Curricular Practices in Computer Science

Introduction

Prior curricular guidelines enumerated issues in the design and delivery of computer science curriculum. Given the increased importance of these issues, in CS2023, peer-reviewed, well-researched, in-depth articles were solicited from recognized experts on how computer science educators could address these issues in their teaching practices. These articles complement the CS2023 curricular guidelines. Whereas curricular guidelines list what should be covered in the curriculum, these articles describe how and why they could best be covered, including challenges, state of the art practices, etc.

The articles may be categorized as covering the following.

- **Social aspects,** including teaching about accessibility, computer science for social good, responsible computing, and ethics in the global souths.
- **Pedagogical considerations,** including CS + X, the role of formal methods in computer science, quantum computing education, and the impact of generative AI on programming instruction.
- **Educational practices,** in varied settings such as liberal arts institutions, community colleges, and polytechnic institutes.

The articles provide a “lay of the land,” a snapshot of the current state of the art of computer science education. They are not meant to advocate specific approaches or viewpoints, but rather help computer science educators weigh their options and make informed decisions about the appropriate option for their degree program.

The computer science education community was invited to provide feedback and suggestions on the first drafts of most of these articles. Several of the articles have been or are in the process of being published in peer-reviewed conferences and journals. In this section, self-contained summaries of most of the articles have been included. The full articles themselves will be accessible at the csed.acm.org website.

In addition, to globalize computer science education, articles were also invited on educational practices in various parts of the world. (See *ACM Inroads*, Special Issue, 15, 1 (March 2024)). It is hoped that these articles will foster mutual understanding and exchange of ideas, engender transnational collaboration and student exchange, and serve to integrate computer science education at the global level through shared understanding of its challenges and opportunities.

Social Aspects

Accessibility is about making computing systems accessible to people with disabilities and designing technical solutions for accessibility problems faced by people with disabilities. The article "**Teaching about Accessibility in Computer Science Education**" explains the practical, intellectual, and social reasons for integrating accessibility into the computer science curriculum.

The article "**Computing for Social Good in Education**" highlights how computing education can be used to improve society and address societal needs while also providing authentic computing environments in education. The authors discuss approaches, challenges, and benefits of incorporating computing for social good into computer science curriculum.

Given the pervasive use of computing in society, educators would be remiss not to teach their students about the principles of responsible computing. How they should go about doing so is explored in the article "**Multiple Approaches for Teaching Responsible Computing**." It uses research in the social sciences and humanities to transform responsible computing into an integrated consideration of values throughout the lifecycle of computing products.

In a globalized world, applications of computing transcend national borders. In this context, making ethics at home in global computer science education is about helping students relate to values within and beyond their own contexts. The article "**Making Ethics at Home in Global CS Education: Provoking Stories from the Souths**" presents storytelling as a mechanism that educators can use to engage students with "ethos building."

Pedagogical Considerations

"**CS + X: Approaches, Challenges, and Opportunities in Developing Interdisciplinary Computing Curricula**" states how interdisciplinary majors that apply computational methods in natural sciences, social sciences, humanities, and the arts can broaden participation in computing and reach a larger group of students.

"**The Role of Formal Methods in Computer Science Education**" makes the case for incorporating formal methods in computer science education. It lists the multiple ways in which formal methods can be incorporated into the undergraduate computer science curriculum and buttresses its advocacy of formal methods with testimonials from the industry.

"**Quantum Computing Education: A Curricular Perspective**" presents the current state of art in quantum computing and uses the results of a pedagogic experiment to illustrate that quantum computing education is within reach of even school children. It presents three curricular approaches for incorporating quantum computing in undergraduate computer science curriculum.

"**Generative AI in Introductory Programming**" explores how generative AI tools based on Large Language Models (LLMs) such as ChatGPT might affect programming education including how these tools can be used to assess student work, provide feedback, and to act as always-available virtual teaching assistants in introductory programming courses.

One issue with the study of databases/data management is that the number of possible topics far exceeds the bandwidth of a single undergraduate computer science course. “**The 2022 Undergraduate Database Course in Computer Science: What to Teach?**” presents multiple viewpoints on what a single undergraduate course in Databases/Data Management should cover.

Educational Practices

No curricular guidelines are complete by themselves. They must be adapted to local strengths, constraints, and needs. In this regard, “**Computer science Curriculum Guidelines: A New Liberal Arts Perspective**” provides a process to adapt CS2023 to the needs of liberal arts colleges that constrain the size of the computer science coursework in order to expose students to a broad range of liberal arts subjects.

Community and polytechnic colleges across the world offer specialized programs that help students focus on specific educational pathways. They award academic degrees that enable students to transfer to four-year colleges and are attuned to the needs of the local workforce. “**Computer Science Education in Community Colleges**” presents the context and perspective of community college computer science education.

Teaching about Accessibility in Computer Science Education

Richard E. Ladner, University of Washington, Seattle, WA, USA
Stephanie Ludi, University of North Texas, Denton, TX, USA
Robert J. Domanski, Hunter College (CUNY), New York, NY, USA

Accessibility, in the context of computer science, is about making computing products accessible to people with disabilities. This means designing hardware and software products that can be used effectively by people who have difficulty reading a computer screen, hearing computer prompts, or controlling the keyboard, mouse, or touchscreen. Thus, accessibility topics should be woven into any course about human-facing applications or websites, such as app and web design/development, software engineering, and human-computer interaction. In addition, accessibility is about creating technical solutions to accessibility problems that people with disabilities encounter in everyday living. These technical solutions may include the use of artificial intelligence, computer vision, natural language processing, or other CS topics. Thus, accessibility topics can be included in technical courses, particularly those that incorporate projects where students attempt to solve accessibility problems using techniques taught in the course. There are practical, intellectual, and social reasons to integrate accessibility into computer science curriculum. From a practical standpoint, employers increasingly include accessibility knowledge in job descriptions because they want their products and services to be accessible to more customers and for legal compliance. From an intellectual standpoint, technical solutions to many accessibility problems often require creativity and a multi-disciplinary approach that includes understanding user needs integrated with technical knowledge. From a social standpoint, accessibility is an important topic in addressing inclusivity and an attractive topic for those students who enter the field to do social good, leading to a broader mix of students in terms of gender, race, ethnicity, and ability.

Helpful Resources:

[1] Catherine Caldwell-Harris, & Chloe Jordan. 2014. Systemizing and special interests: Characterizing the continuum from neurotypical to autism spectrum disorder. *Learning and Individual Differences*. Volume 29, Issue 2014, 98-105. https://doi.org/10.1016/j.lindif.2013.10.005.

[2] CAIR: RIT Center for Accessibility and Inclusion Research; http://cair.rit.edu/projects.html**.** accessed September 7, 2022.

[3] Robert F. Cohen, Alexander V. Fairley, David Gerry, and Gustavo R. Lima. 2005. Accessibility in introductory computer science. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education (SIGCSE '05)*. Association for Computing Machinery, New York, NY, USA, 17–21. https://doi.org/10.1145/1047344.1047367.

[4] Robert F. Dugan Jr (2011) A survey of computer science capstone course literature, *Computer Science Education*, 21:3, 201-267, https://www.tandfonline.com/doi/abs/10.1080/08993408.2011.606118. Accessed March 2024.

[5] Kristen Shinohara, Saba Kawas, Amy J. Ko, and Richard E. Ladner. 2018. Who Teaches Accessibility? A Survey of U.S. Computing Faculty. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18*). Association for Computing Machinery, New York, NY, USA, 197–202. https://doi.org/10.1145/3159450.3159484.

[6] Stephanie Ludi, Matt Huenerfauth, Vicki Hanson, Nidhi Rajendra Palan, and Paula Conn. 2018. Teaching Inclusive Thinking to Undergraduate Students in Computing Programs. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 717–722. DOI: https://doi.org/10.1145/3159450.3159512.

[7] Alannah Oleson, Amy J. Ko, Richard Ladner (Eds.) (2023). Teaching Accessible Computing. Self-Published. https://bookish.press/tac. Accessed November 28, 2023.

[8] PEAT; https://www.peatworks.org/. Accessed January 5, 2023.

[9] Teach Access website, http://www.teachaccess.org. Accessed September 10, 2022.

[10] Kendra Walther and Richard E. Ladner. 2021. Broadening participation by teaching accessibility. *Communications of the ACM* 64, 10 (October 2021), 19–21. https://doi.org/10.1145/3481356.

[11] WCAG https://www.w3.org/WAI/standards-guidelines/wcag/ Accessed November 6, 2022.

[12] Jacob O. Wobbrock, Shaun K. Kane, Krzysztof Z. Gajos, Susumu Harada, and Jon Froehlich. 2011. Ability-Based Design: Concept, Principles and Examples. *ACM Transactions on Accessible Computing* 3, 3, Article 9 (April 2011), 27 pages. https://dl.acm.org/doi/10.1145/1952383.1952384.

Computing for Social Good in Education

Heidi J. C. Ellis, Western New England University, Springfield, MA, USA
Gregory W. Hislop, Drexel University, Philadelphia, PA, USA
Mikey Goldweber, Denison University, Granville, OH, USA
Sam Rebelsky, Grinnell College, Grinnell, IA, USA
Janice L Pearce, Berea College, Berea, KY, USA
Patti Ordonez, University of Maryland Baltimore County, Baltimore, MD, USA
Marcelo Pias, Universidade Federal do Rio Grande, Rio Grande, Brazil
Neil Gordon, University of Hull, Hull, UK

Computing for Social Good (CSG) encompasses the potential of computing to have a positive impact on individuals, communities, and society, both locally and globally. Incorporating CSG into education (CSG-Ed) is especially relevant as computing has more and more impact across all areas of society and daily life. Educators can address CSG-Ed through a variety of means [2]. A simple way to start is by modifying a single assignment within a single course by updating the domain of the assignment to be one with social impact. The use of this domain can then be expanded across several assignments within the same course or across several courses. The domain could also provide the opportunity for collaborations across related departments. Indeed, some countries, such as England, integrate CSG throughout the curriculum starting before higher education studies [6].

Another way that educators may support CSG-Ed is the adoption or creation of a classroom project that solves a social problem either for a campus organization or from the larger community [1]. This approach allows students to see the impact of their work within their own community. On a larger scale, participation in established projects with national or global scope allows students to understand the breadth of influence that computing can have. Such efforts align well with institutions that have a service-learning requirement [4]. In addition, hackathons, code-days, clubs, and other extracurricular activities allow students to understand the social impact of computing outside of the classroom.

There are several challenges to integrating computing for social good into higher education [3]. One challenge is that instructors may not be inclined to incorporate new topics fearing that it could disrupt the curriculum or require course rework. Instructor time is a second barrier where it may take time to understand CSG domains and create new assignments. The interdisciplinary nature of many CSG topics may also require collaborating with other departments, disciplines, or community partners resulting in additional course preparation time. CSG-Ed assignments may result in the discussion of social issues within the classroom that could require instructors to prepare to discuss these issues with students. In addition, there appears to be a shortage of coverage of CSG in textbooks.

While barriers to CSG-Ed adoption exist, this focus of computing education provides multiple opportunities. CSG-Ed provides the possibility for students to connect with real-world problems to understand the complexity of computing while also apprehending the social impact of computing [5]. Students can be motivated by engaging in solving local problems that directly impact themselves or their community. They can also gain a better understanding of global citizenship and responsibility by participating in social projects that have a global scale.

There are several areas of future investigation including creation of a repository of CSG-Ed materials, addressing project-related challenges, exploring open source in CSG-Ed, and approaches for creating and growing an inclusive community to support CSG-Ed.

References

[1] Grant Braught, Steven Huss-Lederman, Stoney Jackson, Wes Turner, and Karl R. Wurst. 2023. Engagement Models in Education-Oriented H/FOSS Projects. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V*. (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 409–415. https://doi.org/10.1145/3545945.3569835

[2] Michael Goldweber, John Barr, Tony Clear, Renzo Davoli, Samuel Mann, Elizabeth Patitsas, and Scott Portnoff. 2012. A framework for enhancing the social good in computing education: a values approach. In *Proceedings of the final reports on Innovation and technology in computer science education 2012 working groups (ITiCSE-WGR '12).* Association for Computing Machinery, New York, NY, USA, 16–38. https://doi.org/10.1145/2426636.2426639.

[3] Mikey Goldweber, Lisa Kaczmarczyk, and Richard Blumenthal. Computing for the social good in education. *ACM Inroads*, 10, 4 (Dec 2019): 24–29.

[4] Janice L. Pearce. Requiring outreach from a CS0-level robotics course. *J. Comput. Sci. Coll.* 26, 5 (May 2011), 205–212.

[5] Lori Postner, Darci Burdge, Stoney Jackson, Heidi Ellis, George W. Hislop, and Sean Goggins. Using humanitarian free and open source software (HFOSS) to introduce computing for the social good. *SIGCAS Comput. Soc.* 45, 2 (June 2015), 35. https://doi.org/10.1145/2809957.2809967.

[6] Computer Science GCSE Subject Content. https://assets.publishing.service.gov.uk/media/5a7e3cb440f0b62305b81b02/Computer_Science_GCSE_-_subject_content_-_final.pdf. Accessed 26 Nov. 2023.

Multiple Approaches for Teaching Responsible Computing

Stacy A. Doore, Colby College, Waterville, ME, USA
Atri Rudra, University at Buffalo, Buffalo, NY, USA
Michelle Trim, University of Massachusetts, Amherst, MA, USA
Joycelyn Streator, Prairie View A&M University, Prairie View, TX, USA & the Mozilla Foundation
Richard Blumenthal, Regis University, Colorado, CO, USA
Bobby Schnabel, University of Colorado, Boulder, CO, USA

Teaching applied ethics in computer science (and computing in general) has shifted from a perspective of teaching about professional codes of conduct and an emphasis on risk management towards a broader understanding of the impacts of computing on humanity and the environment. This shift has produced a diversity of approaches for integrating responsible computing instruction into core computer science knowledge areas and for an expansion of dedicated courses focused on computing ethics. There is an increased recognition that students need intentional and consistent opportunities throughout their computer science education to develop the critical thinking, analytical reasoning, and cultural competency skills to understand their roles and professional duties in the responsible design, implementation, and management of complex computing systems. Therefore, computing education programs are re-evaluating the ways in which students learn to identify and assess the impact of computing on individuals, communities, and societies along with other critical professional skills such as effective communication, workplace conduct, and regulatory responsibilities. One of the primary shifts in the new approach comes from interdisciplinary collaborations, combining computing, social sciences and humanities researchers who work together to help students identify potential biases, blind spots, impacts, and harms in applications or systems and examine underlying assumptions and competing values driving design decisions.

There are examples of how topics within the CS2023 Society, Ethics, and the Profession (SEP) knowledge area can be implemented and assessed with numerous links to current module repositories [1-6], lessons [7-11], and resources [12-21] to embed responsible computing teaching across the CS curriculum. There are specific recommendations and resources that will help address current barriers for moving forward with the integration of responsible computing practices in the classroom [22]. These include ways of being open and confident in honoring all students' prior knowledge and lived experiences in sometimes difficult conversations [23-24] and overcoming student apathy or resistance to embedding responsible computing content [25-26]. These strategies require a willingness to work within an interdisciplinary community to incorporate social science and humanities domain expertise within these classroom interactions [27-29]. There are also recommendations on how to bring undergraduate students into curriculum planning as many of the earliest responsible computing teaching models were co-developed with undergraduate CS students [30-32]. Finally, there are recommendations about distinguishing between often conflated concepts, associated with responsible computing such as social justice [33-35], trust and safety [36-38], and value-sensitive design and co-design [39-40]. The understanding and use of these principles and practices in the classroom communicate the importance of stakeholder groups and impacted community inclusion from the beginning of the technology development lifecycle and affirms the agentive role of that community in development decisions. We hope this contribution will assist instructors as they develop their learning

objectives, activities, and assessments while adding to the growing body of knowledge on the best practices for weaving responsible computing principles and content throughout the evolving ACM/IEEE/AAAI computing curricula.

References

[1] ACM Engage CSEdu Ethics Repository. https://www.engage-csedu.org/ethics-and-computing/repository. Accessed Feb 28, 2024.

[2] Embedded EthiCS @ Harvard University - Modules Repository. https://embeddedethics.seas.harvard.edu/. Accessed Feb 28, 2024.

[3] Computing Ethics Narratives and Modules Repository at Bowdoin College and Colby College. https://computingnarratives.com. Accessed Feb 28, 2024.

[4] Embedded Ethics in Computer Science at Stanford University - Modules Repository. https://embeddedethics.stanford.edu/. Accessed Feb 28, 2024.

[5] Embedded EthiCS Modules Repository at University of Toronto. https://www.cs.toronto.edu/embedded-ethics/modules/index.html. Accessed Feb 28, 2024.

[6] Responsible Computer Science Repository at Bemidji State University. https://www.bemidjistate.edu/academics/departments/mathematics-computer-science/rcs/. Accessed Feb 28, 2024.

[7] Integrating Social Responsibility into Core CS. https://evanpeck.github.io/projects/responsibleCS. Accessed Feb 28, 2024.

[8] Internet Rules Lab University of Colorado Boulder. https://www.internetruleslab.com/responsible-computing. Accessed Feb 28, 2024.

[9] Responsible Computer Science at Washington University at St. Louis. https://www.cse.wustl.edu/~cytron/RCS/. Accessed Feb 28, 2024.

[10] University of Miami Dade Responsible Computing Role Playing Lesson. https://news.mdc.edu/role-playing-scenario-developed-at-entec/. Accessed Feb 28, 2024.

[11] Georgia Tech Responsible Computing Science. https://sites.gatech.edu/responsiblecomputerscience/. Accessed Feb 28, 2024.

[12] Mozilla Responsible Computing Playbook. https://foundation.mozilla.org/en/responsible-computing-challenge-playbook/. Accessed Feb 28, 2024.

[13] Teaching Responsible Computing at University of Buffalo. https://foundation.mozilla.org/en/responsible-computing-challenge-

playbook/https://c4sg.cse.buffalo.edu/projects/Teaching%20Responsible%20Computing.html. Accessed Feb 28, 2024.

[14] Human Context and Ethics at UC Berkeley. https://foundation.mozilla.org/en/responsible-computing-challenge-playbook/ https://data.berkeley.edu/academics/human-contexts-and-ethics. Accessed Feb 28, 2024.

[15] Social & Ethical Responsibilities of Computing at MIT. https://foundation.mozilla.org/en/responsible-computing-challenge-playbook/ https://computing.mit.edu/cross-cutting/social-and-ethical-responsibilities-of-computing. Accessed Feb 28, 2024.

[16] Socially Responsible Computing @ Brown University. https://foundation.mozilla.org/en/responsible-computing-challenge-playbook/ http://ethics.cs.brown.edu/. Accessed Feb 28, 2024.

[17] Embedded Ethics Program at Georgetown University. https://foundation.mozilla.org/en/responsible-computing-challenge-playbook/ https://ethicslab.georgetown.edu/embedded-ethics. Accessed Feb 28, 2024.

[18] Ethical Computer Science at Allegheny College. Accessed Feb 28, 2024. https://foundation.mozilla.org/en/responsible-computing-challenge-playbook/ https://csethics.allegheny.edu/.

[19] Ethics 4 EU - Educational Resources. https://foundation.mozilla.org/en/responsible-computing-challenge-playbook/ https://ascnet.ie/ethics4eu-website/welcome-to-the-bricks/. Accessed Feb 28, 2024.

[20] Human Context and Ethics at UC Berkeley. https://foundation.mozilla.org/en/responsible-computing-challenge-playbook/ https://data.berkeley.edu/academics/human-contexts-and-ethics. Accessed Feb 28, 2024.

[21] Markkula Center for Applied Ethics at Santa Clara University- Technology Ethics. https://foundation.mozilla.org/en/responsible-computing-challenge-playbook/ https://www.scu.edu/ethics/focus-areas/technology-ethics/. Accessed Feb 28, 2024.

[22] Colleen Greer & Marty J. Wolf. 2020. Overcoming barriers to including ethics and social responsibility in computing courses. In *Societal Challenges in the Smart Society*, 131-144. Universidad de La Rioja.

[23] Rodrigo Ferreira & Moshe Y. Vardi. 2021. Deep tech ethics: An approach to teaching social justice in computer science. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 1041-1047.

[24] Michelle Trim & Paige Gulley. 2023. Imagining, generating, and creating: Communication as feminist pedagogical method for teaching computing ethics. In *Proceedings of the 41st ACM International Conference on Design of Communication*, 206-209.

[25] Nina Zuber, Jan Gogoll, Severin Kacianka, Alexander Pretschner, & Julian Nida-Rümelin. Empowered and embedded: ethics and agile processes. *Humanities and Social Sciences Communications*. 9, 1 (2022): 1-13.

[26] Shamika Klassen & Casey Fiesler. Run Wild a Little with Your Imagination: Ethical Speculation in Computing Education with Black Mirror." *In Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*. 1, (2022): 836-842.

[27] Barbara J. Grosz, David Gray Grant, Kate Vredenburgh, Jeff Behrends, Lily Hu, Alison Simmons, & Jim Waldo. 2019. Embedded EthiCS: Integrating ethics across CS education. *Communications of the ACM*, 62, 8 (2019): 54–61.

[28] National Academies of Sciences, Engineering, and Medicine. 2022. Fostering Responsible Computing Research: Foundations and Practices.

[29] Trystan S. Goetze. 2023. Integrating ethics into computer science education: Multi-, inter-, and transdisciplinary approaches. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, 1, (2023): 645-651.

[30] Beleicia B. Bullock, Fernando L. Nascimento, & Stacy A. Doore. Computing ethics narratives: Teaching computing ethics and the impact of predictive algorithms. *In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 2021: 1020-1026.

[31] Nora McDonald, Adegboyega Akinsiku, Jonathan Hunter-Cevera, Maria Sanchez, Kerrie Kephart, Mark Berczynski, and Helena M. Mentis. Responsible computing: A longitudinal study of a peer-led ethics learning framework. *ACM Transactions on Computing Education (TOCE)* 22,4 (2022): 1-21.

[32] Alexandra Gillespie. 2023. Designing an ethical tech developer. *Communications of the ACM,* 66, 3 (2023): 38-40.

[33] Ruha Benjamin. Race after technology, *Social Theory Re-Wired*, 405-415. Routledge (2023).

[34] Rachel Charlotte Smith, Heike Winschiers-Theophilus, Daria Loi, Rogério Abreu de Paula, Asnath Paula Kambunga, Marly Muudeni Samuel, & Tariq Zaman. Decolonizing design practices: towards pluriversality. In *Extended abstracts of the 2021 CHI conference on human factors in computing systems*. 1-5.

[35] Sasha Costanza-Chock. *Design Justice: Community-led Practices to Build the Worlds We Need.* The MIT Press, 2020.

[36] Ben Shneiderman. Bridging the gap between ethics and practice: guidelines for reliable, safe, and trustworthy human-centered AI systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 10,4 (2020): 1-31.

[37] Cansu Canca. 2020. Operationalizing AI ethics principles. *Communications of the ACM*, 63, 12 (2020): 18-21.

[38] International Organization for Standardization. Information Technology–Artificial Intelligence – Management Systems (ISO/IEC Standard 42001-2023). https://www.iso.org/standard/81230.html Accessed Feb 28, 2024.

[39] John M. Carroll. Encountering others: Reciprocal openings in participatory design and user-centered design. Human-computer Interaction 11,3 (1996), 285-290. https://foundation.mozilla.org/en/responsible-computing-challenge-playbook/. Accessed Feb 28, 2024.

[40] David G. Hendry, Batya Friedman, & Stephanie Ballard. Value sensitive design as a formative framework. *Ethics and Information Technology* 23, 23 (2021): 1-6.

Making ethics at home in Global CS Education: Provoking stories from the Souths

Marisol Wong-Villacres, Escuela Superior Politecnica del Litoral, Guayaquil, Ecuador
Cat Kutay, Charles Darwin University, Northern Territory, Australia
Shaimaa Lazem, City of Scientific Research and Technological applications, Alexandria, Egypt
Nova Ahmed, North South University, Dhaka, Bangladesh
Cristina Abad, Escuela Superior Politecnica del Litoral, Guayaquil, Ecuador
Cesar Collazos, Universidad del Cauca, Popayan, Colombia
Shady Elbassuoni, American University of Beirut, Beirut, Lebanon
Farzana Islam, North South University, Dhaka, Bangladesh
Deepa Singh, University of Delhi, New Delhi, India,
Tasmiah Tahsin Mayeesha, North South University, Dhaka, Bangladesh
Martin Mabeifam Ujakpa, Ghana Communication Technology University, Accra, Ghana
Tariq Zaman, University of Technology, Sibu, Malaysia
Nicola J. Bidwell, Charles Darwin University and University of Melbourne, Australia, International University of Management, Namibia

We, a group of thirteen educators in computing programs and researchers in universities, retell the stories of 46 university educators and practitioners in Latin America, South-Asia, Africa, the Middle East, and Australian First Nations who participated in surveys and interviews with us [1]. We use the plural of Global Souths to indicate the multiple and overlapping geographic and conceptual spaces that are negatively impacted by contemporary capitalist globalization and the US–European norms and values exported in computing products, processes, and education. The stories illustrate frictions between local practices, values, and impacts of technologies and the static, anticipatory approaches to ethics that computer science (CS) curricula often promote through codes of ethics. The stories show diverse perspectives on privacy and institutional approaches to confidentiality; compliance with regulations to attain various goals and difficulties when regulations are absent or ambiguously relate to practices; discrimination based on their gender or technical ability and minoritized positions; and, finally, that relational, rather than transactional, approaches to ethics may better suit local ethical challenges.

CS codes of ethics can assist educators by listing factors for consideration and mitigating situations when regulations, laws or policies are not fully developed. Yet the gap between codes of ethics and local realities can also cause harm. Further prevalent codes of ethics are instruments of power that enable actors in the Global North to determine what legitimate CS practice comprises and the position of the Global Souths relative to this. Thus, we advocate for ethical guidance that speaks to and comes “from within” people’s messy realities in the Global Souths not only because connecting ethics to students’ and educators’ values, knowledge, and experiences is vital for learning but also to assert greater recognition and respect for localized ethical judgements.

Making ethics at home in global CS education is about fostering students’ ethical sensibilities and orienting them to engage reflexively with different values and positionalities within and beyond their own contexts. Ethical considerations are always updating as new technologies, new socio-technical

situations and new sensitivities emerge and, thus, we suggest that educators use storytelling about ongoing, real-world events to engage students with “ethos building.” [2] In the epilogue that extends this piece [3], we share two stories that arose when researching and presenting this article to show how ethics is embedded in every action and how as educators we must continuously refine our sensitivity to the varied ways our lives are implicated in technical and socio-technical systems, from local to global scales, and develop confidence to discuss their implications with our students.

Our modest study significantly extends existing research [1] on how CS educators account for the diverse ways ethical dilemmas and approaches to ethics are situated in cultural, philosophical, and governance systems, religions, and languages [1].

References

[1] Janet Hughes, Ethan Plaut, Feng Wang, Elizabeth von Briesen, Cheryl Brown, Gerry Cross, Viraj Kumar, and Paul Myers. Global and local agendas of computing ethics education. In *Proceedings of the Conference on Innovation and Technology in Computer Science* Education, 2020; (ACM, New York, NY, 2020) 239-245.

[2] Christopher Frauenberger and Peter Purgathofer. 2019. Responsible thinking educating future technologists. In *Proceedings of CHI Conference on Human Factors in Computing Systems (CHI’19).*

[3] Wong-Villacres, M., Kutay, C., Lazem, S., Ahmed, N., Abad, C., Collazos, C., ... & Bidwell, N. J. Making ethics at home in Global CS Education: Provoking stories from the Souths. *ACM Journal on Computing and Sustainable Societies*. (2023).

CS + X: Approaches, Challenges, and Opportunities in Developing Interdisciplinary Computing Curricula

Valerie Barr, Bard College, Annandale-on-Hudson, NY, USA
Carla E. Brodley, Northeastern University, Boston, MA, USA
Elsa L. Gunter, UIUC, Urbana-Champaign, IL, USA
Mark Guzdial, University of Michigan, Ann Arbor, MI, USA
Ran Libeskind-Hadas, Claremont McKenna College, Claremont, CA, USA
Bill Manaris, College of Charleston, Charleston, SC, USA

Interdisciplinary computing curricula and majors (often called CS+X) interweave foundational computing concepts with those of specific disciplines in the natural sciences, social sciences, humanities, and the arts. Well-designed CS+X programs have substantially increased diversity and inclusion in computing. They address a rapidly growing need for a computationally sophisticated workforce across many domains that are critical to society. Virtually every discipline has significant challenges and opportunities that require computational methods. Increasingly, many researchers and practitioners in those fields are using computational methods, yet undergraduates in those fields often get little or no computational training deeper than using existing software tools.

Interdisciplinary computing can be implemented in individual courses (e.g., a course that combines both the art and computing concepts for visualization); as a major+minor; or as its own major, where students take some courses from computing, a similar number from another discipline, and one or more integrative courses.

Interdisciplinary courses and majors have several additional benefits. There is ample evidence that such innovative programs significantly broaden participation in computing. For example, interdisciplinary programs can substantially improve gender diversity and, generally, engage diverse populations of students who are unlikely to pursue a within-discipline computing degree [1,2,3,4]. The gender diversity likely depends in part on the X in CS+X. For example, at some institutions with CS+X programs where X is related to the arts, the CS+X major has approximately equal numbers of women and men, which is more than twice the national statistic for CS programs (22% women).

A second benefit of interdisciplinary computing majors is the ability to reach a larger set of students – because of enrollment pressures and course caps in computer science departments, non-majors are often unable to access the computing courses that they seek. CS+X majors can help computing departments (and universities) better manage enrollments. A CS+X major typically will require fewer computing classes than a within-discipline CS major, reducing enrollment pressure on higher-level electives that are often harder to staff.

References

[1] William Bares, Bill Manaris, and Renée McCauley. Gender equity in computer science through Computing in the Arts – A six-year longitudinal study. *Computer Science Education Journal* 28, 3 (September 2018), 191–210. https://doi.org/10.1080/08993408.2018.1519322.

[2] William H. Bares, Bill Manaris, Renée McCauley, and Christine Moore. Achieving gender balance through creative expression. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (Minneapolis, MN, USA) (SIGCSE 2019)*. Association for Computing Machinery, New York, NY, USA, 293-299. https://doi.org/10.1145/3287324.3287435

[3] Carla E. Brodley, Benjamin J. Hescott, Jessica Biron, Ali Ressing, Melissa Peiken, Sarah Maravetz, and Alan Mislove. Broadening participation in computing via ubiquitous combined majors (CS+X). In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (Providence, RI, USA) (SIGCSE 2022).* Association for Computing Machinery, New York, NY, USA, 544–550. https://doi.org/10.1145/3478431.3499352

[4] Zachary Dodds, Malia Morgan, Lindsay Popowski, Henry Coxe, Caroline Coxe, Kewei Zhou, Eliot Bush, and Ran Libeskind-Hadas. A Biology-based CS1: Results and reflections, ten years in. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE 2021)*. Association for Computing Machinery, New York, NY, USA, 796-801.

The Role of Formal Methods in Computer Science Education

Maurice H. ter Beek, CNR–ISTI, Pisa, Italy
Manfred Broy, Technische Universität München, München, Germany
Brijesh Dongol, University of Surrey, Guilford, UK
Emil Sekerinski, McMaster University, Hamilton, Canada

Formal Methods (FM) are available in various forms, spanning from lightweight static analysis to interactive theorem proving. These methods provide a systematic demonstration to students of the application of formal foundations in Computer Science within engineering tasks. The core skill of abstraction, fundamental to computer science, is effectively addressed through FM [1]. Even students specializing in 'Formal Methods Thinking'—the application of ideas from FM in informal, lightweight, practical, and accessible ways—experience notable improvement in their programming skills [2]. Exposure to these ideas also positions students well for further study on why techniques work, how they can be automated, and the development of new approaches.

FM can contribute significantly to teaching programming to novices, complementing informal reasoning and testing methods. They elucidate algorithmic problem-solving, design patterns, model-driven engineering, software architecture, software product lines, requirements engineering, and security, thereby supporting various fields within computer science [3]. Formalisms provide a concise and precise means of expressing underlying design principles, equipping programmers with tools to address related problems.

In industry, FM find widespread application, from eliciting requirements and early design to deployment, configuration, and runtime monitoring [4]. A recent survey [5] involving 130 FM experts, including three Turing Award winners, all four FME Fellowship Award winners, and 16 CAV Award winners, indicates that the most suitable place for FM in a teaching curriculum is in bachelor courses at the university level, as reported by 79.2% of respondents. Furthermore, 71.5% of respondents identify the lack of proper training in FM among engineers as the key limiting factor for a broader adoption of FM by the industry.

The survey highlights the uneven nature of FM education across universities, with many experts advocating for the standardization of university curricula. A recent white paper [6] supports this view, proposing the inclusion of a compulsory FM course in Computer Science and Software Engineering curricula. This recommendation is based on the observation that there is a shortage of Computer Science graduates qualified to apply Formal Methods in industry.

The challenge is twofold: (1) the lack of definitive educational sources that support FM-based courses in Computer Science; and (2) the training of academic staff to teach FM. Help is, however, becoming available (https://fmeurope.org/teaching/), and the future is bright, as more and more educators contribute to the effort of creating and sharing teaching resources.

References

[1] Manfred Broy, Achim D. Brucker, Alessandro Fantechi, Mario Gleirscher, Klaus Havelund, Cliff Jones, Markus Kuppe, Alexandra Mendes, André Platzer, Jan Oliver Ringert, and Allison Sullivan. Does Every Computer Scientist Need to Know Formal Methods? Submitted to *Form. Asp. Comput.* (2023).

[2] Brijesh Dongol, Catherine Dubois, Stefan Hallerstede, Eric Hehner, Daniel Jackson, Carroll Morgan, Peter Müller, Leila Ribeiro, Alexandra Silva, Graeme Smith, and Erik de Vink. On Formal Methods Thinking in Computer Science Education. Submitted to *Form. Asp. Comput.* (2023).

[3] Emil Sekerinski, Marsha Chechik, João F. Ferreira, John Hatcliff, Michael Hicks, and Kevin Lano.. Should We Teach Formal Methods or Algorithmic Problem Solving, Design Patterns, Model-Driven Engineering, Software Architecture, Software Product Lines, Requirements Engineering, and Security? In preparation 2023.

[4] Maurice H. ter Beek, Rod Chapman, Rance Cleaveland, Hubert Garavel, Rong Gu, Ivo ter Horst, Jeroen J. A. Keiren, Thierry Lecomte, Michael Leuschel, Kristin Y. Rozier, Augusto Sampaio, Cristina Seceleanu, Martyn Thomas, Tim A. C. Willemse, and Lijun Zhang. 2023. Formal Methods in Industry. Submitted to *Form. Asp. Comput.* (2023).

[5] Hubert Garavel, Maurice H. ter Beek, and Jaco van de Pol. 2020. The 2020 Expert Survey on Formal Methods. In Proceedings of the 25th International Conference on Formal Methods for Industrial Critical Systems (FMICS'20) (LNCS, Vol. 12327), Maurice H. ter Beek and Dejan Ničković (Eds.). Springer, Germany, 3–69. https://doi.org/10.1007/978-3-030-58298-2_1

[6] Antonio Cerone, Markus Roggenbach, James Davenport, Casey Denner, Marie Farrell, Magne Haveraaen, Faron Moller, Philipp Körner, Sebastian Krings, Peter Csaba Ölveczky, Bernd-Holger Schlingloff, Nikolay Shilov, and Rustam Zhumagambetov. 2021. Rooting Formal Methods Within Higher Education Curricula for Computer Science and Software Engineering – A White Paper. In Revised Selected Papers of the 1st International Workshop on Formal Methods – Fun for Everybody (FMFun'19) (CCIS, Vol. 1301), Antonio Cerone and Markus Roggenbach (Eds.). Springer, Germany, 1–26. https://doi.org/10.1007/978-3-030-71374-4_1

Quantum Computing Education: A Curricular Perspective

Dan-Adrian German, Indiana University, Bloomington, IN, USA
Marcelo Pias, Federal University of Rio Grande, Rio Grande, RS, Brazil
Qiao Xiang, Xiamen University, Xiamen, Fujian, China

At the end of 2023 we are still in the NISQ era [4,5]. The term (Noisy Intermediate-Scale Quantum) was introduced by John Preskill at Q2B in December 2017. Atom Computing first reached 1,000 qubits in 2013 [9], soon thereafter followed by IBM [10]. The milestone marks just how far the industry has come: only 6 years ago, typically, under 10 qubits were available for developers on the IBM Quantum Experience. Long-time quantum pioneer D-Wave remains an outlier in that it has a 5,000-qubit system (Advantage) but it is an analog, not a gate-based system; it is an open question whether gate-based approaches are necessary to get the full power of fault-tolerant quantum computing and D-Wave has recently started developing gate-based technology. On the other hand, adiabatic quantum computing (AQC) and quantum annealing (QA) remain legitimate (and promising) avenues of research in quantum computation. Also, this year, a Harvard-led team developed the first-ever quantum circuit with logical quantum bits [1]. Arrays of “noisy” physical Rydberg qubits were used to create quantum circuits with 48 error-correcting logical qubits, the largest number to date, a crucial step towards realizing fault-tolerant quantum computing. Meanwhile, PsiQuantum continues to pursue unabated the 1,000,000 (physical) qubits mark [7,8]. The competition between the various qubit implementation modalities intensified: superconducting qubits, trapped atoms/ions, spin qubits (Intel has a 12-qubit chip) and photonics are currently in the lead. Debates [6] now abound about the potential (or impending) demise of the NISQ era. The industry remains engaged in a sustained effort of both short-term (upskilling and reskilling workers, and HS teachers) and long-term workforce development. This past summer, researchers at Quantinuum and Oxford University [2,11] established the foundations and methodology for an ongoing educational experiment to investigate the question: ‘From what age can students learn quantum theory if taught using a diagrammatic approach?' The math-free framework in [3] was used to teach the pictorial method to UK schoolchildren, who then beat the average exam scores of Oxford University’s postgraduate physics students. The experiment involved 54 schoolchildren, aged 15-17, randomly selected from around 1,000 applicants, from 36 UK schools (mostly state schools). Teenagers spent two hours a week in online classes and after eight weeks were given a test using questions taken from past Oxford postgraduate quantum exams: more than 80% of the pupils passed and around half earned a distinction. Interest in incorporating quantum architecture topics in the traditional CS curriculum remains high for the next 10-year horizon. A growing consensus is that the CS undergraduate must have a proper appreciation for the quantum mechanical nature of our world. The main prerequisite to such a knowledge unit remains a certain intellectual versatility, manifested in the willingness to be exposed to information from more than one domain/discipline. In quantum computing, labs will be quintessential and will rely on (1) computer-assisted mathematics (e.g., Wolfram Alpha, NumPy, Qiskit, Matplotlib, etc.) as well as CAD/CAM and advanced software emulation (Qiskit Metal), (2) access to actual quantum computers via various cloud platforms (Amazon Braket, IBM Q, Xanadu Borealis, etc.) and (3) occasionally access to a physics lab, fab or foundry. A genuinely interdisciplinary program can only be built if faculty have wide general support towards such a goal. Three curricular approaches have emerged: one is entirely without math but leading into math and lasts about eight weeks. The second is a full semester, 14-week long, and entirely based on linear algebra. The last one

is two semesters long and includes weekly, messy but critical, quantum hardware labs supporting a quantum engineering degree. Incorporating material about all qubit modalities in the curriculum will ensure the material will remain relevant over a reasonably long period of time, if it includes such topics as the design and implementation of qubits (e.g., via Qiskit Metal) and error mitigation and (classical) control.

References

[1] D. Bluvstein, S. J. Evered, A. A. Geim, et al. Logical quantum processor based on reconfigurable atom arrays. In *Nature*. https://doi.org/10.1038/s41586-023-06927-3 (6 Dec. 2023).

[2] Bob Coecke. https://medium.com/quantinuum/everyone-can-learn-quantum-now-even-at-a-cutting-edge-level-and-we-have-the-test-scores-to-prove-49e7fdc5c509 (21 Dec. 2023). Accessed March 2024.

[3] Bob Coecke and Stefano Gogioso. *Quantum in Pictures: A New Way to Understand the Quantum World. Cambridge Quantum*, 1st edition (3 Feb. 2023).

[4] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum* 2, 79 (2018). https://quantum-journal.org/papers/q-2018-08-06-79/. Accessed March 2024; Preprint: https://arxiv.org/abs/1801.00862. Accessed March 2024.

[5] John Preskill. Quantum technology in the short term and long term: the search for applications. https://www.youtube.com/watch?v=TSzpz8N7Xw4 (Q2B 2018 Keynote Address). Accessed March 2024.

[6] John Preskill. Crossing the Quantum Chasm: From NISQ to Fault Tolerance. Q2B 2023 (6 Dec 2023) http://theory.caltech.edu/~preskill/talks/Preskill-Q2B-2023 (slides, video not yet available).

[7] Terry Rudolph. What is the logical gate speed of a photonic quantum computer? (June 21, 2023, via John Preskill's Twitter account and the Quantum Frontiers blog at the Institute for Quantum Information and Matter at Caltech) https://quantumfrontiers.com/2023/06/21/what-is-the-logical-gate-speed-of-a-photonic-quantum-computer/. Accessed March 2024.

[8] John Russell. PsiQuantum's Path to 1 Million Qubits.(21 April 2022, in hpcwire.com) https://www.hpcwire.com/2022/04/21/psiquantums-path-to-1-million-qubits-by-the-middle-of-the-decade/.

[9] John Russell. Atom Computing Wins the Race to 1000 Qubits. (24 Oct. 2023 in hpcwire.com) https://www.hpcwire.com/2023/10/24/atom-computing-wins-the-race-to-1000-qubits/ Accessed March 2024.

[10] The Quantum Mechanic. IBM and UC Berkeley Usher in New Era of Quantum Computing with 1,121 Qubit Machine. Hello IBM Condor. (4 Dec. 2023) https://quantumzeitgeist.com/ibm-and-uc-berkeley-usher-in-new-era-of-quantum-computing-with-1121-qubit-machine-hello-ibm-condor/. Accessed March 2024.

[11] Aleks Kissinger. Research unveils new picture-based approach to teaching physics. (20 Dec. 2023) https://www.cs.ox.ac.uk/news/2280-full.html. Accessed March 2024.

Generative AI in Introductory Programming

Brett A. Becker, University College Dublin, Dublin, Ireland
Michelle Craig, University of Toronto, Toronto, Canada
Paul Denny, The University of Auckland, Auckland, New Zealand
Hieke Keuning, Utrecht University, Utrecht, The Netherlands
Natalie Kiesler, DIPF Leibniz Institute for Research and Information in Education, Frankfurt, Germany
Juho Leinonen, Aalto University, Aalto, Finland
Andrew Luxton-Reilly, The University of Auckland, Auckland, New Zealand
Lauri Malmi, Aalto University, Aalto, Finland
James Prather, Abilene Christian University, Abilene, TX, USA
Keith Quille, TU Dublin, Dublin, Ireland

Generative AI tools based on Large Language Models (LLMs) such as OpenAI's ChatGPT, and IDEs powered by them such as GitHub Copilot, have demonstrated impressive performance in myriad types of programming tasks including impressive performance on CS1 and CS2 problems. They can often produce syntactically and logically correct code from natural language prompts that rival the performance of high-performing introductory programming students—an ability that has already been shown to extend beyond introductory programming [2]. However, their impact in the classroom goes beyond producing code. For example, they could help level the playing field between students with and without prior experience. Generative AI tools have been shown to be proficient in not only explaining programming error messages but in repairing broken code [6], and pair programming might evolve from two students working together into “me and my AI.” On the other hand they could have negative effects. Students could become over-reliant on them, and they may open up new divides due to different backgrounds, experience levels and access issues [9]. Generative AI has been successful in generating novel exercises and examples including providing correct solutions and functioning test cases [11]. Instructional materials are already being produced including a textbook that uses Generative AI from the first day of CS1 [8] that has already been used [4]. Given their ability to provide code explanations [7] they have the potential to assess student work, provide feedback, and to act as always-available virtual teaching assistants, easing the burden not only on the educator but on their human assistants and the broader educational systems where learning takes place [9]. Generative AI could even affect student intakes given its prominence in the media and the effect that such forces can have on who chooses to—and who chooses not to—study computing.

Given that Generative AI has the potential to reshape introductory programming, it is possible that it will impact the entire computing curriculum, affecting what is taught, when it is taught, how it is taught, and to whom it is taught. However, the dust is far from settled on these matters with some educators embracing Generative AI and others very fearful that the challenges could outweigh the opportunities [5]. The computing education community needs to understand more about how students interact with Generative AI [10] and provide tooling and strategies to effectively achieve that interaction [3]. Indeed, during the transformation from pre- to post-Generative AI introductory programming, several issues need to be mitigated including but certainly not limited to those of ethics, bias, academic integrity, and broadening participation in computing [1]. Further study is warranted to explore the long-term effects of

Generative AI on pedagogy, curriculum, student demographics, and the broader educational ecosystem.

References

[1] Brett A. Becker, Paul Denny, James Finnie-Ansley, et al. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V*. 1, 2023. (Toronto, ON, Canada) (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 500–506. https://doi.org/10.1145/3545945.3569759.

[2] Paul Denny, James Prather, Brett A Becker, James Finnie-Ansley, Arto Hellas, Juho Leinonen, Andrew Luxton-Reilly, Brent N Reeves, Eddie Antonio Santos, and Sami Sarsa. 2024. Computing Education in the Era of Generative AI. Commun. ACM 67, 2 (Feb. 2024). https://doi.org/10.1145/3624720. Preprint available: https://arxiv.org/abs/2306.02608. Accessed March 2024).

[3] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett Becker, and Brent Reeves. 2024. Prompt Problems: A New Programming Exercise for the Generative AI Era. In Proceedings of the 55th SIGCSE Technical Symposium on Computer Science Education (Portland, OR USA) (SIGCSE '24). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3626252.3630909. Preprint available: https://arxiv.org/abs/2311.05943. Accessed March 2024).

[4] Katie E. Ismael, Ioana Patringenaru, and Kimberley Clementi. In This Era of AI, Will Everyone Be a Programmer? UC San Diego Today (Dec 2023). https://today.ucsd.edu/story/in-this-era-of-ai-will-everyone-be-a-programmer. Accessed March 2024.

[5] Sam Lau and Philip Guo. 2023. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools Such as ChatGPT and GitHub Copilot. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1* (Chicago, IL, USA) (ICER '23). Association for Computing Machinery, New York, NY, USA, 106–121. https://doi.org/10.1145/3568813.3600138.

[6] Juho Leinonen, Arto Hellas, Sami Sarsa, et al. Using Large Language Models to Enhance Programming Error Messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto, ON, Canada) (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 563–569. https://doi.org/10.1145/3545945.3569770.

[7] Stephen MacNeil, Andrew Tran, Dan Mogil, Seth Bernstein, Erin Ross, and Ziheng Huang. Generating Diverse Code Explanations using the GPT-3 Large Language Model. In Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2 (ICER '22), Vol. 2. Association for Computing Machinery, New York, NY, USA, 37–39. https://doi.org/10.1145/3501709.3544280.

[8] Leo Porter and Daniel Zingaro. 2023. Learn AI-Assisted Python Programming with GitHub Copilot and ChatGPT. Manning, Shelter Island, NY, USA. https://www.manning.com/books/learn-ai-assisted-python-programming. Accessed March 2024.

[9] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Petersen, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education (Turku, Finland) (ITiCSE-WGR '23)*. Association for Computing Machinery, New York, NY, USA, 108–159. https://doi.org/10.1145/3623762.3633499.

[10] James Prather, Brent N. Reeves, Paul Denny, Brett A. Becker, Juho Leinonen, Andrew Luxton-Reilly, Garrett Powell, James Finnie-Ansley, and Eddie Antonio Santos. 2023. "It's Weird That It Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. ACM Trans. Comput.-Hum. Interact. 31, 1, Article 4 (Nov 2023), 31 pages. https://doi.org/10.1145/3617367.

[11] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1 (Lugano and Virtual Event, Switzerland) (ICER '22)*. Association for Computing Machinery, New York, NY, USA, 27–43. https://doi.org/10.1145/3501385.3543957.

The 2022 Undergraduate Database Course in Computer Science: What to Teach?

Mikey Goldweber, Denison University, Granville, OH, USA
Min Wei, Microsoft, Seatle, WA, USA
Sherif Aly, The American University in Cairo, Cairo, Egypt
Rajendra K. Raj, Rochester Institute of Technology, Rochester, NY, USA
Mohamed Mokbel, University of Minnesota, St. Paul, MN, USA

One issue with the study of databases, though maybe it should be labeled data management, or maybe even more precisely, the study of persistent data, is that the number of possible topics far exceeds the bandwidth of a single undergraduate CS course. Yes, there are several institutions with two course sequences. However, most undergraduate curricula, based on CS2013 [2] recommendations or ABET [1] criteria, have at most one database course, or just an elective. So, the question arises as to what to include and what to exclude.

Contributing to this phenomenon are the emergence of new topics (e.g., NoSQL, distributed and cloud-based databases) and the current renewed (and hopefully continuing) emphasis on both security and privacy, as well as societal and ethical issues associated with persistent data.

Another complicating factor is the institutional context. Every institution's curricular viewpoint sits somewhere on the spectrum between computer science as a pure science and computer science as a profession. Institutions are now preparing graduates for careers as Data Engineers, Data Infrastructure Engineers, and Data Scientists, in addition to Computer Scientists.

There are four primary perspectives with which to approach databases.
1. Database designers/modelers: those who model the data from an enterprise and organize it according to the principles of a given data model.
2. Database users: (SQL?) query writers.
3. Database administrators: those involved with tuning database performance through the building of index structures and the setting of various parameters.
4. Database engine developers: those who write the code for database engines.

Four different viewpoints for what an undergraduate CS course in Databases/Data Management should cover are described in [3].

References

[1] ABET (2022). ABET Computing Accreditation Commission: Criteria for Accrediting Computing Programs. https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2022-2023/. Accessed March 2024.

[2] ACM (2013). Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, Association for Computing Machinery and IEEE Computer Society. https://doi.org/10.1145/2534860.

[3] Mikey Goldweber, Min Wei, Sherif Aly, Rajendra K. Raj, and Mohamed Mokbel. The 2022 undergraduate database course in computer science: what to teach? *ACM Inroads* 13, 3 (September 2022), 16–21. https://doi.org/10.1145/3549545.

Computer Science Curriculum Guidelines: A New Liberal Arts Perspective

Jakob Barnard; University of Jamestown; Jamestown, MD, USA
Valerie Barr; Bard College; Annandale-on-Hudson, NY, USA
Grant Braught; Dickinson College; Carlisle, PA, USA
Janet Davis; Whitman College; Walla Walla, WA, USA
Amanda Holland-Minkley; Washington & Jefferson College; Washington, PA, USA
David Reed; Creighton University; Omaha, NE, USA
Karl Schmitt; Trinity Christian College; Palos Heights, IL, USA
Andrea Tartaro; Furman University; Greenville, SC, USA
James Teresco; Siena College; Loudonville, NY, USA

ACM/IEEE curriculum guidelines for computer science, such as CS2023, provide well-researched and detailed guidance regarding the content and skills that make up an undergraduate computer science (CS) program. Liberal arts CS programs often struggle to apply these guidelines within their institutional and departmental contexts [6]. Historically, this has been addressed through the development of model CS curricula tailored for the liberal arts context [1,2,3,4,7]. We take a different position: that no single model curriculum can apply across the wide range of liberal arts institutions. Instead, we argue that liberal arts CS educators need best practices for using guidelines such as CS2023 to inform curriculum design. These practices must acknowledge the opportunities and priorities of a liberal arts philosophy as well as institutional and program missions, priorities, and identities [5].

The history, context, and data about liberal arts CS curriculum design support the position that the liberal arts computing community is best supported by a process for working with curricular guidelines rather than a curriculum model or set of exemplars [5]. Previous work with ACM/IEEE curriculum guidelines over the decades has trended towards acknowledging that liberal arts CS curricula may take a variety of forms and away from presenting a unified “liberal arts” model [6]. A review of liberal arts CS programs demonstrates how institutional context, including institutional mission and structural factors, shape their curricula [5]. Survey data indicates that liberal arts programs have distinct identities or missions, and this directly impacts curriculum and course design decisions. Programs prioritize flexible pathways through their programs coupled with careful limits on required courses and lengths of prerequisite chains [6]. This can drive innovative course design where content from Knowledge Areas is blended rather than compartmentalized into distinct courses [7,8]. The CS curriculum is viewed as part of the larger institutional curriculum and the audience for CS courses is broader than just students in the major, at both the introductory level and beyond.

To support the unique needs of CS liberal arts programs, we propose a process that guides programs to work with CS2023 through the lens of institutional and program missions and identities, goals, priorities, and situational factors. The Process Workbook we have developed comprises six major steps:

1. articulate institutional and program mission and identity;
2. develop curricular design principles driven by program mission and identity, structural factors, and attention to diversity, equity, and inclusion;

3. identify aspirational learning outcomes in response to design principles and mission and identity;
4. engage with CS2023 to select curriculum and course content based on design principles to achieve learning outcomes and support mission and identity;
5. evaluate the current program, with attention to current strengths, unmet goals, and opportunities for improvement;
6. design, implement, and assess changes to the curriculum.

An initial version of the Process Workbook, based on our research and feedback from workshops [9, e.g., 10,11] and pilot usage within individual departments, is available as a supplement to this article [12]. The authors will continue this iterative design process and release additional updates as we gather more feedback. Future work includes development of a repository of examples of how programs have made use of the Workbook to review and redesign their curricula in the light of CS2023.

References

[1] Kim B. Bruce, Robert D. Cupper, and Robert L. Scot Drysdale. A History of the Liberal Arts Computer Science Consortium and Its Model Curricula. *ACM Trans. Comput. Educ.* 10,1, Article 3 (March 2010), 12 pages. https://doi.org/10.1145/1731041.1731044.

[2] Liberal Arts Computer Science Consortium. A 2007 Model Curriculum for a Liberal Arts Degree in Computer Science. *J. Educ. Resour. Comput.* 7,2 (June 2007), 2-es. https://doi.org/10.1145/1240200.1240202.

[3] Henry M. Walker and G. Michael Schneider. A Revised Model Curriculum for a Liberal Arts Degree in Computer Science. *Commun. ACM* 39,12 (Dec. 1996), 85–95. https://doi.org/10.1145/240483.240502.

[4] Norman E. Gibbs and Allen B. Tucker. A Model Curriculum for a Liberal Arts Degree in Computer Science. *Commun. ACM* 29, 3 (March 1986), 202-210. https://doi.org/10.1145/5666.5667.

[5] Amanda Holland-Minkley, Jakob Barnard, Valerie Barr, Grant Braught, Janet Davis, David Reed, Karl Schmitt, Andrea Tartaro, and James D. Teresco. Computer Science Curriculum Guidelines: A New Liberal Arts Perspective. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 617-623. https://doi.org/10.1145/3545945.3569793.

[6] James D. Teresco, Andrea Tartaro, Amanda Holland-Minkley, Grant Braught, Jakob Barnard, and Douglas Baldwin. CS Curricular Innovations with a Liberal Arts Philosophy. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (Providence, RI, USA) (SIGCSE 2022).* Association for Computing Machinery, New York, NY, USA, 537-543. https://doi.org/10.1145/3478431.3499329.

[7] Henry M. Walker and Samuel A. Rebelsky. Using CS2013 for a Department's Curriculum Review: A Case Study. *J. Comput. Sci. Coll.* 29,5 (May 2014), 138-144.

[8] David Reed. Spiraling CS2013 Knowledge Units across a Small CS Curriculum. *J. Comput. Sci. Coll.* 32,5 (May 2017), 125-131.

[9] Amanda Holland-Minkley, Andrea Tartaro, and Jakob Barnard. Innovations and Opportunities in Liberal Arts Computing Education, https://computing-in-the-liberal-arts.github.io/SIGCSE2023-Affiliated-Event/. URL. SIGCSE 2023 Affiliated Event by the SIGCSE Committee on Computing Education in Liberal Arts Colleges.

[10] Jakob Barnard, Grant Braught, Janet Davis, Amanda Holland-Minkley, David Reed, Karl Schmitt, Andrea Tartaro, and James Teresco. Developing Identity-Focused Program-Level Learning Outcomes for Liberal Arts Computing Programs. *J. Comput. Sci. Coll.* 39,4 (October 2023), 97-98.

[11] Jakob Barnard, Grant Braught, Janet Davis, Amanda Holland-Minkley, David Reed, Karl Schmitt, Andrea Tartaro, and James Teresco. Reflective Curriculum Review for Liberal Arts Computing Programs. *J. Comput. Sci. Coll.* 38, 3 (November 2022), 178–179.

[12] SIGCSE Committee on Computing Education in Liberal Arts Colleges. 2023. CS2023 Activity: The Curricular Practices Workbook. https://computing-in-the-liberal-arts.github.io/CS2023/. Accessed March 2024.

Computer Science Education in Community Colleges

Elizabeth Hawthorne, Rider University, Lawrenceville, NJ, USA
Lori Postner, Nassau Community College, Garden City, NY, USA
Christian Servin, El Paso Community College, El Paso, TX, USA
Cara Tang, Portland Community College, Portland, OR, USA
Cindy Tucker, Bluegrass Community and Technical College, Lexington, KY, USA

Community and Technical Colleges serve as two-year educational institutions, providing diverse academic degrees like associate's degrees in academic and applied sciences, certificates of completion, and remedial degrees. These colleges play a crucial role in fostering collaboration between students, workers, and institutions through educational and workforce initiatives. Over the past 50+ years, Community Colleges have served as a hub for various educational initiatives and partnerships involving K-12 schools, four-year colleges, and workforce/industry collaborations.

These colleges offer specialized programs that help students focus on specific educational pathways. Among the programs available, computing-related courses are prominent, including Computer Science degrees, particularly the Associate in Arts (AA) and Sciences (AS) degrees, known as academic transfer degrees. These transfer degrees are designed to align with the ACM/IEEE curricular guidelines, primarily focusing on creating two-year programs that facilitate smooth transferability to four-year colleges.

Furthermore, the computing programs offered by Community Colleges are influenced by the specific needs and aspirations of the regional workforce and industry. Advisory boards and committees play a significant role in shaping these programs by providing recommendations based on the demands of the job market. While the ACM Committee for Computing in Community Colleges (CCECC) and similar entities help address inquiries related to these transfer degrees, there is a desire to capture the challenges, requirements, and recommendations from the Community College perspective in developing general curricular guidelines.

This work presents the context and perspective of the community college education. It emphasizes the importance of understanding the unique challenges faced by Community Colleges and their specific needs while formulating curricular guidelines. Additionally, the work envisions considerations for the next decade regarding curricular development and administrative efforts, considering the evolving educational landscape and industry demands. By doing so, the vision is to enhance the effectiveness and relevance of computing programs offered by Community Colleges and foster better alignment with the needs of students and the job market.

References

[1] "ABET Accredits 54 Additional Programs in 2021, Including First Associate Cybersecurity programs." https://www.abet.org/abet-accredits-54-new-programs-in-2021-including-first-associate-cybersecurity-programs/. Accessed Feb 29, 2024.

[2] William F. Atchison, Samuel D. Conte, John W. Hamblen, Thomas E. Hull, Thomas A. Keenan, William B. Kehl, Edward J. McCluskey, Silvio O. Navarro, Werner C. Rheinboldt, Earl J. Schweppe, William Viavant, and David M. Young. "Curriculum 68: Recommendations for Academic Programs in Computer Science: A Report of the ACM Curriculum Committee on Computer Science." *Communications of the ACM* 11,3 (1968), 151-197. https://doi.org/10.1145/362929.362976.

[3] Jill Denner, Paul Tymann, and Huihui Wang. "Community College Pathways." In *Proceedings of the 2023 CISE EWF PI Meeting.* Georgia Tech Conference Center.

[4] Dennis Foley, Leslie Milan, and Karen Hamrick. 2020. "The Increasing Role of Community Colleges among Bachelor's Degree Recipients: Findings from the 2019 National Survey of College Graduates." Technical Report NSF 21-309. National Center for Science and Engineering Statistics (NCSES), Alexandria, VA. https://ncses.nsf.gov/pubs/nsf21309/. Accessed March 2024.

[5] ACM Committee for Computing Education in Community Colleges (CCECC). 2017. "ACM Computer Science Curricular Guidance for Associate-Degree Transfer Programs with Infused Cybersecurity." 2017. Association for Computing Machinery, New York, NY, USA.

[6] The Community College Presidents Initiative in STEM. 2023. "Community college presidents initiative – STEM – achieving excellence in workforce education." https://www.ccpi-stem.org/. Accessed March 2024.

[7] Robin G Isserles. "The Costs of Completion: Student Success in Community College." JHU Press, 2021

[8] Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. "Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science." ACM, New York, NY, USA, 2013.

[9] A. Kahlon, D. Boisvert, L.A. Lyon, M. Williamson, and C. Calhoun. "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. https://doi.org/10.1145/3159450.3159627.

[10] Amruth N. Kumar and Rajendra K. Raj. "Computer Science Curricula 2023 (CS2023): Community Engagement by the ACM/IEEE-CS/AAAI Joint Task Force." In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2 (Toronto, ON, Canada) (SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1212-1213. https://doi.org/10.1145/3545947.3569591.

[11] Joyce Currie Little, Richard H. Austing, Harice Seeds, John Maniotes, and Gerald L. Engel.. "Curriculum recommendations and guidelines for the community and junior college career program in computer programming: a working paper of the ACM committee on curriculum in computer sciences by the subcommittee on community and junior college curriculum." ACM SIGCSE Bulletin - Special issue on computer science curricula 9, 2 (1977), 1-16. https://doi.org/10.1145/988948.988951.

[12] B. Morrison and A. Settle. "Celebrating SIGCSE's 50th Anniversary!" SIGCSE Bulletin 50,1 (2018), 2-3.

[13] American Association of Community Colleges. 2022. "The Economic Value of America's Community Colleges." https://www.aacc.nche.edu/2022/11/29/the-economic-value-of-americas-community-colleges-report/. Accessed March 2024.

[14] Christian Servín. "Fuzzy Information Processing Computing Curricula: A Perspective from the First Two-Years in Computing Education." In *Explainable AI and Other Applications of Fuzzy Techniques: Proceedings of the 2021 Annual Conference of the North American Fuzzy Information Processing Society, NAFIPS 2021*. Springer, 453-460.

[15] Christian Servin, Elizabeth K. Hawthorne, Lori Postner, Cara Tang, and Cindy Tucker. "Community Colleges Perspectives: From Challenges to Considerations in Curricula Development (SIGCSE 2023)." Association for Computing Machinery, New York, NY, USA, 1244. https://doi.org/10.1145/3545947.3573335.

[16] Christian Servin, Elizabeth K. Hawthorne, Lori Postner, Cara Tang, and Cindy S. Tucker. "Mathematical Considerations in Two-Year Computing Degrees: The Evolution of Math in Curricular Guidelines." In *The 24th Annual Conference on Information Technology Education (SIGITE '23) (Marietta, GA, USA).* ACM. https://doi.org/10.1145/3585059.3611441.

[17] Cara Tang. 2017. "Community College Corner Community colleges in the United States and around the world." *ACM Inroads* 8,1 (2017), 21-23.

[18] Cara Tang. 2018. "Community Colleges and SIGCSE: A Legacy Fueling the Future." *ACM Inroads* 9,4 (2018), 49-52. https://doi.org/10.1145/3230699.

[19] Cara Tang, Elizabeth K Hawthorne, Cindy S Tucker, Ernesto Cuadros-Vargas, Diana Cukierman, and Ming Zhang. "Global Perspectives on the Role of Two-Year/Technical/Junior Colleges in Computing Education." In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. 204-205.

[20] "Celebrating 40++ years of service to computing education communities." [n.d.]. ACM CCECC. https://ccecc.acm.org/correlations/all. Accessed March 2024.

[21] Stuart Zweben, Jodi L.Tims, Cindy Tucker, and Yan Timanovsky. "ACM-NDC Study 2021–2022: Tenth Annual Study of Non-Doctoral-Granting Departments in Computing." *ACM Inroads* 13,3 (2022), 38-54. https://doi.org/10.1145/3544304.

[22] Stuart Zweben and Cindy Tucker. "How Well Did We Keep Students in Computing Programs, Pre-COVID and COVID?" *ACM Inroads* 13,4 (2022), 32-52. https://doi.org/10.1145/3571094.