Encouraging Exploration and Diversity in Computer Science

Moie Uesugi

June 15, 2015

Contents

1 Introduction

Science, technology, engineering, and math (STEM) workers are in demand—this motivates students to pursue STEM degrees [1] and teachers to recruit. Even the United States President's Council of Advisors on Science and Technology has published that one million additional STEM workers are required to meet projected demand from 2012 to 2022 [2]. Similarly, the Bureau of Labor Statistics estimates a 15% employment increase in computer and information scientists alone in that same time span [3]. Yet, even with this high demand in a struggling economy, professionals and scholars are still confronted with an insufficiency of students graduating from undergraduate computer science programs and startling underrepresentation of women, black and latino/a students.

What deters coveted students, before and especially after their entrance into their first computer science classes, from completing the major and eventually building a career in computer science? Many articles in the past ten years have attempted to answer this question, both through theory as well as actual implementation of new types of introductory-level (CS0 and CS1) classes.

2 Barriers to Entry and Retention

Before effective interventions and changes can be implemented, it is important to identify the factors that contribute to this lack of diversity. From early development, parental and societal influences deter females from science-oriented fields. In turn, once these same women enter college-level computer science classes, they are deterred by their apparent lack of experience [1]. Students are not encouraged to solve problems through exploration in supportive class environments, which can have a negative effect on their resilience and self-confidence [4][5][6]. Many secondary school and college classes have been identified as only focusing on contained STEM concepts without showing their practical or humanitarian applications [1][4][7][8][9][10], and do not actively encourage students to find these applications on their own [11]. Outside of the classroom, both students and teachers in computer science departments perpetuate a "hacker culture" from which underrepresented students feel excluded [1]. In fact, the teachers, the literal "gatekeepers" to STEM education and subsequent STEM employment, measure student understanding inaccurately [6], at times even changing the structure or content of the class based on a single voice in the classroom [12]. Many also focus only on traditional teaching styles, attempting to passively transfer learning to students through lectures without active engagement [15], attending to just the most outgoing and comfortable students in need of help [13], and not cultivating peer relationships for mutual instruction [14][16].

3 Efforts to Broaden Participation

3.1 Pre-college Interventions

Tytler *et al.* argue that STEM education interventions should occur before age 14, before which students identify their life aspirations.[4][1]. Students must be exposed to STEM edu-

[1]It is important to note that the education system is different in Australia, where the review was based, and thus while the age of 14 seems inaccurately young from the United States education system, occupation

cation before they choose which classes to take and what paths to pursue.

Instructors play an undeniably large role in encouraging students to explore STEM, especially in the younger grades. By introducing more problem solving and explorative math and science learning, they help students learn early-on to connect academic concepts to real-life applications [4]. Furthermore, Shah *et al*. claim that by providing quality, personalized instruction, educators can maintain the same level of understanding across different students and thus combat race and gender inequity early-on [13] to prevent education disparities upon entrance to college classes. In addition, they also recommend access to curriculum, peer instruction, and role models to create a framework of computer science education that supports all of its students.

3.2 Increasing Self-Efficacy

The lack of diverse participation in computer science can also be explained by influences on students' belief in their own ability to succeed. Because of the more general nature of this topic, this has been related to younger students as well as undergraduate students.

Martin and Marsh and Tytler *et al*. argue for pedagogical reforms that increase this self-belief. Anxiety (negative), self-efficacy, academic engagement, and to a lesser degree, teacher-student relationships are strongly correlated (if not causal) to later academic buoyancy [5]. Tytler *et al*. advocate for building optimism—a similar concept to self-efficacy—though successful problem solving and individual exploration. They propose a project- and understanding-based curriculum in which students learn how different concepts connect and how to apply concepts to unfamiliar situations. [4]. Wiggins and McTighe also highlight the long-term benefits of teaching students to understand new concepts on their own using prior knowledge [6].

By combining the recommendations of Martin and Marsh, Tytler *et al*. and Wiggins and McTighe to combat and/or nurture the mentioned traits, students can build their buoyancy

distinctions in classes generally appear around the beginning of secondary school, just under age 14 [17].

and thus succeed through more challenges in academic settings.

3.3 Changing Computer Science Department Culture

Although Margolis and Fisher wrote *Unlocking the Clubhouse* over a decade ago, their seminal exploration of the role and place of female students in undergraduate computer science programs is still very much relevant [1]. They argue for the dissipation of a common "hacker culture" in computer science departments that supports aggressive, obsessive behavior that is alienating for many students and can contribute to students not finishing their computer science degrees. In addition, Margolis and Fisher advocate making the focus of the curriculum less narrow and instead showing students the social applications of their technical skills as well as the greater computer science context into which their skills would fit. By doing so, students understand that creative and social people can thrive in computer science, thus helping to weaken the idea that only a stereotypical hacker type is welcome in the department and in the field.

3.4 Enhancing Introductory Courses

The practical purposes and applications of a computer science curriculum, especially in introductory courses, can be less apparent than in other subjects such as the natural sciences. Goldweber *et al*. suggest infusing the curriculum with social relevancy as a way to encourage the participation of students from less-represented groups in computer science [7]. They argue for this trend in computer science courses because they believe seeing the knowledge being applied early-on in the major could help students see the greater purpose of studying computer science. They provide 14 examples of projects that demonstrate the wide scope and impact of computer science. Their study found a gender difference in the appreciation of the socially relevant projects that were tested, with females ranking the projects positively, thus suggesting these socially relevant projects as a way to encourage women to continue studying computer science rather than subjects whose practical purposes are more readily apparent, as in the natural sciences.

However, Goldweber *et al.* categorize both philanthropic interest and personal interest under "social relevancy," which may be at the root of the disparities in its results as compared with those of Rader *et al.* [8]. They instead split socially relevant projects between humanitarian ones (*i.e.* an inflation estimator) and those with which they can personally relate (*i.e.* a calorie counter). Students were generally more inclined toward the simple, "fun" projects, but some liked the personally relevant projects where none seemed to prefer the more challenging prospects of the humanitarian projects.

Many new introductory courses have been developed to teach a more overarching view of the applications of computer science before teaching the smaller mechanics and ideas with which they cannot yet do much. Although this idea may seem like a fast new trend, the idea of teaching students how to make sense of new knowledge, as opposed to memorizing a limited amount of new information, is not as recent; Wiggins and McTighe presented a similar recommendation for educators in all fields in 2005 [6]. The two ideas are closely related because the purpose behind teaching students to connect disparate ideas in school is so that they can compile together all of the information to understand the applications of their field as a whole. Thus, this new idea is only a step closer to helping the students complete an older goal.

Both Heintz and Inger as well as Anderson *et al.* have very recently written articles about designing new classes that still teach the traditional computing fundamentals to students in introductory classes while also showing them a much larger overview of the potential of computer science through analyzing real-world, relevant data [9], providing an overview of the field and education at the start of the program, and moving through the curriculum based on concept (teaching each from the basics to the advanced before moving to the next) rather than difficulty [10]. Balasubramanian *et al.* specifically look at how to use extra credit hardware assignments to show the applications of the ideas learned in class in practice [11].

3.5 Teaching to Diverse Fields

Recent articles also point to the importance of successful student-teacher and student-student interactions and provide solutions on how to make them more meaningful. Barker and Gruning look at student-teacher relationships generally in classes and argue for a more representative way for students to provide feedback to the teacher on not only their satisfaction with the course but also their understanding of the material covered, so that misunderstandings can be corrected before the class moves to a new topic [12]. They advocate for equally-weighted responses from all students, using tools such as routine and in-depth questionnaires as well as clickers for in-class "votes", rather than sporadic and biased student feedback garnered through evaluations, direct requests, and nonverbal behavior and performance.

In terms of student-student relationships, recent research looks at the benefit of cooperative interactions and group work. Zarb introduces guidelines for students to pair program successfully together based on examples set by professional pair programmers [14]. Spacco *et al.* study the effects of a peer instruction environment in which students first learn material independently and later use class time to collectively address misunderstandings and student questions [15]. They found that students in the peer instruction class overall were more successful on the final exam, the final measure of student learning, in relation to students who were learning the same material in a comparable class taught in the traditional lecture style. Blaheta argues not for cooperation in class but instead for cooperative homework assignments [16]. Because the class involved in the study was an advanced one, the homework was algorithms-based and each problem did not have a single correct answer. Blaheta proposes the implementation of this system not only decreases the workload of the professor but also produced more students satisfied with the course and more motivated to both complete and revise the assignments.

4 Concluding Remarks

Although computer science pedagogy is changing in different ways, the common theme of this past decade is about expanding—showing students more about the applications, reaching more students earlier, before they get to university, and reforming computer science communities where hacker culture and individuality has been rampant. While most of these changes have yet to be executed on larger platforms, this theme of extending computer science will no doubt continue beyond these studies and hopefully result in long-term changes in how computer science is taught, understood, and experienced.

References

[1] J. Margolis and A. Fisher, *Unlocking the Clubhouse: Women in Computing*, pp. 15–75 and 129–141. The MIT Press, 2002.

[2] J. P. Holdren, E. Lander, W. Press, M. Savitz, R. Bierbaum, C. Cassel, C. Chyba, S. J. G. Jr., M. Gorenberg, S. A. Jackson, R. C. Levin, C. Mirkin, M. Molina, E. J. Moniz, C. Mundie, E. Penhoet, B. Schaal, E. Schmidt, D. Schrag, D. E. Shaw, A. Zewail, D. Stine, D. Evers, and A. H. Scholz, "Report to the president; engage to excel: Producing one million additional college graduates with degrees in science, technology, engineering and mathematics," tech. rep., President's Council of Advisors on Science and Technology, 2012.

[3] Bureau of Labor Statistics, U.S. Department of Labor, "Occupational outlook handbook, 2014-15 edition." http://www.bls.gov/ooh/computer-and-information-technology/computer-and-information-research-scientists.htm.

[4] R. Tytler, J. Osborne, G. Williams, K. Tytler, and J. C. Clark, "Opening up pathways: Engagement in stem across the primary-secondary school transition," tech. rep., Australian Department of Education, Employment and Workplace Relations, 2008.

[5] A. J. Martin and H. W. Marsh, “Academic buoyancy: Towards an understanding of students’ everyday academic resilience,” *Journal of School Psychology*, vol. 46, pp. 53–83, 2008.

[6] G. Wiggins and J. McTighe, *A simple algorithm for homeomorphic surface reconstruction*, pp. 13–55. Merrill Education/Prentice Hall, 2 ed., 2005.

[7] M. Goldwber, R. Davoli, J. Barr, S. Mann, S. Portnoff, T. Clear, and E. Patisas, “A framework for enhancing the social good in computing education: A values approach,” *ITiCSE-WGR ’12 Proceedings of the Final Reports on Innovation and Technology in Computer Science Education 2012 Working Groups*, pp. 16–38, 2012.

[8] C. Rader, D. Hakarinen, B. M. Moskal, and K. Hellman, “Exploring the appeal of socially relevant computing: Are students interested in socially relevant problems?,” *SIGCSE ’11 Proceedings of the 42th ACM Technical Symposium on Computer Science Education*, pp. 423–428, 2011.

[9] F. Heintz and I. E. Klein, “The design of sweden’s first 5-year computer science and software engineering program,” *SIGCSE ’14 Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pp. 199–204, 2014.

[10] R. E. Anderson, M. D. Ernst, R. Ordóñez, P. Pham, and B. Tribelhorn, “A data programming cs1 course,” *SIGCSE ’15 Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pp. 150–155, 2015.

[11] R. Balasubramanian, Z. York, M. Dorran, A. Biswaws, T. Girgin, and K. Sankaralingam, “Hands-on introduction to computer science at the freshman level,” *SIGCSE ’14 Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pp. 235–240, 2014.

[12] L. Barker and J. Gruning, “The student prompt: Student feedback and change in teaching practices in postsecondary computer science,” *Frontiers in Education Conference 2014 IEE*, pp. 1–8, 2014.

[13] N. Shah, C. M. Lewis, R. Caires, N. Khan, A. Quereshi, D. Ehsanipour, and N. Gupta, “Building equitable computer science classrooms: Elements of a teaching approach,” *SIGCSE ’13 Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, pp. 263–268, 2013.

[14] M. Zarb, “Industry-inspired guidelines to improve students’ pair programming communication,” *ITiCSE ’13 Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pp. 135–140, 2013.

[15] J. Spacco, J. Parris, and B. Simon, “How we teach impacts student learning: Peer instruction vs. lecture in cs0,” *SIGCSE ’13 Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, pp. 199–204, 2013.

[16] D. Blaheta, “Reinventing homework as cooperative, formative assessment,” *SIGCSE ’14 Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, pp. 301–206, 2014.

[17] Australian Bureau of Statistics, “Primary and secondary education,” *Year Book Australia, 2009-10*, vol. 1301, 2010.