Approaches to Broadening Participation with AP Computer Science Principles

Audrey Le Meur
lemeu001@umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

The Advanced Placement Computer Science Principle (AP CSP) course framework was created with the intention of broadening participation in computing. Research has produced mixed results on whether or not the framework succeeds in that goal [14, 19]. Given that teachers have significant freedom in how they choose to teach the AP CSP content, students can have a variety of experiences that may or may not impact their continued participation in CS. In this paper, I compare four different approaches to the AP CSP framework by examining their impact on AP exam scores, self-efficacy and confidence, belongingness and identity, and persistence and interest, to examine how these approaches might impact those traditionally underrepresented in CS. I also discuss how social and curricular interventions may differ in outcomes.

Keywords: computer science education, K-12 education, broadening participation

1 Introduction

The Advanced Placement Computer Science Principles (AP CSP) framework and exam was introduced in 2016 to serve more diverse populations than the older Computer Science A (AP CSA) framework [14]. Specifically, the College Board sought to create a course that would cover broader CS concepts, aimed towards students who are not intending to major in CS and students who are traditionally underrepresented in computing. In contrast to AP CSA, which focuses on programming and algorithmic thinking, AP CSP expands the curriculum to include the study of computational solution design, abstraction, code analysis, computing innovations, and responsible computing [3].

Research has produced mixed results on whether AP CSP achieves its goal in broadening participation in computing. The College Board found that students who had taken AP CSP were more likely to have majored in CS in college than students who had not taken AP CSP [24]. This result also held up among female, Black, Hispanic, and first-generation students respectively. That said, Sax et al. found that students who took an AP CSP course were less likely to want to major in or have a career in CS than students who took AP CSA [19]. These students were also less confident in their computing abilities. It is therefore unclear whether the AP CSP framework itself contributes to future participation for those traditionally underrepresented in CS.

Since AP CSP is only a framework, teachers have significant freedom to choose how they teach the content. This paper looks at specific approaches that teachers can take to make an AP CSP course more accessible to diverse audiences.

2 Background

To evaluate the outcomes of a particular AP CSP approach in relation to the other approaches, it can be helpful to look at the results of individual studies using a shared set of criteria. Since the different studies do not all use the exact same criteria, but often report similar results, I aggregate them into four criteria: exam results, self-efficacy and confidence, belongingness and identity, and persistence and interest.

2.1 Exam Results

The College Board evaluates AP CSP students based on two assessments: a multiple-choice exam and a project called the "Create performance task." The multiple-choice exam consists of 70 questions over 120 minutes which counts for 70% of the student's final score. For the "Create performance task," Students develop code either independently or with others before individually making a video to explain and demonstrate their project and completing written prompts. Students are allotted at least twelve hours in class to complete the "Create performance task." The project counts for the remaining 30% of the student's score [2]. The resulting score is on a scale from 1 to 5, where a 3 is considered passing the exam and the student may be awarded college credit [3].

2.2 Self-efficacy and Confidence

Bandura defines self-efficacy as "people's beliefs about their capabilities to produce designated levels of performance that exercise influence over events that affect their lives [4]." In computer science education research, we look at people's beliefs in their ability to complete tasks that require a computer or that require computational thinking [5, 20].

In contrast with self-efficacy, confidence is defined much more loosely. Hoegh evaluates "students' confidence in their own ability to learn computer science skills", but doesn't provide a firm definition for confidence [10]. In this case, we

can consider confidence to be synonymous with the *strength* of self-efficacy, which Compeau defines as the confidence an individual has in their ability to complete a task [5]. Therefore, a student's confidence in their computer science skills is the strength of their computer science self-efficacy.

2.3 Belongingness and Identity

There are several ways to measure a student's sense of belonging in relation to their identity within computer science. *Belongingness* is the "feeling that you fit in and there are others like you in CS [17]." Alternatively, we can look at students' views of their own identity in comparison to the identities of STEM professionals [18]. Researchers have also examined how students see their race and/or gender as an indicator of success in CS [6]. Each of these evaluations generally tell us about how students' identity impacts their participation in CS.

2.4 Persistence and Interest

We are also interested in knowing if students will continue pursuing CS. We can measure this two ways: *persistence* (whether students want to continue studying or doing CS) and *interest* (whether students are interested in CS content). We can measure persistence by asking students if they plan on taking more CS courses, majoring or minoring in CS in college or having a career in CS. Measuring interest is usually more straightforward; we can ask students how interested they are in CS.

3 Approaches

In this paper, I distinguish between two types of approaches to teaching AP CSP. First, a *social* approach is an approach in which the researchers hope to improve outcomes by changing the way that students interact with others within the classroom. Second, a *curricular* approach is an approach in which the content of the course is designed to be appealing and inclusive of a diverse audience. I will describe two social approaches and two curricular approaches and discuss the methods and results of studies evaluating their efficacy. Due to the nature of an AP framework, all studies evaluated students using their AP exam scores.

3.1 Supporting Students through Peer Learning Communities

One approach to broadening participation is to create social environments in which traditionally underrepresented students feel included and supported. Peer Learning Communities can provide students with valuable relationships with other peers, near-peers, and teachers of the same gender and of a similar racial and/or ethnic background. The LEGACY project provided Black female students with a culturally-relevant, project-based preparatory curriculum for AP CSP in the context of a Peer Learning Community (PLC) [6].

3.1.1 Methods. Escobar et al. recruited 40 young Black women in a southern U.S. state who were enrolled in an AP CSP course for the following school year [6]. The students participated in a five-day long course introducing the key concepts of the AP CSP framework taught by Black female teachers. Students later presented a project to their peers in another two-day session. Both sessions included social activities and opportunities to meet Black female role-models. During the school year, teachers and students stayed in touch through a Moodle site where they could post questions and connect with other students. Students also had the opportunity to meet their peers and teachers in-person and virtually to discuss AP CSP content and practice for the exam.

Students completed assessments before and after the summer portion of the program and the end of the school year. The researchers used a computational thinking self-efficacy questionnaire, adapted from Weese and Feldhausen [20], to evaluate students' self-efficacy. They also evaluated students' sense of identity in CS using the Gender and Racial Attitudes Toward Computing inventory. An adaptation (called CS-PIO) of the STEM Professional Identity Overlap (STEM-PIO) [18] was used to further evaluate students' sense of identity and examine students' persistence.

3.1.2 Results. 87.5% of students who participated in the LEGACY PLC and took the AP exam passed. This is significantly higher than the national average of all students, male students, White male students, Black female students and Black male students [6]. Exam scores were positively correlated with attending more PLC sessions. Students saw increases in self-efficacy for algorithmic thinking and control flow both throughout the summer and the school year, but saw no difference in their confidence in the importance of computing or in their ability to organize complex tasks. Students generally had more positive attitudes about the ability of people from all racial backgrounds and women to succeed in CS as the program progressed. Students' self-identification with their personal image of a CS professional also increased over the course of the study. Finally, 59% of students said they intended to major or minor in CS in college.

3.2 Encouraging Cooperative Learning

Classroom management can also be a way through which teachers can manipulate the social environment in which their students learn. Gray et el. sought to examine how the use of cooperative learning affects outcomes in an AP CSP classroom [8]. Kagan and Kagan define *cooperative learning* to be an instructional approach which follows four principles: positive interdependence, individual accountability, equal participation, and simultaneous interaction [12]. Positive interdependence means students are working together "on the same side," while individual accountability means every student is required to publicly and individually demonstrate

Table 1. Cooperative Learning Structures used in [8]

Name	Description
Round Robin [12]	Students each speak for an equal amount of time in response to a prompt.
Debate Team Carousel [9]	Students are asked to write down their opinion on a question. They then pass their papers around and give reasons for and against other students' opinions.
Jot Thoughts [13]	Students individually brainstorm a topic for a set amount of time while writing their ideas down on slips of paper and laying them around the table. They then processes their ideas by discussing and rearranging their slips of paper.

their learning. Additionally, students have to participate relatively equally (equal participation) and must be actively engaged for the entire duration of the activity (simultaneous interaction).

3.2.1 Methods. Gray et al. consider pair programming to be an example of cooperative learning in the CS specific context [8]. Other structures are described in Table 1. Note that Gray and al. study used other structures in their study and these are just a few examples.

The study provided 27 teachers with a professional development course about using collaborative learning structures in an AP CSP classroom. Teachers filled out surveys describing how they used CL in their class. The researchers used these surveys to calculate an "opportunity to learn collaboratively" (OTLC) score for each class. 6,492 students in classes where the teacher had completed the professional development also completed a computing self-efficacy survey adapted from Compeau and Higgins [5].

3.2.2 Results. The students in the cooperative learning (CL) classrooms passed the AP exam at a greater rate than the national average (76.6% vs 72.3%). Specifically, underrepresented minority students passed the AP exam at a significantly greater rate than the national average (66.3% vs 55%). Use of CL structures (OTLC) was only a predictor of AP score in classrooms where the teacher had less than three years experience teaching CS. Gray et al. also found no significant gains in self-efficacy for students in an AP CSP course using CL structures [8].

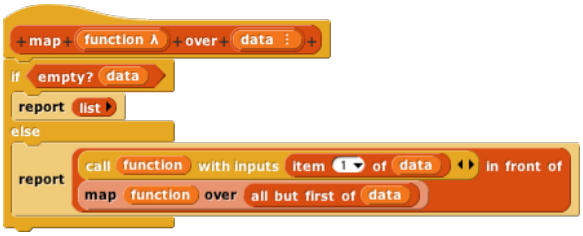

Figure 1. An example block of Snap*!* code [1].

3.3 Finding the *Beauty and Joy in Computing*

In line with AP CSP's larger goal of broadening participation in CS, the designers of the *Beauty and Joy of Computing* (BJC) curriculum aimed to create an environment where students could freely develop their competence, confidence, and creativity [7]. They made two major design choices to support this goal: use of a visual programming language and an emphasis on the social implications of technology. BJC uses Snap*!*, a block-based programming language visually similar to Scratch [16] which allows for more advanced programming skills such as "recursion, higher-order functions, complex data structures, object-oriented programming, and lambda [7]." For example, Figure 1 shows a higher-order function implemented in Snap*!* which utilizes a map data structure. The visual nature of the language allows students to focus on learning programming concepts rather than spending too much time on syntax, Further, block-based languages have been linked with greater learning gains and interest in CS in comparison to text-based languages [21]. Students also learned about the social implications of technology to build their agency within the future of computer technology [7].

3.3.1 Methods. The BJC team provided teachers in NYC with professional development for teaching the BJC curriculum [17]. These teachers taught 311 students at 24 NYC high schools. Students completed a CS attitude survey adapted from three previous works [10, 15, 22] both before and after they took the course. These surveys evaluated the students' confidence, interest, belongingness and identity in relation to CS [17]. The researchers also received AP score data from the NYC Department of Education on all 2,854 students in NYC who had taken the AP CSP exam, regardless of whether they had been taught using the BJC curriculum [17].

3.3.2 Results. Mark et al. analyzed AP scores by comparing the scores of BJC students with those of non-BJC students in NYC. In their initial analysis, 67.2% of BJC students passed the AP exam while 72.7% of NYC students passed the exam [17]. The researchers decided to exclude the data of two schools who were demographically distinct from most NYC schools. These schools had fewer students from underrepresented minority groups (URG), fewer students with individualized education plans (IEP), and fewer students in poverty. Additionally, these schools had significantly higher

Table 2. Results of the four studies.

Approach	Exam Results	Self-efficacy and Confidence	Belongingness and Identity	Persistence and Interest
Social Approaches				
Supporting Students through Peer Learning Communities [6]	✓	✓[1]	✓	✓
Encouraging Cooperative Learning [8]	✓[2]	✗	N/A	N/A
Curricular Approaches				
Finding the Beauty and Joy in Computing [7, 17]	✓	✓	✓[1]	✗
Engaging Students through Mobile Computing [11]	✓	N/A	N/A	✓[13]

[1] Gains only seen in some measures
[2] Gains only seen for students with less experienced teachers
[3] Gains only seen for underrepresented minority students but not female students

pass rates in comparison to other NYC schools. Excluding these two schools, 1,192 students from the remaining 75 schools took the AP exam. Under their new analysis, BJC students passed the AP exam at a significantly higher rate than non-BJC students (54.2% vs. 37.7%).

Students saw significant gains in confidence after taking a BJC course [7]. They also saw significant gains in identity but not in belongingness. Students did not gain interest in CS after taking the course. Female and URG students generally had similar differences in pre and post surveys as male and non-URG students.

3.4 Engaging Students through Mobile Computing

The Mobile Computer Science Principles (Mobile CSP) curriculum sought to engage traditionally underrepresented students through mobile app design and programming [11]. Similarly to BJC, Mobile CSP used a visual-programming language called App Inventor [23] to make programming more accessible. In the first half of the course, students used App Inventor to learn about programming and algorithms by creating mobile applications. The second half of the course focused on non-programming concepts, including social impact. Finally, students completed their "Create performance task" by creating a mobile app which they were encouraged to build in a "socially useful" way. The goal was for students to connect CSP content to their own lives through a familiarity of mobile devices and allowing them to choose social topics that were relevant to their own community.

3.4.1 Methods. 275 participating teachers received about 100 hours of professional development on how to teach Mobile CSP [11]. Students in their classes completed a survey before and after the course which asked about their attitudes and interest in CS.

3.4.2 Results. Students that were taught the Mobile CSP curriculum did pass the exam at a slightly higher rate than the national average (78% vs. 74% in 2017 and 76% vs. 69% in 2018), yet, the authors did not note if this result was statistically significant [11]. Women, Hispanic/Latino, and multiracial students who completed Mobile CSP performed better on the AP exam than the national average for their respective groups both years that the curriculum was administered. Black/African American students performed better than the national average for Black/African American students only on the 2017 exam.

Overall, a majority of (59%) of students reported feeling more interested in CS after taking the Mobile CSP course [11]. More specifically, majorities of female (56%), Black (56%), and Latino (66%) students expressed more interest in CS as a result of the course. Additionally, students generally showed a desire to continue studying CS. 64% of students said they were interested in majoring in CS or pursuing CS as a career and a similar proportion (62%) of underrepresented minority students held the same view. That said, female students expressed a desire to continue in CS at a lower rate (48%).

4 Conclusion

4.1 Discussion

All four approaches resulted in improved AP exam scores for at least some subset of students (See Table 2). No approach saw improvements across all four criteria. For self-efficacy and confidence, belongingness and identity, and persistence and interest, there was improvement in at least one social and

one curricular approach. This indicates that neither social or curricular approaches have an advantage when it comes to improving measures that impact participation.

Escobar et al. argued that "curriculum alone does not drive engagement and interest among those underrepresented in CS [6]." While the BJC curriculum did not improve interest, the Mobile CSP curriculum did increase persistence for underrepresented minority students. The Mobile CSP did not have the same effect on female students' persistence. Contrary to Escobar, this suggests that the content of the curriculum may influence students' interest. The social nature of Mobile CSP and the Peer-Learning Communities approach indicate that social connections may be key to engaging female students. There seems to be increased interest from female students when they have a social connection between CS and their peers or their community.

It may be necessary to use multiple strategies, mixing social approaches with curriculum, to improve outcomes for underrepresented groups. For example, Peer Learning Communities could be implemented with either the Mobile CSP or BJC curriculum to improve students' persistence and interest.

4.2 Limitations

The variety of methods used across the studies makes it difficult to compare numerical results between studies. Each study did their own statistical analysis using different instrumentation. Additionally, not all studies were conducted during the same academic year, with the Peer-Learning Communities study being conducted during the COVID-19 pandemic. That said, it is still possible to compare results by looking at the general trend rather than the number.

The Mobile CSP study did not analyze whether the data showed any statistically significant differences, which makes knowing if the curriculum actually had a positive effect on AP scores, interest, and persistence difficult. It would be preferable for the authors to clarify how they did their statistical analysis, so we can be more sure about the significance of their results.

All studies were conducted with the United States so it is unclear if these results would extend to international educational contexts. While the Cooperative Learning and Mobile Computing studies engaged teachers and students across the United States, the Peer-Learning Communities and BJC studies were concentrated in one community.

4.3 Future Work

Many of these studies end when the student has completed the AP exam. Future work could look at undergraduate students who took the AP CSP exam and the relationship between the AP curriculum they completed and their current CS participation. We could also examine other AP CSP curricula developed commercially, such as Apple's Develop In Swift curriculum, Microsoft's MakeCode curriculum, or Carnegie Learning's Zulama curriculum.

Acknowledgments

Special thanks to my advisor, Professor K.K. Lamberty, for her feedback and support. I would also like to thank my senior seminar instructor, Professor Elena Machkasova, for guiding my peers and I through the course. Finally, thanks to Ariel Cordes for reviewing my paper and Professor Kiel Harell for providing me with sources.

References

[1] [n.d.]. Unit 8 lab 4: Building higher order functions. https://bjc.edc.org/bjc-r/cur/programming/8-recursive-reporters/4-building-higher-order-functions/2-generalizing-the-map-pattern.html?topic=nyc_bjc%2F8-recursive-reporters.topic&course=bjc4nyc.html&novideo&noassignment

[2] 2020. AP Computer Science Principles: Course and Exam Description. https://apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf

[3] 2022. About AP scores. https://apstudents.collegeboard.org/about-ap-scores

[4] Albert Bandura. 1994. Self-efficacy. *Encyclopedia of human behavior* 4 (1994), 71–81.

[5] Deborah R Compeau and Christopher A Higgins. 1995. Computer Self-Efficacy: Development of a Measure and Initial Test. , 189-211 pages. Issue 2.

[6] Martha Escobar, Jeff Gray, Kathleen Haynie, Mohammed A. Qazi, Yasmeen Rawajfih, Pamela McClendon, Donnita Tucker, and Wendy Johnson. 2021. Engaging Black Female Students in a Year-Long Preparatory Experience for AP CS Principles. *SIGCSE 2021 - Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* May 2020 (2021), 706–712. https://doi.org/10.1145/3408877.3432560

[7] Paul Goldenberg, June Mark, Brian Harvey, Al Cuoco, and Mary Fries. 2020. Design principles behind beauty and joy of computing. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE* (2020), 220–226. https://doi.org/10.1145/3328778.3366794

[8] Jeff Gray, Owen Astrachan, Kathy Haynie, Chinma Uche, Siobhan Cooney, Fran Trees, and Richard Kick. 2019. Infusing cooperative learning into AP computer science principles courses to promote engagement and diversity. *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (2019), 1190–1196. https://doi.org/10.1145/3287324.3287421

[9] William Himmele and Pérsida Himmele. 2012. How to know what students know. *Educational Leadership* 70, 1 (2012), 58–62. https://pdo.ascd.org/LMSCourses/PD15OC007M/media/M2-TPTReading2.pdf

[10] Andrew Hoegh and Barbara M. Moskal. 2009. Examining science and engineering students' attitudes toward computer science. *Proceedings - Frontiers in Education Conference, FIE*. https://doi.org/10.1109/FIE.2009.5350836

[11] Beryl Hoffman, Ralph Morelli, and Jennifer Rosato. 2019. Student Engagement is Key to Broadening Participation in CS. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. 1123–1129. https://doi.org/10.1145/3287324.3287438

[12] Spencer Kagan and Miguel Kagan. 2009. *Kagan Cooperative Learning*. Kagan Publishing.

[13] Spencer Kagan, Miguel Kagan, and Laurie Kagan. 2015. *59 Kagan Structures - Proven Engagement Structures*. Kagan Publishing.

[14] Richard Kick and Frances P. Trees. 2015. AP CS Principles: Engaging, Challenging, and Rewarding. *ACM Inroads* 6, 1 (Feb 2015), 42–45. https://doi.org/10.1145/2710672

[15] Clayton Lewis, Michele H. Jackson, and William M. Waite. 2010. Student and Faculty Attitudes and Beliefs about Computer Science. *Commun. ACM* 53, 5 (May 2010), 78–85. https://doi.org/10.1145/1735223.1735244

[16] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. *ACM Trans. Comput. Educ.* 10, 4, Article 16 (Nov 2010), 15 pages. https://doi.org/10.1145/1868358.1868363

[17] June Mark and Kelsey Klein. 2019. Beauty and joy of computing: 2016-17 findings from an AP CS principles course. *SIGCSE 2019 - Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (2019), 627–633. https://doi.org/10.1145/3287324.3287375

[18] Melissa M. McDonald, Virgil Zeigler-Hill, Jennifer K. Vrabel, and Martha Escobar. 2019. A Single-Item Measure for Assessing STEM Identity. *Frontiers in Education* 4, July (2019), 1–15. https://doi.org/10.3389/feduc.2019.00078

[19] Linda J. Sax, Kaitlin N.S. Newhouse, Joanna Goode, Max Skorodinsky, Tomoko M. Nakajima, and Michelle Sendowski. 2020. Does ap cs principles broaden participation in computing? an analysis of apcsa and apcsp participants. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE* (2020), 542–548. https://doi.org/10.1145/3328778.3366826

[20] Joshua Levi Weese and Russell Feldhausen. 2017. STEM Outreach: Assessing Computational Thinking and Problem Solving. In *2017 ASEE Annual Conference & Exposition*. ASEE Conferences, Columbus, Ohio. https://peer.asee.org/28845

[21] David Weintrop and Uri Wilensky. 2017. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education* 18, 1 (2017), 1–25. https://doi.org/10.1145/3089799

[22] Eric N. Wiebe, L. Williams, K. Yang, and C. Miller. 2003. Computer Science Attitude Survey. *Computer Science* January 2003 (2003), 5. arXiv:arXiv:1011.1669v3 http://www4.ncsu.edu/$\sim$wiebe/www/articles/prl-tr-2003-1.pdf

[23] David Wolber. 2011. App Inventor and Real-World Motivation. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, USA) *(SIGCSE '11)*. Association for Computing Machinery, New York, NY, USA, 601–606. https://doi.org/10.1145/1953163.1953329

[24] Jeff Wyatt, Jing Feng, and Maureen Ewing. 2020. AP Computer Science Principles and the STEM and Computer Science Pipelines. December (2020). https://apcentral.collegeboard.org/pdf/ap-csp-and-stem-cs-pipelines.pdf?course=ap-computer-science-principles