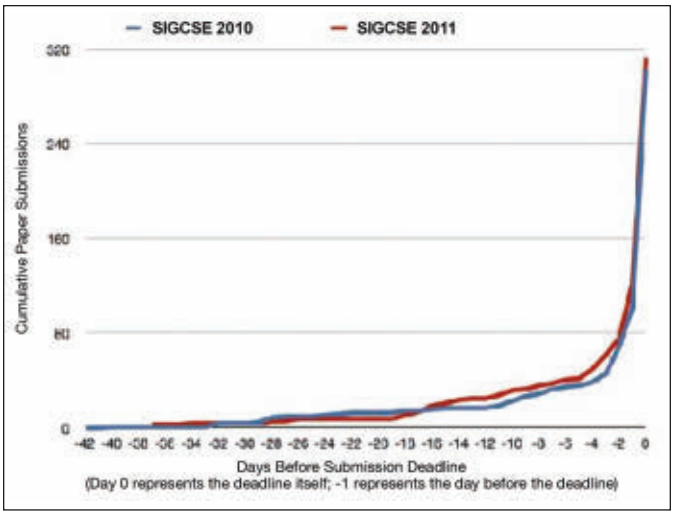


Figure 5: Number of Papers Submitted versus Days Before the Submission Deadline

6. SYMPOSIUM COMPONENTS

Over the years, SIGCSE Technical Symposia have evolved in numerous ways. Each year marks experimentation with some new ideas, refinement of past practices, adjustments due to local circumstances, responses to [unforeseen] challenges, etc. With so many details and program elements in this ever-evolving event, a complete record of every activity and practice would fill volumes, and thus is well beyond the scope of this "Brief History." Instead, this section outlines selected themes and activities that may provide insights and highlight trends that reflect the ongoing development of this annual event.

6.1. SYMPOSIUM FIRSTS

The following are some of the "firsts" for the Symposium, most gleaned from the formal reports that are submitted to the SIGCSE Board and the ACM by the Symposium chair or co-chairs.

- 1983: Severe snow complications, not in Orlando (the host city), but along the East Coast, so previous hotel guests could not leave
- 1984: Birds of Feather sessions
- 1989: Saturday workshops
- 1990: Symposium luncheon, email system available to attendees
- 1991: Faculty poster session
- 1994: Demonstrations, Friday evening workshops
- 1998: Doctoral Consortium

The SIGCSE Board has committed to moving the Symposium location around the country, giving more cities the chance to celebrate the ***first*** time that it has come to town. The fifty symposia have been held in 36 different cities in 22 different states and the District of Columbia. Atlanta and St. Louis are first on the list of most symposia hosted, tied at four. Missouri is by far the most popular state, with seven symposia being held there.

6.2. EXHIBITS

The exhibits have been a key part of the Symposium from early in its history. One can track the rise and fall of the components of the corporate world that support university-level computer science education by studying the list of exhibitors. The 1980's and 1990's show strong reliance on textbooks, with as many as 24 book publishers having booths on the exhibit floor. Gradually the industry has consolidated, and the book exhibitors have decreased in number to about eight. At the same time there has been an increase in the number of groups selling software systems as IDEs, or to grade programs, or to coach and tutor students. Also, the large computing corporations are present as exhibitors to display their systems that support the hot areas of computing—virtual reality, artificial intelligence, machine learning, internet of things.

Over the years, SIGCSE Technical Symposia have evolved in numerous ways. Each year marks experimentation with some new ideas, refinement of past practices, adjustments due to local circumstances, responses to [unforeseen] challenges, etc.

6.3. THE RECEPTION

The SIGCSE community has often been described as an extended family. At SIGCSE symposia, technical sessions are valuable and provide many professional insights, but networking among attendees also is quite important. Often relatively new symposium attendees focus upon formal sessions, as they start to make contacts and get to know some members of the community. Veteran symposium attendees may participate in most/all plenary sessions and some technical sessions, but conversations and collaborations in the hallways and meeting areas often are equally important.

With such an emphasis on community, meeting people, and networking, SIGCSE symposia have included an evening reception dating back to the 1990s or before. For example, when a SIGCSE symposium was one of several events during ACM Computing Week in the early and mid-1990s, the ACM and SIGCSE typically co-sponsored a reception—often near the start of the symposium.

When the SIGCSE symposium became independent in

1997, the tradition of an opening reception continued. Over the years, details of the reception were driven by at least three, sometimes-competing, interests.

- All parties (the SIGCSE organization, the symposium committee, veteran attendees, and first-timers) had a keen interest in fostering community and encouraging personal interactions. A reception, particularly near the beginning of a symposium, brought people together and helped begin the process of social inclusion.
- Symposium leadership want the reception to be a welcoming and comfortable opening event, with inviting and tasteful food and beverage options to meet attendees needs, but also within in the [sometimes-significant] constraints of the overall budget.
- Attendees often participate in the symposium with modest funding and limited travel budgets, and this substantial group has a strong preference for a reception that can replace some or all of an evening meal.

For the most part, symposia through SIGCSE 2000 maintained a balance of food and beverage through lovely receptions of modest scale. In 2001, however, the symposium had to pay a substantial penalty to a hotel, due to an error in the contract signed by ACM, but the hotel also agreed that the required payment could be used to cover an expanded reception. As a result, SIGCSE 2001's reception was substantially more extensive than similar events in the past—and, as might be expected, the SIGCSE 2001 reception was greeted with considerable enthusiasm by many symposium attendees.

Of course, once attendees experience a (somewhat) extravagant reception, expectations were raised. Thus, after 2001, Symposia leadership have faced the challenging assignment of organizing an evening reception that would fulfill expectations of attendees while also allowing the overall symposium to stay within a designated budget. Of course, this planning is just one of the many competing options that must be addressed by the leadership of each SIGCSE symposium.

6.4. NIFTY ASSIGNMENTS

One of the most popular sessions at the Technical Symposium is the Nifty Assignments organized by Nick Parlante from Stanford. The session started in 1999 and has run every year since then, except at SIGCSE 2000. Usually six presenters describe an assignment they have used for a CS1 or CS2 course and comment on its difficulties, its successes, and its appeal to the students. The assignments, almost 150 of them, are archived at nifty.stanford.edu. Some are complex because of their layered structure, but each layer is rather simple. Some involve clever uses of data structures. Others involve intriguing applications and evoke concepts from the computing in context effort.

6.5. BIRDS OF A FEATHER

"Birds of a Feather" or BOF sessions provide opportunities for individuals with common interests to gather and discuss their perspectives and insights within an informal framework, achieving the spontaneity noted in the Electronic Group Interactive Session on SIG activities [2]. Within the SIGCSE symposia, SIGCSE 1984 seems to be the first program mentioning BOFs. At the time SIGCSE symposia represented one component of ACM Computing Week, with SIGCSE events largely scheduled during the day on Thursday and Friday. For SIGCSE 1984, however, a new program entry appears, "8:00 pm -- 10:00 pm BIRDS -OF-A-FEATHER SESSIONS SALON 10", with the additional notation, "Post your proposed topic with a leader's name on the bulletin board or sign up to attend one already posted. Rooms will be assigned to fit the group size." At the symposium, groups would identify themselves through an informal sign up, and ad-hoc discussions could be scheduled over a 2-hour block on Thursday evening [9]. A similar notation appears for SIGCSE 1985.

Neither a time nor sign-up process appears in the SIGCSE 1986 Proceedings, but by SIGCSE 1987 a specific contact was identified for those interested in organizing a BOF, and the program schedule again identifies an evening meeting time. This general arrangement continued through SIGCSE 1993. Interestingly, BOFs apparently were becoming progressively more popular through this period, and SIGCSE 1994 identified BOF sessions on both Thursday and Friday nights—the formal symposium closing session was 5:45-6:15 pm on Friday, but an after-symposium BOF period still was reserved (in the conference hotel) from 7:30-9:00 pm that evening.

SIGCSE 1995 returned to schedule BOF sessions one evening during the symposium, and the published Proceedings for SIGCSE 1996 and 1997 both included descriptions of six BOF sessions. Interest in BOFs continued to expand in the following years. By approximately 2008, the number of requests for BOF sessions exceeded the possible spaces and logistics, as shown in the graph in Figure 6.

Also, by SIGCSE 2008, two "flocks" of BOF sessions were held, back-to-back, to accommodate as many BOF sessions as possible.

6.6. FIRST-TIMERS LUNCH

The SIGCSE organization has had a long-standing commitment to developing a sense of community among its members and seeking to incorporate new members. As early as SIGCSE 1998, the symposium committee included two people (Dick Austing and Cathy Bareiss) who were in charge of first-timers activities. Before long, an important activity included a First-Timers Lunch, when first-time attendees could meet each other and become acquainted with some veteran SIGCSE members (e.g., SIGCSE Board members and others who had extensive experience with the SIGCSE organization).

By SIGCSE 2009, the number of first-timers reached about 340 attendees, and all received invitations to the luncheon—held on the first day of the symposium to connect with first-time attendees early in the symposium. To further connect first-timers with veteran members of the computing-education community,

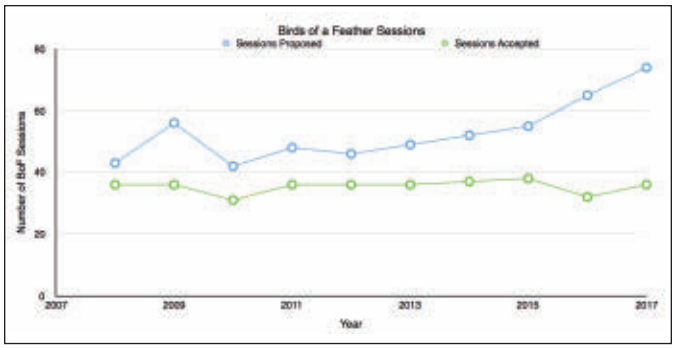


Figure 6: Birds of a Feather Proposals Submitted and Accepted

at least by 2009, the First-Timers Lunch was the venue for an address by the winner of the SIGCSE Award for Lifetime Service to Computer Science Education. With the Award winner presenting an address, any symposium attendee could attend the luncheon, but first-timers attended free of charge.

7. CONCLUSION AND LOOKING AHEAD

The history of the Special Interest Group is different from that of the Symposium. SIGCSE, the Special Interest Group, supports many efforts and collaborations to help the thousands of faculty members at all levels provide their students with the most current ideas in computing delivered with effective pedagogy. Join your colleagues at SIGCSE 2019 in Minneapolis, February 27 through March 2, 2019. Share your recollections of Symposia in the past with *ACM Inroads* to help the Editors expand the story of the Symposium and its influence on the computing education community.

So, while we have come a long way, many challenges remain for us to make our mark including issues around the lack of diversity in our classrooms, support for CS education researchers, and national K-12 CS education and teacher certification.

For additional information, Susan Rodger maintains a wonderful website chronicling the history of SIGCSE at [7], building upon data compiled by Henry Walker at [12]. Barbara Owens and Vicki Almstrum are leaders of the Computing Educators Oral History Project whose work is archived at [6]. ❖

References

1. ACM Computer Science Conference Focus and Goals, undated.
2. ACM, *DataPlan: Report on Electronic Group Interactive Session (EGIS)*, Prepared by Glenn H. Tecker, Consultants (Trenton, NJ, May 3, 1991).
3. Casterson, M. (SIGCSE 2017 Symposium Co-Chair), An Improved Process for the SIGCSE Technical Symposium. *SIGCSE Bulletin Newsletter*, 48, 2 (April 2016), 5.
4. Martin, C.D., Chair's Message: CQL and ACM Conferences. *ACM SIGCAS (Computers and Society)* Newsletter, 25, 4 (December 1995), 1.
5. House, C.H., ACM97: An Event to Remember. *Communications of the ACM*, 31, 5 (May 1997), 31–34.
6. Owens, B.B., and Almstrum, V. Welcome to the Computing Educators Oral History website. CEOHP: *Computing Educators Oral History Project*; http://www.southwestern.edu/departments/mathcompsci/OHProject/. Accessed 2018 March 6.
7. Rodger, S. (continuing from H.M. Walker), *SIGCSE Schedule of Conferences and Symposia*; https://users.cs.duke.edu/-rodger/sigcseconferences.html. Accessed 2018 February 10.
8. SIGCSE, *The SIGCSE Technical Symposium*; https://sigcse.org/sigcse/events/symposia. Accessed 2018 February 10.
9. SIGCSE 1984, *The Papers of the Fifteenth SIGCSE Technical Symposium on Computer Science Education*, February 16–17, 1984, SIGCSE Bulletin, 16, 1(1984).
10. Walker, H.M. and Dooley, J.F. The SIGCSE submission and review software: 10 (hexadecimal) lessons. *Journal of Computing Sciences in Colleges* 25, 5 (2010 May), 196–206.
11. Walker, H.M., The SIGCSE submission and review software: 10 (hexadecimal) lessons. (an update of the Walker/Dooley paper [10]), *Thursday extra*, Grinnell College; http://www.cs.grinnell.edu/-walker/talks/software-lessons-grinnell/. Accessed 2018 February 14.
12. Walker, H.M., SIGCSE Schedule of Conferences and Symposia; http://www.cs.grinnell.edu/-sigcse/conference-index.html. Accessed 2018 March 6.
13. Walker, H. M., Ma, W. and Mboya, D. Variability of Referees' Ratings of Conference Papers. ITiCSE '02 *Proceedings of the 7th annual conference on Innovation and technology in computer science education*, (Aarhus, Denmark, 2002), 178–182.

Robert E. Beck
Department of Computer Science
292 A Mendel Science Center
Villanova University
800 E. Lancaster Avenue
Villanova, PA 19085-1699
robert.beck@villanova.edu

Henry M. Walker
Department of Computer Science
Grinnell College
1116 Eighth Avenue
Grinnell, IA 50112
walker@cs.grinnell.edu

DOI: 10.1145/3276301

Four Reflections on the History of ITiCSE

Lillian (Boots) Cassel, *Villanova University,* Mats Daniels, *Uppsala University ,*
Michael Goldweber, *Xavier University,* and Judy Sheard, *Monash University*

In 1996, recognizing a growing worldwide interest in community education, SIGCSE started a new conference in Europe—now known as the Innovation and Technology in Computer Science Education conference (ITiCSE). Following the success of the first ITiCSE, it has been held annually, typically in Europe, ever since. ITiCSE has developed its own special character, providing unique and valuable experiences for participants. In this article we reflect upon ITiCSE, its establishment and development into a leading venue for computing education researchers and practitioners.

ITICSE – THE EARLY DAYS (REFLECTIONS FROM LILLIAN (BOOTS) CASSEL)

ITiCSE has its roots in the movement of SIGs to hold their conferences in places other than the United States on a regular basis. Some were alternating US and Europe. Some were including a conference site in Asia as part of the rotation. As chair of SIGCSE at the time, I was sympathetic to the idea of making the conference more accessible to our growing number of members who did not live in the United States, but was also conscious of limited travel funds available to many of our SIGCSE symposium attendees. The idea of moving the Symposium out of the country seemed like taking a very serious risk for our flagship event. After much discussion, the SIGCSE Board agreed to try to add a second conference to our list. We had a good number of European members and decided to put the new conference in Europe. We did not choose to move it around radically, because we wanted it to develop a community. Of course, there was no guarantee that the conference would be successful. To play it safe, we enlisted the collaboration of Jim Hightower, then chair of the ACM SIGCUE (Computer Uses in Education). To make the new conference something that would appeal to our combined memberships, we named it *Integrating Technology into Computer Science Education*. SIGCUE no longer exists, but their role in the beginning of ITiCSE was significant.

The first ITiCSE was held in Barcelona, Spain 2–5 June 1996. Bill Fleischman of Villanova had an ongoing working relationship with a colleague in Barcelona, Emilio Luque. Together they undertook the role of organizing committee and Emilio also organized local arrangements at the conference site—University Autonoma of Barcelona. Early member registration cost $295 for ACM members of SIGCSE or SIGCUE. Dick Austing did registration, Gordon Davies did publications, Harriet Taylor organized the working groups, and Jim Hightower oversaw the Demonstrations and Posters. Jim Hightower and I co-chaired the conference, and the following year Mats Daniels and I co-chaired the second ITiCSE in Uppsala, Sweden. Official airlines of the Barcelona conference were TWA and Iberia; both offered discounts to conference travelers.

One of my fondest memories of the first ITiCSE was that first working group poster session. As conference attendees left their sessions and came into the reception area, every single person ignored the wine and tapas and went to see what the working groups were doing!

Harriet Taylor and I wrote the first Call for Participation for ITiCSE. Since our European colleagues were often familiar with the IFIP (International Federation for Information Processing) format, we incorporated some features of those conferences. For example, the half-day break on the middle day of the conference is a regular component of the IFIP meeting schedule. We did expect that some people would travel to ITiCSE who had not previously done a lot of travel and we wanted time to see the conference site. We included the main meal of the day in the conference schedule with a two-hour time slot.

IFIP conferences include working groups. However, they are very different from what we built into ITiCSE. At IFIP conferences, everyone joins a working group as part of the conference attendance. The groups meet in a few short sessions set aside

from the conference time. We wanted our groups to have time to do something more substantial and so designed a different model. That initial model has been changed, for good reasons; however, the basic idea still works. The plan was to have two overlapping, but somewhat distinct events—the conference and the working groups. The working group members began their work by email communication before the conference and arrived a day early to meet in person and continue their work. The groups continued to meet during the conference, with breaks to attend plenary sessions and members stepping out to go to a session of particular interest. On Monday and Wednesday afternoons, the working groups presented status reports in poster form at a wine and tapas reception.

The enthusiastic conference evaluations from Barcelona provided the encouragement to continue and expand on the idea of a conference focused on pragmatic applications of techniques and technologies as we adapted to the presence of the web, social media, constant connectivity, and an increasingly tech-savvy student body.

One of my fondest memories of the first ITiCSE was that first working group poster session. As conference attendees left their sessions and came into the reception area, every single person ignored the wine and tapas and went to see what the working groups were doing! I knew then that we had something of value. People did enjoy the wine and snacks, of course, but the lively conversations at the working group posters continued through the session time.

There were five working groups that first year, with seven to ten members in each. The working group chairs were required to turn in their report before leaving from the conference on Thursday afternoon. A committee of editors then took over to put the reports into a final form suitable for publication. That first year, the committee met in a lovely seaport town and edited the reports on Spanish computers—with Spanish menus and keyboard layouts. An essential language lesson for the editors was that *guardar* means save, and *guardar como* means save as.

The conference program had three parallel tracks, two for papers and one for demonstrations. There were ten long papers, 35 short papers, and 12 demonstrations. Two panel sessions looked at *Challenges of Using Groupware to Teach Groupware* and *Computer Science Education and the World Wide Web*. In addition to the Working Group posters, 13 other posters rounded out the program. Each day began with a keynote speaker—Elliot Soloway of the University of Michigan, *The Nintendo Generation Goes to College*; Scott Teel of Sun Microsystems, *The role of Java in the future of computer science education*; and Diana Laurillard of the Open University UK, *Making multimedia user-active*.

Conference evaluation forms were positive, with some good feedback on the program and logistics to guide the next year.

Mats Daniels of Uppsala University joined Boots Cassel of Villanova as co-chairs of the second ITiCSE. It was again co-sponsored with SIGCUE. Working groups met 1–5 June 1997 and the conference began with a reception on the evening of 1 June and continued through Wednesday, 4 June. Jonas Barklund organized posters and demonstrations, while Vicki Almstrum did the working groups.

The early conferences called for submission of extended abstracts, some of which were chosen for expansion into full papers. Others were selected for presentation as short papers or posters. The call for participation asked submitters to indicate whether or not they would be able to expand the abstract into a paper if it was chosen. Work in progress could be submitted as a poster, without going through the extended abstract path. A submission required four copies of the material—so that it could be shared with the reviewers. Many submissions came on paper, though some came in email. Fax submissions were allowed, but discouraged.

Over time, many changes helped define ITiCSE as it is now. SIGCUE disappeared and ITiCSE quickly became a SIGCSE sponsored event. The willingness of the leadership of SIGCUE to co-sponsor the early conferences allowed SIGCSE to take the risk of starting a new conference. As the conference evolved without SIGCUE, the name became less relevant. Keeping the same acronym, ITiCSE became Innovation and Technology in Computer Science Education.

The burden on the Working Groups to produce a substantial report in five days and take the time and effort to make presentations at the conference became untenable. The editing of the final report in the days right after the conference proved impractical. Now, the working group leaders work with their teams to complete the report and submit at a specified date after the conference ends. The working group presentations as posters during breaks was replaced by a single presentation during the conference. In the very early working groups, members solicited help from conference attendees in the form of surveys and interviews to incorporate into their work. Now, more commonly, these take the form of online surveys conducted before the conference begins.

After the first two conferences, I stepped away from the committees. Happily ITiCSE has flourished under leadership from a broad cross section of the SIGCSE community. The initial committee members, particularly Mats Daniels in Sweden and

Gordon Davies in the UK, reached out to colleagues and connections to encourage interest in hosting the conference. As more people from Europe attended, the interest in hosting increased and gained a life of its own. The Tips and Techniques session became a special feature after the first few conferences and has become a popular session. Local tours, sometimes including a walking tour of the city of the conference, complement the technical component of the conference and create groups who bond over the shared experiences. The conference banquet reflects the culture of the conference site, providing experience with the local cuisine, but also entertainment typical of the area.

The enthusiastic conference evaluations from Barcelona provided the encouragement to continue and expand on the idea of a conference focused on pragmatic applications of techniques and technologies as we adapted to the presence of the web, social media, constant connectivity, and an increasingly tech-savvy student body. Those of us who were privileged to be there at the beginning have seen our infant grow and flourish and mature into a popular contributor to the computing education community.

Figure 1: Bruce Klein and Boots Cassel at ITiCSE 2000

Figure 2: Mats Daniels (far right) at ITiCSE 2000

ITICSE – A EUROPEAN EXPERIENCE (REFLECTIONS FROM MATS DANIELS)

It has indeed been a privilege to be part of this journey. It is for me a case of small coincidences that had a huge impact on my life. It started with Vicki Almstrum wanting to spend a year in Uppsala, Sweden and me being able to offer her an opportunity to teach at our department. While discussing details about her stay she introduced me to Boots and the idea of arranging the second ITiCSE. This was quite an alien idea to me, but they talked me into it and accepting this challenge is something for which I thank my lucky star.

I attended the first conference in Barcelona as a preparation and looking back I can see that much has changed, but also that the Barcelona conference left some lasting characteristics. The special form of working groups has already been mentioned, but I would like to point out the long breaks. Coffee and lunch breaks were long, maybe because it couldn't be otherwise when in Spain. Being influenced by the local culture of the site is another hallmark of ITiCSE. Anders Berglund, who oversaw local arrangements in Uppsala made a valiant effort to match the efforts of the people in Barcelona regarding making sure the attendees knew they were in Uppsala.

A side comment here is that the weather was excellent and people actually asked me if I couldn't turn off the sun, and, in fact, most ITiCSEs have had excellent weather.

I continued to get a heavy dose of ITiCSE by being program chair for the third, in Dublin, and co-program chair for the fourth, in Krakow. I continued to attend the conference and was offered a position on a committee to scout out potential sites for future conferences. I started doing this with Bruce Klein in 2005 and since he retired have been doing this with Alison Clear and Mikey Goldweber. This has, apart for giving me many treasurable memories, been an opportunity to meet and interact with people deeply engaged in issues regarding computing education. Which in many ways is what the long breaks did in Barcelona, to give opportunities to meet and interact.

Meeting potential organizers of the conference, we could assess if they could indeed arrange the conference, ascertaining if the facilities would be adequate and if the local support was in place. The main task, as I see it, was to provide the people to run the conference with confidence to put their local identity on the conference while still providing the core character. Providing a local character—in terms of food, facilities, and special area of interest regarding computing education—is one of the characteristics we wanted conference organizers to adhere to. Other aspects that we typically bring up are the importance of providing time and space for meeting other attendees, preparing for a truly international cohort, and striving for a mix of experience and "new blood" in the conference committee.

The selection of location has been strongly dependent on having someone who really wants to run the conference, but we have tried to influence people to provide variation. The main idea is that the conference should be held in Europe, and we've been attempting to alternate between north and south. We have also had it in our minds

PHOTOS: ©ITICSE2000 LOCAL ORGANIZING (2)

that the location could draw in new members to the community. One example of having old while introducing new was the 2013 conference. This conference was planned to be in Belfast, Northern Ireland, in a setting that Bruce Klein and I in our site scouting felt would be inspiring and new, but organizational changes after our visit meant that there was no real local support to rely on. To successfully deal with that situation on short notice is a sign of the strength of the community. The solution was to revisit an earlier site, in this case University of Kent at Canterbury, UK., where the 2001 conference was held. This was not viewed as a secondary option, but rather as a revisiting of old favorites, where other examples are 2014 in Uppsala (1997), and 2017 in Bologna (2006).

Figure 3: ITiCSE 2000 in Helsinki

Figure 4: ITiCSE 2001 Organizing Team: from l. to r. Paul McGrath, Carl Erikson, Bruce Klein (facing!), Roger Boyle, Xristine Faulkner, Janet Carter, Mike McKraken, Fintan Culwin, Sally Fincher, Michael Caspersen

PHOTOS: ©ITiCSE2000 LOCAL ORGANIZING (TOP); © IAN UTTING (BOTTOM)

ITICSE – THE JOURNEY TO THE PRESENT (REFLECTIONS FROM MICHAEL GOLDWEBER)

Like most people who get heavily involved in a conference or an organization, it was really just luck that provided the unique opportunities and circumstances. Regardless, I have been very fortunate, and honored to have held a variety of ITiCSE leadership roles, dating from 2003 through the present. While I was not one of the cadre who gave birth to ITiCSE, I have had the wondrously fulfilling opportunity to help nurture ITiCSE through its youth into its current mature state—23 years and going strong.

As has already been described, the working group experience is a cross between closed door jury deliberations, a hackathon, and a graduate school weekend seminar.

After the first two or three conferences (the third ITiCSE was held in Dublin, Ireland), ITiCSE, began to feel less like an experiment and more like a young—yet still growing, established community. Year after year, ITiCSE experienced steady growth in both submission rates and attendance rates. Just past ITiCSE's ten-year mark, attendance regularly exceeded 200 and acceptance rates hovered around a very respectable 30%.

While we in the academy in general, and in the SIGCSE community specifically, like to behave in a fashion "above" geopolitical politics, we are not immune from world events. ITiCSE 2003, held in Thessaloniki, Greece was an outlier with respect to ITiCSE's growth pattern. It appears that the USA's announcement of and eventual invasion of Iraq in May of 2003 discouraged potential authors and delegates. I can recall many a conversation with potential authors futilely pointing out the distance between Thessaloniki and Baghdad. Another geopolitical event that affected ITiCSE was the souring of relations between Israel and Turkey in early 2010 which kept the usually large Israeli contingent away from ITiCSE that year.

Possibly the event with the greatest impact on ITiCSE was the worldwide Great Recession. Regardless of where an ITiCSE is located, the vast majority of delegates travel, often great distances, to attend. Sadly, as the effects of the Great Recession percolated through various countries' economies, researcher travel budgets shrank. It took many years for ITiCSE submission rates and registration numbers to return to pre-recession levels.

While ITiCSE was initially conceived as a European conference, with the first twenty ITiCSEs held either in Europe or adjacent to Europe (Turkey and Israel), ITiCSE has evolved into a truly international conference. Prior to Australia gaining entry to the Eurovision Song Contest the joke was that ITiCSE could be held in any country that competes in Eurovision. Nonetheless, ITiCSE 2016 was held in Arequipa, Peru. There were two primary reasons for this. First and foremost, ACM specifically asked SIGCSE to skip a year of holding ITiCSE in Europe so that ACM Europe could hold their own experimental conference; which never materialized. Second, the SIGCSE leadership was excited to try to bring members of the active Latin Amer-

[T]he always welcoming and ever expanding ITiCSE community, at least to my eye, has felt, even from the first ITiCSE, like a place where the quality of discourse among colleagues and the freely shared criticisms and suggestions regarding each other's projects is the primary reason researchers return year after year.

ican computer science education community into the ITiCSE community. Even though ITiCSE-Arequipa was a very successful conference, enough unhappiness with the experiment was measured to prompt the SIGCSE Board to commit to keeping ITiCSE within its Eurovision boundaries through 2021. Though I suspect that means Australia is off the table, at least for now.

ITiCSE has grown in other ways as well. Early ITiCSEs had a predominately USA-based attendee list. At the most recent ITiCSE (2017 in Bologna, Italy) submissions came in from over 40 countries, and only 29% of the 228 registered delegates hailed from the USA, and only 47% came from European countries. Hence a quarter of the attendees came from regions outside of Europe and the USA.

All successful conferences have their "special sauce." For ITiCSE, I believe it is a combination of the working group process coupled with the conference's size. As has already been described, the working group experience is a cross between closed door jury deliberations, a hackathon, and a graduate school weekend seminar. When I elected to pursue a career in the academy I had visions of days filled with intellectual collegial discussions on the technical issues of the day—the Life of the Mind. As you can see I was very poorly advised! However, participating in a working group allows one to connect with that vision. Being, sometimes literally, locked in a room with anywhere from six to twelve passionate colleagues discussing, debating and sometimes arguing over the technical issues of computer science education over a five-day period is both exhilarating and exhausting.

This crucible of an experience has led to many a lifelong friendship as well as some of the most respected publications in computer science education. Some of my closest professional friends were colleagues I met participating in one of the working groups at the first ITiCSE in Barcelona. While co-leading a working group at the second ITiCSE, some participants' passion for the topic at hand almost led to a physical confrontation. Fortunately, the working group format has changed. By no longer requiring a finished report at the end of the five-day period, the pressure placed on working groups has lessened. All I can say is that I don't miss the days of post-midnight assembly-line proof reading and am thankful for BibTeX!

ITiCSE is also an intimate conference with registration counts typically between 175 and 225. With country representation often over 40, attendees are virtually guaranteed to meet new colleagues from distant lands. More important than just country of origin, it seems that ITiCSE delegates come looking for projects to join or collaborators to work with. A very typical ITiCSE story: two researchers meet face to face for the first time participating in a working group. A year or two later they have co-authored a paper on a project they first discussed when they met at ITiCSE. A couple more years pass and one of the researchers is taking a sabbatical at the other's institution. Personally speaking, all but one of my sabbaticals was spent working overseas with colleagues I first met at an ITiCSE. A perennial discussion among the ITiCSE community is how to measure the full impact of ITiCSE.

Much has been written about the publication model in computer science. Conferences are no longer places where communities gather to discuss ideas or works in progress. Instead, conferences with their eyes on ever lowering acceptances rates and ever increasing registration numbers often feel more like events surrounding the publication of an annual journal. While ITiCSE is not immune to these pressures, the always welcoming and ever expanding ITiCSE community, at least to my eye, has felt, even from the first ITiCSE, like a place where the quality of discourse among colleagues and the freely shared criticisms and suggestions regarding each other's projects is the primary reason researchers return year after year.

Twenty three years going, ITiCSE still has its "special sauce."

Figure 5: ITiCSE 2001 McKraken et al. Working Group

Figure 6: ITiCSE 2002 Naps et al. Working Group

PHOTOS: ©IMARK WHEADON (TOP); © ITiCSE2000 LOCAL ORGANIZING (BOTTOM)

ITICSE WORKING GROUPS – (REFLECTIONS FROM JUDY SHEARD)

At each ITiCSE there are groups of people who meet to work together on a computing education project, culminating in a final report some months later. The reports are peer reviewed and, if accepted, published in the ACM digital library. These groups are known as 'Working Groups.' Working Groups are a special feature of ITiCSE and have been part of the conference since its beginning. Working Groups provide a unique opportunity for computing education researchers to work intensively on a topic of interest with international colleagues.

The format of Working Groups has remained essentially the same since the first ITiCSE in 1996. A Working Group comprises a team of researchers, typically five to ten although there have been larger groups up to 17.

Signing up for a Working Group is a commitment to hard work … but brings rewarding and unforgettable experiences. The members of an ITiCSE 2003 Working Group still remember being barred from leaving their room for three hours one afternoon while striking students occupied the access stairs. There have been many long lasting collaborations and friendships resulting from ITiCSE Working Groups.

Working Groups reports are some of the most highly cited computing education publications in the ACM digital library. Working Groups have produced seminal reports that have had profound influence on the exposure and direction of computing education research. For example, the Working Group of McCracken et al. [7][1] conducted an international study of introductory computing students and concluded that at the end of their introductory courses many students don't know how to program. This has inspired many studies to investigate the learning and teaching of programming, including another Working Group of Lister et al. [5][2] that tested students' abilities at code reading exercises, concluding that many students have a weak grasp of the skills necessary for programming.

Working Groups have produced many valuable resources. For example, the Working Group of Naps et al. [8][3] developed a taxonomy of learner engagement with visualization technology. This taxonomy has since been used in hundreds of studies of the effectiveness of visualization in education. Some Working Groups have contributed to our knowledge of the computing education research field through extensive surveys of the literature. For example the Working Group by Pears et al. [9][4] surveyed and classified the literature on introductory programming, identifying key research studies and suggesting areas for further research.

Figure 7: ITiCSE 2015 Ihantola et al. Working Group

Figure 8: ITiCSE 2017 Cutts et al. Working Group

Over the years Working Groups have provided unique opportunities for international research into a variety of topics in computing education. For example, a study by Dick et al. [2] in 2002 explored cheating and plagiarism in computing courses from the perspective of academics, presenting a range of practices that can be used to reduce, detect, and respond to cheating, and in 2003 Carter et al. [1] surveyed computing academics to gain insights into their assessment practices and their views on computer-aided assessment. In 2007 Liccardi et al. [4] investigated the role that social networks play in computing students' learning experiences in their university courses.

Although most Working Groups have focused their work in the university sector, some have researched in other contexts. For example, Mannila et al. [6] in 2015 investigated the coverage of computer science, in particular computational thinking, in K-9 education.

More recently, in 2015, a large Working Group of 16 researchers, Ihantola et al. [3], investigated the use of educational data mining and learning analytics in the teaching and learning of programming, exploring the challenges in collecting and sharing programming data.

Up to and including 2017 there have been 111 Working Groups involving hundreds of computing education researchers and introducing many new researchers to the field. Over recent years the popularity of Working Groups is increasing with a record nine groups in 2017 involving over 70 researchers. In 2018, again nine Working Groups convened at ITiCSE. ITiCSE Working Groups are now well established as a key activity in computing education research. ❖

[1] 792 cites in Google Scholar, 4/Oct/2018 [2] 471 cites in Google Scholar, 4/Oct/2018 [3] 689 cites in Google Scholar, 4/Oct/2018 [4] 445 cites in Google Scholar, 4/Oct/2018

PHOTOS: ©DANIEL TOLL (TOP); © JUDY SHEARD (BOTTOM)

Four Reflections on the History of ITiCSE

References

1. Carter, J., Ala-Mutka, K., Fuller, U., Dick, M., English, J., Fone, W. and Sheard, J.. How shall we assess this? *In Working group reports from ITiCSE on Innovation and technology in computer science education* (ITiCSE-WGR '03), David Finkel (Ed.). (ACM, New York, NY, 2003), 107–123.
2. Dick, M., Sheard, J., Bareiss, C., Carter, J., Joyce, D., Harding, T. and Laxer, C. Addressing student cheating: definitions and solutions. In *Working group reports from ITiCSE on Innovation and technology in computer science education* (ITiCSE-WGR '02). (ACM, New York, NY, 2002), 172–184.
3. Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S.H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M.A., Sheard, J., Skupas, B., Spacco, J., Szabo, C. and Toll, D. Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. In *Proceedings of the 2015 ITiCSE on Working Group Reports* (ITICSE-WGR '15). (ACM, New York, NY, 2015), 41–63.
4. Liccardi, I., Ounnas, A., Pau, R., Massey, E., Kinnunen, P., Lewthwaite, S., Midy, M. and Sarkar, C. The role of social networks in students' learning experiences. In *Working group reports on ITiCSE on Innovation and technology in computer science education* (ITiCSE-WGR '07), Janet Carter and June Amillo (Eds.). (ACM, New York, NY, 2007), 224–237.
5. Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B. and Thomas, L. A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bull.* 36, 4 (June 2004), 119–150.
6. Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L. and Settle, A. Computational Thinking in K-9 Education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference* (ITiCSE-WGR '14), Alison Clear and Raymond Lister (Eds.). (ACM, New York, NY, 2014), 1–29.
7. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.D., Laxer, C., Thomas, L., Utting, I. and Wilusz, T. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *Working group reports from ITiCSE on Innovation and technology in computer science education* (ITiCSE-WGR '01). (ACM, New York, NY, 2001), 125–180.
8. Naps, T.L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. and J. Á Velázquez-Iturbide, J.Á. Exploring the role of visualization and engagement in computer science education. In *Working group reports from ITiCSE on Innovation and technology in computer science education* (ITiCSE-WGR '02). (ACM, New York, NY, 2002), 131–152.
9. Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M. and Paterson, J. A survey of literature on the teaching of introductory programming. In *Working group reports on ITiCSE on Innovation and technology in computer science education* (ITiCSE-WGR '07), Janet Carter and June Amillo (Eds.). (ACM, New York, NY, 2007), 204–223.

Lillian (Boots) Cassel
Department of Computing Sciences
Villanova University
800 Lancaster Avenue
Villanova, PA 19085-1699
http://csc.villanova.edu/~cassel

Mats Daniels
Department of Information Technology
Uppsala University
Box 337
751 05 Uppsala
mats.daniels@it.uu.se

Michael Goldweber
Department of Computer Science
Xavier University
3800 Victory Parkway
Cincinnati, OH 45207-4441
goldweber@xavier.edu

Judy Sheard
Faculty of Information Technology
Monash University
900 Dandenong Road
Caulfield East VIC 3145
Australia
Judy.Sheard@monash.edu

DOI: 10.1145/3276302

The International Computing Education Research (ICER) Conference

Sally Fincher, *University of Kent*

Although the subject of much speculative discussion in the early 2000s, there was a very real moment when ICER was brought to life. It was in a bar in the conference venue of the SIGCSE Symposium. I was talking with Richard Anderson, some of his PhD students from the University of Washington, and participants of the "Bootstrapping Research in Computer Science Education" workshop series. We started to talk about why there wasn't a dedicated Computing Education Research conference. I said, *"If I'm re-elected to the SIGCSE Board, I'll work to establish one."* Richard said, *"If you do, I'll support it".*

At their fall meeting 2004, the SIGCSE Board gave the go-ahead to start a new research-focused conference, and the first ICER was hosted at the University of Washington, Seattle on October 1–2, 2005 (scheduled to avoid football games). From the outset, ICER was built on the idea that research is formed and sustained by discourse—a research community needs places to publish (and read) and to meet (and talk). The nascent computing education research community needed ICER to be a high-quality venue, but also a high value one, and this informed all our design decisions.

INFANCY

THE TRIUMVIRATE

At the start, we could see no way to separate the work this conference needed into traditional roles. There were too few people interested in computing education research to go around. So ICER was formed and led by the Triumvirate: three researchers who would all serve for three years, each hosting the conference at their own institution once during that time. The first Triumvirate was composed of Richard Anderson, Mark Guzdial, and me.

LOCATION

A research community is not located in a single geographical area. Unlike the Symposium, it would be important for the conference to be in different areas of the world, while acknowledging the larger population of American SIGCSE members. We chose the location pattern of North America, Europe, North America, Australasia; the pattern proved satisfactory and persists to this day.

REVIEWING

Around this time (2004) there was considerable disquiet and debate about reviewing for the Symposium, with no control on who could sign up as a reviewer. In establishing ICER we took the view that reviews should be conducted by a community of peers, so an early decision was to keep reviews within the named program committee. Thus, as an author, you knew in a broad sense who was reviewing your paper and so you could have confidence in their qualification to do so (even if you might not like their opinion!). In the early days, we also took the view that the opinions of the reviewers were there to inform the judgement of the Chairs.

In founding ICER we had to establish norms and standards for submission and reviewing. Initially, Richard, Mark, and I read every paper and all the reviews—if we, as Chairs, liked the paper, it was accepted, even if it involved contradicting some (or all) reviewers' opinions and over-ruling their recommendations (and vice versa). So, from the outset we added additional comments when returning reviews to authors. This practice was largely successful, and generally appreciated, as one 2005 author responded "Thanks for the detailed comments, both from the PC and from the reviewers. I'm blown away by the thoroughness and thoughtfulness of the reviews. I wish all conferences maintained such high reviewing standards!"

This "closed reviewer pool" model has, in some years, led to extraordinary loads for reviewers—and rebellion—and different methods have been tried to balance that. In some years, a more mechanistic approach has been adopted ("adding up the reviewer scores") more recently this has been addressed by the introduction of tiers of review with a pool of "meta-reviewers" each assigned to groups of papers.

EXPERIENCE

If a conference is to help support and nourish a community, then people must exchange views. So, from the start, ICER was conceived as a place to have conversations. ICER was designed as, and

remains, a single-track conference. Not only does this mean that everyone, from PhD student to faculty, hears everything, but also that everyone can have an opinion on everything, can discuss everything. We tried several ways to introduce discussion sessions, but the most successful has been to have regular short, focused discussion periods. This practice was enhanced at ICER 2011 (Rhode Island) where we were hosted not in a lecture theatre, but in a room with round tables; this has become the model for all subsequent ICERs. After the presentation of each paper, delegates turn to each other and talk in small groups for about five minutes about what they have heard. They say what they liked about the paper, and what they didn't understand. In that way, when the plenary question period starts, some of the low-level "obvious" issues have already been dealt with, and more energetic debate follows. This can be noisy. On occasion it can be challenging. But it allows everyone to understand points of contention and interpretation; to situate their opinions; to hear great questions; to hear about other relevant work and find out who knows about it. It improves the quality of the discourse. And community forms through the exchange of views.

The Fool's Award was established in Sydney in 2008. Named for the Tarot Card "the Fool," it symbolizes a *willingness to take risks and venture into the unknown, with the possibility of achieving great things*. The conference votes on the most "out there" paper of the year, and the winner is awarded a location-themed hat.

ADOLESCENCE

THE TRIUMVIRATE (REPRISE)

Running ICER for three years was, it is true, a lot of work. But enjoyable work! However, Richard, Mark and I had no wish to run it forever, nor to be "ghosts at the feast." So, in 2008 we passed the ICER mantle entirely over to a second triumvirate: Raymond Lister, Mike Clancy and Michael Caspersen. We didn't get this quite right—it was an overly-abrupt transition, and caused a bit of a wobble, with insufficient knowledge passing from one set of organizers to the next. For a while we patched this with an ICER "shepherding" committee, to provide support and continuity, a more satisfactory solution was a "rolling membership" for Chairs, with one member retiring and one joining each year. More recently, with the growth and maturation of the community, from 2019 the roles of local Chair and technical Chair have been separated, so you are no longer expected to host the conference at your own institution.

THE DOCTORAL CONSORTIUM

In 2008, when it was clear that ICER was healthy enough to survive infancy, the SIGCSE Board decided that the Doctoral Consortium (DC) should transition from being co-located with the Symposium to being co-located with ICER. That meant there were two doctoral consortia in that year: one at the Symposium (Portland) in March and one at ICER (Sydney) in September. The DC has been with ICER since then, and grown from strength to strength, with 20 participants in 2017, all focused on CSEd research (not always the case in the early days). Gratifyingly, many who started out in the DC have become regular ICER attendees and authors, some of them award-winning.

THE FOOL'S AWARD

The Fool's Award was established in Sydney in 2008. Named for the Tarot Card "the Fool," it symbolizes a *willingness to take risks and venture into the unknown, with the possibility of achieving great things*. The conference votes on the most "out there" paper of the year, and the winner is awarded a location-themed hat. Old-timers value this quirky expression of community, and it remains an ICER feature. In 2012 the more conventional Chair's Award was added, solely in the gift of the Chairs and decided in a less-observable (although, it is argued, more-informed) process. There has, on-and-off, been discussion of adding a "10 years later" award, which would recognize those papers that have proved most influential in our thinking.

LOOKING FORWARD

The world is a different place today. The focus on teaching computing in schools, in "computational thinking," in the idea that everyone should learn computation as a literacy, has meant increasing funding and an increasing number of people wanting to read—and talk about—Computing Education Research. And ICER is there, ready to stand as an established and distinctive focus for their work. ❖

Acknowledgements

ICER would never have happened without Richard and Mark. It would never have happened without the incredibly supportive SIGCSE Board Chairs Bruce Klein and Henry Walker. It would never have happened without delegates who wanted and needed it to exist—as I write there are still two of us who have attended every ICER, and while no more can join us, there will surely be some who come to surpass us in the amount of ICER notches on their belt.

Sally Fincher
School of Computing
University of Kent
Cornwallis Building
Canterbury, CT2 7NF, United Kingdom
s.a.fincher@kent.ac.uk

DOI: 10.1145/3230698

Community Colleges and SIGCSE: A Legacy Fueling the Future

Cara Tang, *Portland Community College*

As 2018 marks the 50th anniversary of the ACM Special Interest Group in Computer Science Education (SIGCSE), it is appropriate to contemplate the role of two-year institutions—also known as community colleges—in the past, present, and future of SIGCSE and in computing education generally. For the first time in SIGCSE history, one of this year's SIGCSE Program co-chairs, and SIGCSE 2019's Symposium co-chairs, is a current full-time community college faculty member. This "first" is representative of the growing recognition in SIGCSE and the computing education community of the importance of community colleges in computing education pathways, and the contributions that community college educators can and do offer as members of SIGCSE and the larger community. Those of us who have been involved in community colleges have known this all along. As Joyce Currie Little, former two-year college educator and SIGCSE Board member, commented, "You don't ever leave the love of the two-year college level of education. It is the most important aspect of the American dream of being able to better oneself and move up in the world." This article will briefly review the history of two-year schools in ACM and SIGCSE, describe some of the current exciting efforts coming from community colleges to address contemporary issues in computing education, and offer some thoughts on the future and role that two-year institutions can and will play in meeting the challenges of tomorrow.

PHOTO: ©WWW.ISTOCKPHOTO.COM/CRISSERBUG

HISTORY

Community colleges have been a part of SIGCSE since its founding. The original petition to create SIGCSE (then a SIC—Special Interest Committee) was signed by 20 ACM members in 1968, including at least one faculty member from a two-year college—Joyce Currie Little from the Community College of Baltimore. The first membership flyer for the fledgling organization listed four topics to be addressed, among them "What is appropriate training at the junior college (now community college) level?" [15]

In 1975 SIGCSE supported the formation of the ACM Committee on Curriculum for Community and Junior College Education, chaired by Joyce Currie Little. More than a dozen community college educators participated in the first meeting and set curriculum work as the priority for the committee. The year 1981 saw the publication of *Recommendations and Guidelines for an Associate Level Degree Program in Computer Programming: A Report of the ACM Committee on Curriculum for Community and Junior College Education* [13]. The committee's curriculum work was encouraged and supported by Richard Austing, chair of SIGCSE from 1973–1977 and SIGCSE board member until 1981, and Gerald Engel, vice chair of SIGCSE from 1973–1975 and SIGCSE at-large member from 1977–1981.

It was at the 1986 SIGCSE Symposium in Cincinnati, OH where three colleagues met and discussed a vision for continuing to support programs at the community college level. Joyce Currie Little, Richard Austing, and John Impagliazzo joined later with Helen Chlopan to form an ad hoc committee that in 1991 became a standing committee of the ACM Education Board. Chartered as the Two-Year College Education Committee (TYCEC) with Karl Klee of Jamestown Community College as its first chair, the committee changed its name in 2011 to the Committee for

Community Colleges and SIGCSE: A Legacy Fueling the Future

Computing Education in Community Colleges (CCECC). A more detailed history of this committee can be found in Elizabeth Hawthorne's December 2015 *ACM Inroads* column "Celebrating 40 Years of ACM's Commitment to Community Colleges." [10] Over the years the committee has published numerous reports and curriculum volumes, received NSF awards, hosted workshops, presented at conferences, and organized networking events in support of community college programs and educators.

PRESENT DAY

The global mission of the CCECC is to serve and support community and technical college educators in all aspects of computing education. The SIGCSE Symposium is one of the primary venues in which the committee operates to carry out its mission. The CCECC staffs a booth in the exhibit hall each year, providing opportunities to connect, disseminate materials, and bring visibility to community colleges among the larger SIGCSE audience. Members of the CCECC and other community college educators give presentations at SIGCSE, including papers, posters, special sessions, and BOFs, bringing topics of interest to the two- and four-year communities, as well as highlighting issues and challenges common to two-year and four-year programs while facilitating dialogue. SIGCSE 2018's Symposium theme, "CS for All" resonates with the open-access mission of community colleges and there were numerous presentations relevant for two-year schools. Each year the CCECC provides a list of Highlights and Recommendations for Community College Educators attending SIGCSE.

Members of the CCECC and other community college educators give presentations at SIGCSE, including papers, posters, special sessions, and BOFs, bringing topics of interest to the two- and four-year community, as well as highlighting issues and challenges common to two-year and four-year programs while facilitating dialogue.

If you have attended a SIGCSE Symposium recently, you may have noticed a check-box on the registration form for community college educators, and you may have seen some attendees wearing a community college ribbon on their badge. You may also have noticed the Community College Reception in the program on Friday evening. These are a few of the ways that the CCECC and SIGCSE have been working to welcome, support, and highlight community college educators at the Symposium. Approximately 10% of SIGCSE attendees check the community college box, representing an important group within SIGCSE. Building community among two-year college educators is part of the CCECC mission and the committee has organized informal events at SIGCSE as well, such as group lunches. Similar efforts and events in support of community college educators, both formal and informal, take place at the SIGITE conference each year.

Community college faculty have served on the SIGCSE Symposium planning committees in a variety of roles. For the first time in SIGCSE history, one of the SIGCSE 2018 Program co-chairs and SIGCSE 2019 Symposium co-chairs is a current full-time community college faculty member—Elizabeth Hawthorne from Union County College in Cranford, New Jersey. Professor Hawthorne is an ACM Distinguished Educator, a past chair and member of the CCECC, and has been serving two-year computing education communities, ACM, and SIGCSE for many years.

Community and technical college educators have contributed to SIGCSE's Conference on Innovation and Technology in Computer Science Education (ITiCSE) as well. At ITiCSE 2016 in Arequipa, Peru, the CCECC organized and presented an invited panel titled "Global Perspectives on the Role of Two-Year/Technical/Junior Colleges in Computing Education," exploring education systems around the world. Panelists from five nations discussed the types of schools in their country and the role of technical/junior colleges in computing education. For more on this topic, see the *ACM Inroads* column [16] "Community Colleges in the United States and Around the World."

Groups outside academia are paying attention to community colleges, including industry groups. Intel has been a sponsor for the Community College Reception at SIGCSE since its inception. Numerous companies have been involved in curricular efforts for associate-degree programs, specifically contributing to the development of the *Information Technology Competency Model of Core Learning Outcomes and Assessment for Associate-Degree Curriculum* [3], published by ACM in 2014. Ten industry representatives from companies including Google, Dell, Oracle, Cisco Systems, and others served on the task force. Additional companies agreed to be champions of the guidance.

COMMUNITY COLLEGES ADDRESSING CONTEMPORARY ISSUES IN COMPUTING EDUCATION

Community colleges are working hard to address many contemporary issues in computing education. I will highlight two areas here.

CYBERSECURITY WORKFORCE SHORTAGE

Cybersecurity continues to grow in importance and is recognized by ACM as an emerging discipline as well as a thread

that needs to be woven throughout existing computing disciplines. Led by the CCECC, task groups of community college educators have worked to infuse cybersecurity throughout the recent ACM curriculum guidelines for two-year programs. Most recently the ACM CCECC published *Computer Science Curricular Guidance for Associate-Degree Transfer Programs with Infused Cybersecurity* (CSTransfer2017) [4]. Based on CS2013 [2], CSTransfer2017 is specially designed to aid in the smooth transfer from associate degrees to baccalaureate degrees. The curriculum contains 17 of CS2013's 18 knowledge areas, and a variety of knowledge units appropriate in the first two years of a computer science degree. The guidance comprises over 200 learning outcomes, 64 of which are infused with cybersecurity. A paper highlighting CSTransfer2017 and examples of three community college programs mapping to CSTransfer2017 (El Paso Community College, Bluegrass Community and Technical College, and Folsom Lake College) was presented at SIGCSE 2018 [17].

ACM's *Information Technology Competency Model of Core Learning Outcomes and Assessment for Associate-Degree Curriculum* [3] for IT programs addresses industry needs by offering 50 core outcomes for any IT program, with flexibility to meet varying local industry needs. Influential in producing these guidelines was the growing importance of cybersecurity and specific cybersecurity outcomes are included among the 50.

Two Community College Corner *ACM Inroads* columns in 2013 and 2014 highlight cybersecurity at community colleges in both AAS (Associate of Applied Science) and AS (Associate of Science) transfer degrees. "Multifarious Initiatives in Cybersecurity Education" [7] looks at the variety of associate degree programs in cybersecurity, focusing on those that prepare students for jobs, and offers several resources. "Creating 2+2 Education Pathways in Cybersecurity" [8] focuses on transfer pathways in cybersecurity, highlighting specific examples.

The Joint Task Force on Cybersecurity Education, a collaboration between ACM and other major international computing societies, published *Cybersecurity Curricula 2017* (CSEC2017), guidelines for post-secondary degree programs in cybersecurity [11]. The current curriculum effort by the CCECC, codenamed CSEC2Y, will provide guidelines for two-year programs in cybersecurity based on CSEC2017.

DIVERSITY IN COMPUTING

A continuing challenge in computing disciplines is the lack of diversity. The SIGCSE 2018 Symposium, with a theme of CS for All, had numerous presentations on the topic. With the greater diversity found at two-year schools compared with four-year schools in the United States, attention is turning to community colleges to help meet this challenge [14]. According to the AACC (American Association of Community Colleges), 41% of all undergraduate students in the U.S. attend community college. Over half of all community college students are non-white [1].

ACM-W has been embracing the participation of community colleges [6]. In March of 2013 the first-ever community college regional Celebration of Women in Computing was held in Kentucky. This successful conference has continued every other year with the next one to be held in 2019. KYCC-WiC (Kentucky Community Colleges Women in Computing) attracts attendees from across Kentucky, Tennessee, Ohio, and Indiana. Corporate sponsors help keep costs low, especially for students [9]. A continuing feature at SIGCSE promoting women in computing has been the Town Meeting of the Committee on Expanding the Women-in-Computing Community, a BOF for several years running. The meeting provides a venue for both university and community college educators to brainstorm, discuss, and disseminate ideas on successful gender issues projects [18].

In January 2018, an NSF-funded workshop titled, "Authentic Inclusion of Community Colleges in Broadening Participation in Computing" was held in Sunnyvale, CA. A report from the workshop is forthcoming, with some preliminary results presented at a SIGCSE 2018 special session where the conversation continued [12]. One idea suggested by attendees of the workshop was an alliance of community colleges and universities to jointly address the issues.

Community Colleges and SIGCSE: A Legacy Fueling the Future

FUTURE AND CHALLENGES

The issues and challenges in computing education will be best addressed by universities, community colleges, and K-12 working together. Serving all students well in the face of enrollment surges [5] is one critical area where cooperation is needed. Creating smooth pathways for students in computing, which may involve transfers and nontraditional educational situations, is an important challenge. The ACM Education Policy Committee's 2018 report, "Transition Pathways for Community College Students in Computing" (forthcoming as of this writing) discusses the role that community colleges are playing, and best practices in articulation agreements and transition paths. One case study describes the development of an AB (Applied Baccalaureate) degree in Information Systems at Western Oregon University (WOU) designed to allow students with an AAS in computing from a community college to transfer and complete their AB in two more years. Students at Chemeketa Community College have been benefiting from the program and WOU is reaching out to additional community colleges in Oregon to create smooth transitions.

SIGCSE ... must continue to be a forum for educators from all levels of computing education to engage in addressing contemporary issues in computing education.

It is crucial that universities and community colleges work together to serve students. SIGCSE as a SIG, along with its publications and conferences, must continue to be a forum for educators from all levels of computing education to engage in addressing contemporary issues in computing education. If you are a community college educator, get involved in SIGCSE. If you are a university educator, get involved in SIGCSE and support your local community college. ❖

References

1. American Association of Community Colleges (AACC). *Fast Facts 2017*; https://www.aacc.nche.edu/research-trends/fast-facts. Accessed 2018 February 27.
2. ACM and IEEE Computer Society. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. (ACM, New York, 2013).
3. ACM Committee for Computing Education in Community Colleges. *Information Technology Competency Model of Core Learning Outcomes and Assessment for Associate-Degree Curriculum*. (ACM, New York, 2014); DOI: 10.1145/2686614.
4. ACM Committee for Computing Education in Community Colleges. *Computer Science Curricular Guidance for Associate-Degree Transfer Programs with Infused Cybersecurity*. (ACM, New York, 2017).
5. Computing Research Association (CRA). *Generation CS: Computer Science Undergraduate Enrollments Surge Since 2006* (2017); http://cra.org/data/Generation-CS. Accessed 2018 March 2.
6. Hawthorne, E. K. ACM-W Embraces Community College Participation. *ACM Inroads* 4, 2 (2013), 38–39; DOI: 10.1145/2465085.2465097.
7. Hawthorne, E. K. Multifarious Initiatives in Cybersecurity Education. *ACM Inroads* 4, 3 (2013), 46–47; DOI: 10.1145/2505990.
8. Hawthorne, E. K. Creating 2+2 Education Pathways in Cybersecurity. *ACM Inroads* 5, 2 (2014), 33–35; DOI: 10.1145/2740968.
9. Hawthorne, E. K. KYCC-WiC 2015: An ACM-W Celebration of Women in Computing specially for Community Colleges. *ACM Inroads* 6, 3 (2015), 44–46. DOI: 10.1145/2783443.
10. Hawthorne, E. K. Celebrating 40 Years of ACM's Commitment to Community Colleges. *ACM Inroads* 6, 4 (2015), 33–35; DOI: 10.1145/2822901.
11. Joint Task Force on Cybersecurity Education. *Cybersecurity Curricula 2017* version 0.95; https://www.csec2017.org/. Accessed 2018 February 10.
12. Kahlon, A., Boisvert, D., Lyon L.A., Williamson, M., Calhoun, C. The Authentic Inclusion and Role of Community Colleges in National Efforts to Broaden Participation in Computing. In *Proceedings of the 2018 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '18). (ACM, New York, 2018); DOI:10.1145/3159450.3159627.
13. Little, J.C. et al. *Recommendations and Guidelines for an Associate Level Degree Program in Computer Programming: A Report of the ACM Committee on Curriculum for Community and Junior College Education*. (ACM, New York, 1981); DOI: 0-89791-057-5.
14. Lyon, L.A. and Denner, J. Community Colleges: A Resource for Increasing Equity and Inclusion in Computer Science Education. *Communications of the ACM* 60, 12 (2017), 24–26; DOI: 10.1145/3152914.
15. Morrison, B. and Settle, A. Celebrating SIGCSE's 50th Anniversary! *SIGCSE Bulletin* 50, 1 (2018), 2–3.
16. Tang, C. Community Colleges in the United States and Around the World. *ACM Inroads* 8, 1 (2017), 21–23; DOI: 10.1145/3007576.
17. Tang, C., Tucker, C.S., Servin, C., Geissler, M. Computer Science Curricular Guidance for Associate-Degree Transfer Programs. In *Proceedings of the 2018 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '18). (ACM, New York, 2018); DOI: 10.1145/3159450.3159536.
18. Townsend, G.C., Gabbert, P., Powley, W. A Town Meeting: SIGCSE Committee on Expanding the Women-in-Computing Community (Abstract Only) In *Proceedings of the 2018 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '18). (ACM, New York, 2018); DOI: 10.1145/3159450.3162184.

Cara Tang
Computer Information Systems
Portland Community College
12000 SW 49th Ave
Sylvania, TCB 312
Portland, OR 97219 USA
cara.tang@pcc.edu

DOI: 10.1145/3230699

SIGCSE – Who We Are: A Brief History of Conference Registration and Demographics

Cary Laxer, *Rose-Hulman Institute of Technology,* Larry Merkle, *Air Force Institute of Technology,* and Frank H. Young, *Rose-Hulman Institute of Technology*

If you have attended either SIGCSE or ITiCSE within the last two decades, the chances are very good that you've seen either one of the three of us or Lynn Degler smiling back at you from behind the registration desk. Now, as SIGCSE celebrates its 50th anniversary, it is interesting to look back at how our conference registration has evolved and at how attendance at its conferences has changed over the years.

The SIGCSE Technical Symposium on Computer Science Education is the flagship conference of ACM SIGCSE, the organization. Frank and Cary can recall attending their first SIGCSE symposia in the late 1970s and 1980s. At that time SIGCSE was co-located with the ACM Computer Science Conference, and the two events ultimately became ACM Computing Week. In those days approximately 600 people attended SIGCSE. The Symposium was of great value to the many instructors in those early years who had limited formal academic background in computing and who used SIGCSE (and SIGCSE generated contacts) as a valuable substitute for such formal training. Also, SIGCSE in the 1970s and 1980s was affordable for people whose institutions did not provide adequate support for conference attendance, and it strives hard to remain that way today.

Now, as SIGCSE celebrates its 50th anniversary, it is interesting to look back at how our conference registration has evolved and at how attendance at its conferences has changed over the years.

When Cary and Frank started their SIGCSE attendance, Dick Austing was the registrar for the conference, and he used an MS-DOS based system. Frank, upon hearing that Dick was retiring from registration, suggested to Cary that it would be a good thing for the two of them to step up and bring it to Rose-Hulman Institute of Technology. Cary thought Frank was crazy (as if we weren't busy enough back at Rose-Hulman!), but agreed that it might be a good thing, benefitting both us and Rose-Hulman. So, for the 1996 symposium in Philadelphia we took over registration, using the (antiquated!) MS-DOS system we inherited from Dick Austing. We kept that system together with baling wire and duct tape (well, the computing equivalents thereof) for a couple of years. Dick received SIGCSE's first Lifetime Service Award in 1997 for his service as an editor on several curriculum recommendation documents and as the conference registrar (Frank had to wait until 2015).

In 1999 Frank and Cary convinced Lynn, the department secretary, to join the fun and she came along to New Orleans and helped with registration for the first time. Later that year Cary went on sabbatical to UNITEC Institute of Technology in Auckland, New Zealand. While there he learned of a new database system called Jade. This was an object-oriented, relational database system. As Cary learned about it and played with it, he thought this could be a good system for writing a new registration program. He pitched the idea to Frank, including the idea of getting a few of Rose-Hulman's best and brightest computer science students to work on it as an independent study project. And away we went.

During the 2000 conference in Austin, Frank remembers that we spelled Edsgar Dijkstra's name wrong on his name badge and he (Frank) had to modify the database three times to correct the data item that was used to print the name badge. This uncovered a serious problem with the registration system being used then—it taught us why data items should appear in the database only once!

SIGCSE – Who We Are: A Brief History of Conference Registration and Demographics

A couple of years later we decided to bring online credit card payment to the registration system, which required us to re-write the system to be a web browser based system (think another student project!), and then deal with the hassles of getting a credit card payment system to work with the registration system. Frank retired from Rose-Hulman in 2002 (he continued working registration through 2006) and as one of his last official acts he hired Larry away from the United States Air Force Academy.

Having attended the previous two SIGCSEs, Larry had fallen in love with the conference and the community, so he immediately inserted himself into the registration team. Not long after that, he managed to convince Cary that the web-based aspects of the database system would be much easier to manage if we wrote our own system specifically tailored to our needs based on the LAMP stack (Linux, Apache, MySQL, and PHP). We used the system that Larry developed with significant student help for several years before acquiescing to switch to RegOnline, which ACM contracted with for conference registration services. At the time of this writing, RegOnline is scheduled for retirement and ACM is deliberating about its replacement, so it looks like we will be moving to the next generation system in time for Minneapolis.

All of us would agree that the things we remember and value the most about working SIGCSE conference registration are the camaraderie among the registration staff, the conference leadership, the student volunteers, and the strong group of volunteers who help us stuff packets and bags each year, as well as the abundant opportunities to interact with the SIGCSE community. Many good friends have come from those experiences.

So what do our conferences look like now? SIGCSE 2018, located in Baltimore, MD, was the 49th Technical Symposium on Computer Science Education. At that conference (the last conference we have full information on as of the writing of this article), attendance broke 1,700 for the first time. Some "by the numbers" facts for this conference:

- 697 people (40%) attended the SIGCSE Symposium for the first time;
- 103 people (6%) were K-12 teachers;
- 57 people (3%) were Community College faculty;
- 299 people (17%) were students, 107 of whom joined SIGCSE;
- 25 people (1%) were retired SIGCSE members;
- 150 people (9%), representing 35 countries, came from outside the United States; and
- 235 people (14%) joined SIGCSE as new professional members at the conference.

ITiCSE was started in 1996 as a summer conference, located outside the United States, intended to get more of our international colleagues involved in SIGCSE. One of the unique things about ITiCSE is the collection of working groups that bring together colleagues from several countries to work on a research problem that interests them.

ITiCSE 2018, located in Larnaca, Cyprus, was the 23rd Annual Conference on Innovation and Technology in Computer Science Education. The conference attracted 180 attendees:

- 71 people (39%) attended ITiCSE for the first time;
- 73 people (41%) participated in one of nine Working Groups;
- 23 people (13%) were students;
- 65 people (36%) were from the United States;
- 115 people (64%) represented 27 other countries; and
- 17 people (9%) joined SIGCSE as new professional members at the conference.

ICER, the International Computing Education Research conference, began in 2005. It alternates locations between the United States one year and outside the United States the following year. We do not handle registration for ICER, so we are not able to give any demographics about it. However, we wanted to make sure it was mentioned in this article, as it is yet another very worthwhile SIGCSE-sponsored conference.

The SIGCSE conferences are great venues for people to come meet old friends, make new ones, and discuss how computer science education is evolving. We hope to see you and chat with you at one of our future conferences! ❖

Cary Laxer
Department of Computer Science and Software Engineering (retired)
Rose-Hulman Institute of Technology
Terre Haute, IN USA
laxer@rose-hulman.edu

Larry Merkle
Department of Electrical and Computer Engineering
Air Force Institute of Technology
Wright-Patterson Air Force Base, OH USA
xphileprof@gmail.com

Frank H. Young
Department of Computer Science and Software Engineering (retired)
Rose-Hulman Institute of Technology
Terre Haute, IN USA young@rose-hulman.edu

DOI: 10.1145/3231746

A Personal Narrative of My Relationship with SIGCSE

Nell B. Dale, *University of Texas at Austin, retired*

This paper is an unabashedly personal memoir of my 35 years with SIGCSE. To my older colleagues, I trust it will bring back fond memories. To my younger colleagues, I hope it will provide you with a picture of the evolution of SIGCSE, from a very small group whose main conference (the SIGCSE Technical Symposium) was a two-day add-on to another conference to the major organization you know today.

SIGCSE and my professional life were intricately interwoven for these 35 years.

SIGCSE and my professional life were intricately interwoven for these 35 years. I attended my first SIGCSE Technical Symposium in 1976 in Williamsburg, Virginia. I don't remember too much about the conference, but I remember vividly my visit to Colonial Williamsburg and my lovely lunch at the hotel at the center of town. I also remember the people I met—Norm Gibbs, Robert Aiken, Della Bonnette, and Dick Austing. Many of them have remained friends over the years.

At one of my first conferences, I was asked to join a group going out to dinner. They had been doing so together for several years and I felt very privileged to be asked to join them. I enjoyed a SIGCSE night with this group for the next 30 years. The group changed and shifted, but it was always made up of people who loved fine food. The person who lived nearest to the next conference site was tasked with finding the restaurant for the next year. Having such a tradition made the world of difference to me. I sincerely hope others of you have found a group and have carried on similar traditions. Thank you, Bob Aiken, for asking me to join you.

Another tradition that meant a lot to me was having breakfast once each conference with Angela Shifflet. When Angela and her husband went to Oxford the year after my husband died, they took the time to go by his old college and take a picture of where some of his ashes were buried. I cherish those pictures. SIGCSE technical symposia should help you make professional friends, not just provide intellectual stimulation.

By this time, my list of SIGCSE friends had risen to include Boots Cassel, Harriet Taylor, Joe Turner, Henry Walker, Barber Boucher Owns, John Impagliazzo, Joyce Currie Little, Bill Bullgrin, Jane Prey, and many, many more.

HISTORY OF SIGCSE TECHNICAL SYMPOSIA

The SIGCSE Technical Symposium was originally held as an add-on to the annual ACM Computer Science Conference. The Department Chair from a neighboring university applied to run the 1991 ACM Computer Science Conference and gave my name as the person chairing the accompanying SIGCSE Technical Symposium. Imagine how surprised I was when the conference was awarded to San Antonio and I found out that I was chairing the SIGCSE portion. Fortunately, I had a lot of help from John and Laurie Werth, fellow faculty members. They became Co-Program Chairs. (Yes, this really did happen!)

Historically, the Symposium Chair took a suite in the Conference Hotel and held an unofficial reception during the symposium. It wasn't announced, it was just advertised by word of mouth. In San Antonio, we had a lovely suite with a sitting room and a huge bathroom with a big bathtub. I remember Paul (last name omitted on purpose) bringing three cases of pink champagne up the back stairs of the hotel. Then he brought up the ice to fill the bathtub. All enjoyed the reception.

However, I was concerned about SIGCSE's liability running such an unofficial party in the hotel. By the next year, the reception was run openly at the conference hotel, and its existence was published in the program.

I do not remember the exact year, but eventually the SIGCSE's Technical Symposium became the tail wagging the dog. Our conference had continued to grow as the ACM Computer Science Conference had declined, so we cancelled that collaboration and went out on our own.

I particularly remember the Technical Symposium in Orlando in 1983. It was there that my editor presented me with the first copy of my first book: *Programming and Problem Solving in Pascal*. The Exhibits at the Technical Symposia were always one of my favorite parts of the conference. I would visit each publisher's booth, talk with the reps, examine the new books, and sometimes come home with one or two. While making this circuit, I would stop and chat with friends, perhaps sharing a cup of coffee and technical gossip.

A Personal Narrative of My Relationship with SIGCSE

THE STATE OF SIGCSE IN 1992

In 1992, when I was Chair, Boots Cassel was Vice-Chair, and Harriet Taylor was Treasurer of SIGCSE, we got together to discuss the future of our organization:

- Who are we?
- How do we rate current services?
- Where should we be going?

To find the answers to these and other questions, we devised a questionnaire or survey, which we distributed to our membership. Four hundred and forty-five SIGCSE members took the time to fill out the 6-page questionnaire. The following paragraphs give a summary of the responses and the changes that resulted from members' suggestions.

Frankly, 1991 was a time of turmoil in CS education. There really was no consensus as to what should be taught in the first course.

It was not surprising that 88.7% of the respondents were associated with educational institutions. Others were in publishing, industry, government or some other type of institution.

The educational institutions ranged from high schools to PhD-granting institutions with most of the institutions four-year colleges. Of our members in teaching institutions, 41.1% taught all levels in the undergraduate program, 14.6% taught only lower-division classes, and 25.2% taught both undergraduate and graduate courses.

The first *SIGCSE Bulletin* was published in 1969. *ACM Inroads* was first published in 2010. Thus, the questions about the quality of our publications only reflected the *Bulletin* and the Conference Proceedings. First, we asked if the respondents read the *Bulletin*—75.6% read at least the Table of Contents and selected articles; 15.5% read it from cover to cover; one person tossed it on the bookshelf without looking at it; 93.5% rated the Bulletin as good, very good, or excellent. The respondents seemed equally satisfied with the Technical Symposia and the post-Symposia workshops.

The questions relating to the future of our organization were broken into two parts—what should we be doing and where should we be doing it? Curriculum '91 had just been introduced and obviously was of great interest to this group. A large majority of members were looking closely at the recommendations with intent to adopt some or all of them. As the recommendations are of only historical interest, I won't go into them. However, the concept of closed laboratories was introduced there and did have a lasting impact.

Many of the SIGCSE members were interested in, and worked closely with, the Advanced Placement Exam in Computer Science (APCS). Many of our members had been on planning committees and graders for the APCS. During the time of this questionnaire, the APCS changed its format and split the exam into two parts: A and AB. The A part was supposed to mirror what was taught in the first course in CS, and AB was to mirror the second semester.

Frankly, 1991 was a time of turmoil in CS education. There really was no consensus as to what should be taught in the first course. Curriculum 1991 was a step in the right direction, but educators still argued over what language to use in the first course. When the APCS was first instituted using Pascal, local high-school teachers were up in arms—of course Basic was the language to use. What was Pascal? Who ever heard of it? (I personally ran a worship to prepare high-school teachers to teach a course for the first AP exam. I wanted to talk about teaching concepts; they wanted to learn about Pascal.)

During the first twenty years of SIGCSE, there were other computer-education related conferences and many of our members participated in them. A listing looks like alphabet soup: WCCE, NECC, EDUCOM, ACM CSC, CCSC, AFIPS. At the time of the 1992 survey, the National Education Computer Conference (NECC) was still active. Twenty-nine percent of our members had attended an NECC conference, but only 12% in the last year (1991). This reflected NECC's focus on pre-college education. However, most respondents felt that SIGCSE should stay involved with NECC in some way.

CHANGES IN SIGCSE

There were strong feelings that SIGCSE should sponsor summer activities—workshops, refereed conferences, and/or non-refereed conferences. Others cautioned that they could only afford one conference a year. However, in response to this strong support for a summer activity, the ITiCSE Conference was initiated and held in Barcelona, Spain, in June of 1996. This conference has been held each year since in Europe, Australia, Turkey, and Israel. There were no submitted papers the first year, only refereed workshops. The results of the workshops were published in the Bulletin.

The answers to the Free Response questions of the survey highlighted our professional diversity:

- The papers are too technical/The papers are too low level.
- Having a paper accepted brings support for going to the symposium, so take more papers/
  - The quality of the papers is poor, so don't take as many.
- We need more papers on how to teach/We need more technical papers.
- Symposium should be in cheaper places/The sites are dull.

So, how did we, as an organization, solve some of these contradictions? A look at the research areas represented in the survey gave us a clue—involved in technical research, 24.6%, involved in CS Education research, 21.4%, and focused only on teaching, 39.9%. There were two distinct audiences represented in the survey: the teachers and the researchers. Those

who wanted teaching tips should be provided with the best of this type of paper and encouraged to submit papers of their own. One participant interested in research asked, "What can SIGCSE do to add validity to research in Computer Science Education?" We focused on this question as a solution to generating better technical papers. We started offering workshops on educational research techniques. In June of 1998, I began a column in the *Bulletin* on "Research in Computing Education."

The results of the push for meaningful educational research into CS Ed culminated in a new conference ICER (International Computing Education Research) in 2005. The SIGCSE Doctoral Consortium, where Ph.D. students could present their doctoral research, began as part of the SIGCSE Technical Symposium in 1998 and later moved to ICER. We were becoming respectable in the research world.

Of course, we couldn't solve all the diversity problems. We do now have a fund to help first time participants pay their expenses, which is useful for getting more first-time attendees. We also started having a first-timer luncheon to honor those first-time participants. Perhaps one of you reading this personal history will reproduce the 1992 survey with current SIGCSE members and activities—updated, of course.

In closing, I would like to thank my SIGCSE friends for contributing so much to both my professional life and my personal life. The SIGCSE community was my home. ❖

Nell B. Dale
Computer Sciences
University of Texas at Austin, retired
Austin, TX 78712 USA
nelldale35@gmail.com

DOI: 10.1145/3276303

Reflections on an Introductory CS Course, CS15, at Brown University

Andries van Dam, *Brown University*

Here I report on the largest of Brown's four introductory CS courses, CS15, a course which focuses on object-oriented programming in Java, with JavaFX as the 2D interactive graphics library. Included are my reasons for having traded off a strict non-collaboration policy for no exams and close to 200 hours per week of one-on-one help available from 50 trained undergraduate TAs. While my purpose is to report, not to advocate for my specific way of running this course, I do advocate for the pervasive use of undergraduate TAs and provide details about their responsibilities and training.

INTRODUCTION

This article reflects on my years of teaching CS15, Introduction to Object-Oriented Programming and Computer Science, at Brown University. I've been refining my introductory course since starting to teach it at Brown in 1965, and share my ideas and experiences as they evolved over the decades. I'm reporting, not proselytizing for my approach, which gives me personal satisfaction and results in student evaluations that over the years have ranged on average between very good and excellent.

I continue to teach in a rather traditional way, i.e., doing a twice-a-week 80-minute lecture to a starting audience of over 400 students, most of whom have never programmed before. I use Java as a ubiquitous, disciplined (if verbose) language, and Java FX as the 2-D interactive graphics library. Students continuing their study of computer science will take CS16, the second semester course in Algorithms and Data Structures, in which they learn Python. Since I taught my first introductory course to high school students and their teachers in 1962, while still a graduate student at University of Pennsylvania [1], I have seen (and used) many different approaches and languages to turn students on to computational thinking and computer science, and I strongly believe there is no obvious "right way" to do so—what is paramount to turning students on is to convey one's passion for the material and to help them cultivate good habits. Currently, in addition to OOP concepts and techniques, I teach basic data structures (arrays, arraylists, various forms of linked lists, stacks and queues—as special cases implemented as arrays or array lists—trees, sets, hash sets and hashing), and do "sneak preview" lectures on sorting/searching and big O, topics covered in detail in CS16.

Below I describe some of the mechanics I use for running this largest of our introductory CS courses (currently over 400 students). There is a parallel introductory sequence, CS17/18, that also assumes no prior programming experience and that features functional programming first semester and introduces OOP with Scala and Java in the second semester. Unlike in CS15, there is use of pair-programming. Further, there is a single semester course, CS 19, that provides a concentrated introduction to functional programming, algorithms and data structures for students with prior experience. And, this year for the first time there is a fourth option, CS 11, a slightly slower-paced introductory course which spreads two semesters of material plus a little extra on data science over three semesters. Students that complete any of these introductory sequences are ready and eligible to take intermediate courses in the department.

My CS15 real-time narration and annotation of PowerPoint slides via my tablet PC is recorded and available for playback after the lecture. I know the literature that suggests that live lectures (as opposed to the carefully prepared materials presented in lectures) are probably the least important learning resource for students, especially in my course, which features learning-by-doing, and where there are no exams and the final grade is entirely based on the quality of the submitted programs. However, I see lectures primarily as being useful in motivating students to learn foundational concepts and in getting them to start thinking about their assignments. Actual implementation for most students starts with reviewing the detailed lecture PowerPoint decks with their annotated code examples and listening to parts of lectures that are used in the assignment.

To get hands-on experience and more personal support from a large cadre of undergraduate TAs, students also participate in weekly sections, where they do interactive labs or engage in design discussions where TAs reinforce design principles appli-

cable to specific programming assignments. I think of CS15 as a practicum course, and of this interactive component as fundamental to reinforcing and teaching concepts. I don't grade on a curve and we assign grades using a "contract style"—if you meet the specifications of the assignment, i.e., the contract, with a well-structured, well documented program that has no significant bugs or flaws, you get the A you deserve; the clear majority of the class does indeed get A's. The TAs use detailed grading rubrics to deduct points for bad design choices and provide written feedback; it may take up to an hour per program to analyze and grade the later, more complex programs and provide helpful critiques.

These programming assignments ramp up in complexity and size, and after the initial two assignments, are all based on various kinds of graphics-based games that most students find fun and that lend themselves so well to the OOP paradigm and interactive UIs:

- Fruit Ninja, to practice polymorphic use of interfaces and inheritance;
- Cartoon, a real-time animated graphics program of the student's choice;
- DoodleJump, which has a gravity simulation component;
- Tetris, which uses arrays; and
- a 3-week long capstone final project.

For the final project we offer a choice of a Sketchpad (with undo/redo and file I/O), Othello (using a mini-max algorithm), PacMan (using breadth-first search), or a student-proposed independent project. The final projects may range between 800 and several thousand lines of code. All these programs have fully interactive UIs and are the complete, playable versions of the games. After the final project experience, many students report that they are amazed at how much they learned in one semester, since being baffled by the first simple assignment. And as they progress through their undergraduate years, they tell me it was one of the hardest courses they took but also one of the most rewarding (and formative) ones.

EVOLUTION OF COURSE CONTENT

In my first few years I taught a "bottom-up" course in which I started with gates and logical design, then a brief intro to machine language followed by assembler and a higher-level language, initially PL/I, later PL/C. In the two-semester introductory course I taught for a decade or so, I also taught data structures, parsing and compilers, a bit of information retrieval, but not theory or AI. SNOBOL was taught as a second (string processing) high-level language. My aim was to dispel magic and show how to build up levels of abstraction. Over the years the course split into two semester courses, with someone else, initially Bob Sedgewick, teaching algorithms and data structures in the second semester. I successively switched to Pascal, and when object-oriented programming started becoming mainstream, switched to Object Pascal [2], and from there to Java. Today's CS15 celebrates "magic" in the sense of having students learn to build on top of libraries and TA-written support code whose inner workings they do not need to understand (the support code, used in the first three projects, is provided in a black-boxed .jar file). This allows students to deal with the design, implementation and debugging complexity of relatively large programs, large at least for an introductory course aimed at novices. Students have told me that the interactive nature of the programs have shown them some of the practical, concrete uses of computer science, which motivated them to go on and take additional courses in the department.

While the course focuses on programming and design, it also introduces *fundamental concepts in CS*, and I try to make it clear that while programming is a key component of CS, it is a relatively small one.

I teach "objects first," i.e., all the basic OOP design principles before covering flow-of-control constructs and data structures. I was converted to the "objects first" pedagogical strategy after several years of gradually introducing OOP concepts throughout the semester and finding that by the end of the semester students still did far more functional decomposition than object decomposition. My approach also means a minimum of unfamiliar syntax and keeps students who took AP CS more engaged than starting with arithmetic and flow-of-control since many of them haven't learned OOP concepts well enough in high school, even if they use some classes. In contrast to our book based on my lecture notes [3], I now emphasize the use of composition and interfaces (factoring out) over inheritance (factoring up), and have at least one assignment that targets the use of polymorphism in parameter passing.

While I have moved from a bottom-up to a top-down approach (students learn basic logic and assembler in their third course), and from procedural to OOP languages, some aspects of my teaching style have remained invariant over more than 5 decades.

- *Computational thinking* through problem-solving via programming: while my course uses games as a motivator (without getting into game design), picking other application areas would, of course, be just as useful and motivational (health-care, multi-media…).
- While the course focuses on programming and design, it also introduces *fundamental concepts in CS*, and I try to make it clear that while programming is a key component of CS, it is a relatively small one.
- *Divide and conquer* to manage complexity: we strongly encourage students to use UML to diagram their designs and develop their code base incrementally—I encourage the

use of stubs and drivers, *hand-simulation* of pseudocode, and stepwise refinement. Most students don't do this until they confront the complexity of the last few assignments and see the benefits—some never do, regrettably

- Gradual introduction of *tools* to have students understand what is really going on when they use a tool, and to avoid creating a dependence on them. At times I have tried no tools, just an editor, or swinging the pendulum the other way with the Eclipse IDE from the start—neither strategy worked well enough and we now move from a text editor, Atom, to Eclipse over the semester.
- *No tests*, grades determined purely by individual work, no pair programming except supervised collaboration in sections/labs; instead we offer extensive TA hours—see the following section.
- *Coping with complexity* by writing fewer large programs as opposed to many small ones—all projects in CS15 use interactive UIs built on simple 2D graphics. A significant part of the grade is for the design/architecture of the underlying program. We talk a lot about program design, and as the projects progress, we gradually give less of the design away at weekly discussion sections.
- *What is not covered*: regrettably, even in 14 weeks of the semester, no second language, no concurrent programming, no teaching of how to design tests before starting implementation, no use of pre- and post-conditions, contracts, integrated analysis (big-O). I am sticking with my approach refined over many decades because it works well in conjunction with CS16.

THE UNDERGRADUATE TA SYSTEM

I arrived at Brown in 1965 in the Division of Applied Math as the first person there to have officially studied a new discipline called variously Computer Science or Computer and Information Science; I have the second designated Ph.D. in this subject (1966). My charter was to teach the elements of this new field to undergraduates, and to start a research group (in the even newer field of interactive computer graphics.) But there were essentially no students with computational background. Given a class size of around 50 people, I knew there was no way I could help that many students with their programs nor read the programs closely to grade them, so I advertised for half a dozen undergraduates both to take the new course and to help me help the other students. This initial experiment, born out of necessity, that used undergraduates essentially as TAs, proved so successful, especially with the students taking the course, that I kept using undergraduate TAs in my courses, and formalizing the system over time; eventually it was adopted by other faculty members and is now pervasive in our undergraduate curriculum. Our department will employ over 300 undergraduate TAs (UTAs) for the 2018 fall semester, and nearly that many for the spring semester. Additionally, nearly all undergraduate courses will have one to four "Head TAs," who are undergraduates who have almost always TA'd the course before and who are heavily involved in curriculum design, TA team organization, and course management. More advanced undergraduate courses may add graduate TAs to the mix and the most advanced courses will generally be TA'd by graduate TAs. A main reason for continuing to use only undergraduate TAs for the introductory courses is that they remember their initial struggles and have appropriate empathy, know the assignments well, and can relate to their peers in ways somewhat older and more removed graduate TAs can't. If a graduate student would really like to learn how to teach an introductory course, we'd accommodate that, of course.

For the fall 2018 semester I have a cohort of 46 TAs, led by a team of four head TAs who were TAs the previous year. The 46 were chosen from a field of over 150 applicants who did well in the course, using 12-minute interviews conducted by my new head TAs and supervised by me. Applicants are asked a few general questions such as "why do you want to TA," and "why do you think you would be a good TA?" They then hold mock TA hours and are asked to explain fundamental concepts such as polymorphism with a cogent example, not from class, and then to do some real time debugging of a small code fragment to see how Socratic they are in their approach. Most students cite as their reason for applying that as novices, they never would have made it through the course without the help provided by their TAs, and they now want to give back to the next generation (and learn the material even better).

We select our team based on interview performance (especially considering their "user interface," a balance of approachability and friendliness, helpfulness, clear communication, etc.), and on their ability to successfully demonstrate the Socratic teaching method—the smartest students don't necessarily make the best TAs. I also practice what I have always called "strong affirmative action," to get as diverse and balanced a team as possible and provide relatable role models for a diverse population of students. My fall 2018 team is about evenly divided between men and women and has about 15% underrepresented minorities—breaking down barriers for underrepresented minorities taking the course and subsequently TAing remains an ongoing effort and a top priority for me personally and for the department each year.

My TAs work an average of 12 to 15 hours a week for a combination of pay and course credit, and together provide almost 200 hours of one-on-one help to students each week in the course during "TA hours." Additionally, they grade and comment on the assignments and run the weekly design and lab sections for no more than 20 students at a time. They also have "TA jobs" specific to the course, e.g., A/V TA (dealing with our complex real-time recording of tablet PC interactions), scripts and grades TA (managing our extensive record keeping and grading infrastructure), Piazza TA (monitoring the flow, answering questions on this online forum typically within minutes), humor TA (writing, revising and rehearsing main skits and drink skits), etc. In short, they learn to take ownership of nearly all aspects of the course and provide a more comprehen-

sive course for the students than I could possibly do alone, even aided by a handful of graduate TAs. Additionally, they relieve me of having to deal with routine mechanics so that I can focus on curriculum design (we are constantly tinkering with the course to try to improve outcomes), supervising the head TAs and UTA staff, and exception handling, e.g., cases of students experiencing a variety of difficulties and needing accommodation, and regrettably, despite our best efforts to stamp it out, collaboration cases.

I meet with the head TAs nearly daily and am in constant communication with them; I meet with the entire team once a week to go over pedagogical strategy and issues arising, typically at our open TA hours. Through this structure, the course is a better learning experience for the students and allows undergrad TAs and Head TAs to take on challenging leadership and teaching roles that many wouldn't otherwise have until years into their careers. Many TAs go on to (head)TA in other courses, so my course is a kind of feeder program, as are the other introductory courses in our program.

To bring new TAs up to speed on their responsibilities and to provide appropriate training, we have a weeklong TA camp before the start of school. TAs have multiple social events to get to know each other and build esprit de corps. They also redo several assignments so that they are very familiar with them, go over detailed grading rubrics for each assignment to try to achieve uniformity of grading, and do role-playing where the head TAs demonstrate how not to and how to do Socratic helping of students at TA hours. I also give a short lecture on what it means to be a professional and introduce my four extra commandments: thou shalt not flake, thou shalt not assume, thou shalt be proactive, and thou shalt maintain a professional relationship—e.g., no power tripping, no hitting on students, respecting privacy, etc. We also emphasize that the course is very much oriented towards novices, many of whom are intimidated, even downright scared of this course, which is well-known for its intensity. We stress that we are oriented towards diversity and inclusion, and want to create a welcoming environment rather than an old-fashioned "weeding out" course—we strive for maximal retention, and I take great pride in the significant number of students each year who switch into CS even though they had no plans to do so, because of the introduction to the field they had. Additionally, the department runs TA training sessions for all TAs for all courses in which some of these same points are covered, especially the diversity and inclusion theme.

For many TAs, TA'ing is a formative experience in that they not only learn the material far more deeply than they did as students and learn the useful skill of code reading and debugging, but also sharpen their oral and written communication skills, and their teamwork skills. Those who work with me on course development, including researching and drafting new lectures, find that is an especially valuable experience. Many TAs get hooked on teaching and I'm proud of the legacy of more than 60 former TAs who have gone into academia or some other form of teaching either in high school or within companies. Some of the most technical TAs wind up in program or product management where they get to leverage their software engineering skills as well as their communication and management skills. At one time, former students who had either TA'd with me and/or had done research as undergraduates with me, chaired departments at MIT, Princeton, Waterloo, University of Washington, Maryland, Virginia, and New Hampshire. I also count a key designer of the Intel x86 line (and Intel Fellow), the Microsoft executive who launched early web browsers and Windows 95, and the current director of research at Google. All credit their early undergraduate experiences as TAs or RAs.

COLLABORATION POLICY

I am not a fan of time-pressured in-class exams, feeling they are too hit-and-miss. To accommodate different learning styles and speeds, I prefer the previously mentioned contract style of grading, and abandoned exams after multiple years in the late 1960s. The price students pay for not having to worry about exams and quizzes is that all written work must be strictly their own—we don't allow pair programming except in supervised labs, and no discussion of assignments, let alone solutions, between students. Instead I prefer students to get Socratic advice from qualified TAs during the nearly 200 hours of one-on-one help available during the week, augmented by both private and public Piazza posts. Last year average response was 30 minutes, but often, especially during lectures, it may only be a few minutes. Repeated private questions and common misperceptions will be noticed by TAs and result in a public response.

In my upper-level graphics course, there are small-group final projects so my strategy of having students get help from TAs instead of doing pair-programming is only for my introductory course. During lecture I strongly encourage discussion among students to answer the frequent clicker questions, to help decrease the sense of isolation. We also foster strong student interactions, including supervised pair programming, in the weekly sections. Students alternate writing the code and "dictating" what should be done next. Since students stay in the same small group sections led by the same TAs, they get to know each other and this type of community building helps combat the sense of isolation that the collaboration policy can have as an unwanted side effect. Students this fall were also able to choose a mentor TA who can provide some social support in addition to answering general questions.

We use the Stanford software MOSS (Measure Of Software Similarity) [4]—to check for undue similarity in code submissions (including checking against the previous three years' submissions) and show students early on in lecture how we use it, so that they are clear about our vigilance and process. I found over the years that it takes multiple years to hone an assignment (and its write-up) so that it is fun, appropriately challenging but not too difficult, builds on the concepts taught in class, doesn't have too much "make work," etc.; thus, an assignment like Tetris has been used for several decades now. Despite all my warnings

and the MOSS demonstration, some students still try to get away with various degrees of collusion, even code copying, and we subject those program pairs identified by MOSS as highly identical to a detailed hand inspection to make sure of our case. When we are sure of substantial collusion, we bring the case to Brown's academic code committee. Conviction typically results in a directed No Credit on their transcript and parental notification, for a first offense. Documenting and prosecuting these cases are by far the worst parts of teaching this course. The recent year when I relaxed the collaboration policy slightly to allow high level design collaboration on well circumscribed portions of the last few assignments, there was a three-fold increase in the number of cases I had to prosecute; the following years we reverted to a strict "no collaboration of any kind" policy and the number of cases was reduced dramatically.

During Brown's two-week "shopping period" where students can attend as many courses as they like, I make it clear that they have a choice between CS15's no tests, no collaboration policy and CS17's pair programming plus tests policy. There is also the significant difference between CS 15's "magic is great" with OOP and interactive graphics, and CS17s "reduce magic as much as possible" and functional programming. Thus, students have a clear choice.

MOTIVATORS

We've used a variety of mechanisms over the decades to keep students interested and coming to class; even so, class attendance inevitably decreases over the semester, as it does in most courses, and I must compete against the recording of my lectures. Our primary mechanism is well-rehearsed skits which happen every couple of weeks and are somewhat tied to concepts in the course such as linked lists. The opening day skit is usually based on some topical movie or television show, and is centered on making fun of ourselves, CS15, Brown, pop-culture, etc. There is usually an all-hands dance number in the 15-minute-long opening day skit to culminate it. Among recent themes were X-Men, Mean Girls, and Game of Thrones, where I played Daenerys in blonde wig and ball gown. Each lecture also has a 2-minute "water skit," featuring a sequence of increasingly silly plots and drink containers. Students often report that they took and stayed in the course because it looked like great fun, and because the TAs seem to have such a good time performing the skits. I want students to see that we don't take ourselves too seriously, and they consistently report that this bit of levity is worth the loss of lecturing time. The skits don't just serve to amuse, they also "humanize" us and make us seem much more approachable. We hope this encourages students to seek out TA help when they need it.

Two other short preludes to lectures are bringing back former TAs, who are typically on campus to recruit for their companies or to visit friends, to talk very briefly about "life after Brown," and "IT in the news," a 1-2 slide sequence on something just in the news. This year's topics included massive data breaches, Cambridge Analytica and the manipulation of our election, advances in Machine Learning (Alpha Go Zero, accidents with Self-Driving Vehicles) and automation, the future of work, and Universal Basic Income. Detailed mid-term and end-of-semester questionnaires measuring every aspect of the course consistently show student appreciation for these deviations from a strict lecturing mode.

CONCLUSION

After multiple decades of progressive refinement, we like to think that our current course runs like a well-oiled machine, and overall has a growing, well-satisfied clientele. Our highly detailed mid-term and final questionnaires and TA interviews tell us that the course works for the clear majority of students taking it, and that for most it is a very demanding but equally rewarding course. A large proportion of students since I started teaching reported that they had no idea whether they were interested in CS and then decided to take more courses, possibly major in CS. The TA system is largely responsible for helping students cope and is consistently ranked as one of the most valuable aspects of the course. So, while the specific style and mechanisms I use for running the course are just one of many that work to have students learn and turn them on to computational thinking, I believe the undergraduate TA system I use is well worth adapting for other courses, as is done in the Brown CS department. ❖

Acknowledgements

I gratefully acknowledge the profound contributions that five decades of TAs and head TAs have made to my courses, and especially to my introductory courses. I here have room to name only a few of the most important ones who worked with me on course development or co-authored books, but am immensely grateful to the many others who also contributed ideas, examples, content, and above all, devotion to our common cause—in the early years, Robert Munck (less a head TA, far more my co-instructor), in the middle years David Niguidula, Robert Duvall, and Brook Connor, and more recently, Samantha Squires, Nick Turley, Alexa van Hattum, and Divya Mahadevan. For those I should also have listed, please accept my apologies for having had a senior moment and remember that I've had the privilege of working closely with several hundred head TAs over the years

References

1. Asher, D., van Dam, A., and Prener, D. Computer and information sciences program for high school students. in *Proceedings of the 1964 19th ACM national conference* (ACM '64). (ACM, New York, NY, 1964), 111.301-111.305.
2. Connor, D. L., Niguidula, D., and van Dam, A. *Object Oriented Programming in Pascal*. (Addison-Wesley, 1995).
3. Sanders, K. E. and van Dam, A. *Object-Oriented Programming in Java: A Graphical Approach*. (Addison-Wesley, 2005).
4. Stanford MOSS (Measure Of Software Similarity); https://theory.stanford.edu/-aiken/moss/. Accessed 2018 June 7.

Andries van Dam
Thomas J. Watson, Jr. University
Professor of Technology and Education
and Professor of Computer Science
Brown University
Providence, RI 02912 USA
andries_van_dam@brown.edu

DOI: 10.1145/3284639

What We Care About Now, What We'll Care About in the Future

Mark Guzdial, *University of Michigan*

ACM SIGCSE is the oldest and best-known organization supporting computing education in the world today. Over 1700 attendees participated in the 2018 SIGCSE Technical Symposium, an indication of the importance of SIGCSE. SIGCSE's 50 years almost perfectly matches the history of computing education. SIGCSE has defined the focus of computing education for the last 50 years. Predicting the future of SIGCSE is predicting the future of computing education.

WHAT WE CARE ABOUT IN CS

We teach what we care about. What we teach in CS has changed surprisingly little in the 50 years of SIGCSE. Programming languages come into fashion and then depart, and the topic names might change. The core of *Curriculum 68: Recommendations for Academic Programs in Computer Science* [2] is much the same as the current curriculum volume, *Computer Science Curricula 2013* [3].

Both emphasize programming, of course, but the details differ. Machine language is mentioned prominently in Curriculum 68. It's still there in CS 2013 but on an equal footing with assembly language. Discrete mathematics does not appear in *Curriculum 68* by that name, but the content of a modern discrete mathematics course appears in several of the required courses. Calculus was considered a peer introductory course, right next to "Introduction to Programming" in *Curriculum 68*. Today, it's not a requirement, but the four dozen times it appears in the *CS 2013* document tells you that a calculus perspective still pervades our CS curriculum.

Teaching computer science in post-secondary education was designed to provide well-prepared software developers for the growing software industry. SIGCSE was created to support these efforts to prepare computer scientists at college. The original objective of SIGCSE was "To create a forum to discuss common problems among college educators attempting to develop and implement programs and courses in computer science."[1]

[1] With thanks to Briana Morrison for finding this for me.

WHAT WE CARE ABOUT IN COMPUTING

The mission of SIGCSE is different today [4]:

The scope of SIGCSE is to provide a global forum for educators to discuss research and practice related to the learning, and teaching of computing, the development, implementation, and evaluation of computing programs, curricula, and courses at all education levels, as well as broad participation, educational technology, instructional spaces, and other elements of teaching and pedagogy related to computing.

We in SIGCSE value *computing* which we define as much broader than *computer science*. Computing is an umbrella term that includes computer science and related disciplines, such as computer engineering, information systems, information technology, software engineering, cybersecurity, and data science. Computing does not even have to be about programming at all. The 2018 SIGCSE award for outstanding contributions to education went to Tim Bell, who famously developed ways of teaching computing concepts without any computer at all, embodied in his wonderful *CS Unplugged* curricular materials.

Think of all the people who develop and implement computer programs today. The writers of *Curriculum 68* would likely be surprised at who programs and why. Today, there are many "computational" fields. Computational journalists write code to mine data in order to enrich their stories or discover new stories. Computational photographers invent new things to do with their digital cameras, like stitching together multiple shots to make panoramas. Computational science has been called the third branch of science, after theoretical and empirical.

The computer scientists who started SIGCSE would not be surprised at any of these applications. The course "Large scale information processing systems" which appears in *Curriculum 68* might be called "Big Data" today. Computer graphics was an elective course in Curriculum 68. Alan Perlis, the first Turing Award winner, talked about doing computational science with simulations in 1961. What would have been hard to predict 50 years ago is specifically what parts of computer science would become the core of a new computational practice, and that the practitioners of these new practices would not be computer scientists.

Not all of these new "Computational X" programmers know machine language, calculus, or discrete mathematics. The broad range of people who use computing, who might be called

computationalists,[2] are taking the pieces they find to be useful from computer science and programming. They do not need to be computer scientists to find and make value in computing. The new computationalists care about computing, but not necessarily all of computer science.

EXAMPLE: SCRATCH AS COMPUTING THAT COMPUTATIONALISTS CARE ABOUT

SIGCSE members of 50 years ago likely would not have imagined 25 million programmers, many of them schoolchildren, using a programming language that does not involve typing. Scratch is amazing on many levels. It's a real programming language that people use in dozens of countries to do programming that they find personally meaningful and useful. Scratch is also a website with 25 million users sharing almost 30 million projects.

Scratch is one of a family of *blocks-based programming languages* that have been developed for use by computationalists, not just computer scientists. Scratch programs are created by dragging into place blocks that look a bit like jigsaw puzzle pieces. These blocks are the primitives of the language, and the Scratch library includes blocks to control program flow, change graphical elements, and manipulate data.

The empirical results on blocks-based programs are impressive. Learners can build something personally useful and interesting faster in blocks-based programming languages than typical text-based languages. They learn more and faster with blocks-based languages [5]. If they ever switch to text-based languages, the knowledge of programming elements, like control flow and variables, transfers from blocks to text—the students remember and successfully use ideas like repetition.

Scratch computationalists are building programs that they want to build. Many Scratch projects just tell stories, with moving characters, sound, and text. Other Scratch projects are games. There is a wide variety of Scratch projects—each of which is meaningful and useful to someone.

Empirical studies of Scratch projects suggest that there is not a whole lot of computer science there. There is actually little use of iterative control structures in Scratch programs. Scratch programmers rarely use Booleans. Some studies say that Scratch projects "smell" in the software engineering sense, e.g., too many literals and lots of duplicated code [1]. Computer scientists may not all value Scratch.

The important part of Scratch is that computationalists find value in it, i.e., that they can make something that they care about in Scratch. What we see in Scratch is the same process we see among the computationalists in computational photography, journalism, and science. They don't need all of computer science. They can find value and make something useful with just some parts of computing. Scratch projects smell wonderful to Scratch computationalists.

WHAT WE WILL CARE ABOUT IN THE FUTURE

Computer science education will likely be important for the next 50 years, and SIGCSE will likely grow and support this important endeavor. We will always need computer scientists. We will also need many kinds of computationalists, and they will need different kinds of education. Computing education will likely continue to fork, just as computing does. S**IGCSE will likely be best known in the future for supporting the broader goals of computing education**.

SIGCSE members are involved in all aspects of computing education, in concert with the mission statement of SIGCSE. We have members involved in computing education in schools, for professional development of teachers, for computationalists in a wide variety of disciplines, as well as for our those studying computing at college. We care about all of computing education.

The world has a growing appetite for computing education, but that appetite is finicky. Computationalists are picky. Not everyone likes tomatoes or mushrooms. Not all computationalists want discrete data structures or machine language. Computing education is so important that everyone wants some of it. Computing education is so important that even small parts of it are valuable and useful.

I can't predict what the next big "Computational X" will be, but based on past experience, I know that it's already here. Someone will come along and realize that just some parts of the computing that we have today will be useful to someone else tomorrow. They will care about just a segment of the CS education described in *Curriculum 68* and *CS 2013*. Just one part of what we do will become the heart of a new computational practice. That one part will be important and the seed of great innovation and value. William Gibson said it well, "The future is already here— it's just not very evenly distributed."

SIGCSE members will be helping computationalists to learn those parts. Our mission is to teach people about the computing that they care about. SIGCSE members will make important contribution in the increasingly-computational world. **We help people to understand their world and to build new things in it**. ❖

References

1. Aivaloglou, E., & Hermans, F. *How Kids Code and How We Know: An Exploratory Study on the Scratch Repository*. Paper presented at the Proceedings of the 2016 ACM Conference on International Computing Education Research, (Melbourne, VIC, Australia, 2016).
2. Atchison, W. F., et al. Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science. *Commun. ACM*, 11, 3 (1968)), 151–197. doi:10.1145/362929.362976.
3. Sahami, M., & Roach, S. (2014). Computer science curricula 2013 released. *Commun. ACM*, 57, 6 (2014), 5–5. doi:10.1145/2610445.
4. SIGCSE Mission; https://sigcse.org/sigcse/about/profile. Accessed 2018 September 9.
5. Weintrop, D., & Wilensky, U. (2017). Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Trans. Comput. Educ.*, 18, 1 (2017), 1–25. doi:10.1145/3089799.

Mark Guzdial
University of Michigan
Computer Science & Engineering Division
2260 Hayward Street
Ann Arbor, MI 48109
mjguz@umich.edu

[2] A term that Charles Isbell first introduced me to.

DOI: 10.1145/3276304

SIGCSE: Remembrance and Looking Forward

Lillian (Boots) Cassel, *Villanova University*

Sometimes looking ahead means looking back to see where the path began and then projecting that forward, with thoughts about possible forks in the road.

My first SIGCSE symposium will always be a defining element in what SIGCSE is and has been to me. I was young, and a brand new department chair of a brand new department. It was 1981 and we did not have enough faculty to meet the demand for courses. I came to the symposium in hopes of finding help. I was shy and quiet, planning to listen and learn. There was a session about preparing faculty, led by Larry Jehn. The discussion seemed to focus on what kind of credit people should receive for doing training courses. I had not said a word the whole time I was at the conference, but I finally broke and spoke up— "I don't need credit," I said. "I need help. I have to teach all the standard courses and I don't know how to choose books or build assignments or prepare classes." Larry, whom I had not met, looked at me, pointed to someone else in the room, and said "You need to meet him." The someone else was Dick Austing, then another stranger. As the session ended, Dick introduced himself, told me about the SIGCSE reception, and told me that I needed to be there. In those days, the reception was a small, informal affair in the conference chair's hotel room. Yes, really. People learned about it by word of mouth. I would never have known about it, and would have been too shy to go if I had heard about it. Dick took me there and introduced me to Charlie Shub, then disappeared. Charlie asked what I needed. I said I needed, for example, to know a good book to teach operating systems, and what assignments would work with undergraduates, and where to find teaching resources. Charlie started telling me about books and offered to share his materials. Within a few moments, a crowd had gathered around us, because a bunch of other people needed to know similar things. And that is SIGCSE to me.

Charlie started telling me about books and offered to share his materials. Within a few moments, a crowd had gathered around us, because a bunch of other people needed to know similar things. And that is SIGCSE to me.

Dick Austing was Symposium chair in 1984 in Philadelphia, and he asked me to be program co-chair with Joyce Currie Little. I said I had never done anything like that. He said "Neither had I, the first time." The three of us were the entire symposium committee then. The Symposium was organized in conjunction with the Computer Science Conference, which provided local arrangements. There was an immense career fair for interviewing faculty candidates. The ACM programming contest was also co-located with the conferences. The exhibits were nearly all publishers, promoting the latest text books. Many of the books were closely aligned with the most recent Computer Science Curriculum recommendation. That was Curriculum 78—which gave us the labels CS1 and CS2, still used now. Papers were submitted on paper then, with duplicate copies to be mailed to reviewers. Joyce and I got the final pages for the accepted papers, and actually printed a long list of numbers, cut them out, and pasted them on the pages to make the page numbers. There was a physical box of materials that was passed from one program chair to the next to explain what needed to be done, the schedule and the hard deadlines, the special paper for final copies, etc.

From the beginning of my involvement in SIGCSE, this group of people has been about sharing: sharing experiences, sharing resources, sharing problems and sharing solutions. The symposium was much smaller, about 350 people, and it was easy to feel connected to everyone. Nell Dale once commented that what intrigued her about SIGCSE was the number of people she considered friends, whom she saw just once a year. She said it

felt like she ended a conversation one year and picked it up again the following year as though there had been no intervening time.

By the mid-1990s, I was on the SIGCSE Board. At the business meeting at the symposium, we collected email addresses. We created the first SIGCSE mailing list—it was initially a forwarding address at Villanova, where I had moved some years earlier. The list grew and interest grew and eventually we were able to get the list moved to ACM. Now, of course, it is available to all members without having to attend a business meeting to sign up.

TIMES CHANGE.

There is no paper involved in submitting papers. Reviewers and meta reviewers work from online copies. There is a template for formatting the final accepted papers and no pieces of paper with blue outlines of the print area. The symposium has grown, and there have been many changes. In 2018, the official attendance figure was 1,735. There is a first-timers luncheon that now seems larger than the attendance in those early days. Old timers like me are encouraged to come and greet the newcomers and help them feel welcome. This year I met a young woman who said what amazed her about the symposium was that everyone was so willing to share what they have and what they know. Times have changed, but she was experiencing the modern equivalent of my experience with Dick Austing and Charlie Shub. It made me feel that the essential core of SIGCSE is intact.

Of course, the symposium has changed in more ways than the size. There is now a significant group of SIGCSE members who do computer science education research. Their papers are more formal than the traditional descriptions of experiences and ideas for innovations. They are building a body of work that informs the community based on reproducible results, suitably evaluated. The symposium now has a track for experience reports, what used to be virtually all of the papers. High school teachers now comprise a meaningful component of the attendance, as the teaching of computer science has become a regular part of the high school curriculum in many places.

There are fewer publishers in the exhibit hall. People now can find very valuable resources online. There are YouTube videos and interactive tutorials on a wide range of subjects. The Ensemble computing education portal [1] has collections of teaching resources in a number of computing topic areas, and is part of the National Science Digital Library [2]. There are also spaces for communities to work on resource development. The SIGCSE exhibit area is still large and lively, but has very few publishers.

Perhaps an exciting area of change in computing education for the immediate future is the trend to interdisciplinary collaborations. We are seeing computing as an essential component of nearly every discipline. In addition to the tremendous growth in the number of majors, interdisciplinary collaborations are bringing us more students who want a significant computing education without necessarily doing the major. They are demanding more than the traditional computing for non-majors courses. Often, they are looking to what were previously advanced and specialized courses. Examples include machine learning and other components of artificial intelligence.

New fields that meld computer science with other disciplines now attract a lot of attention. Massive amounts of information, whether from telescopes or census records, genomes or voting records, demand new approaches to how we experiment and make new discoveries. Data science (and the closely related area of analytics) requires meaningful computer science combined with statistics and some connection with one or more domains of application. Digital humanities give a new face to ancient fields and new challenges to the computing and information sciences. These, perhaps, provide a glimpse of the computing fields of the future—still rich in the study of how we can interact with digital devices to solve many types of problems, but also intimately connected to other disciplines.

Of course, the future of computing depends on attracting and retaining a diverse workforce. The need for a multitude of views seems finally to have become apparent, and we are beginning to see a broader base of student interest. Perhaps moving the beginning courses in real computer science to lower levels of schooling will help previously missing populations find that this is a field that touches everything. No matter what interests you, computer science is there and is an integral part of the field. At Villanova, we are seeing women making up 35–40% of our majors and we see increasing numbers of minority students as well. Not only does this represent a good, healthy development for the career prospects of these students, it means that computing products will benefit from multiple perspectives and will better serve the whole population.

Interdisciplinary collaboration, a more diverse community, new challenges in dealing with massive data sets, they all represent an exciting new time for the computing field and for SIGCSE. SIGCSE has always been about supporting each other, growing together, welcoming change and moving forward. My best hope and sincere expectation is that SIGCSE will greet the new challenges and continue to be what it has always been—a mutually supportive community, ever willing to share and to support the newcomers and to challenge the old timers. ❖

References

1. Ensemble computing education; http://computingportal.org/. Accessed 2018 August 11.
2. National Science Digital Library; https://nsdl.oercommons.org/. Accessed 2018 August 11.

Lillian (Boots) Cassel
Department of Computing Sciences
Villanova University
800 Lancaster Avenue
Villanova, PA 19085-1699
http://csc.villanova.edu/~cassel

DOI: 10.1145/3230701

Overheard at SIGCSE '68

Zach Dodds, *Harvey Mudd College*

REGISTRATION

At last—we made it! I can see why SIGCSE's never been to LA before. What I *can't* see is why 2068 was the year to start. That was flood-fill traffic! And for a company whose slogan is 'Go Lyber - go free,' that ride sure didn't *feel* free."

I think they generate their slogans algorithmically, Alice.

Ugh. Like their driving. And weren't we promised *flying* cars by now? Aha—there it is, Bob: 'Welcome to SIGCSE 100!' Let's register! Who's keynoting?

This morning? It's Ray Kurzweil. Registration's up there to the right.

Ray K. *again*? Why don't you take in that one for the both of us, Bob?

No problem—I'll share the highlights.

Here we are. Good morning, Cary!

Great to see you this year, Alice! Sending the conference program to your spex. Be sure to check out all the hundredth-anniversary exhibits. And this year: all-day kombucha and krullers in the exhibition hall.

Kombucha! And I love the exhibition! Which way, Bob?

Past the registration, turn into to Hall B. The kombucha's at the far end …

EXHIBITION

Every year, there are more exhibitors! Look at this one, Bob: 'iEye: Captain your Career' ...

Avast there, Alice! Have ye heard of iEye? In just the past decade, it be told, committees' expectations have grown fourfold. And with iEye, you'll increase professional sightations by an order of magnitude.

Thanks, mate, but that's a plank I've already walked. Now, if you could help with this week's trove of *student projects*—or with any of the treasure I'm buried in from *last* week, I'd be the first on board.

"Ah—there it is, Bob: 'Welcome to SIGCSE 100!'"

Aargh! Sorry, Alice. As they say, 'Education's free. It's the teaching that costs.' Maybe try *Tend2IT* over there…

Hey, Alice! Do you use software gardening? With Tend2IT, your students will raise their software and their self-efficacy—all while helping you plow through their projects. From sowing to growing, Tend2IT is your platform.

In fact, we *do* take a software-gardening approach. What's more, this year I had my students spend a week *composing* Python, instead of *training* it—just for the old-school feel. Do you support scripting?

Whoa there—that's not something everyone gets to try! Using voice? Or *typing*!?

Yes, typing! Just for a couple of programs. Lots of them loved it. And they said their parents *loved* they were doing it. They were astonished, though, at the slowness—and all the typos. Most don't believe that so much software—or *anything*—was ever written by hand!

And they better appreciate software's growth mindset. After all, humans never really 'write' software, Alice. They 'nurture' it. They 'teach' it.

SIG-C

Hmmm. *Teaching* software… I like it—it's an idea worth thinking about, Bob. Ah—here's one of those SIG-C history booths: 'Experience SIGCSE 2018.'

Overheard at SIGCSE '68

Welcome, Alice! How about a guided tour of SIGCSE, 50 years ago?

2018? That must have been something, CS'ing in '18.

It's incredible! Think of it—back in 2018, there were still six Millennium problems and only eight Millennium Falcons. Isn't the upcoming *Star Wars: Force Fed* the twentieth? Rey's going to make a cameo, I hear.

Right—in 2018 the independence of P vs. NP wasn't known. There was no Axiom of Nondeterminism yet...

From my experience, it's the most obvious of the human axioms.

And 2018 was before Panop was running. How'd you re-stage SIGCSE 2018 without all of today's cameras?

They let us gather all the pixels taken at the 50th conference. You know, most of 2018 was still in photos? We image-wove the 3d model. It's pretty coarse, with lots of blind spots; our gaming team has invested some creativity in the gaps. But the best part of SIGCSE '18 is chatting with the attendees. They're all Markov-animated using social media sources of the day. You'll feel like you're talking to your grandparents!

I love it! I got this. How about, 'Far out! CS is groovy!'

That's too far back, Alice…

'What about, 'SIGCSE is hella high-key!'

Not sure about that either, Alice…

OK. Whatever, Bob.

Spot on. You're good to go. And remember, Alice, in 2018 "person" was a synonym for "human."

Good point, Bob. No splaining needed. We're both people—but better to let me lead on this.

Hi—sorry to interrupt, but I'd really recommend it. I ran the stage yesterday, it was great! I'm Charlee, by the way—nice to me you.

Hi—I'm Alice.

The stage has all the details. In '18 they served pretzels—and coffee!

Coffee? Wow—I'm glad we're in '68 and not '18. Do you work out here, Charlee?

No, I'm at Amazon U.'s east-coast campus.

I see! I teach at Alphabet.

But the best part of SIGCSE '18 is chatting with the attendees. They're all Markov-animated using social media sources of the day. You'll feel like you're talking to your grandparents!

Nice! Actually, that was a frustration with SIGCSE '18. Company schools weren't a thing back then. If you try to share your experiences with the attendees, you just get non-sequiturs back. Or at least *I think* that's what was happening.

Well, I've got too many tabs open to run the stage right now anyway. We're in the midst of student projects.

For us, those all start next week—I'm steeling myself. Are you headed to the keynote, Alice? It's Kurzweil, I hear.

KEYNOTE

Ray K.? Let's go! Which way, Bob?

Out past the kombucha. Take the escalator up and into the next SIGCSE story.

Hah—you know, Bob, you really *are* a daemon.

True. But it's what makes us such a good team, Alice! ❖

Zachary Dodds
Harvey Mudd College - Computer Science
Claremont, California 91711 USA
dodds@cs.hmc.edu

DOI: 10.1145/3233229

Looking Forward by Looking Back

Eric S. Roberts, *Stanford University*

Difficult to see. Always in motion is the future.
—Yoda, The Empire Strikes Back, 1980

On the occasion of SIGCSE's fiftieth anniversary, a few of us who have been around SIGCSE a long time have been asked to think about the organization's future over the next half century. Gazing into a crystal ball that far in advance is, of course, impossible to do with any hope of accuracy. If nothing else, consider the projections about our field that serious people have offered in the past. Although there is controversy about whether Bill Gates actually uttered the oft-quoted line that "640K ought to be enough for anybody," there is compelling evidence for Digital Equipment Corporation founder Ken Olsen's 1977 claim that "there is no reason for any individual to have a computer in his home" or networking pioneer Dave Walden's assessment that the limit of 127 nodes in the ARPANET was "a reasonable approximation of infinity." As a field, we haven't been particularly good at predicting how our discipline will evolve, and it is unlikely that we can do much better looking forward from 2018, at least in terms of specifics.

The breakthroughs in computing in recent years are staggering beyond anything we could have imagined 50 years ago.

What we can do, however, is use our understanding of how computing and computer science education have evolved in the past to make general predictions with high confidence, even fifty years ahead. For example, I am entirely safe in offering the following claim.

Over the next fifty years, computer science will become increasingly central to our lives, our culture, and our economy in ways that no one can possibly foresee at present.

That prediction, after all, would have held up perfectly from the time of SIGCSE's founding to the present day.

The remaining sections of this paper seek to extrapolate current trends into the future in three areas.

1. The excitement that continues to characterize computing in the modern age.
2. The increasing centrality of computing to both the modern economy and the conduct of our daily lives.
3. The challenges and opportunities we face as computer science educators in this time of great change.

In each of these sections, I start by looking at common trends in the past and that can serve as reliable general guideposts to the future.

THE EXCITEMENT OF THE DISCIPLINE

I have had the good fortune to be involved in computing for more than the fifty years that SIGCSE has existed. In 1964, I had a volunteer job soldering connections on the backplane of a room-sized analog computer that was being built in the electrical engineering department at my father's university. My first exposure to digital computers began a year later as an assembly-language programmer for the IBM 1401. In high school, the Sputnik-inspired campaign to prepare more students in science and engineering gave me wonderful opportunities. I took the three courses our school offered in electronics and spent most of a summer at a NSF-sponsored summer course that introduced me to the IBM 1620—a machine with a teletype console that one could use as a personal computer, as long as one was willing to stay up into the wee hours of the morning when no one else was using it. Through those experiences, I fell in love with programming and have maintained that devotion ever since.

Starting college in the year the first computer network came into existence, I was fortunate enough to be part of the networking revolution from its infancy. I have had an email address since 1971 when the first email programs appeared. I also worked for several years as a graduate student at Bolt Beranek and Newman, the company that built the ARPANET. My doctoral thesis at Harvard drew on my work at BBN with one of the earliest multiprocessor systems, a field of research that is still important today.

It was undeniably exciting to have the chance to witness first-hand those early days in computing. Even so, I feel confident in making the following claim and prediction.

The computing field has never advanced as quickly as it is moving today. The power that computing offers and the excitement that comes from being able to harness that power will continue to grow over the next fifty years.

The breakthroughs in computing in recent years are staggering beyond anything we could have imagined fifty years ago. In 1957, computing pioneers Herb Simon and Allen Newell predicted "that within ten years a digital computer will be the world's chess champion." [6] That forecast was overly optimistic. It took forty years for chess algorithms to beat world-champion Garry Kasparov. Five years ago, however, hardly anyone would have predicted that a computer program could win against the world's top Go player any time in the next several decades, but AlphaGo and its deep-learning strategies accomplished that milestone in 2016. The leading edge of technology just moves faster and faster as computers become more central to the modern world.

THE GROWING CENTRALITY OF COMPUTING

Research and development in computer science—no matter whether it is carried out in universities or industry—has a profound impact on the world in which we live. If nothing else, computing has become the main driver of the world economy. Figure 1 makes this point clear by listing the five largest industrial corporations in 2007 and 2017.

In just ten years, the list of the top five industrial corporations has shifted from including no computing companies to being entirely composed of them.

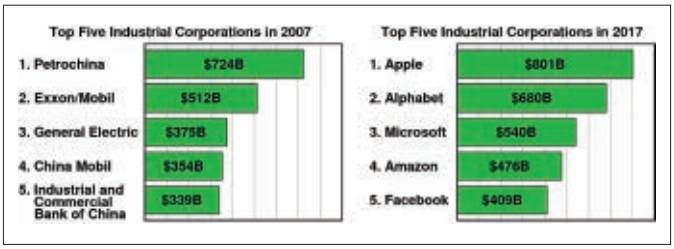


Figure 1: Top five industrial corporations by market valuation. (Financial Times Global 500 rankings)

Similarly, every survey of employment suggests that computing jobs dominate the employment needs in the science, technology, engineering, and mathematics (STEM) fields and will continue to do so for the foreseeable future. The charts in Figure 2 show the fraction of current employment, projected growth, and projected job openings for five broad categories of STEM fields:

As Steve Lohr reported in a recent *New York Times* column, there are more than enough graduates to fill existing jobs in most STEM disciplines. "Computing," he notes, "is the only exception." [4]

In recent years, more and more high-paying jobs require not just computer literacy but significant coding skills. Burning Glass Technologies, a consultancy that analyzes employment data in computing produced the graph in Figure 3, which shows that, even today, half the jobs in the highest-income quartile require coding skills.

Students today recognize the importance of computing education to their ability to succeed in a 21st-century economy and are therefore flocking to computer science courses at an unprecedented rate. I see no evidence to suggest that this heightened interest will subside at any point in the next fifty years. I therefore believe the following prediction is sound.

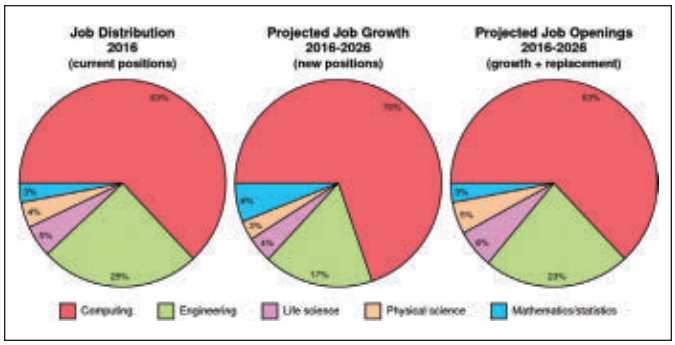


Figure 2: Employment growth in STEM disciplines. [1]

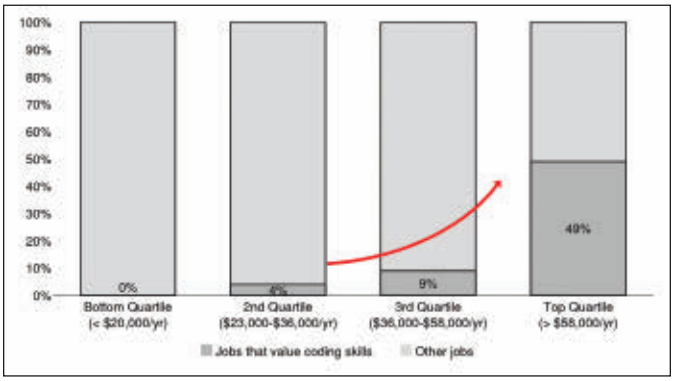


Figure 3: Fraction of jobs in each income quartile that required coding skills. [2]

Computing will continue to grow in importance relative to other fields, particularly as more disciplines become more computational. Students will be attracted to computing in growing numbers for the foreseeable future.

The recent report from the National Academies Committee on the Growth of Computer Science Undergraduate Enrollments [5] supports my prediction in the following findings.

FINDING 2: Enrollments in CS courses and the number of CS majors have risen markedly since 2005 at many institutions, and there is no indication that enrollments will fall in the near term. Both CS majors and non-majors have contributed significantly to the recent growth in enrollment in undergraduate CS courses. Information about current program enrollment trends suggests that the boom in enrollments has only begun to register in the data on CS degree production, and that CS bachelor's degree completions will rise sharply for at least the next few years

in the absence of institutional actions to limit or discourage participation in the major.

FINDING 5: Computing is pervasive, and its penetration is deep and growing in virtually all sectors of the economy, all academic disciplines, and all aspects of modern life. The broad opportunities in computing, both in the labor market and for enabling a host of intellectual pursuits, will continue to be drivers of increasing enrollments in undergraduate computer science, from both majors and non-majors. While there will probably be fluctuations in the demand for CS courses, demand is likely to continue to grow or remain high over the long term.

THE FUTURE OF COMPUTER SCIENCE EDUCATION

Assuming that student demand for computing education continues to grow, it seems natural to conclude that university programs in computer science will experience a similar rate of growth. Unfortunately, the evidence from history suggests that the expansion of university-level computer science programs is by no means assured. The importance of computing has grown steadily over time, as have the economic advantages that accrue to people with the necessary skills. Degree production, however, has been remarkably episodic, as shown in Figure 4, which graphs the production of U.S. bachelor's degrees in computer science since 1975.

This graph shows two periods of rapid increase in degree

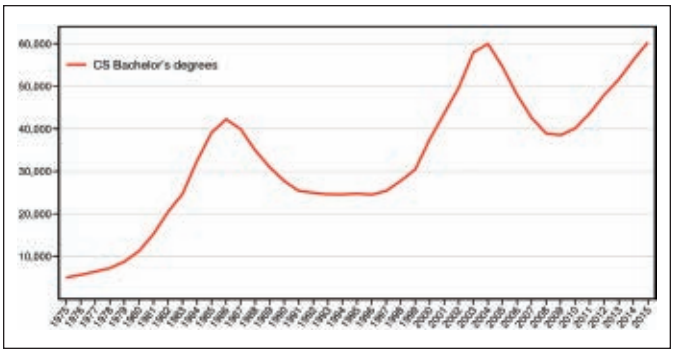


Figure 4: Historical data showing the number of bachelor's degrees in computer science. [5]

production followed by equally rapid declines. Since 2009, degree production has again followed a rapidly rising trajectory. Any attempt to predict the future of computer science education must surely be able to anticipate what will come next on this graph. Student interest and societal demand will both remain high. Do those observations suggest that we have seen an end to this cyclical pattern?

As many economists have written over time, the four most dangerous words in finance are "this time is different." Computer science programs at universities and colleges face much the same pressures that they have in past expansionary times. Consider, for example, the situation facing the PhD-granting institutions covered by the annual Taulbee survey shown in Figure 5.

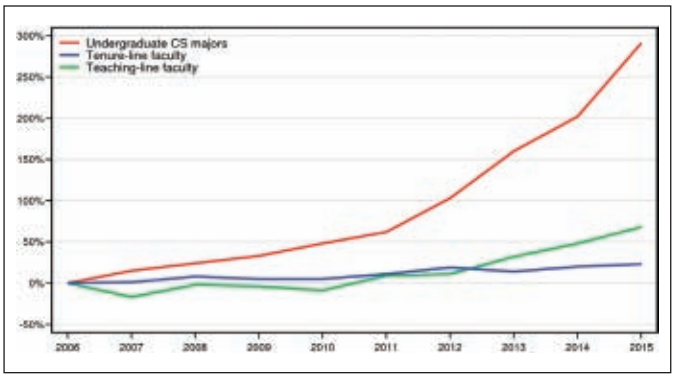


Figure 5: Relative growth rates of undergraduate majors and faculty lines at the Taulbee institutions. [3]

Since 2006, the number of undergraduate majors has nearly quadrupled while the number of tenure-line faculty has remained essentially flat. Recent years have shown some growth in teaching-line faculty, but at nothing like the rate student numbers have expanded.

Similar disparities between the number of students and the number of faculty members preceded each of the historical declines in CS degree production. Although the decline in the early 2000s was triggered in part by the dot-com collapse and a media-fueled fear of offshoring that had no basis in reality, there is no similar economic explanation that fits the decline in the early 1980s. Institutions lacked the capacity to serve the growing number of students and were forced, as suggested in Finding 2 from the National Academies report, to take "institutional actions to limit or discourage participation in the major."

As a longtime observer of this phenomenon who spent the early 1980s chairing a computer science department at a liberal arts college when it was simply impossible to find enough faculty, I believe that the following prediction is likely to characterize the next fifty years of computer science education.

Until the nation can produce a much larger community of computer science educators willing to forgo the economic advantages of industry and universities can support computer science at a level more closely approximating its growing importance, degree production in computer science and related fields will continue to endure boom-and-bust cycles that will leave many students without access to this vital body of knowledge.

Fortunately, the future need not repeat the past if we can learn from it. With a concerted effort on the part of universities, industry, and government, it will be possible to avoid the capacity problems of the past and ensure that all students can pursue their interest in our increasingly important field. ❖

References

1. Bureau of Labor Statistics. 2016-26 *Employment Projections and Occupational Outlook Handbook*. October 2017; http://www.bls.gov/emp/ep_table_107.htm. Accessed 2018 September 10.
2. Burning Glass Technologies. *Beyond Point and Click: The Expanding Demand for Coding Skills*. June 2016; http://www.burningglass.com/research-project/coding-skills/. Accessed 2018 September 10.
3. Computing Research Association. *Generation CS: Computer Science Enrollments Surge Since 2006*. February 2017; http://cra.org/data/Generation-CS/. Accessed 2018 September 10.
4. Lohr, Steve. Where the STEM jobs are (and where they aren't). *New York Times*, November 1, 2017; http://www.nytimes.com/2017/11/01/education/edlife/stem-jobs-industry-careers.html. Accessed 2018 September 10.
5. National Academies of Science, Engineering, and Medicine. Assessing and Responding to the Growth of Computer Science Enrollments. Prepublication report, October 2017; http://www.nap.edu/catalog/24926/. Accessed 2018 September 10.
6. Simon, Herbert A. and Newell, Allen. Heuristic problem solving: The next advance in operations research. *Operations Research*, 8, 1 (January-February 1958).

Eric S. Roberts
Department of Computer Science
Stanford University
Stanford, CA USA
eroberts@cs.stanford.edu

DOI: 10.1145/3284175

Computing Education Will Not Be One Size Fits All

Valerie Barr, *Mount Holyoke College*

Computing Education in the 21st century will have to better account for the many different educational areas and approaches that come under that term. It will also have to evolve to better ensure that those who develop technology applications in and for the developing world help to improve conditions, rather than increase inequities.

INTRODUCTION

Having been invited to opine on "The 5 Big Open Questions in Computing Science Education," in the context of looking toward the next 50 years of SIGCSE, my first step was to consider what I thought those questions might be. I address here two areas that I believe represent big open questions or issues.

1. There are currently many large curricular areas included under *computing science education*, but what will the field be called in the future, and does the name influence what is taught?
2. The large economic gap between the developed and developing world correlates with a large technology gap. Yet need exists in the developing world for increased applications of technology and increased access to technology, in general, and to the internet specifically. Should this potential be considered when structuring and changing computing education in both the developed and developing world?

WHAT'S IN A NAME?—PART 1

As I began to think about this article, I realized that I tend toward a very computer science (CS) centric perspective on computing education, even though ACM has long been involved in the development of curricula for computer science as well as curricula for information systems, information processing, and software engineering. These curricula have been updated periodically: 1968, 1978, and 1983 [6]; 1991 [16]; 2001 [1]; 2008 [2]; and 2013 [3]. Today we see new fields evolving with new program titles, such as information science and data science. Given these many fields, all of which involve computing, *computing science education* could serve as an umbrella term that covers all of them. It is also the case that, increasingly, those with a computer science degree are being asked to carry out work that lies at the intersection of disciplines and might, therefore, fall into the category now typically thought of as information science. Perhaps we will begin to see an increasing number of academic programs change their names to reflect a focus that is more on computing and applications of computing. In much of the discussion that follows, I have further shortened the *computing science education* umbrella term to *computing education* which, in my mind, is even broader as it can encompass the education that is necessary to tackle applications of computing across the widest possible range of areas. Beyond the name, there is the question of what will drive the content of computing education. But first it's interesting to look at where we have come from.

WHAT'S IN A NAME?—PART 2

Rather than wade into lengthy debate about what starting point to choose, I settled on 1948, the year the Computer History Museum cites [7] as when a program first ran on a computer—the Manchester Baby. This also gives us a nice round 70 years of history to contemplate. I divide this period into three large time blocks, each of which has helped shaped the content and development of computing education.

- 1948–1980: computer development up to the introduction of mass market personal computers (in the developed world).
- 1981–2003: the personal computer (PC) became ubiquitous in business and at home (in the developed world).
- 2003–present: social media and e-commerce have grown; the number and type of individual tech devices have increased (including the launch of Internet of Things (IoT) applications); and personal data is increasingly monetized (both data that identifies the user and data that is not directly tied to an individual user identity). An equally important trend has been the increased application of computing across numerous disciplines and industries.

What have been the concomitant developments in computing education? The first period required a focus on those aspects of the field that were necessary for us to be able to *do* computer science. CS education focused on the development of the discipline and the actual hardware (hence use of the term *computer science*). Curricula included classes on computer organization and assembly language, systems program-

ming, and compilers. As the underlying theory evolved, new data structures were designed, algorithmic analysis came of age, language paradigms and new languages were created, and courses were developed on these topics.

The PC era changed the way we teach computing because it became easier to provide equipment for students, we could have closed labs like other sciences, and students could do class work outside of formal academic spaces. Meanwhile, the curriculum continued to evolve. Hardware improvements and new applications supported interest in topics such as computer graphics, human computer interaction, and artificial intelligence, while courses on software engineering and algorithmic analysis became more common at the undergraduate level. This period also saw an increase in discussions about computer literacy and information literacy, along with a seemingly perennial debate about the role of computer science departments in post-secondary institutions, particularly in liberal arts colleges.

It would be helpful if the 21st century saw the various disciplinary communities coalesce around a name that serves as an effective and flexible umbrella over these many perspectives on computing, one that points effectively toward the underlying disciplinary (and, increasingly, interdisciplinary) content.

In the current period the demand for computing education has increased significantly and it is no longer only at the undergraduate and graduate level. The ubiquity of computing across application areas means that many people feel the need to know something about computing. Meanwhile, personal and IoT devices require more widespread knowledge about hardware, embedded computing, and security. These are no longer niche areas. We also must reconsider ethical concerns considering the deluge of data that is available and the issues that arise when designing and building devices that reside in people's homes and pockets [9,13].

But wait a moment! Everything I have written above is a very Western developed-world view of computing education. In the history laid out above, computing education evolved at the undergraduate and graduate levels, and is now trickling down into elementary and secondary education. But when we consider the rest of the world, with an eye toward the future, there is much more that should shape computing education.

WHAT'S IN A NAME?—PART 3

There's a very important term missing from the disciplines listed above, namely ICT (Information and Communication Technologies). ICT is a term that has been adopted globally and is used much the way that "IT" (information technology) is used in the U.S. ICT has proven to be a very flexible umbrella term, adjusting over time to include television, fixed-line phones, computers, radios, laptops, mobile phones, satellites, routers, etc. ICT, as a term and in the understanding that users have of it, tends to encompass both hardware and software, all under the guise of usage and information flow [20]. This term avoids the possible narrowness of *computing science*, and avoids the potential machine focus of *computer science*. On the other hand, there is nothing inherent in the term that makes it clear that computing is involved at all, which can make it difficult to talk about what we are educating people to do, what we are educating them about. It would be helpful if the 21st century saw the various disciplinary communities coalesce around a name that serves as an effective and flexible umbrella over these many perspectives on computing, one that points effectively toward the underlying disciplinary (and, increasingly, interdisciplinary) content.

IT'S A BIG WORLD OUT THERE!

As argued previously, much of computing education to date has been driven by both the historical development of the field and the recent development of personal uses of technology and interdisciplinary applications of computing. During the last 15 years much of the computing industry in the developed world has focused on monetizing the user. This is done through the development of many consumer products which presumably improve our lives (talk to your TV remote! let your thermostat learn your habits so that you never have to set it! let your refrigerator tell you what to buy at the store! let your appliances directly order products!). Monetizing is also done through analysis of user data, both at the individual user level and in aggregate, while we are left with little control over personal data once it heads out to commercial entities (see the work of Debra Estrin [8]). As technologically interesting as these developments might be, they are fundamentally intended to make more comfortable lives that are already fairly comfortable. Yet for every new application that we presumably cannot live without in the developed world, there is an article or book that describes the negative effects on social interaction [18], the stunted development of empathy and communication skills among today's children [17], the need for digital detox [5], and the potential uses of the vast quantity of data we are generating [12].

What about the rest of the world? There is clear evidence [15,20] that a facile uncritical application of certain technologies in developing world contexts will be unsuccessful. In some cases, such as the introduction of computers into el-

ementary classrooms in India, a project is unsuccessful because the local infrastructure and local conditions (political will, lack of trained personnel) cannot support the technology [15]. In other cases, the introduction of technology serves as a proxy for and a way to avoid or postpone real systemic change. Unwin [20] makes a compelling argument that, despite the potential to improve the quality of life, increased use of technology in developing nations often results in an increase in poverty and inequality. Those who are the most marginalized face the greatest impediments, resulting in skewed demographics, such as the fact that only 29% of internet users in India are women and there are areas of the country in which girls are prohibited from using mobile phones and social media [19]. Furthermore, 60% of African youth are not online, compared with just 4% of young people in Europe, and overall 1.7 billion females in low- and middle-income countries do not own mobile phones [10].

A central question to consider, and one that could be mind-twisting within computing education, is the question of what comes first. Is technology the driver for development, or should development be driving the uses of technology?

What does this mean for computing education? Entire books have been written about ICT for development (ICT4D), and degree programs have been launched. In this short treatise, one cannot do justice to the full topic of ICT4D. It is possible, however, to briefly consider the ways in which computing education can respond to ICT4D, both for the developers of tomorrow and those who will inform the developers. For too long those who develop computing technologies and those who teach future developers have seemed to hold the attitude that the eventual use of and impact of technologies are of minimal concern. The 21st century calls for a deeper understanding in the technology world of the economics and politics of international development and the role of technological change therein. It calls for technologists who can work collaboratively *with* those in government and public service, *with* those who understand developing world infrastructure, *with* those who understand the day-to-day realities of poverty and marginalization (including in the developed world where there are still impoverished and marginalized communities). It calls for technologists who are willing to recognize that we must start by examining human problems in conjunction with those who live in the developing world and determine in collaboration with them whether there are appropriate technical solutions. We should not start with a technology and the presumption that it can be made to fit a specific human situation. Technologists also must be willing to address fundamental questions, such as whether internet access should be treated as a public utility rather than being privately controlled and profit generating. This is not about making people in computing feel bad if they are not on some moral high ground. Rather it is about ensuring that we understand enough about *all* possible end users and, therefore, are capable of building technologies that are useful, relevant, and empowering, applying them in appropriate and useful ways, not deciding what ought to be useful based on the narrow set of experiences of a small sector of possible end users.

There are some developed world efforts focused on computing and "social good" (see, for example, [14] which discusses the CS + Social Good student group at Stanford). Many of these focus on ethical implications of computing, but do not fundamentally address issues of inequality. The ACM Code of Ethics [4] discusses promoting fundamental human rights and minimizing negative consequences of computing systems. It also states that "When the interests of multiple groups conflict, the needs of those less advantaged should be given increased attention and priority." This is important, and all computing students should study the ACM document. But computing education should also explicitly teach future developers that a potential risk of technology use is the further marginalization of people who are already impoverished. Those of us in computing may not be the right people to teach these lessons, but we can ensure that these lessons are taught by others. We must also stress the software engineering principle that end users must be consulted, that requirements analysis must be carried out in a way that involves *all* intended end user groups. Future developers must be prepared to consider factors that may not be relevant when thinking only about a developed world marketplace. A central question to consider, and one that could be mind-twisting within computing education, is the question of what comes first. Is technology the driver for development, or should development be driving the uses of technology [20]? Identifying what will make the most immediate and sustainable improvement in the lives of the most impoverished people should influence the decisions of what technology to put in place and where to do so. The computing education community must make sure that we prepare computing students to be productive contributors to that decision-making process.

There are a few current initiatives that may provide helpful starting points. One is the Harvard Humanitarian Initiative [11] which includes efforts to ensure "that new technologies make their way to the front lines of serving communities at risk by supporting evidence-based strategies to effective humanitarian response." Another interesting effort is Ethical OS [9] which presents "a guide to anticipating the future impact of today's technology." They ask developers to consider who will have access to tech-

nology, what differences can develop between those with access and those without access, and who will have access to collected data. While they are not encouraging people to drive technological decisions based on developmental needs, they at least make explicit those areas that, if disregarded, can lead to unintended negative consequences from new technology development.

Computing education also must provide a suitable knowledge base for those in the developing world who are the local experts, who should be equipped to be part of the conversations about technological needs and implementation, that is, the conversations about appropriate uses of technology to solve local problems. This is a complex area since local needs vary tremendously, technology deployment is dependent on the state of and improvements in necessary underlying infrastructure, and education efforts need to align with the specific types of technology employed. One thing is certain—we cannot view computing education in the developing world only through the western model of post-secondary computing education that then trickles down into elementary and pre-secondary schooling. Computing education must be a rich mix that is geared locally toward many constituencies, including end-users, policy makers, and future developers.

Less certain to take place, but potentially more important, are changes in computing education that will ultimately help developers work collaboratively with all sorts of end-users, including those in the developing world, to contribute to efforts that effectively decrease inequality and improve quality of life.

CONCLUSION

The next 50 years will see a large focus on the use of computing to solve problems in a range of fields. It is likely that the very name of what we call our educational endeavors will change to better reflect this interdisciplinary role of our field. Less certain to take place, but potentially more important, are changes in computing education that will ultimately help developers work collaboratively with all sorts of end-users, including those in the developing world, to contribute to efforts that effectively decrease inequality and improve quality of life. ❖

Acknowledgements

More than 20 years ago, Anita Borg first presented to me the idea that people are experts in their own lives, and technology can be used to solve problems in people's lives only when developers talk with those experts. This is as true today as it was then. Significant thanks to Revi Sterling for feedback on a preliminary draft and for many discussions over the years, particularly one in late 2017 that helped me crystallize my thoughts about ICT and computing education. I also thank the participants in the December, 2017, meeting of the UN ITU EQUALS Project (https://www.equals.org/) research group, particularly Tim Unwin, for the speediest ever introduction to the rich area of ICT4D (ICT for Development). They also underscored the lesson learned from Anita so many years earlier, that those of us in computing should never presume that we can dictate what is needed by others.

References

1. 2001. Computing Curricula 2001; https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2001.pdf; Accessed 2018 August 4.
2. 2008. Computer Science Curriculum 2008: An Interim Revision of CS 2001; https://www.acm.org/binaries/content/assets/education/curricula-recommendations/computerscience2008.pdf. Accessed 2018 August 4.
3. 2013. Computer Science Curricula 2013; https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf. Accessed 2018 August 4.
4. ACM Code of Ethics; https://www.acm.org/code-of-ethics. Accessed 2018 August 4.
5. Acton, Annabel. Why You Need a Digital Detox. Forbes, October 19, 2017; https://www.forbes.com/sites/annabelacton/2017/10/19/why-you-need-a-digital-detox/#177bcab57b7c. Accessed 2018 February 17.
6. Atchison, W.F., Conte, S.D., Hamblen, J.W., Hull, T. E., Keenan, T. A., Kehl, W.B., McCluskey, E.J., Navarro, S.O., Rheinboldt, W.C., Schweppe, E.J., Viavant, W., and Young, Jr., D.M. *ACM Curricula Recommendations for Computer Science. Technical Report*. (ACM, New York, NY, 1983).
7. Computer History Museum. Timeline of Computer History; http://www.computerhistory.org/timeline/computers/. Accessed 2018 February 17.
8. Estrin, Debra. http://destrin.smalldata.io/. Accessed 2018 February 17.
9. Ethical OS; https://ethicalos.org/. Accessed 2018 August 12.
10. GSMA Connected Women Global Development Alliance; https://www.gsma.com/mobilefordevelopment/connected-women/ . Accessed 2018 August 29.
11. Harvard Humanitarian Initiative; https://hhi.harvard.edu. 2018 August 12.
12. Hill, K. and Mattu, S. The House That Spied On Me. *Gizmodo*; https://gizmodo.com/the-house-that-spied-on-me-1822429852. Accessed 2018 February 17.
13. Singer, N. Tech's Ethical 'Dark Side': Harvard, Stanford and Others Want to Address It. *New York Times*, Feb 12, 2018.
14. Stolzoff, S. Are Universities Training Socially Minded Programmers? *The Atlantic*, July 24, 2018.
15. Toyama, K. *Geek Heresy: Rescuing Social Change from the Cult of Technology*. (PublicAffairs, 2015).
16. Tucker, A.B., Aiken, R.M., Barker, K., Bruce, K.B., Cain, T.J., Conry, S. E., Engel, G L., Epstein, R.G., Lidtke, D.K., Mulder, M.C., Rogers, J.B., Spafford, E.H., Turner, A.J.. Computing Curricula 1991: Report of the ACM/IEEE-CS Joint Curriculum Task Force. Technical Report. (ACM, New York, NY, 1991).
17. Turkle, S. *Alone Together: Why We Expect More from Technology and Less from Each Other*. (Basic Books, 2011).
18. Turkle, S. *Reclaiming Conversation: The Power of Talk in a Digital Age*. (Penguin Press, 2015).
19. UNICEF. *The State of the World's Children 2017: Children in a Digital World*; https://www.unicef.org/sowc2017/. Accessed 2018 August 16.
20. Unwin, T. *Reclaiming Information and Communication Technologies for Development*. (Oxford University Press, 2017).

Valerie Barr
Department of Computer Science
Mount Holyoke College
50 College Street
South Hadley, MA 01075
vbarr@mtholyoke.edu

DOI: 10.1145/3276305

Five Big Open Questions in Computing Education

Kim B. Bruce, *Pomona College*

Certainly, there is no shortage of problems in computer science education. The difficulty is narrowing them down to just a few of the most important. Before I lay out those I find most significant and challenging, let me explain where I'm coming from. First, and most importantly, my research specialty is the theory and design of programming languages, rather than computer science education, though I do have a deep commitment to computer science education. My contributions have mainly been in the areas of curricular design and the design of introductory courses in computer science. This background should help put in context the concerns I highlight here.

1. HOW CAN WE DEAL WITH SKYROCKETING ENROLLMENT IN CS UNDERGRADUATE COURSES?

Certainly, the most pressing problem for the field now is how to deal with the overwhelming increases in enrollments in computer science. While the degree of change differs between institutions, nearly all are seeing increases, with some seeing the number of CS majors increasing by as much as factors of 5 to 10. While it's not completely clear what the cause is—residual effects of the great recession, an understanding of the centrality of computing to many parts of society, an increase in students from lower-income families, or second-generation immigrants who are more occupationally focused—the effects are plain to see. Virtually every institution is struggling to offer more sections, capping classes, increasing the size of classes, or all three.

The possible solutions for colleges and universities fall into three broad categories—increase the size of the faculty, cap the number of students, or increase the ratio of students to faculty. Others have written eloquently about this problem and I refer you to their analyses [3,7,8]. Thus, I'll limit myself to a few brief remarks. Expanding faculty is hard, slow, and will take serious incentives to attract more people to the profession. Capping enrollments frustrates everyone and goes against the huge demand for CS education. Students learn less effectively in large classes. Modern techniques like peer instruction can help as classes become even larger but can't make up for the difference between large and small classes.

Finally, alternative ways of educating students (e.g., MOOCs and code academies) are unlikely to have a major impact on the problem. MOOCs seem most effective with college graduates who are highly motivated to learn specific skills. Code academies vary widely in quality [8], but most aim at teaching narrow skills with current technology, rather than the depth of knowledge typically obtained with a degree in computer science. Similarly, asking faculty from other departments to teach CS (as they often do with statistics) generally causes more articulation problems than it helps.

It appears there are no simple answers, but this is a crisis we must address.

2. HOW CAN WE MAKE CS COURSES MORE INCLUSIVE?

Look around at nearly any computer science department and you will immediately notice that the students are mainly White and Asian males. While most CS departments do not actively discourage women and under-represented minorities, there are actions that can be taken to make underrepresented groups more successful. For example, research indicates that those who are less confident in their abilities in an area (no matter what their actual ability level) are more likely to give up in frustration and quit, especially when society views them as less capable [14].

Providing support at all stages of education can make a significant difference in retaining those capable of succeeding, but lacking confidence. Having an instructor who knows the student and can provide encouragement and support where needed can change a student's trajectory from failure to success. Sadly, the huge increase in enrollments discussed earlier seems to be causing a reversal of some of the improvements we had been seeing in CS classes.

There has been a lot of work in recent years to make computer science more welcoming of women, with some success at institutions like Carnegie-Mellon University and Harvey Mudd College. Yet, with the percentage of female CS majors under 20% across the country, there is clearly much room for improvement. Departments that have succeeded have had more control over admissions than most departments and have devoted a great deal of energy to make women feel welcome and an important part of the department.

Similar work has been undertaken with members of under-represented minorities, first-generation college students, and others, especially with learning communities. An extra challenge is that statistics show that fewer Black students are exposed to computer science before college than White or Hispanic students [4].

While further research is surely needed to find out more about how to help all students succeed, it is important that we all better understand the existing research and carry out best practices to provide all groups with equal opportunity.

How do we train and support the teachers who will be teaching this pre-college level material?

3. HOW CAN WE DEVELOP AND SUPPORT TEACHERS FOR PRE-COLLEGE INSTRUCTION IN COMPUTING?

There have been big pushes in the US and elsewhere to integrate computing education into pre-college or university curricula. An example in the US at the high end is the relatively new CS Principles advanced placement exam in the U.S., designed for junior and senior high school students. That course has undergone extensive development, with several variants, but seems clearly designed to introduce students to the ideas behind computing and computational thinking, teaching some programming, but not having that be the focus of the course. The US government also announced a CS for All project in 2016 with a stated goal of "offering every student the hands-on computer science and math classes that make them job-ready on day one."

England introduced revised computing material into their school curricula in 2014 after several years of preparation. Coverage of computing material is now mandatory for students aged 5 to 16. The 2017 report [13] addresses the successes and disappointments in this program. The report indicates that a major problem is finding and/or training enough teachers to present this material.

How do we train and support the teachers who will be teaching this pre-college level material? There will likely be few CS majors who will be interested in teaching computing in the schools, so most of the teachers will need to be trained with the appropriate skills and will need resources (both master teachers on call and materials) to succeed.

4. HOW CAN WE GET STUDENTS TO SERIOUSLY ADDRESS THE ETHICAL IMPLICATIONS OF COMPUTING?

It's hard to imagine anyone in this era believing that computing is value-free. Computing can be used for great good—and it can be used for evil—and, perhaps even worse, it can inadvertently have grave unintended consequences for society. A few negative examples in the news lately include cyber-hacking, privacy violations, election tampering, addiction, the increasing gap between the haves and have-nots, and machine learning perpetuating prejudices.

Starting with the 1978 ACM Curriculum recommendations, computing curricula have specified coverage of ethical issues in computing. The 1978 recommendations specify discussion of social, philosophical, and ethical considerations of applications in the data structures course, while in the Computing Curricula 1991 there was a specification of 11 hours of coverage of social, ethical and professional issues. Later curricula recommendations as well as ABET accreditation continued to specify similar amounts of coverage of these issues. We don't know how many departments' curricula satisfied these recommendations, as they could be spread over many courses and it's hard to verify if faculty included these discussions. Certain electives like AI and security are more likely to at least give lip service to these concerns, though we suspect not much notice was paid in introductory or even core courses.

Given the issues mentioned above, it has certainly become clear that we as the society at large—and even more, we as computer scientists—need to pay attention to these issues. A recent article in the *New York Times* [12] describes new ethics courses in CS departments at Harvard, MIT, Stanford, University of Texas, and Cornell.

Of course, it is not clear this is just a computer science problem. This is a general issue with the impact of technology on society. It's also not clear that computer scientists are the best people to teach such courses, as most of us have no training in these issues, and it may be more effective to have a mixture of CS and non-CS students in such a course. Instead it might be more effective to work with local philosophers and others to develop a course on ethics and the social impact of technology. It might be even more effective to have such a course co-taught by a computer scientist and a specialist in ethics.

That leaves the problem of getting students interested in such a course. Until recently, that seemed to be the last thing on the minds of our students. Hopefully these new developments will wake all of us up to the possible dangers if we don't think through the consequences of our development and use of technology.

5. HOW CAN WE ALLEVIATE THE HIGH DROPOUT RATE IN INTRODUCTORY CS COURSES?

It seems strange to be lamenting the high dropout rate in computing at the same time we are suffering from over-enrollment in CS courses. However, large and increasing numbers of university students need at least some exposure to computing ideas and skills, and we should be providing them with that experience—even if we don't want them all to be majors!

Only two groups seem to have gathered much data on pass rates in introductory programming classes [1,16]. While the

data is a bit sparse, they estimated a pass rate of approximately 67%, with an average pass rate of 80% for small classes compared to 65% for large classes. However, there were wild variations between institutions—including at least one large university with a failure rate of 72%. While we do not have comparable data for introductory courses in other disciplines, it seems that the failure rate in introductory courses in CS is higher than most other disciplines.

How can we increase the success rate in computer science courses? Small classes seem to make a significant difference, but in the current over-enrollment crisis, that may not be obtainable without blocking many students from CS courses. Modern teaching techniques like peer instruction will not overcome all the problems with large courses but can allow classes to increase from 100 to 400 or more with little additional degradation of quality [5].

My own approach to increase success rates in introductory courses has been to increase student interest by making it possible for them to write more interesting programs without being overwhelmed by inessential complexity of tools. For example, early on my colleagues and I found the value of programs in the introductory course that used graphics and animation. When we shifted our introductory course to Java, we found that the standard Java graphics were too complex for novices, so we (along with many other groups) developed a custom graphics library for our classes. This library was designed to cut the programming overhead for students, and, with the visual feedback, was very effective in helping students find errors in their code. We later found out that faculty teaching undergraduate courses nearly uniformly resisted using non-standard Java classes (e.g., for graphics), and even resisted an officially ACM SIGCSE-endorsed Java library [11], regardless of whether it helped students learn to program more effectively.

I've always been firmly of the opinion that our goal as instructors is to figure out what we want to teach our students and to find the most effective way to help them learn it. Sometimes that will be using industry standard methods and tools (languages, ide's, debuggers, etc.), but more often tools developed for professionals who are already expert are not the right tools for novices.

While it's clear to most that providing graduates with useful skills relevant to the job market is a positive thing, it is less clear that this is necessary for students just starting out at the university. Long ago, we taught introductory courses in languages like BASIC and Pascal that were designed for educating novices. Each of those was so successful that they were scaled up to professional languages that were used for important systems (e.g., Object Pascal for the original Macintosh operating system). Unfortunately (in my opinion), the pull of industry compatibility overcame those benefits and many introductory courses moved from Pascal to C in the mid-80'sand then to C++, trading simple, relatively clear syntax and semantics for languages designed for low-level systems programming. The educational language Blue [6] was introduced in the mid-90's but was swept away by the enthusiasm for the new language Java, which was certainly simpler than C++, but, as an industrial language, had its own complexities—which have gotten worse as the language has evolved. While the language Blue was abandoned, the related BlueJ [7] programming environment for Java has been quite successful as a tool for novices.

Long ago, we taught introductory courses in languages like BASIC and Pascal that were designed for educating novices. Each of those was so successful that they were scaled up to professional languages that were used for important systems (e.g., Object Pascal for the original Macintosh operating system).

In pre-college education, visual (typically blocks-based) languages have been widely adopted for younger children and seem to engender great enthusiasm among those learning to program. Typical examples of such languages are Alice and Scratch. The entertaining scenarios and the ability to program without having typing skills combines to help make this a big success, especially for younger children. Moreover, the shapes of the program components make it nearly impossible to make syntactic mistakes. However, few colleges and universities use visual languages for more than a few weeks before moving on to industrial-strength languages.

There are still some colleges and universities that start students out programming in a complex language like C++ using only a text editor (e.g., emacs or vim) and the command line. While the more motivated students can succeed with that kind of an approach, it's my opinion that if we wish to help more students succeed in the introductory course (especially those from underrepresented groups), we need to provide a more supportive environment. There are many ways to do this. Tutors and support groups can help, but tools that are designed to help novices can make a difference for those who are uncertain of their abilities and easily scared away. The tool BlueJ, mentioned previously, is such a tool for helping novices learn to program in Java. It provides a view of programs and objects that are most useful to a novice. When students enter more advanced courses they can learn to use more professionally

oriented IDE's or even fall back to vanilla text editors and the command line.

As noted earlier, in the 70's and 80's introductory classes were taught in languages designed for that purpose. Why not return to that practice? After all, programming languages now are typically even more complicated than the industri-

Don't we, as educators, have an obligation to investigate (and develop as necessary) the tools that will help our students learn the key concepts of our discipline? Limiting ourselves from the very first day to tools that are in use by highly trained professionals might prevent us from discovering new and better ways of helping our students learn.

al-strength languages of that era. Several of my research colleagues and I found this a compelling argument several years back and were inspired to work on designing and implementing a language, Grace (see http://web.cecs.pdx.edu/~grace/doc/), designed for teaching novices the object-oriented style of programming.

While we have been very happy with the results, we were very dismayed that few would even consider using a language designed for teaching. It wasn't that they looked at Grace and decided it didn't meet their needs. Instead, they wouldn't consider teaching in a language that wasn't in wide usage in industry. Have things changed so much since the 70's and 80's that we should no longer teach using a language designed for novices?

Yes, our students will eventually benefit from learning tools (whether IDE's, languages, or others) that are in common use in industry. However, that does not have to happen during the first semester of their degree programs. Don't we, as educators, have an obligation to investigate (and develop as necessary) the tools that will help our students learn the key concepts of our discipline? Limiting ourselves from the very first day to tools that are in use by highly trained professionals might prevent us from discovering new and better ways of helping our students learn. ❖

Acknowledgements

I'd like to thank my friends and colleagues who responded generously to my request for their views on the important problems in CS education—Andrea Danyluk, Scot Drysdale, Michael Kölling, and Kathy Fisler.

References

1. Bennedsen, J., and Caspersen, M.E., Failure rates in introductory programming. *SIGCSE Bull.* 39, 2 (2007), 32–36. DOI=http://dx.doi.org/10.1145/1272848.1272879.
2. Boyer, K.E., et al., A case for smaller class size with integrated lab for introductory computer science. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education* (SIGCSE '07), 341–345. DOI: https://doi.org/10.1145/1227310.1227430.
3. Computing Research Association. *Generation CS: Computer Science Undergraduate Enrollments Surge Since 2006.* (2017); https://cra.org/data/Generation-CS/. Accessed 2018 Aug 11.
4. Gallup Poll, U.S. Minority Students Less Exposed to Computer Science, 10/18/2016; http://news.gallup.com/poll/196307/minority-students-less-exposed-computer-science.aspx. Accessed 2018 Aug 11.
5. Liao, S.N., Griswold, W. G., and Porter, Leo. Impact of Class Size on Student Evaluations for Traditional and Peer Instruction Classrooms. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*, 375–380. DOI: https://doi.org/10.1145/3017680.3017764.
6. Kölling, Michael and Rosenberg, John. Blue—a language for teaching object-oriented programming. In *Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education (SIGCSE '96)*, Karl J. Klee (Ed.). (ACM, New York, NY, 1996), 190–194. DOI: https://doi.org/10.1145/236452.236537.
7. Michael Kölling and John Rosenberg. Guidelines for teaching object orientation with Java. In *Proceedings of the 6th annual conference on Innovation and technology in computer science education (ITiCSE '01)*. (ACM, New York, NY, 2001), 33–36. DOI=http://dx.doi.org/10.1145/377435.377461.
8. McBride, S., Want a Job in Silicon Valley? Keep Away from Coding Schools, *BloombergTechnology*, 12/6/2016; https://www.bloomberg.com/news/features/2016-12-06/want-a-job-in-silicon-valley-keep-away-from-coding-schools. Accessed 2018 Aug 11.
9. Nager, A., and Atkinson, R. D. *The Case for Improving U.S. Computer Science Education*, May 2016; http://www2.itif.org/2016-computer-science-education.pdf. Accessed 2018 Aug 11.
10. National Academies of Sciences, Engineering, and Medicine. 2017. *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments.* (The National Academies Press, Washington, DC, 2017). https://doi.org/10.17226/24926.
11. Roberts, E., et al. The ACM java task force: final report. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education (SIGCSE '06)*. (2006), 131–132. DOI=http://dx.doi.org/10.1145/1121341.1121384
12. Singer, N., Tech's Ethical 'Dark Side': Harvard, Stanford and Others Want to Address It, *New York Times*, Feb 13, 2018; https://www.nytimes.com/2018/02/12/business/computer-science-ethics-courses.html. Accessed 2018 Aug 11.
13. Royal Society, *After the Reboot – Computing Education in UK Schools*, 2017; https://royalsociety.org/-/media/policy/projects/computing-education/computing-education-report.pdf . Accessed 2018 Aug 11.
14. Claude M Steele, *Whistling Vivaldi: How Stereotypes Affect Us and What We Can Do*. (W. W. Norton & Co., 2011).
15. Smith, M., *Computer Science for All*; https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all. Accessed 2018 Aug 11.
16. Watson, C. and Li, F.W.B. Failure rates in introductory programming revisited. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (ITiCSE '14), 39–44. DOI: http://dx.doi.org/10.1145/2591708.2591749.

Kim B. Bruce
Pomona College - Computer Science
333 N. College Way
Claremont, California USA 91711
kim@cs.pomona.edu

DOI: 10.1145/3230697

Building a Creative, Computationally-Competent Future

Allyson Kennedy, *AAAS science and Technology Policy Fellow at the National Science Foundation* and Janice Cuny, *National Science Foundation*

> **"I don't just want my students to be ready for the 21st century, I want them to create it."**
>
> ***—Jordan Budisantoso,***
> High school Exploring Computer Science and CS Principles teacher

Computational skills and competencies are becoming imperative in our increasingly digital world. While the past ten years have seen considerable progress in terms of access and participation in computing education, the challenges that remain must be addressed to achieve true equity for all students.

FRAMING THE MOST PRESSING ISSUES IN COMPUTER SCIENCE EDUCATION

The landscape for computer science (CS) education is rapidly changing. Just a decade ago, rigorous CS courses were only offered as electives, if they were available at all, and only nine states conferred math or science graduation credit on them. Even fewer states had reasonable CS standards or certification pathways for teachers. The students who did take CS, were overwhelmingly white and Asian males—the only Advanced Placement® (AP) CS course at that time, CS-A, had by far, the worst gender imbalance of any AP classes.

Much has changed over that decade. Today, 29 states have adopted K-12 CS standards, 28 now have CS teacher certification pathways, eight require all high schools to offer at least one academic CS course, and 16 allow CS to satisfy math or science graduation requirements [2].

These changes are in response to the growing recognition that computing skills and competencies are increasingly fundamental to a wide array of disciplines, careers, and student interests. Most—if not all—of today's students will need,

> ... the ability to make digital technology do whatever, within the possible, one wants it to do—to bend digital technology to one's needs, purposes, and will, just as [we bend] ... words and images. [7]

Figure 1: Elementary school students learn to code through a hands-on Time4CS lesson in Broward County, FL.

Whether they will become software engineers, scientists or educators, architects or engineers, journalists or historians, musicians or artists, students will need to be computationally savvy. Not only will they need to understand the basic concepts of computation and their applications in problem solving, but they will also need to comprehend the social and ethical

implications of computing, and the basics of cybersecurity. Moreover, it is essential that we as educators show students the breadth of computing and its relevance to their lives. By illustrating the potential for computing to transform their world (Figure 1), we will give students the opportunity to experience the "passion, beauty, joy and awe of computing." [1]

The changes in the field over the past ten years have been swift, creating notable progress while also simultaneously exposing glaring gaps in access. Here, we briefly cover five open questions centered around the need to ensure equitable access and participation in rigorous, engaging, and even inspiring, computing education.

QUESTION 1: CULTURE CHANGE IS HARD. HOW CAN WE CHANGE THE STILL PERSISTENT NARRATIVE ABOUT WHO SHOULD AND WHO SHOULD NOT STUDY COMPUTER SCIENCE?

Despite all the attention focused on CSforAll, many students are still constrained by obsolete notions about who does and does not belong in computing classes. Earlier this summer, teachers of the new AP Computer Science Principles (CSP) course in three different states reported (via private communication with author):

- "At the girl's high school where I teach, only students with a 95% or above in math are allowed to take CS,"
- "A Guidance Counselor walked into my CSP class and commented, 'These aren't AP kids. They shouldn't be in here,' and
- "School officials and the parents of a blind child all argued that she should not be allowed to enroll in my CS class despite the fact that the student and I were both enthusiastic about her participation."

It is time to discard the narrative that CS is only for a select few. Girls, students with disabilities, and students from minority groups far too often receive the subtle (and sometimes not so subtle) message that computing is not for them. They see it in news coverage of the appalling lack of diversity in technology companies and the "pale male," nerdy images of computer scientists in popular media. They hear it from school officials, guidance counselors, teachers, peers, and sometimes parents.

Figure 2: Computer Scientists Marvin Andujar (University of South Florida) and Chris Crawford (University of Alabama) demonstrate brain-computer interfaces at the Washington Leadership Academy (Washington DC).

This narrative is, of course, changing (Figure 2). Code.org, and its extraordinarily successful Hour of Code, has shifted the perception of who is capable of computing. Similarly, there has been an explosion of programs attempting to reach underrepresented groups along the education pipeline. For example, NCWIT's Aspire IT program recognizes and supports the IT accomplishments of girls, TECHNOLOchicas showcases Latinas in technology, and Tapestry supports classroom teachers in the recruitment and retention of girls. National organizations, like the Boys and Girls Clubs, 4H, Code Interactive, Code2040, Black Girls Who Code, and Code Stars have also adopted technology-related activities in the after-school space. Finally, major foundations and corporations have begun targeted efforts, such as Google's pilot programs in underserved areas of Harlem and Oakland.

Despite the progress, much more needs to be done. If we are to truly change the narrative for students, we must also change the reality of CS classrooms.

QUESTION 2: HOW DO WE ENSURE THAT CS CLASSROOMS ARE INCLUSIVE AND SUPPORTIVE OF DIVERSITY?

Inclusion and equity need to pervade CS instruction. Embedding these into the classroom should be intentionally designed into curricula, pedagogy, classroom environment, and teacher professional development (PD) from the very beginning.

The Exploring Computer Science (ECS) course and many of the new AP CSP courses have done just that—and the results are impressive:

- Of the more than 4,000 students who took ECS as an elective in Los Angeles last year, 49% were female and 84% were either Latino or African American [Jane Margolis, Private communication]; and
- After CSP joined CS-A as an official AP course last year, the number of women, Latinos, and African Americans taking AP CS exams more than doubled [8].

Of course, these numbers are not where we need them to be; the success of ECS in LA is not entirely reflected in its success nationally, and the overall AP numbers are still well below what we'd expect based on student population demographics.

While we have begun making inroads into creating inclusive CS learning environments, more is needed on both teacher training and classroom implementation to reproduce the successes of these programs and programs like them, at the national level.

QUESTION 3:
HOW DO WE PREPARE GREAT CS TEACHERS AND DO IT IN WAYS THAT ARE SCALABLE AND SUSTAINABLE?

Most current preK-12 teachers do not have formal training in CS. To get CS established in all schools, we will need to prepare an unprecedented number of teachers—there are over 3 million elementary and secondary U.S. public schools [5]! Furthermore, we will need to ensure that when the current wave of enthusiasm and targeted funding from both federal and private sources subsides, schools and school districts are ready to continue the work themselves. Preparation of in-service high school instructors to teach the ECS and CSP courses will be key to establishing a network of trained CS teachers.

For many, this is a tall order. The successful implementation of these curricula requires that teachers not only have content knowledge in CS and programming skills, but also the ability to create an inclusive classroom. This entails promoting the growth mindset and exposing students to a wide range of computing applications as well as hands-on, project- and inquiry-based, and differentiated instruction. Collaborative projects in culturally relevant topic areas of the students' choosing are also key in the recruitment and retention of a diverse generation of computer scientists.

Figure 3: Exploring Computer Science professional development workshop held at Tuskegee University, AL.

There are currently a number of PD projects that aim to provide such training. The National Science Foundation (NSF), for example, has funded projects for both the ECS and CSP curricula (including endorsed CSP courses: Beauty and Joy of Computing, Mobile CSP, UTeachCS, and CSMatters). In addition, private organizations such as Code.org, the National Math and Science Initiative (NMSI), Project Lead the Way (PLTW), and the Infosys Foundation USA have also developed PD resources for their ECS and CSP curricula.

At the preK-8 level, PD for these courses is accomplished through one to two-day workshops where teachers learn how to incorporate CS content into their regular classroom activities (Figure 3). Preparation of high school teachers, on the other hand, involves one full week of face-to-face training and several follow-up sessions over the course of the subsequent year. This is often much more extensive, and much more expensive, than most school districts can cover on their own.

In the long term, districts will have to develop their own sustainable mechanisms for training CS teachers, whether that be via train-the-trainer, Master Teacher, Professional Learning Communities, online PD, or some other model. The burden, however, will be significantly lessened once Schools of Education begin offering pre-service and Masters' degree programs in CS education. This has already begun to happen in some universities, and it is particularly encouraging that in some cases, pre-service CS training is likely to soon be required of all teachers.

Continuing support will be a vital part of effective teacher preparation. Though the fundamental concepts of computing are stable, the technology, applications, and social implications change rapidly. Both the Computer Science Teachers' Association (CSTA) and the CSforAllTeachers Community of Practice are positioning themselves to be providers for such a teacher support network; however, research is still needed to determine which strategies are most efficient, practical, and compelling.

The CSforAll movement has made strides in strengthening the preK-12 pipeline and creating a generation of college-ready students with solid CS backgrounds. But as these students prepare for graduation, will colleges be ready for them?

QUESTION 4:
HOW WILL COLLEGES AND UNIVERSITIES, ALREADY STRESSED BY RECENT SURGES IN CS COURSE ENROLLMENTS, COPE WITH POTENTIALLY LARGER SURGES OF STUDENTS ARRIVING BECAUSE OF CSFORALL?

Recently, CS departments across the country have seen a dramatic, and consistent, rise in student enrollment. The average number of CS majors reported by departments in the Taulbee Survey has tripled since 2006 and doubled since 2011 [9]. Even more striking is that the number of non-majors taking representative courses primarily intended for majors, has also increased at a rate equal to, and sometimes greater than, that of CS majors (See Table 1) [6].

Table 1: Percent increase since 2006 for Majors and Non-Majors entering computing courses at undergraduate institutions.

Representative Course	Increase in Majors	Increase in Non-majors
Intro to Major	152%	177%
Mid-Level	152%	251%
Upper-Level	165%	143%

We can expect these numbers to grow dramatically as the CSforAll generation starts arriving on campuses. These students will be diverse not only in gender, ethnicity, and disability, but also in their interests and academic needs. How institutions react to this growth will be pivotal in creating true diversity and equity in the field, and in establishing the increasingly interdisciplinary role of computation in our world and on our campuses.

In past cycles of high enrollment, many CS departments responded by simply capping enrollments; however, this quick-fix solution unintentionally resulted in a significant decrease in diversity. Given the increasing relevance of computing to all disciplines and aspects of our lives, it is hard to imagine that the current influx of students is a temporary problem. In fact, there is mounting evidence that this cycle is different from those in the past [6]. As computation becomes increasingly ubiquitous and powerful, students are more likely to come to CS courses with strong interdisciplinary goals. This is creating an opportunity for CS departments and universities to re-envision the role of computing on their campuses [3] and

> … [to] consider their missions and the constituencies they serve, and to determine what role computing should play in the experience, knowledge, and skills of its graduates of 2025 and beyond.

The four questions addressed above highlight some of the most pressing needs in computer science education; however, the growing and urgent need for computer science education *research* underpins them all. Without this, we will not be able to develop the most effective ways to engage both students and their teachers in computing.

QUESTION 5: HOW CAN WE BRING COMPUTING EDUCATION RESEARCH (CER) TO ADDRESS THE MOST URGENT ISSUES IN CS EDUCATION?

CS Education has not benefited from the long histories that other math and science disciplines have with pedagogical and learning inquiry. However, the emerging field of CER has the potential to empirically determine best practices for teaching CS to a broad audience effectively, inclusively, at scale, and with evidence-based methods.

Computing is particularly well positioned to address these issues because many of the educational challenges can leverage, and be built upon, ideas and solutions from the field itself. Mike Resnick's work at MIT investigating how to scale programming instruction to K-12 students, for instance, resulted in the creation of Scratch [4]. This platform has since reached hundreds of thousands of learners, providing a scalable foundation for teaching computing across the world.

Despite the opportunities to address these interesting, relevant, and challenging problems, there are few academic career pathways for computing education researchers. The scarcity of faculty appointments, coupled with an expectation of CER faculty to take on higher teaching loads, has created a culture in which CER is not regarded as a research discipline in computer science on equal footing as others in the university tenure track.

However, increased research funding is helping to grow a dynamic community of computing education researchers. Starting with its first Faculty Early Career Development Program (CAREER) awards for CER faculty in 2014, NSF has been a leader in this effort through continued support in areas of broadening participation, effective teacher training, and improving undergraduate CS education [4]. Importantly, the Graduate Research Fellowship Program (GRFP) supports graduate students pursuing computing education research.

To fully explore the challenging research questions facing computing education requires the expertise from computer scientists and CS teachers, as well as from education and learning science researchers. For too long, researchers and practitioners have operated in silos, with the result that many research findings were not widely adopted because they failed to address the problems that teachers were encountering then. Now, NSF requires that all proposals responding to its CSforAll solicitation (NSF 18-537) come from researcher-practitioner partnerships that leverage the two groups of experts to close this gap in what we know about the teaching and learning of computing and to get the best practices into our classrooms.

In conclusion, society will benefit from the preparation of *all* our citizens to take full advantage of—and become creators in—a future shaped by computational skills and competencies. ❖

References

1. Booch, G. Readn', writ'n, 'rithmetic…and code'n in *Proceedings of the 38th Special Interest Group on Computer Science Education Technical Symposium on Computer Science Education*. Covington, KY, USA, March 07–11, 2011.
2. Code.org. *K-12 Computer Science Policy and Implementation in States*; https://code.org/promote. Accessed 2018 April 24.
3. Computing Research Association. *Generation CS: Computer Science Undergraduate Enrollments Surge Since 2006*; http://cra.org/data/Generation-CS/. Accessed 2018 April 24.
4. Cooper, S., Forbes, J., Fox, A., Hambrusch, S., Ko, A., Simon, B. The Importance of Computing Education Research. *Computing Community Consortium Catalyst*, 2 (2016); https://cra.org/ccc/wp-content/uploads/sites/2/2015/01/CSEdResearchWhitePaper2016.pdf. Accessed 2018 April 24.
5. National Center for Education Statistics; https://nces.ed.gov/fastfacts/display.asp?id=372. Accessed 2018 April 24.
6. National Academies of Science, Engineering and Mathematics. *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*; http://www.nap.edu/24926. Accessed 2018 April 24.
7. Prensky, M. *From Digital Natives to Digital Wisdom: Hopeful Essays for 21st Century Learning*. (SAGE Ltd, California, 2012).
8. The College Board. *Program Summary Report 2017*; https://secure-media.collegeboard.org/digitalServices/pdf/research/2017/Program-Summary-Report-2017.pdf. Accessed 2018 April 24.
9. Zweben, S. and Bizot, B. 2016 Taulbee Survey: Generation CS Continues to Produce Record Undergrad Enrollment; Graduate Degree Production Rises at both Master's and Doctoral Levels. *Computing Research News* 29, 5 (2017).

Allyson Kennedy
AAAS Science and Technology Policy Fellow
Directorate for Computer and
Information Science and Engineering
National Science Foundation
2415 Eisenhower Ave.
Alexandria, VA 22314
aykenned@nsf.gov

Janice Cuny
Directorate for Computer and
Information Science and Engineering
National Science Foundation
2415 Eisenhower Ave.
Alexandria, VA 22314
jcuny@nsf.gov

DOI: 10.1145/3232539

Paving a Path to More Inclusive Computing

Mehran Sahami, *Stanford University*

The need for computing science (CS) education has reached unprecedented levels, creating myriad opportunities for computing science educators to have significant impact on the educational landscape. From riding the wave of growing enrollments, to better understanding modalities for success in CS education through research, to embracing a broad perspective of computing as a substrate for work in other fields, there is no shortage of opportunities to pursue. We must, however, also be mindful of the social structure surrounding our field and inculcate a healthy, inclusive culture in computing if we are to fully realize the promise that broad computing education offers.

INTRODUCTION

Perhaps the greatest challenge in trying to identify "The Five Big Questions in Computing Science Education" is limiting the list to only five topics. Indeed, computing science educators have tremendous opportunities ahead to have significant impact in addressing the need for large-scale computing education. The past few years have seen unparalleled growth in computing science enrollments at a national level in the United States and the recent emphasis on broad availability of computing education at the K-12 level will only accelerate this trend. While daunting, the demand for more computing education creates more possibilities for CS educators to have greater impact. It also brings to the fore significant questions for our field as we try to fully embrace this opportunity. Here we raise five of these questions that we believe are among the most pressing. While these questions may seem disparate when considered separately, we believe their common thread is that they all play a role in promoting a more diverse pipeline of students in computing, either implicitly or explicitly. Thus, we believe that in order to pave a path to more inclusive computing, we need to address the following questions.

- How can we successfully meet the capacity challenges for computing education?
- How can computing education research (CER) be more broadly accepted as a research area within computer science, and how can CER results be more fully utilized in practice?
- What are ways we can harness computing to better understand and enhance the educational process, especially within the computing domain?
- What educational models can we successfully employ to develop computing as a substrate for work in other fields?
- Perhaps most importantly, how can we better promote diversity and broaden participation in computing, and inculcate a healthy and inclusive culture?

Here we raise five of these questions that we believe are among the most pressing. While these questions may seem disparate when considered separately, we believe their common thread is that they all play a role in promoting a more diverse pipeline of students in computing, either implicitly or explicitly.

MEETING THE CAPACITY CHALLENGES IN COMPUTING EDUCATION

As those working in computing education are aware, the past decade has witnessed unprecedented growth in computer science enrollments in colleges and universities in the United States and elsewhere. For example, the 2016 CRA Taulbee Survey [13] shows that the number of CS majors per academic unit in US PhD-granting institutions with CS programs grew by over 250% from 2006 to 2015. A more recent report by the National Academies of Sciences, Engineering, and Medicine [6] finds that this enrollment growth shows no signs of abating and goes on to highlight the challenges of dealing with such growth. Particularly worrying is the Generation CS report by the Computing Research Association [2], which states that "the increase in the number of tenure-track faculty and teach-

ing faculty in no way matches the growth in the number of undergraduate CS majors."

With this backdrop, a significant question for our community is: how can we successfully meet the capacity challenges for computing education? Admirably, the National Academies' report not only describes this problem, but also provides a set of recommendations aimed at trying to address this issue. We do not repeat their full set of recommendations here (the interested reader is strongly encouraged to read the original report). Rather, we highlight one of the more significant issues raised in the report, which is that the seemingly simple approach of limiting enrollment in computing programs to deal with the capacity challenge may lead to the unintended consequence of limiting student diversity. This is a critical factor for programs to consider, both for purposes of equity as well as having a diversity of viewpoints in the downstream workforce.

Somewhat unintuitively (at least until one examines the statistical dynamics more closely), we found at Stanford University that as the total number of students majoring in CS increased, the percentage of women in the program also increased (Figure 1). This indicates that overall enrollment growth caused an acceleration in the number of women relative to men choosing to major in CS. Such accelerations may be due to a number of factors, including an increased sense of community and belonging when there are greater numbers of women in the program. This larger community of women results in even more women feeling a sense of common identity in pursuing CS.

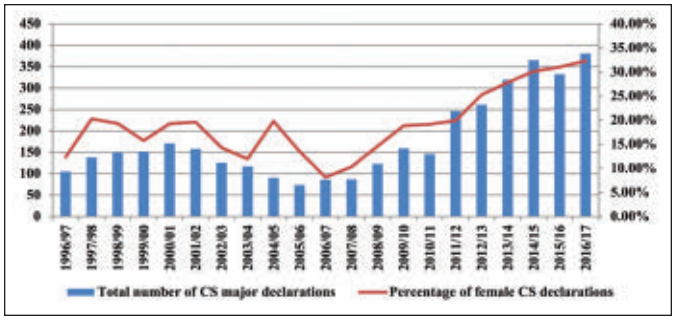


Figure 1: Total number of CS major declarations, including men and women (bar graph, left axis) and the percentage of female CS declarations (line graph, right axis) at Stanford University from academic year 1996/97 to 2016/17.

Limits on enrollment typically also take the form of minimum GPA requirements in introductory computing courses or other early performance hurdles to allow for admission into a computing major. Such barriers tend to favor students with prior experience in computing, reinforcing the existing dominant demographic of students and thus potentially limiting student diversity in these programs.

Ultimately, to maintain a commitment to making computing education broadly accessible, we must answer the question of how we scale our educational offerings to make them available to all students who are interested. This is a question not only about resources—of whose limitations we are painfully aware—but also one of research into understanding the implications of different models for addressing the capacity challenge. We need to study the impact of different responses to the capacity crisis to better inform school administrators and policy makers about approaches that can successfully scale opportunities in computing education without sacrificing educational quality or student diversity—otherwise, we stand to lose both.

COMPUTING EDUCATION RESEARCH AS A RESEARCH AREA WITHIN CS

Computing education research (CER) faces an identity crisis in much of higher education. Faculty in many computing departments believe that CER, as educationally-focused research, belongs in a school of education. Conversely, many schools of education, focusing on K-12 education, education policy, or more "established" subjects such as mathematics education, believe that CER, with a focus on computing science, belongs in a computing science department. Practically, this means that very few schools have CER faculty, much less any programs in that area, giving rise to the question: how can computing education research be more broadly accepted as a research area within computer science, and how can CER results be more fully utilized in practice?

In many ways, the story arc for CER matches that of Human-Computer Interaction (HCI) with a 25 year lag, despite the fact that SIGCSE predates SIGCHI by more than a decade. Indeed, in the early 1980's much work in the HCI community was being done by social scientists, psychologists, and human factors researchers. Many computing departments at the time did not see this work as "core" computer science and, as a result, it was difficult to justify hiring HCI faculty within a CS department. Fast forward to today and HCI is a well-established area in many CS departments and included as a "core" knowledge area in the CS2013 curricular recommendations [1].

In a similar way, we must promote CER to become a more accepted area of research within computing departments. To this end, CRA released a white paper aptly titled "The Importance of Computing Education Research" [3] outlining various approaches to promoting CER. Generalizing one of the report's recommendations (originally: "Apply CER to improve departmental courses"), we would argue that for CER to become more accepted as a research area within CS, it needs to not only define common problems that computing educators would care about (e.g., assessing modalities for teaching), but also define benchmarks that can consistently and objectively be measured to make studies more comparable and show progress in the field. The use of benchmarks, while not to be considered the only measure of progress in a field, have been used to help show progress in areas such as machine learning, computer vision, information retrieval, and systems for years. CER could certainly benefit from the same.

CER needs to extend beyond the notion of an idea working in a researcher's own classroom to establishing practices that

are reproducible using such benchmarks by those not invested in the original research. There has been strong work in this vein, for example, in measuring the efficacy of active learning [9] as well as developing language independent CS1 knowledge assessments [7]. Yet more work needs to be done to solidify CER as a field whose results move others to change their educational practices to achieve better outcomes. When CER results are sufficiently compelling and general that data-minded CS faculty (and other educators) can no longer ignore the opportunity to produce better educational outcomes, then it will become possible to convince more CS faculty that computing education research should be happening in their own departments. Ultimately, this will benefit students who will be the recipients of more effective and inclusive computing education.

By making programming education more accessible through such modeling and personalization, we also have the potential to support a greater array of students in their initial forays into programming.

HARNESSING COMPUTING TO BETTER UNDERSTAND EDUCATION IN COMPUTING

A benefit of working in computing is that we can utilize tools from our own field to analyze data related to educational processes and outcomes. The 1980's saw growing research in trying to model and understand how students learn to program [10,12]. And, on a general level, research in computer-based automated tutoring systems has existed for many years in several domains. More recently—mostly in the past decade—the confluence of large educational data sets (often available via on-line learning platforms), advances in machine learning techniques such as probabilistic modeling and deep learning, and greater computational infrastructure, has made it possible to create more expressive models of students' learning in many educational activities, including programming. To this end, we face the question: what are ways we can harness computing to better understand and enhance the educational process, especially within the computing domain?

There is no question that online learning provides a significant means to scale the availability of educational opportunities. But serious questions remain regarding the quality of learning outcomes and the impact of the online setting on factors like students' motivation and persistence [4]. In the computing domain, the push to provide broadly accessible educational opportunities (e.g., the Hour of Code [5]) and the pull of growing demand for computing education has provided greater opportunities to bring analytical tools to bear to help scale and improve computing education. As just one example in this regard, Piech et al. [8] analyzed a dataset of over 2 million student program submissions from the Hour of Code to create a model capable of autonomously generating hints for learners having difficulty completing the programming exercises. While we are still in the early days of building scalable and deployable models of students learning to program, this research area does hold palpable promise for helping us to create more scalable and robust learning opportunities for novice programmers. By making programming education more accessible through such modeling and personalization, we also have the potential to support a greater array of students in their initial forays into programming.

PROVIDING COMPUTING EDUCATION AS A SUBSTRATE FOR WORK IN OTHER FIELDS

As computing plays a larger role in other fields, from computational biology to digital humanities, we must assess how to provide appropriate computing education to an increasingly diverse audience. For example, what alternate entry-level courses might we envision that provide students both a meaningful introduction to computing while also preparing them for work in domains other than computing. More generally, it raises the question: what educational models can we successfully employ to develop computing as a substrate for work in other fields?

As computing plays an increasing role in work in other domains, it becomes more critical for computing education to not only consider what computing techniques a student should learn, but also what the appropriate abstractions are for modeling problems in a particular domain. For example, a history major using computational techniques to analyze changing word usage in scientific writing need not know the details of the underlying trie data structure being used to efficiently store word counts over time. Rather this student needs to know an appropriate set of computational tools that might be brought to bear in helping answer questions in their domain.

Providing diverse learning opportunities in computing to match the needs of an audience from multiple disciplines forces us to think more critically about the fact that not all computing education will be done in computer science (or related) departments. Rather, we must consider new models in which computing education is more readily incorporated throughout other disciplines, where the selection of the appropriate techniques and abstractions to teach can be made by those with requisite domain knowledge. Broadening computing education in this way presents the challenge of teaching those in other domains enough to understand the tools that computing has to offer so that they can in turn teach those tools in a generally applicable way to others in their field. It also provides the opportunity for the computing education community to motivate our own students with respect to the potential of their work in solving a wide range of problems across disciplines.

PROMOTING DIVERSITY AND A HEALTHY, INCLUSIVE CULTURE IN COMPUTING

As computing science educators, we sometimes tend to focus on the content that is being taught rather than the broader context in which that content will eventually be used. While we have known for many years that women and many minorities groups are underrepresented in computing (both in academia and industry), the past few years have raised public consciousness with regard to just how toxic the culture in the high-tech sector is to women and underrepresented groups [11]. As computing science educators, we have a moral obligation to answer the question: how can we better promote diversity and broaden participation in computing, and inculcate a healthy and inclusive culture?

Rather, we must consider new models in which computing education is more readily incorporated throughout other disciplines, where the selection of the appropriate techniques and abstractions to teach can be made by those with requisite domain knowledge.

For all its potential, computing has been a field with a significant lack of diversity for many years. And while there are recent signs that the gender disparity is improving, albeit slowly—the National Center for Women and Information Technology reports that the percentage of bachelor's degrees awarded to women increased from 12% in 2010 to 16% in 2014—there is clearly much more that needs to be done in continuing to increase diversity. Many schools and universities, often with public and private support, have been working tirelessly to broaden participation in computing. Yet, by itself, broadening participation is not enough. We need to promote the idea that everyone must play a role in creating a healthy and inclusive culture in computing. We need to find ways to evolve beyond a culture of competition and tribalism (e.g., programming language wars) and social division (e.g., "brogrammers") to a culture that understands the value of diversity both intrinsically (i.e., fairness and social justice) and for its utility (i.e., including different viewpoints helps to produce better solutions). We need to be advocates for our students, our colleagues, and ourselves to build a culture that not only seeks to welcome more diversity to our field, but actively works to maintain it. Gains in attracting underrepresented groups to computing will have little long-term impact if those gains are eroded by cultural toxicity that eventually pushes those same individuals out of the field. We need to work on ways to effectively improve the culture in computing to be more inclusive and healthy, not just because we are educators, but because we are members of that community.

CONCLUSION

There are myriad ways to answer the questions raised above and the views here are by no means comprehensive. The challenges we face as computing educators are great, but the potential opportunity to impact the future for millions of learners—many of whom might not otherwise have access to computing education or consider a career in computing—is even greater. If we are to realize the full impact that our field can have on shaping current and future generations, then it is incumbent on us to not shy away from the hard questions, but rather embrace them as the means for pushing our field further forward in a more inclusive and supportive way. ❖

References

1. ACM/IEEE-CS Joint Task Force on Computing Curricula. Computer Science Curricula 2013. (ACM Press and IEEE Computer Society Press, 2013); DOI: http://dx.doi.org/10.1145/2534860.
2. Computing Research Association. Generation CS: Computer Science Undergraduate Enrollments Surge Since 2006. (CRA, 2017); https://cra.org/data/Generation-CS/. Accessed 2018 Aug 13.
3. Cooper, S., Forbes, J., Fox, A., Hambrusch, S., Ko, A., and Simon, B. The importance of computing education research. (2016); arXiv preprint retrieved at arXiv:1604.03446.
4. Hart, C. Factors associated with student persistence in an online program of study: A review of the literature. *Journal of Interactive Online Learning*, 11,1 (2012), 19–42.
5. Hour of Code; https://hourofcode.com/. Accessed 2018 Aug 13.
6. National Academies of Sciences, Engineering, and Medicine. *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*. (The National Academies Press, Washington, DC, 2017); DOI: https://doi.org/10.17226/24926.
7. Parker, M.C., Guzdial, M., and Engleman, S. Replication, validation, and use of a language independent CS1 knowledge assessment. *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. (ACM, New York, NY, 2006), 93–101. DOI: https://doi.org/10.1145/2960310.2960316
8. Piech, C., Sahami, M., Huang, J., and Guibas, L. Autonomously generating hints by inferring problem solving policies. *Proceedings of the Second ACM Conference on Learning at Scale (L@S '15)*. (ACM, New York, NY, 2015), 195–204. DOI: http://dx.doi.org/10.1145/2724660.2724668
9. Porter, L., Lee, C. B., and Simon, B. Halving fail rates using peer instruction: a study of four computer science courses. *Proceeding of the 44th ACM technical symposium on Computer Science Education (SIGCSE '13)*. (ACM, New York, NY, 2013), 177–182. DOI: http://dx.doi.org/10.1145/2445196.2445250
10. Reiser, B., Anderson, J., Farrell, R. Dynamic student modeling in an intelligent tutor for LISP programming. *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI '85)*. (1985), 8–14.
11. Scott, A., Klein, F. K.., and Onovakpuri, U. Tech Leavers Study. (Kapor Center for Social Impact, 2017); https://www.kaporcenter.org/wp-content/uploads/2017/08/TechLeavers2017.pdf. Accessed 2018 Aug 13.
12. Soloway, E. and Spohrer, J. *Studying the novice programmer*. (L. Erlbaum Assoc. Inc., Hillsdale, NJ, 1988).
13. Zweben, S. and Bizot, B. Taulbee Survey. *Computing Research News*, 29, 5 (2017); https://cra.org/crn/wp-content/uploads/sites/7/2017/05/May_2017_CRN.pdf. Accessed 2018 Aug 13.

Mehran Sahami
Stanford University
Gates Building, Room 180
Computer Science Department
Stanford University
Stanford, CA 94305
sahami@cs.stanford.edu

DOI: 10.1145/3230700

Experiences with SIGCSE

Frank H. Young, *Rose-Hulman Institute of Technology (retired)*

Excerpts from the First-Timers Luncheon talk given at SIGCSE 2015

By 1973, five years after getting my PhD in mathematics, I had completed three summer institutes on topics in computer science, continuing a process of moving my primary teaching from mathematics to computer science. I learned that computer science was both more interesting and more difficult than abstract algebra.

I also found out about SIGCSE. SIGCSE was a lifesaver for me. I discovered that I was not alone. I met others who were trying, sometimes with very little support from their institutions, to initiate, expand, and improve the teaching of computing at their institutions. I was able to learn from those who had more experience than I did. I was able in some small ways to encourage people from different institutions to cooperate with curriculum development and do some needed cooperation.

SIGCSE helped me develop as an educator. It provided me with an opportunity to organize and participate in panels that helped others do what I was doing. I was able to interact with and get needed help from others, help that was unavailable at my first small college position. SIGCSE was (and continues to be) an economical way to get assistance to become a better teacher of computing. I could not have given my students what they needed and deserved without SIGCSE. If in a small way I have been successful, it is because of my colleagues in SIGCSE, their inspiration, and their generous support and assistance. Learning from my SIGCSE friends has been very important for me professionally.

If in a small way I have been successful, it is because of my colleagues in SIGCSE, their inspiration, and their generous support and assistance.

SIGCSE is very different from most other professional organizations. SIGCSE would not succeed without volunteers. The SIGCSE Symposium is exceptionally successful (and rather inexpensive!) because of its many volunteers and a strong tradition of volunteering.

In conclusion, thank you for your patience and your kind reception. The SIGCSE Award for Lifetime Service to the Computer Science Education Community is truly an honor. But the best way for SIGCSE members to honor my work is to continue it after I am obsolete (which is probably pretty soon now). Let me encourage all of you to volunteer both for SIGCSE and computing education (and for any other things that strike your fancy). You will experience the joy and satisfaction of paying back and paying forward. Even though I will not be around, I am confident that you will take SIGCSE to new heights. Good luck. The future awaits, and it will be exciting. ❖

Frank H. Young
Rose-Hulman Institute of Technology (retired)
125 Villa Lane
Terre Haute, IN, USA
young@rose-hulman.edu

DOI: 10.1145/3230693

SIGCSE, Goldilocks and the Three Bears

MaryAnne L. Egan, *Siena College*

Finding the right fit for a professional organization can be a little like "Goldilocks and the Three Bears," minus the breaking and entering issues that always bothered me about that story. Attending an organization's associated conferences gives one an excellent introduction into the general philosophy and tenor of the group. Reflecting on over 20 years of attending various SIGCSE conferences, I realize how similar my story is to Goldilocks. After attending conferences associated with other professional organizations that were either too "hard"-core or too "soft," I was happy to find one that was "just right" when I attended my first SIGCSE conference in 1998.

WHAT MAKES IT JUST RIGHT

It was at that first conference in Atlanta, Georgia, where I presented a paper about a visualization tool for fuzzy clustering. Not only did I meet amazing people throughout the conference, beginning with the shuttle ride from the airport, but I brought back information, tools, and techniques that were readily incorporated into my classroom. Pulling out my copy of that particular proceedings, I was surprised by the number of papers flagged and highlighted. It was a reminder of the overwhelming decision making process necessary to select which sessions to attend for every conference. In addition to the information gleaned from other attendees' presentations and papers, I received useful, positive feedback about my work which was then incorporated into future iterations of that project. But, what I remember most are the people, the collegiality, and the now 20-year friendships.

At one point I was going to list every SIGCSE conference I attended and mention a memorable event or speaker from each. But when compiling the information, I realized that I have attended 17 SIGCSE, ITiCSE, or CCSC conferences over the last 20 years and who's going to read through all of that? Instead I want to focus on three reasons why I love SIGCSE conferences—the amazing people, cultural experiences, and technical knowledge acquired. I will illustrate each of these reasons with select examples.

THE PEOPLE

How do you describe to others the excitement, respect, and willingness to help others, that is so often witnessed at SIGCSE conferences? Attendees are genuinely excited to see one another, they share information about new pedagogies or tools in CS education, they offer advice, and act as resources even after the conference is over. I still remember the SIGCSE conference where I saw this crazy person running around inviting everyone to a tea party… Alice's Tea Party, where Alice was introduced! I also remember sharing a hot air balloon basket over Turkey with over a dozen other conference attendees. How awesome is that!

Sessions or BOFs about departmental and university initiatives are enlightening and informative. Additionally, it is through these sessions that one develops a network of colleagues from other institutions with similar interests and/or challenges. The value of these networks is immense, especially for faculty from smaller schools who have less resources readily available. It is through this network of colleagues that I was encouraged to take on more responsibility both at my own institution and within the larger professional community and I am forever grateful.

TECHNICAL CONTENT

The number of tips, techniques, tools, and pedagogy that are applicable to one's classroom and research initiatives is overwhelming! There is an art to selecting just a few concepts from a conference on which to focus or incorporate into your teaching, otherwise one risks running out of time before the next conference comes around and more ideas are added to your list! I admit that I have not yet perfected this art as people present many innovative and interesting ideas.

A very short list to illustrate a few things picked up at SIGCSE conferences and still used in my classes today include—introducing students to CS with multimedia concepts in Python, CS Unplugged, CS "magic" tricks, and interactive programming with groups and easel paper.

CULTURAL OPPORTUNITIES

While SIGCSE and CCSC allow one to get to know the people and places of the host city, there is nothing like the cultural experiences of ITiCSE. I love the fact that there is time set aside during the conference to take a tour of the city, a local university, or some other cultural excursion, and that the conference dinners introduce you to the food, dance, and history of the host country. The tours before or after the conference are

a great way to get to know the country (and your colleagues) better! Who can forget eating haggis in a castle before learning how to do Scottish dancing? Or learning how to belly dance in Turkey? Or discovering geo-caching in Cappadocia? Or finding out what Parisians do during hot summer nights with no air conditioning? The answer to that last question is to hang out on the banks of the Seine with a picnic basket and good friends, which is exactly what we did!

We learn how other countries teach and offer computer science to their students, we learn about cultural mores that could be a stumbling block when working on global projects, and we learn that computing does not happen in a cubicle with limited human interaction.

These are amazing opportunities that open our eyes to the world around us. We learn how other countries teach and offer computer science to their students, we learn about cultural mores that could be a stumbling block when working on global projects, and we learn that computing does not happen in a cubicle with limited human interaction. In other words, it helps us, as educators, bring this world-view back into the classroom.

LAST WORDS

The impact the SIGCSE community has made in my professional career, and my life, is tremendous. I now have an amazing network of mentors, colleagues and friends; enough new tips, tools and techniques to keep me busy far into retirement; and discovered that some people can pack for two weeks of international travel with only carry-on luggage. I bet Goldilocks packed more when she broke into the Three Bears house! ❖

MaryAnne L. Egan
Department of Computer Science
Siena College
515 Loudon Rd.
Loudonville, NY USA
maegan@siena.edu

DOI: 10.1145/3276306

My SIGCSE: Reflections of a Computing Educator

Briana B. Morrison, *University of Nebraska at Omaha*

I will never forget my first SIGCSE Conference—1999, New Orleans. My first was supposed to be in 1998 in my home town of Atlanta, but I had given birth to my second child a week before the conference was scheduled. That year didn't work out, but due to my department chair's encouragement I tentatively went in 1999. Now here I was, standing in the registration line alongside the authors of the textbooks I was using in class! Who could have predicted what followed—that one conference led to a career change, new research trajectory, lifelong friends, and an organization that I consider my community of practice.

MY PATH

I transitioned to academia from industry. I was a non-tenure track instructor, teaching at a technical university because it allowed me a flexible schedule to spend time with my children. It was just a temporary gig until I went back to industry. My department chair encouraged me to explore the current research in teaching computer science, to see what others were doing to help their students succeed. He suggested I attend the SIGCSE Technical Symposium (TS) and learn from the experts. At my first TS, I found what I was looking for and so much more. I learned about language choices, lab assignments, and pedagogical techniques. I met the most incredibly friendly and helpful people, who were not at all like the arrogant academics I had been warned about. I had a wonderful time and left rejuvenated and excited to continue teaching.

That single experience—attending the Technical Symposium in 1999—changed my view of computer science education forever. I realized that others were exploring how to improve computing education and making it accessible to students beyond computing majors. I wanted to be a part of that. I wanted to be a proud, contributing member of that group.

Because I joined SIGCSE and the listserv, I kept current on CS education discussions and read about opportunities for summer workshops and ways to further my knowledge. Through one of those announcements in 2003 I found out about an NSF sponsored project, Bootstrapping Research in Computer Science Education. That project changed my entire professional life. It was through that project that I discovered *research* in computer science education, and that *I* could do that research. Because of SIGCSE and the Bootstrapping project, I decided to pursue a PhD in Computer Science Education and contribute to the research discourse. I was no longer satisfied just teaching, I wanted to research *why* students learned the way they did and what could be done to help them succeed. I knew that SIGCSE was my community to host these discussions.

Since 2003, I have attended every SIGCSE Technical Symposium except 2013 in Denver. I missed Denver because it was one week before qualifying exams for my PhD. While I really wanted to attend, I made the difficult decision to stay home and continue studying but followed everything on Twitter! Every year I look forward to seeing longtime friends and catching up. I look forward to having my brain stimulated with new ideas, fresh perspectives, and learning what others are doing to improve computer science education. I never fail to leave the Symposium with several ideas that I can't wait to try in the classroom. I present my own research now and discuss the implications with others. I can literally "see" the discipline moving forward!

Yet I have also learned that SIGCSE is so much more than just the TS. I've been to ITiCSE once and ICER several times. I have served as a volunteer for the organization—as a reviewer, conference help, and now Board Member-At-Large. Through volunteering, I continually meet amazing people, all of whom care deeply about computer science education and sacrifice time and energy for that cause. I am pleased to give back to the organization that has given me so much.

There is no measure for the impact that the SIGCSE organization has had on my life. It opened my eyes to a new professional path that I had not considered possible, and now I am happily journeying down that path. I am exceptionally proud to be a SIGCSE member; proud of the accomplishments of the organization and excited for its very bright future. The next 50 years are going to be incredible.

If attending the TS inspires you in your teaching, why not invite a friend next year? Are your colleagues SIGCSE members? Perhaps that invitation from you might just result in a life changing event, just as SIGCSE forever changed my life. ❖

Briana B. Morrison
Department of Computer Science
College of IS&T
University of Nebraska at Omaha
6001 Dodge Street, PKI 172
Omaha, NE 68182-01162
bbmorrison@unomaha.edu

DOI: 10.1145/3230688

2006-2018 Same Issues Same Challenges

Judith Gal-Ezer, *The Open University of Israel*

In 2006 I was awarded the SIGCSE Award for Outstanding Contribution to Computer Science Education—a milestone in my professional career. In my keynote presentation I thanked my family, teachers, colleagues, and university. From a professional perspective, I mentioned issues that were challenging at the time, among them:

- placing computer science on a par with other sciences within school systems,
- increasing the number of women interested in computing disciplines, and
- recruiting well-educated, enthusiastic teachers certified to teach the discipline.

If we look around today, we can observe that indeed some progress has been made, yet we are still confronting the same basic challenges.

INTRODUCTION

My first SIGCSE conference was in February 1993 in Indianapolis. The Technical Symposium was co-located with the ACM conference. I was there to attend the ACM conference, where my friend and colleague David Harel was awarded the Karl Karlstrom Award for his outstanding contribution to Computer Science Education. At that time, we had finished designing the computer science high-school curriculum and started its pilot program. During these years I became involved in computer science school education research. This is the year when I became acquainted with SIGCSE, and got to know my current friends and colleagues, who are too numerous to mention.

This was no doubt a milestone in my professional career, and since then I have attended almost every Technical Symposium on Computer Science Education (SIGCSE).

PHOTO: ©WWW.ISTOCKPHOTO.COM/HANIBARAM

2006-2018 Same Issues Same Challenges

"TO TEACH IS TO TOUCH LIVES FOREVER"

This was the title of the keynote presentation I gave in 2006; it was the name of a small booklet I had purchased several years previously. It emphasized the importance of the teachers' role in our lives, and it expressed my belief that teachers constitute the cornerstone of a successful implementation of any program of study, at any level. When we designed the high-school curriculum at that time, we pointed out that teachers who were certified to teach computer science must have a formal computer science education as well as the required methodological and pedagogical skills. Indeed, following various reports published in recent years, the lack of qualified teachers is the bottleneck inhibiting the introduction of computer science into school systems. We are very familiar with the global efforts to recruit well-educated and enthusiastic teachers to assist in this enormous challenge.

High school computer science studies pave the way for higher education in this field, that is, those students who took high school computer science courses are more likely later to pursue one of the computing disciplines. Interestingly, this is even more pronounced with female students.

GENDER IN THE CONTEXT OF COMPUTER SCIENCE

Another issue I addressed in my keynote was the gender issue in the context of computer science. Here is what I said.

> Actually, I never thought this to be an issue I would get involved in. However, in recent years I came to realize that the gender issue is really something that ought to be dealt with in earnest. And I am not the only one. To quote Fran Allen, the first woman to win the Turing Award, "The decline of the number of women entering computer science is a serious, national concern," and I should add a world-wide concern. Indeed, the dearth of women represented in computer science studies and in professional occupations in the computing and hi-tech world is a fact in many western countries.

Eleven years have passed, and we are still almost in the same situation. We are still trying to address the issue of underrepresented communities, including women, in the computer science field.

It is also the duty of our community to ensure that the requirements of teachers certified to teach computer science will follow high standards like other disciplines.

However, in the studies we have conducted we found some encouraging results—high school computer science studies pave the way for higher education in this field, that is, those students who took high school computer science courses are more likely later to pursue one of the computing disciplines. Interestingly, this is even more pronounced with female students. This can consequently lead to the realization that the efforts of introducing computer science into schools are worthwhile and may help in encouraging more women to become interested in the discipline and to continue their computer science studies at institutions of higher learning.

In addition, the perception of gender was found to be irrelevant to computer science abilities among both males and females. This indicates that something has been changed, since males and females seem to agree on females' abilities regarding computer science and perceive them without gender bias.

FUTURE TRENDS

It is our community's duty to continue making progress. We believe that all students should be given the opportunity to be exposed to computer science at school, preferably at an early age. The computer science curriculum should reflect the scientific aspects of the discipline, as well as its more practical applications. It should be on a par with other STEM disciplines. Such an effort will not only assist students to decide whether to take AP exams in computer science or pursue this discipline further in their future education—it may also help in attracting more women into the discipline and possibly alleviate the pay gap between genders.

It is also the duty of our community to ensure that the requirements of teachers certified to teach computer science will follow high standards like other disciplines. Neither formal requirements nor proper methodological training should be omitted. Difficulties in recruiting qualified teachers should not lead to the temptation to reduce high standards and requirements.

Many challenges await us now and in the future. ❖

Judith Gal-Ezer
Department of Mathematics and Computer Science
The Open University of Israel
Israel
galezer@cs.openu.ac.il

DOI: 10.1145/3230689

A Student in SIGCSE-land or How I Discovered Teaching

E. Anne Applin, *Southern Maine Community College*

From my first days in high school at least, I thought my teachers were sort of mystical beings. They walked into a room and started speaking and they just KNEW everything. The knowledge was there on the tips of their tongues. They held court, and mostly I hung on every word. When I went back to college majoring in Computer Science my professors were titans in my mind. Their very thought processes amazed me. And then I went to SIGCSE.

SAN ANTONIO, 1991

As chair of both my university student ACM and our chapter of UPE, going to the national UPE meeting sounded like a great idea. The faculty advisors for both clubs encouraged me to attend. It was the ACM Annual Computer Science Conference which overlapped with the SIGCSE conference 1991 in San Antonio, Texas. I remember reading the programs and marking interesting sessions to attend. I remember hearing one plenary session that was mostly over my head, but bits of it have stuck with me to this day and although I don't remember the speaker's name, I quote him in Computer Organization every term. "They shouldn't have called it a bus," he said, "they should have called it a cab because it moves one piece of data at a time."

When I went back to college majoring in Computer Science my professors were titans in my mind. Their very thought processes amazed me. And then I went to SIGCSE.

What I remember so vividly from that SIGCSE conference was not the papers or the panels which were not yet of interest to me since I thought I would graduate and go into industry. What I remember are the passionate discussions that happened in the hallways and during breaks. The discussions of what to teach, when to teach it, and how to approach it. I came back from that conference and looked at faculty in a whole new light. This thing that they did every day, that they made look so effortless was a labor of love. There is no doubt in my mind that even those who don't attend educational conferences, still struggle with how to best approach a topic—they just don't have the support of my "800 closest friends."

This year is my 26th year in the classroom, it will be my 24th SIGCSE. I was at the first ICER (Seattle) and I've attended three ITiCSE conferences (Italy, Spain, and Scotland). Where else can a casual conversation in a bathroom lounge end up as a project that is still active 15 years later? I have learned much and continue to be a student in this community. The picture? A memento from the reception that first year. ❖

E. Anne Applin
Department of Computer and Information Sciences
Southern Maine Community College
2 Fort Road
South Portland, ME USA
aapplin@smccme.edu

DOI: 10.1145/3230690

My SIGCSE: It's the Community!

Michael Clancy, *University of California, Berkeley*

I decided early on that I liked to teach. After a lot of teaching as a graduate student, I was hired by U.C. Berkeley's EECS department as a teaching faculty in charge of the introductory programming courses. My understanding of good teaching, however, was somewhat limited: giving interesting lectures and inventing interesting homework that appropriately exercised the topic of the week.

Then I discovered SIGCSE! Reading the *Bulletin* and attending the Symposia exposed me to hundreds of people interested in computer science education, compared with only a few at home. Arnie Dyck showed me how to manage teaching and support staff. David Kay and I traded ideas for teaching functional programming in CS 1. Owen Astrachan, Rich Pattis, and Stuart Reges had opinions about book writing, curriculum design, and almost any other topic related to computer science. These people, and numerous others, became my collaborators, partners, and friends.

Flash to 2018. SIGCSE members continue to explore advances in computer science education, but challenging questions remain. How can students and teachers make best use of working collaboratively? Which format—e.g. MOOC, lab-centric, self-paced, flipped—offers the best pedagogical options for large-enrollment courses? What activities and tools most effectively contribute to learning? Answers are being devised by SIGCSE members.

Along with the scope of SIGCSE activities, I came to value the *accessibility* of the community. It has been easy for me to collect ideas and contribute a few of my own. Everyone is happy to share.

Then I discovered SIGCSE! Reading the *Bulletin* and attending the Symposia exposed me to hundreds of people interested in computer science education, compared with only a few at home.

Retirement hasn't stopped me. I've been to every Symposium since 1985, and Symposium attendance is a valued part of my year. This is due partly to the technical content, where SIGCSE is pushing the state of the art in computer science education, and partly to the people who contribute ideas, experiments, and feedback. This is the SIGCSE community, and I'm still excited to be part of it. ❖

Michael Clancy
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776
clancy@cs.berkeley.edu

DOI: 10.1145/3231744

A High School Teacher Attends His First SIGCSE Symposium

Alfred C Thompson II, *Bishop Guertin High School*

Attending SIGCSE for the first time as a high school teacher can be a bit intimidating. First off there are all those impressive people with PhDs who teach at schools where you probably would not have been accepted as a student let alone a faculty member. The people who write the textbooks you teach from are likely to be there. How are you, a lowly high school teacher, going to be accepted by those who live in the lofty towers of academia? It will turn out pretty well for the most part, but you don't know that yet.

Add to that I was giving a sponsored talk for a major company in front of an audience of high-powered academics and my stress level was through the roof at my first SIGCSE. How would the talk be received? Would someone blast me from the audience as an industry toady? Would my sponsor company who was paying what seemed like a lot of money for my travel and hotel be disappointed in me? Let's face it there were a lot of things that could go wrong.

That first presentation went well. Some people I had been warned about asked good questions in a tone of voice that indicated interest and respect rather than distain and dismissal. Even better, they seemed happy with my responses. Good thing I was well prepared.

More relaxed after my presentation I was able to attend many useful sessions and learn quite a bit that was useful for my own teaching practice. It turns out that there really is a lot of overlap between first year university CS courses and advanced high school computer science like the Advanced Placement Computer Science courses. Teaching techniques also have many similarities at all levels of teaching.

SIGCSE was an opportunity to connect with people on a more personal level. In many ways, the conversations in the hallways and break areas are as useful as the sessions. When an attendee uses them to build a personal network of friends, as many do, these conversations become even more useful over time. Over the years, I have developed friendships with university faculty members, other high school teachers, industry researchers, and other people interested and committed to advancing computer science education. SIGCSE has had a major role in developing my teaching practice, my involvement with promoting computer science education, and me as a person. I've come a long way thanks in part to SIGCSE. ❖

SIGCSE has had a major role in developing my teaching practice, my involvement with promoting computer science education, and me as a person.

Acknowledgements
The author is very grateful to all the SIGCSE attendees who have made him feel welcome and taught him so much.

Alfred C Thompson II
Computer Science Department
Bishop Guertin High School
Nashua, NH
Act2@acthompson.net

DOI: 10.1145/3276307

From Reno to Baltimore: Life in the Booth

James H. Cross II, *Auburn University*

For many of us in computing education, the annual SIGCSE Technical Symposium is an academic highlight of our year. As a member of ACM for 34 years and faculty member at Auburn University, attending SIGCSE is an event that seemed to grow in importance with each passing year. As with most attendees, I looked forward to the papers and panel sessions for new ideas, the breaks for good conversation, and the exhibits for exploring and more conversation.

After what seems like many years of presenting papers, the software project that Dean Hendrix, Larry Barowski, and I were working on had progressed to a shareable "tool" state. To have a more meaningful exchange with other faculty about using our software tool and getting feedback, we explored the idea of demoing in the exhibits area in 2000 (Austin). I approached the Addison-Wesley folks about setting up for an hour or two on one of their small round tables and they agreed. I demoed the tool for folks as they came to peruse Addison-Wesley's textbooks, answered questions, and discussed potential new features. In 2001 (Charlotte) and 2002 (Covington, KY), Dean Hendrix and I were invited to demo in the SIGAda booth. This was a natural fit since the first language our tool supported was Ada. Next, we added support for C and C++ since the tool itself was written in C/C++. By 1999, we had redesigned and implemented our tool in Java, and we had begun using Java in our CS1 and CS2 courses in Fall 2000. At this point, we began focusing most new features of the tool on supporting teaching and learning with Java.

By 2003 (Reno), we made the leap into the big-time—we finally got our own booth. Dean Hendrix and I showed up on Wednesday late afternoon to set up with some rolled up posters, a few handouts, and 150 CDs, and by late that evening, after having scrounged enough scrap cardboard to serve as poster board (who knew one couldn't just unroll the posters and tape them to the curtains), we were ready to go. The next morning when the exhibits opened at the break after the plenary session, there was the rush to the food, and then . . . well, we had a nice growing number of faculty at our booth. There were lots of quick conversations, and the handouts and CDs were going quickly. When the break ended, there was time for more relaxed discussions with lingering faculty around questions and concerns like— is this free? what's the catch? When we finally had a bit of a break, I remember looking at Dean and saying, with no sarcasm whatsoever, "wow, that went well!" The following year (2004 Norfolk), we added our College of Engineering table top display, and then in 2005 (St. Louis) we upgraded to a proper booth kit.

As a final note, after much consideration, we have decided to give our tool a proper name in 2019.

So here we are with 2019 (Minneapolis) approaching, and we'll have our booth for the 17th consecutive year. As we do every year, we look forward to seeing old friends and colleagues and meeting newcomers. I encourage anyone with an interest in dissemination of their work in computer education to consider exhibiting at SIGCSE— just a great way to share your work and meet others with similar interest. As a final note, after much consideration, we have decided to give our tool a proper name in 2019. ❖

James H. Cross II
Auburn University
Computer Science and Software Engineering
3101 Shelby Center
Auburn, AL 36849-5347
crossjh@auburn.edu

DOI: 10.1145/3230696

Tallying up SIGCSE

Amruth N. Kumar, *Ramapo College of New Jersey*

I remember my first SIGCSE. It was snowing in Philadelphia. My friend Liz Adams was the program chair. I had submitted a paper for the first time to the conference. And it was accepted. I thought, "Well, that wasn't too hard!" Little did I realize it was beginner's luck. I have spent the two decades since, riding the roller-coaster of acceptance highs and rejection lows. But, this is par for the course, and we all know, in Computer Science education, it is SIGCSE or bust! A couple of years later, I found myself wandering the side alleys by the Ramblas in Barcelona with Liz and company, looking for a good restaurant. That was my first ITiCSE conference. And a friendship had been made.

Two years later, SIGCSE was held in Atlanta. The land of Coca Cola and CNN. My friend Jane Prey was the program chair. I had managed to wangle the role of Workshops Chair. Java and object-oriented programming were all the rage in workshop proposals! The proposals were submitted by email as ASCII text! Looking back, yes, this was the last millennium in computing as well. A few years later, I would run into Jane again while waiting for the conference tour bus in Leeds. I was the program co-chair at that ITiCSE. And she was happy to be back after a health scare. And a friendship was cemented.

I could go on and on about the friends I made at SIGCSE and ITiCSE—with whom, I have carried on an annual conversation that has lasted decades. That is what SIGCSE and ITiCSE conferences have been to me—a Facebook group before there was Facebook. A MOOC for Computer Science educators before there were MOOCs. I would return from each conference with three-pages of closely scrawled notes—ideas to improve my teaching, opportunities for collaboration, etc. Just like a new year's resolution, I would find myself hopelessly falling behind in catching up on that list within a month of return. No more notes. Now, I look for three—three good ideas per conference. Just three on which I can follow up. And I have never been disappointed.

I remember the heydays of SIGCSE, when each conference tended to do one better than the one before in terms of attendee bags. When exhibitors had so many goodies to give away, including T-shirts, pens, keychains, and whatnots, you wanted to be at the exhibitor's booth on Thursday morning before the best stuff got scooped up! I remember when the conference first introduced computer terminals to check our email for those of us who were going through email-withdrawal symptoms—this was before the age of WiFi, smartphones, and ubiquitous laptops. Inevitably, there would be a long waiting line in front of each terminal. I remember when, beginning in my teaching career, I would collect at least half a dozen examination copies of textbooks at the exhibitor's booths, only to regret having to lug them back home. I am grateful airlines did not charge for checking luggage back then! I remember when submitting a paper meant sending multiple hard copies by regular mail; when we used transparencies and telescoping pointers during presentations; and the conference proceedings was yet another bulky book that I had to carry back home, only to receive a second copy of it a few weeks later in the mail in the form of SIGCSE Bulletin. I do not miss those days. But, I do find it remiss that conference attendees these days are often physically present, but electronically otherwise engaged, with their heads buried in the glow of their laptop screens. Maybe we are taking after our students after all?

Travel is one of the perks of being an educator. In that respect, ITiCSE has been a god-send. When I finally hang up my dry-erase markers and kick back in the rocking chair, I am sure I will be able to track my years in the profession not by the courses I taught, but by recalling the places I visited, courtesy of ITiCSE: Dover castle, dinner cruise on the Seine, Fishamble street, Macchu Picchu, Hagia Sophia, Bahai template in Haifa, Wieliczka salt mine, and the list goes on. Who says you cannot have your cake and eat it too?

SIGCSE and ITiCSE conferences have been the metronome of my professional life for the last two decades. Twice a year at these conferences, I get to stick my pedagogical finger up in the air and sense the direction in which computer science education currents are blowing. Reminiscing about these events has reminded me how pivotal they have been to my career. And how much my academic life revolves around them. ❖

Amruth N. Kumar
Computer Science
Ramapo College of New Jersey
505 Ramapo Valley Road
Mahwah, NJ, USA
amruth@ramapo.edu

DOI: 10.1145/3276308

Much Better Late than Never

Jodi L. Tims, *Baldwin Wallace University*

I was a little bit late arriving to the SIGCSE party. I had a good reason, of course. For the first fourteen years of my CS education career, I had been teaching full time and working on my PhD part time, which left no room for other types of professional activities. After receiving a PhD in 1998, I took a few years to catch my breath and then began to look for a professional community that would allow me to further develop as a computer science educator. In 2002, I was traveling to Ohio to interview for open faculty positions when I realized that SIGCSE was being held in nearby Covington, Kentucky. I had heard positive things about the symposium, so decided to attend and see what it had to offer. What I found was a community that quickly drew me in and became the professional family whose party I now look forward to every year. I haven't missed one since.

Like any good family, SIGCSE is supportive of me and my career. In 2010, Susan Williams (Georgia Southern University) and I were awarded a SIGCSE special projects grant that we used to conduct a survey of non-PhD granting departments. The goal was to produce data like that reported by the Taulbee study of PhD granting institutions. This project, along with a similar project led by Mikey Goldweber (Xavier University) led to the launching of the *ACM Annual Study of Non-Doctoral-Granting Departments in Computing (NDC)*. I feel privileged to have stayed involved with the project along with colleagues Stu Zweben (Ohio State University), Yan Timanovsky (ACM Education Manager), and Jane Prey (ACM Education Board). SIGCSE continues to support this project by publishing the NDC report in *ACM Inroads* each September.

A few years later, the opportunity to give something back to the community came. In 2013, I served on my first SIGCSE symposium organizing committee as Poster Chair. Additional opportunities soon followed—Program Chair (2015), Symposium Chair (2016), and Sponsor Liaison (2018). Planning symposia is a LOT of work, but the reward has been well worth the effort. My network of colleagues and friends continues to grow to include those who have also served SIGCSE in this way and I look forward to seeing all of them when we gather each spring.

The future of SIGCSE is indeed a bright one. I, for one, can hardly wait so see what's ahead as the work of this community continues to transform computer science education.

I encourage anyone who is asked to serve in this way to jump at the chance.

SIGCSE has seen considerable growth in recent years, largely due to increased participation of those from the K-12 and community college sectors. This growth brings some challenges to the organization. As the annual conference evolves to accommodate larger attendance and provide a program that offers something for every type of attendee, some may find it more difficult to spot among the larger crowd the colleagues they only run into once a year. It may be more difficult to choose from the growing number of parallel sessions in the program and it is impossible attend all sessions of interest. But the payoff for SIGCSE is well worth the individual effort to adapt as the inclusion of new perspectives enriches our organization, giving all members the opportunity to understand the full spectrum of computer science education.

The future of SIGCSE is indeed a bright one. I, for one, can hardly wait so see what's ahead as the work of this community continues to transform computer science education. ❖

Jodi L. Tims
Department of Computer Science and Engineering
Baldwin Wallace University
275 Eastland Rd
Berea, OH USA
jltims@bw.edu

DOI: 10.1145/3230694

My SIGCSE → ITiCSE

Mats Daniels, *Uppsala University*

My SIGCSE is rather my ITiCSE, since this conference has become part of my identity. This is not just professional, it is also very personal. Education has always been special to me and attending conferences, such as SIGCSE and ITiCSE, has been essential to get a better foundation for my strong drive to improve education, both at Uppsala University and elsewhere. Thinking back to my first conferences the main impression is "coming home," to be where I belong among positive and supporting people discussing matters that I deeply care about. This is also evident in my line of research. I started my academic career as a researcher in formal methods in computer science, specifically in timing aspects in computer architecture and computer networks and subsequently changed to the area of discipline-based education research. Attending SIGCSE and ITiCSE has been instrumental in this shift as well as the creation of the Uppsala Computing Education research Group (UpCERG).

My ITiCSE engagement started at SIGCSE 1996, when Vicki Almstrum introduced me to Boots Cassel, chair of SIGCSE. Vicki was about to spend a sabbatical at my department and had planted the idea in Boots that the second ITiCSE could be run in Uppsala. We had a constructive meeting and as a consequence I went to the first ITiCSE in Barcelona 1996 and took on the role as co-chair for ITiCSE in Uppsala 1997. I was the program chair for ITiCSE in Dublin 1998 and co-program chair 1999 when it was held in Krakow. So, I went from not knowing about ITiCSE to getting a solid dose in a few years. I really enjoyed the character of the conference and the community it was forming. The people organizing the first conference in Barcelona set the tone of a place where people met and where those attending felt that they were visiting a special place. The sense of being somewhere special was not least due to providing an opportunity to experience the local cuisine.

Other characteristics of ITiCSE were that attendees and presenters came from a wide variety of countries and that the special format of the working groups in many cases provided a strong bond between its members. It has been part of my agenda to preserve and develop those characteristics in my role as a member of the team that scouted out potential future sites for the conference. I started doing this 2005 under the excellent mentorship of Bruce Klein, and since his retirement I have continued to do this together with Alison Clear and Michael Goldweber. This position has strengthened my bond to ITiCSE and provided numerous opportunities for interesting and enjoyable discussions with those wanting to host the conference. There have been discussions about opportunities and potential difficulties, the inspection of locations, and learning about local strengths. Another important aspect of placing the conference is the expansion of the community worldwide, and particularly in Europe. The community aspect is also addressed by discussing how to form a conference committee, where we have encouraged conference chairs to aim for a mixture of experience and "new blood." This scouting is all in all a very inspiring task from which I will soon resign. Two newly appointed persons for this exciting and, as I see it, important role are Anna Eckerdal and Ari Korhonen. ❖

I started my academic career as a researcher in formal methods in computer science, specifically in timing aspects in computer architecture and computer networks and subsequently changed to the area of discipline-based education research. Attending SIGCSE and ITiCSE has been instrumental in this shift …

Mats Daniels
Department of Information Technology
Uppsala University
Box 337
751 05 Uppsala, Sweden
mats.daniels@it.uu.se

DOI: 10.1145/3231745

SIGCSE: A Pause to Look Back Over the 50 Year Journey

Elva J. Jones, *Winston-Salem State University*

As I recall my over forty-year experience with SIGCSE, I realize the exhilaration of innovation that attracted me to the dynamic field of computing, also attracted me to SIGCSE year after year. SIGCSE was my source for continuing education and leadership development, and it formed my professional community.

FIRST IMPRESSIONS–LASTING FRIENDSHIPS

During the early years the SIGCSE Technical Symposium was small enough to allow development of strong professional affiliations. I was welcomed to the symposium by Della Bonnette, Joe Turner, and Gerald Engel at varying stages during those early years. I had no idea these were the leaders of the organization. Yet, these individuals continued to impact my professional development for many years after my initial introduction to them at SIGCSE. My years of membership and service to SIGCSE have strengthened the importance of professional association involvement.

CONTINUING EDUCATION

The past fifty years have seen a changing landscape for SIGCSE and provided a dynamic window on future trends for teaching computer science. As a professor at a small Historically Black College/University (HBCU), the most challenging barrier to currency and professional development was funding for travel. ACM SIGCSE not only provided a high-quality conference, but at a reasonable cost for a professional who only received funding to attend one conference per year. This conference allowed faculty to interact with the largest and most innovative technical companies, with faculty from all over the world, as well as provided an opportunity to expose our undergraduate students to a professional conference at only the cost of transportation to the venue. The conference was always preceded and followed by outstanding workshops and these workshops provided a source for high quality continuing education for attendees.

LEADERSHIP DEVELOPMENT

SIGCSE was, and remains, a vehicle for expanding influence beyond the boundaries of the conference. I began my journey with SIGCSE when I was a lecturer and have remained active with the organization throughout my professional journey to Assistant Professor, Associate Professor, Professor, and Department Chairperson. Not only have there been individual benefits, but organizational benefits as well. Features such as Birds-Of-A-Feather (BOF) allowed a diverse group of HBCU faculty to meet and brainstorm ways to advance our computing curriculums. One of the oldest computing organizations for minority serving institutions, the Association of Computer Information Sciences and Engineering Departments at Minority Institutions (ADMI), was formed out of discussions held at ACM SIGCSE conferences. ADMI is in its twenty third year of sponsoring its annual research symposium. For many years after the organization was formed, SIGCSE was a site for one of the ADMI Board Meetings focused on planning the annual symposium for the following year. BOF sessions allowed ADMI to connect with faculty from HBCUs, to collaborate with faculty with majority serving institutions, and connect with professionals from funding agencies such as the National Science Foundation. Additionally, many HBCUs received their introduction to program accreditation at BOFs held at SIGCSE conferences. Consequently, the number of ABET accredited institutions has grown continually since the early 1990s.

SIGCSE was, and remains, a vehicle for expanding influence beyond the boundaries of the conference

SIGCSE has also been the site for collaborative grant writing. Two NSF Broadening Participation grants on which I served as Co-PI (ARTSI and iAAMCS) were born out of "collaborative SIGCSE" sessions. ARTSI (Advancing Robotics Technology for Societal Impact) provided the support for a group of HBCU faculty and students to collaborate with Research-1 faculty in delivering robotics instruction to our students and providing summer research opportunities at national labs and R-1 institution labs. Faculty from Hampton, Winston-Salem State, University of the District of Columbia (UDC), Norfolk State, Spelman, Tennessee State, Jackson State, FAMU and other HBCUs found SIGCSE the perfect place to brainstorm, and thus became the perfect mid-year site for meeting with R-1 faculty

from CMU, Duke, Rice, Brown, Georgia Tech, University of Alabama, University of Pittsburg and many others.

The SIGCSE collaborations continue, as we find our current iAAMCS ("I am CS"—Institute for African American Mentoring in Computing Sciences) grant uses SIGCSE as one of the key conferences for disseminating our research findings. The NSF program manager for both these grants is an avid SIGCSE participant. SIGCSE continues to be a networking hub!

Participation in SIGCSE has ensured continued growth and education as well as providing the opportunity to participate in the broader computing community.

PROFESSIONAL COMMUNITY

Participation in SIGCSE has ensured continued growth and education as well as providing the opportunity to participate in the broader computing community. SIGCSE has made great strides in promoting industry/academia interaction by engaging and leveraging industry conference presentations as well as exhibit hall demonstrations and presentations. Some of the largest and most influential industry leaders such as IBM, Google, and Microsoft have been present and provided substantial impact on the success on the symposia. Over the years I have collaborated with individuals and groups from these industry partners to leverage internships for our students, software and equipment grants for our programs, and to gain access to a wealth of speakers and consultants for continually improving our academic programs.

I hope that by sharing my SIGCSE journey, I can show that this professional organization not only impacts the established players, but also those new to the field. SIGCSE had reinforced my belief that it is possible to achieve amazing things and one must leverage all the resources at one's disposal. As we celebrate the 50th anniversary, I encourage our computing colleagues to get engaged with SIGCSE activities and volunteer to advance the impact of this organization on our community. ❖

Elva J. Jones, Professor
Winston-Salem State University
College of Arts Sciences Business and Education
Department of Computer Science
Winston-Salem, NC 27110
jonese@wssu.edu

DOI: 10.1145/3230702

COMMON ACRONYMS

AACC	American Association of Community Colleges *https://www.aacc.nche.edu/*
ABET	Accreditation Board for Engineering and Technology *http://www.abet.org/*
ACM	Association for Computing Machinery *https://www.acm.org/*
ACM-W	ACM Council on Women in Computing *https://women.acm.org/*
ADMI	Association of Computer/Information Sciences and Engineering Departments at Minority Institutions *https://www.admiusa.org/*
AIS	Association for Information Systems *https://aisnet.org/*
AISES	The American Indian Science and Engineering Society *http://www.aises.org/*
CPSR	Computer Professionals for Social Responsibility *http://cpsr.org/*
CRA	Computer Research Association *https://cra.org/*
EngageCSEdu	Engage CS EDU *https://www.engage-csedu.org/*
HBCU	Historically Black Colleges and Universities *https://www.niche.com/blog/list-of-hbcu-schools-in-america-2/*
iAAMCS	Institute for African-American Mentoring in Computing Sciences ('i am cs') *http://www.iaamcs.org/*
IEEE	Institute of Electrical and Electronics Engineers *https://www.ieee.org/index.html*
MAA	Mathematical Association of America *https://www.maa.org/*
MAES	Latinos in Science and Engineering *http://mymaes.org/*
NCWIT	National Center for Women and Technology *https://dev.ncwit.org/*
NSBE	National Society of Black Engineers *http://www.nsbe.org/home.aspx*
SHPE	Society of Professional Hispanic Engineers *http://www.shpe.org/*
WICS	Women in Computer Science *https://www.computerscience.org/resources/women-in-computer-science/*

ACM Special Interest Groups

SIGACCESS	Accessibility and Computing	*http://www.sigaccess.org/*
SIGCAS	Computers and Society	*http://www.sigcas.org/*
SIGCHI	Computer-Human Interaction	*http://www.sigchi.org/*
SIGCSE	Computer Science Education	*http://www.sigcse.org/*
SIGGRAPH	Computer Graphics	*http://www.siggraph.org/*
SIGHPC	High Performance Computing	*http://www.sighpc.org/*
SIGITE	Information Technology Education	*http://www.sigite.org/*
SIGMIS	Management Informaton Systems	*http://www.sigmis.org/*
SIGPLAN	Programming Languages	*http://www.sigplan.org/*
SIGSOFT	Software Engineering	*https://www.sigsoft.org/index.html*

DOI: 10.1145/3284669

ACM ON A MISSION TO SOLVE TOMORROW.

Dear Colleague,

Without computing professionals like you, the world might not know the modern operating system, digital cryptography, or smartphone technology to name an obvious few.

For over 70 years, ACM has helped computing professionals be their most creative, connect to peers, and see what's next, and inspired them to advance the profession and make a positive impact.

We believe in constantly redefining what computing can and should do.

ACM offers the resources, access and tools to invent the future. No one has a larger global network of professional peers. No one has more exclusive content. No one presents more forward-looking events. Or confers more prestigious awards. Or provides a more comprehensive learning center.

Here are just some of the ways ACM Membership will support your professional growth and keep you informed of emerging trends and technologies:

- Subscription to ACM's flagship publication ***Communications of the ACM***
- Online books, courses, and videos through the **ACM Learning Center**
- Discounts on registration fees to ACM Special Interest Group conferences
- Subscription savings on specialty magazines and research journals
- The opportunity to subscribe to the **ACM Digital Library**, the world's largest and most respected computing resource

Joining ACM means you dare to be the best computing professional you can be. It means you believe in advancing the computing profession as a force for good. And it means joining your peers in your commitment to solving tomorrow's challenges.

Sincerely,

Cherri M. Pancake
President
Association for Computing Machinery

Association for Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING.
JOIN ACM TODAY.

ACM is the world's largest computing society, offering benefits and resources that can advance your career and enrich your knowledge. We dare to be the best we can be, believing what we do is a force for good, and in joining together to shape the future of computing.

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

- ❑ Professional Membership: $99 USD
- ❑ Professional Membership plus ACM Digital Library: $198 USD ($99 dues + $99 DL)
- ❑ ACM Digital Library: $99 USD (must be an ACM member)

ACM STUDENT MEMBERSHIP:

- ❑ Student Membership: $19 USD
- ❑ Student Membership plus ACM Digital Library: $42 USD
- ❑ Student Membership plus Print *CACM* Magazine: $42 USD
- ❑ Student Membership with ACM Digital Library plus Print *CACM* Magazine: $62 USD

❑ **Join ACM-W:** ACM-W supports, celebrates, and advocates internationally for the full engagement of women in computing. Membership in ACM-W is open to all ACM members and is free of charge.

Priority Code: CAPP

Payment Information

Name

ACM Member #

Mailing Address

City/State/Province

ZIP/Postal Code/Country

❑ Please do not release my postal address to third parties

Email

❑ Yes, please send me ACM Announcements via email

❑ No, please do not send me ACM Announcements via email

Purposes of ACM

ACM is dedicated to:

1) Advancing the art, science, engineering, and application of information technology
2) Fostering the open interchange of information to serve both professionals and the public
3) Promoting the highest professional and ethics standards

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc., in U.S. dollars or equivalent in foreign currency.

❑ AMEX ❑ VISA/MasterCard ❑ Check/money order

Total Amount Due

Credit Card #

Exp. Date

Signature

Return completed application to:
ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

Prices include surface delivery charge. Expedited Air Service, which is a partial air freight delivery service, is available outside North America. Contact ACM for more information.

Satisfaction Guaranteed!

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.

Association for Computing Machinery

1-800-342-6626 (US & Canada)
1-212-626-0500 (Global)

Hours: 8:30AM - 4:30PM (US EST)
Fax: 212-944-1318

acmhelp@acm.org
acm.org/join/CAPP

BACKPAGE

Decade Matching

By John Barr, *Ithaca College*

John Barr, Back Page Editor
Department of Computer Science
Ithaca College
953 Danby Road
Ithaca, New York 14850 USA
barr@ithaca.edu

Instructions: Have you played Asteroids in an arcade?
Programmed a PDP-11? > Taught Pascal? > ***Then you should ace this puzzle!***

Connect the articles with the people, computers, and games from the same decade!
Note that there may be multiple images in a decade for each category.

COMPUTER IMAGES COURTESY OF THE COMPUTER HISTORY MUSEUM (HTTP://WWW.COMPUTERHISTORY.ORG/).
ILLUSTRATIONS: ©WWW.ISTOCKPHOTO.COM/HOMUNKULUS28 (STAR FIELD); GSSHOT (BINARY BACKGROUND); HARDIK PETHANI (CIRCUITY BACKGROUND); LUYALI (GAME BACKGROUND); NICESCENE (PICKACHU TOY CHARACTER).

DOI: 10.1145/3230685

Back Page Solution: Decade Matching

COMPUTER IMAGES COURTESY OF THE COMPUTER HISTORY MUSEUM (HTTP://WWW.COMPUTERHISTORY.ORG/). ILLUSTRATIONS: ©WWW.ISTOCKPHOTO.COM/HOMUNKULUS28 (STAR FIELD); GSSHOT (BINARY BACKGROUND); HARDIK PETHANI (CIRCUITY BACKGROUND); LUYALI (GAME BACKGROUND); NICESCENE (PICKACHU TOY CHARACTER).

Association for Computing Machinery

ACM Chuck Thacker Breakthrough in Computing Award

The "ACM Breakthrough Award"

Nominations Solicited

Nominations are invited for the inaugural 2018 ***ACM Charles P. "Chuck" Thacker Breakthrough in Computing Award*** (the "ACM Breakthrough Award").

ACM Turing Laureate Charles P. (Chuck) Thacker (1943–2017) received the 2009 ACM A.M. Turing Award for "the pioneering design and realization of the first modern personal computer—the Alto at Xerox PARC—and seminal inventions and contributions to local area networks (including the Ethernet), multiprocessor workstations, snooping cache coherence protocols, and tablet personal computers."

The award was established in recognition of Thacker's pioneering contributions in computing.

These contributions are considered by the community to have propelled the world in the early 1970s from a visionary idea to the reality of modern personal computing, providing people with an early glimpse of how computing would deeply influence us all. The award also celebrates Thacker's long-term inspirational mentorship of generations of computer scientists.

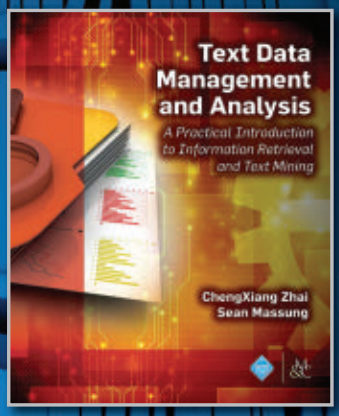
The Breakthrough Award will recognize individuals with the same out-of-the-box thinking and "can-do" approach to solving the unsolved that Thacker exhibited. The recipient should be someone who has made a surprising or disruptive leapfrog in computing ideas or technologies that provides a new capability or understanding that influences the course of computing technologies in a deep and significant manner through its numerous downstream influences and outcomes. Due to the breakthrough nature of the award it is expected that it will be presented biennially and will not be presented if there is no candidate who meets the criteria in a particular year.

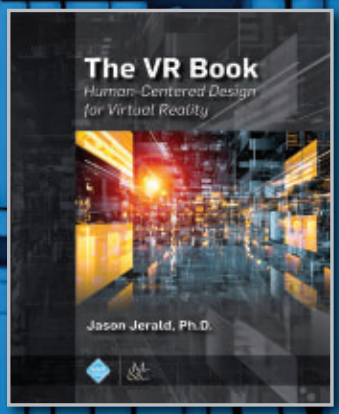
The award is accompanied by a prize of $100,000 and would be presented at the annual ACM Awards Banquet. The award recipient would be expected to give the *ACM Breakthrough Lecture* at a major ACM conference of his or her choice during the year following the announcement. The travel expenses of the recipient, and a companion, to attend the Lecture are supported by the award. Financial support of the Thacker Award is provided by Microsoft.

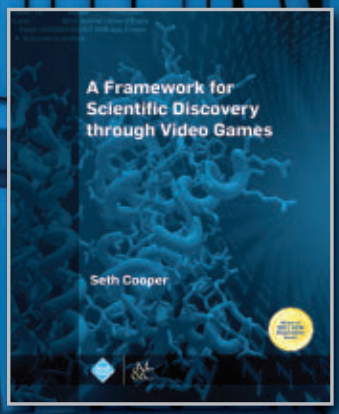
Nomination information and the online submission form are available on:
https://awards.acm.org/thacker/nominations

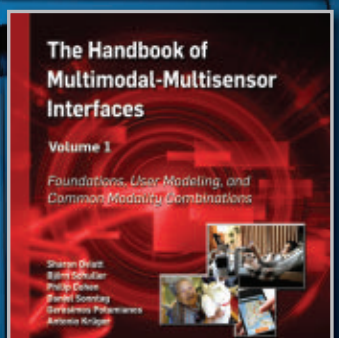
The deadline for nominations/endorsements is:
January 15, 2019, End of Day, AoE, UTC-12 hours.

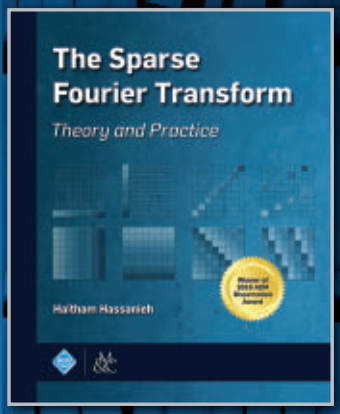
For additional information on ACM's award program please visit:
www.acm.org/awards/

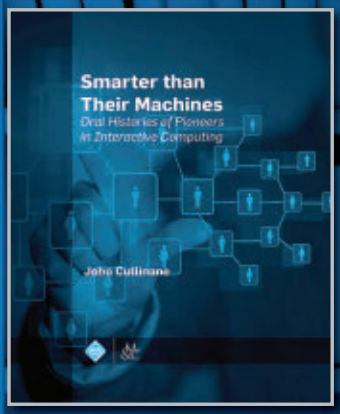


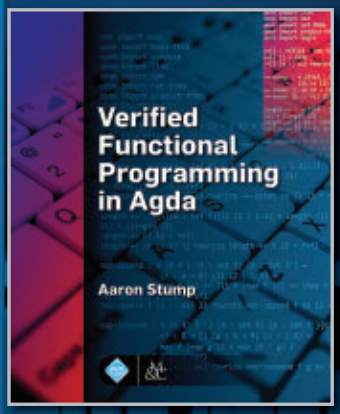












In-depth. Innovative. Insightful.

Inspired by the need for high-quality computer science publishing at the graduate, faculty and professional levels, ACM Books is affordable, current, and comprehensive in scope.

For more information, please visit
http://books.acm.org

Association for Computing Machinery
2 Penn Plaza, Suite 701, New York, NY 10121-0701, USA
Phone: +1-212-626-0658 Email: acmbooks-info@acm.org